딥러닝 에스프레소

딥러닝을 위한 베이지안 통계 Day3

VAE, BBB, MCDropout, Uncertainty, GAN

2020

멀티캠퍼스

박수철

v1.0

Variational Auto-Encoders

Variational Auto-Encoders

- VAE (Variational Auto-Encoders)는 GAN과 더불어 딥러닝의 대표적인 생성모델(Generative Models)입니다.
- Auto-Encoder가 Z-variables의 분포를 알 수 없는 단점을 개선한 모델입니다.
- Z-variables의 분포를 Isotropic Gaussian 등으로 제한하여 자유롭게 컨트롤할 수 있다는 특징을 가집니다.
- VAE는 또한 Probabilistic PCA의 conditional distribution이 linear transform 모델에 제한된다는 단점을 개선한 모델로 생각할 수 있습니다.
- Neural Networks의 non-linearity를 posterior와 conditional distribution을 형성하는데 사용해 기존 머신러닝에서 보여줄 수 없었던 유연한 표현 능력을 가집니다.

Variational Auto-Encoders

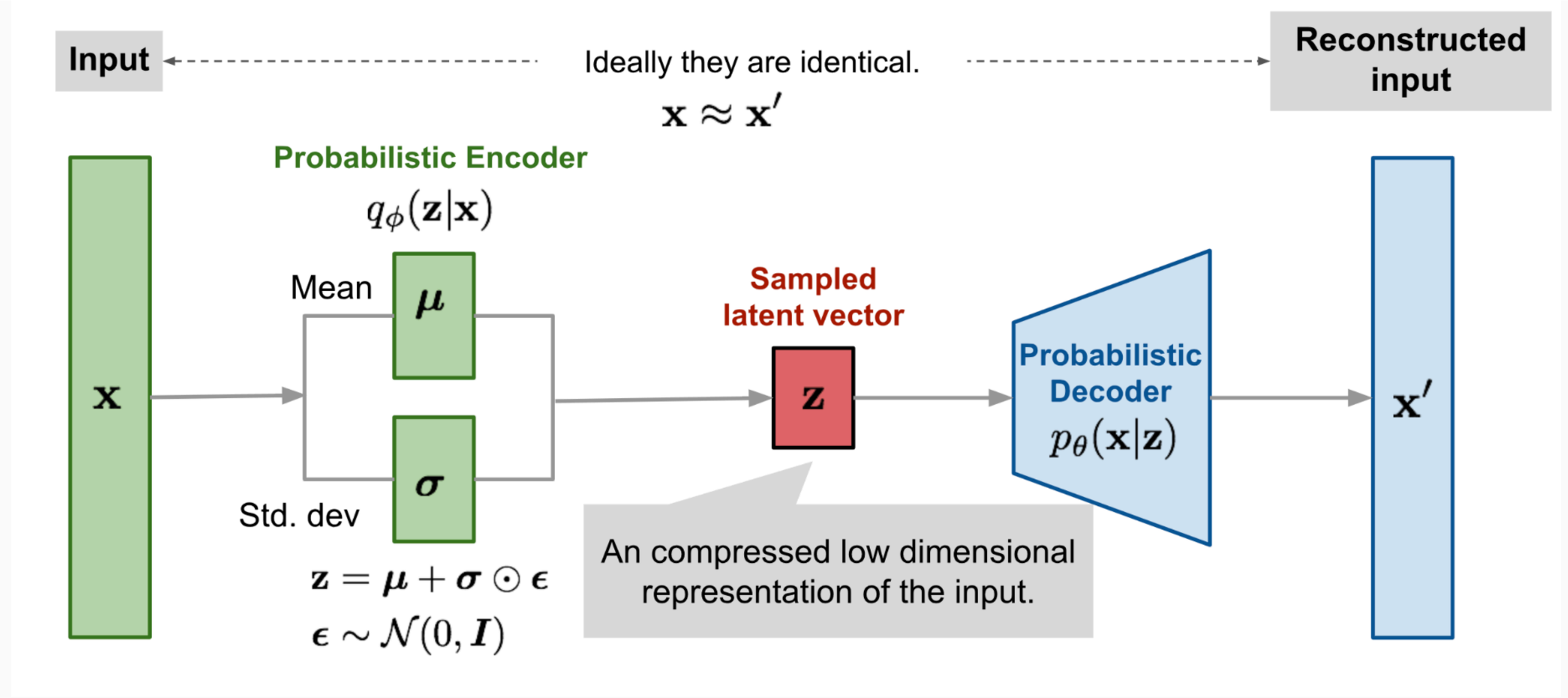


Fig. 9. Illustration of variational autoencoder model with the multivariate Gaussian assumption.

https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html

Variational Auto-Encoders

Called a variational method because it derives from the **Calculus of Variations**.

Functions:

- Variables as input, output is a value.
- Full and partial derivatives $\frac{df}{dx}$
- E.g., Maximise likelihood p(x|θ) w.r.t. parameters θ

Functionals:

- Functions as input, output is a value.
- Functional derivatives $\frac{\delta F}{\delta f}$
- E.g., Maximise the entropy H[*p(x)*] w.r.t. *p(x)*

We exploit both types of derivatives in variational inference.

Shakir Mohamed. Tutorial on Variational Inference for Machine Learning. https://www.shakirm.com/papers/VITutorial.pdf

Variational Auto-Encoders

Abstract

How can we perform efficient inference and learning in directed probabilistic models, in the presence of continuous latent variables with intractable posterior distributions, and large datasets? We introduce a stochastic variational inference and learning algorithm that scales to large datasets and, under some mild differentiability conditions, even works in the intractable case. Our contributions is two-fold. First, we show that a reparameterization of the variational lower bound yields a lower bound estimator that can be straightforwardly optimized using standard stochastic gradient methods. Second, we show that for i.i.d. datasets with continuous latent variables per datapoint, posterior inference can be made especially efficient by fitting an approximate inference model (also called a recognition model) to the intractable posterior using the proposed lower bound estimator. Theoretical advantages are reflected in experimental results.

Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes

Variational Auto-Encoders

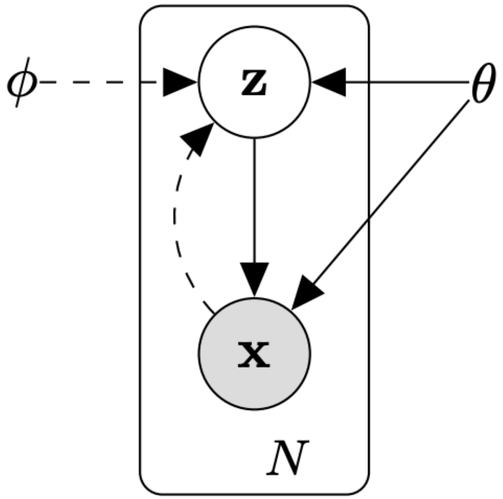


Figure 1: The type of directed graphical model under consideration. Solid lines denote the generative model $p_{\boldsymbol{\theta}}(\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$, dashed lines denote the variational approximation $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ to the intractable posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$. The variational parameters $\boldsymbol{\phi}$ are learned jointly with the generative model parameters $\boldsymbol{\theta}$.

Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes

Variational Auto-Encoders

- VAE에서 해결하고자 하는 문제의 정의

2.1 Problem scenario

Let us consider some dataset $\mathbf{X} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$ consisting of N i.i.d. samples of some continuous or discrete variable $\mathbf{x}$. We assume that the data are generated by some random process, involving an unobserved continuous random variable $\mathbf{z}$. The process consists of two steps: (1) a value $\mathbf{z}^{(i)}$ is generated from some prior distribution $p_{\boldsymbol{\theta}^*}(\mathbf{z})$; (2) a value $\mathbf{x}^{(i)}$ is generated from some conditional distribution $p_{\boldsymbol{\theta}^*}(\mathbf{x}|\mathbf{z})$. We assume that the prior $p_{\boldsymbol{\theta}^*}(\mathbf{z})$ and likelihood $p_{\boldsymbol{\theta}^*}(\mathbf{x}|\mathbf{z})$ come from parametric families of distributions $p_{\boldsymbol{\theta}}(\mathbf{z})$ and $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$, and that their PDFs are differentiable almost everywhere w.r.t. both $\boldsymbol{\theta}$ and $\mathbf{z}$. Unfortunately, a lot of this process is hidden from our view: the true parameters $\boldsymbol{\theta}^*$ as well as the values of the latent variables $\mathbf{z}^{(i)}$ are unknown to us.

Very importantly, we *do not* make the common simplifying assumptions about the marginal or posterior probabilities. Conversely, we are here interested in a general algorithm that even works efficiently in the case of:

Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes

Variational Auto-Encoders

- VAE에서 문제를 해결함에 있어 부딪히는 난관

1. *Intractability*: the case where the integral of the marginal likelihood $p_{\boldsymbol{\theta}}(\mathbf{x}) = \int p_{\boldsymbol{\theta}}(\mathbf{z}) p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\, d\mathbf{z}$ is intractable (so we cannot evaluate or differentiate the marginal likelihood), where the true posterior density $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) p_{\boldsymbol{\theta}}(\mathbf{z}) / p_{\boldsymbol{\theta}}(\mathbf{x})$ is intractable (so the EM algorithm cannot be used), and where the required integrals for any reasonable mean-field VB algorithm are also intractable. These intractabilities are quite common and appear in cases of moderately complicated likelihood functions $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$, e.g. a neural network with a nonlinear hidden layer.

2. *A large dataset*: we have so much data that batch optimization is too costly; we would like to make parameter updates using small minibatches or even single datapoints. Sampling-based solutions, e.g. Monte Carlo EM, would in general be too slow, since it involves a typically expensive sampling loop per datapoint.

Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes

Variational Auto-Encoders

- 문제를 해결하기 위한 방법의 개요

For the purpose of solving the above problems, let us introduce a recognition model $q_{\phi}(\mathbf{z}|\mathbf{x})$: an approximation to the intractable true posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$. Note that in contrast with the approximate posterior in mean-field variational inference, it is not necessarily factorial and its parameters $\boldsymbol{\phi}$ are not computed from some closed-form expectation. Instead, we'll introduce a method for learning the recognition model parameters $\boldsymbol{\phi}$ jointly with the generative model parameters $\boldsymbol{\theta}$.

From a coding theory perspective, the unobserved variables $\mathbf{z}$ have an interpretation as a latent representation or *code*. In this paper we will therefore also refer to the recognition model $q_{\phi}(\mathbf{z}|\mathbf{x})$ as a probabilistic *encoder*, since given a datapoint $\mathbf{x}$ it produces a distribution (e.g. a Gaussian) over the possible values of the code $\mathbf{z}$ from which the datapoint $\mathbf{x}$ could have been generated. In a similar vein we will refer to $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ as a probabilistic *decoder*, since given a code $\mathbf{z}$ it produces a distribution over the possible corresponding values of $\mathbf{x}$.

Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes

Variational Auto-Encoders

- ELBO(Evidence Lower BOund) 설명

2.2 The variational bound

The marginal likelihood is composed of a sum over the marginal likelihoods of individual datapoints $\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(N)}) = \sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})$, which can each be rewritten as:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) = D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}^{(i)})) + \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) \tag{1}$$

The first RHS term is the KL divergence of the approximate from the true posterior. Since this KL-divergence is non-negative, the second RHS term $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$ is called the (variational) *lower bound* on the marginal likelihood of datapoint i, and can be written as:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) \geq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[-\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) + \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})\right] \tag{2}$$

which can also be written as:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = -D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z})) + \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z})\right] \tag{3}$$

Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes

Variational Auto-Encoders

- 왜 ϕ에 대해 ELBO를 최대화하면 variational posterior $q_\phi(\mathbf{z}\,|\,\mathbf{x}^{(i)})$가 true posterior $p_\theta(\mathbf{z}\,|\,\mathbf{x}^{(i)})$를 근사하게 되는가?
-
$$p_\theta(\mathbf{z}\,|\,\mathbf{x}^{(i)}) = \frac{p_\theta(\mathbf{z})p_\theta(\mathbf{x}^{(i)}\,|\,\mathbf{z})}{p_\theta(\mathbf{x}^{(i)})}$$
-
$$\arg\max_\phi \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log \frac{p_\theta(\mathbf{z})p_\theta(\mathbf{x}^{(i)}\,|\,\mathbf{z})}{q_\phi(\mathbf{z}\,|\,\mathbf{x}^{(i)})}\right]$$

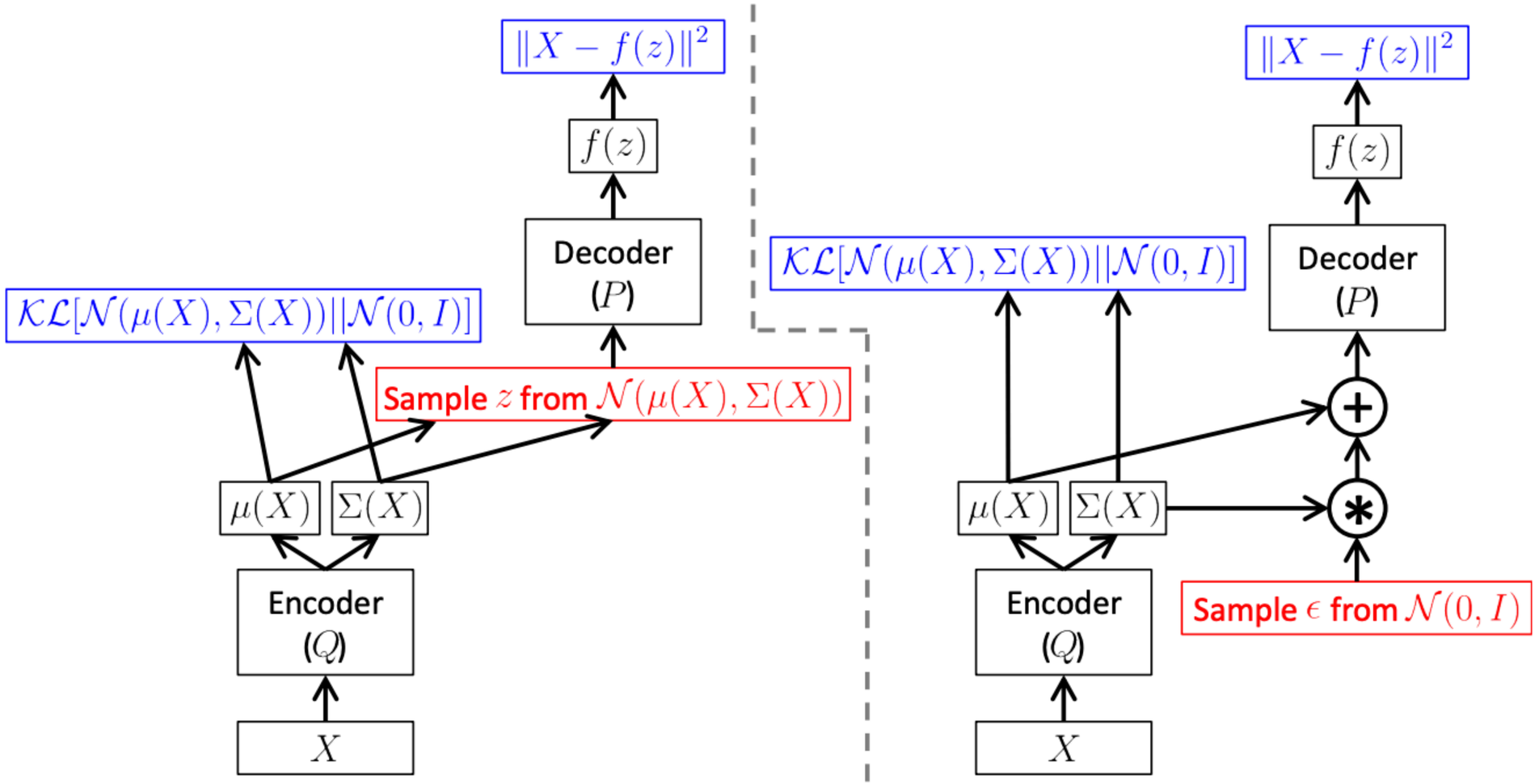
Variational Auto-Encoders

- ELBO를 ϕ에 대해 미분하고자 할 때 생기는 문제점 지적

We want to differentiate and optimize the lower bound $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$ w.r.t. both the variational parameters $\boldsymbol{\phi}$ and generative parameters $\boldsymbol{\theta}$. However, the gradient of the lower bound w.r.t. $\boldsymbol{\phi}$ is a bit problematic. The usual (naïve) Monte Carlo gradient estimator for this type of problem is: $\nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}\left[f(\mathbf{z})\right] = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}\left[f(\mathbf{z}) \nabla_{q_{\boldsymbol{\phi}}(\mathbf{z})} \log q_{\boldsymbol{\phi}}(\mathbf{z})\right] \simeq \frac{1}{L} \sum_{l=1}^{L} f(\mathbf{z}) \nabla_{q_{\boldsymbol{\phi}}(\mathbf{z}^{(l)})} \log q_{\boldsymbol{\phi}}(\mathbf{z}^{(l)})$ where $\mathbf{z}^{(l)} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})$. This gradient estimator exhibits exhibits very high variance (see e.g. [BJP12]) and is impractical for our purposes.

Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes

Variational Auto-Encoders



Carl Doersch, Tutorial on Variational Autoencoders

Variational Auto-Encoders

- Reparameterization trick과 Monte Carlo estimation을 이용하여 ELBO 계산

2.3 The SGVB estimator and AEVB algorithm

In this section we introduce a practical estimator of the lower bound and its derivatives w.r.t. the parameters. We assume an approximate posterior in the form $q_\phi(\mathbf{z}|\mathbf{x})$, but please note that the technique can be applied to the case $q_\phi(\mathbf{z})$, i.e. where we do not condition on $\mathbf{x}$, as well. The fully variational Bayesian method for inferring a posterior over the parameters is given in the appendix.

Under certain mild conditions outlined in section 2.4 for a chosen approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$ we can reparameterize the random variable $\tilde{\mathbf{z}} \sim q_\phi(\mathbf{z}|\mathbf{x})$ using a differentiable transformation $g_\phi(\boldsymbol{\epsilon}, \mathbf{x})$ of an (auxiliary) noise variable $\boldsymbol{\epsilon}$:

$$\tilde{\mathbf{z}} = g_\phi(\boldsymbol{\epsilon}, \mathbf{x}) \quad \text{with} \quad \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon}) \tag{4}$$

See section 2.4 for general strategies for chosing such an approriate distribution $p(\boldsymbol{\epsilon})$ and function $g_\phi(\boldsymbol{\epsilon}, \mathbf{x})$. We can now form Monte Carlo estimates of expectations of some function $f(\mathbf{z})$ w.r.t. $q_\phi(\mathbf{z}|\mathbf{x})$ as follows:

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[f(\mathbf{z})\right] = \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[f(g_\phi(\boldsymbol{\epsilon}, \mathbf{x}^{(i)}))\right] \simeq \frac{1}{L}\sum_{l=1}^{L} f(g_\phi(\boldsymbol{\epsilon}^{(l)}, \mathbf{x}^{(i)})) \quad \text{where} \quad \boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon}) \tag{5}$$

Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes

Variational Auto-Encoders

- SGVB(Stochastic Gradient Variational Bayes) A type

We apply this technique to the variational lower bound (eq. (2)), yielding our generic Stochastic Gradient Variational Bayes (SGVB) estimator $\widetilde{\mathcal{L}}^A(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) \simeq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$:

$$\widetilde{\mathcal{L}}^A(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = \frac{1}{L} \sum_{l=1}^{L} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}, \mathbf{z}^{(i,l)}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}^{(i,l)} | \mathbf{x}^{(i)})$$

$$\text{where} \quad \mathbf{z}^{(i,l)} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(i,l)}, \mathbf{x}^{(i)}) \quad \text{and} \quad \boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon}) \tag{6}$$

-> Bayes By Backprop.에서도 등장

Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes

Variational Auto-Encoders

- SGVB(Stochastic Gradient Variational Bayes) B type
  posterior와 prior 간의 KL-Divergence를 closed form으로 계산

Often, the KL-divergence $D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z}))$ of eq. (3) can be integrated analytically (see appendix B), such that only the expected reconstruction error $\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z})\right]$ requires estimation by sampling. The KL-divergence term can then be interpreted as regularizing $\boldsymbol{\phi}$, encouraging the approximate posterior to be close to the prior $p_{\boldsymbol{\theta}}(\mathbf{z})$. This yields a second version of the SGVB estimator $\widetilde{\mathcal{L}}^{B}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) \simeq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$, corresponding to eq. (3), which typically has less variance than the generic estimator:

$$\widetilde{\mathcal{L}}^{B}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = -D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z})) + \frac{1}{L}\sum_{l=1}^{L}(\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}^{(i,l)}))$$

$$\text{where} \quad \mathbf{z}^{(i,l)} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(i,l)}, \mathbf{x}^{(i)}) \quad \text{and} \quad \boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon}) \tag{7}$$

-> VAE에서 사용하는 일반적 방식

Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes

Variational Auto-Encoders

- SGVB를 이용한 AEVB(Auto-Encoding Variational Bayes) 알고리즘 pseudo code

Algorithm 1 Minibatch version of the Auto-Encoding VB (AEVB) algorithm. Either of the two SGVB estimators in section 2.3 can be used. We use settings $M = 100$ and $L = 1$ in experiments.

$\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Initialize parameters
repeat
  $\mathbf{X}^M \leftarrow$ Random minibatch of M datapoints (drawn from full dataset)
  $\boldsymbol{\epsilon} \leftarrow$ Random samples from noise distribution $p(\boldsymbol{\epsilon})$
  $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} \widetilde{\mathcal{L}}^M(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}^M, \boldsymbol{\epsilon})$ (Gradients of minibatch estimator (8))
  $\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Update parameters using gradients $\mathbf{g}$ (e.g. SGD or Adagrad [DHS10])
until convergence of parameters $(\boldsymbol{\theta}, \boldsymbol{\phi})$
return $\boldsymbol{\theta}, \boldsymbol{\phi}$

Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes

Variational Auto-Encoders

- Dataset 전체의 likelihood, batch size와 sampling 갯수

Given multiple datapoints from a dataset $\mathbf{X}$ with N datapoints, we can construct an estimator of the marginal likelihood lower bound of the full dataset, based on minibatches:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}) \simeq \widetilde{\mathcal{L}}^M(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}^M) = \frac{N}{M} \sum_{i=1}^{M} \widetilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) \tag{8}$$

where the minibatch $\mathbf{X}^M = \{\mathbf{x}^{(i)}\}_{i=1}^M$ is a randomly drawn sample of M datapoints from the full dataset $\mathbf{X}$ with N datapoints. In our experiments we found that the number of samples L per datapoint can be set to 1 as long as the minibatch size M was large enough, e.g. $M = 100$. Derivatives $\nabla_{\boldsymbol{\theta}, \boldsymbol{\phi}} \widetilde{\mathcal{L}}(\boldsymbol{\theta}; \mathbf{X}^M)$ can be taken, and the resulting gradients can be used in conjunction with stochastic optimization methods such as SGD or Adagrad [DHS10]. See algorithm 1 for a basic approach to compute the stochastic gradients.

Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes

Variational Auto-Encoders

- SGVB B-type의 직관적 해석

A connection with auto-encoders becomes clear when looking at the objective function given at eq. (7). The first term is (the KL divergence of the approximate posterior from the prior) acts as a regularizer, while the second term is a an expected negative reconstruction error. The function $g_{\boldsymbol{\phi}}(.)$ is chosen such that it maps a datapoint $\mathbf{x}^{(i)}$ and a random noise vector $\boldsymbol{\epsilon}^{(l)}$ to a sample from the approximate posterior for that datapoint: $\mathbf{z}^{(i,l)} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(l)}, \mathbf{x}^{(i)})$ where $\mathbf{z}^{(i,l)} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})$. Subsequently, the sample $\mathbf{z}^{(i,l)}$ is then input to function $\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}^{(i,l)})$, which equals the probability density (or mass) of datapoint $\mathbf{x}^{(i)}$ under the generative model, given $\mathbf{z}^{(i,l)}$. This term is a negative *reconstruction error* in auto-encoder parlance.

Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes

Variational Auto-Encoders

- VAE 모델의 구체적 예시

3 Example: Variational Auto-Encoder

In this section we'll give an example where we use a neural network for the probabilistic encoder $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ (the approximation to the posterior of the generative model $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$) and where the parameters $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ are optimized jointly with the AEVB algorithm.

Let the prior over the latent variables be the centered isotropic multivariate Gaussian $p_{\boldsymbol{\theta}}(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$. Note that in this case, the prior lacks parameters. We let $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ be a multivariate Gaussian (in case of real-valued data) or Bernoulli (in case of binary data) whose distribution parameters are computed from $\mathbf{z}$ with a MLP (a fully-connected neural network with a single hidden layer, see appendix C). Note the true posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ is in this case intractable. While there is much freedom in the form $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, we'll assume the true (but intractable) posterior takes on a approximate Gaussian form with an approximately diagonal covariance. In this case, we can let the variational approximate posterior be a multivariate Gaussian with a diagonal covariance structure[2]:

$$\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)}) = \log \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}^{(i)}, \boldsymbol{\sigma}^{2(i)}\mathbf{I}) \tag{9}$$

where the mean and s.d. of the approximate posterior, $\boldsymbol{\mu}^{(i)}$ and $\boldsymbol{\sigma}^{(i)}$, are outputs of the encoding MLP, i.e. nonlinear functions of datapoint $\mathbf{x}^{(i)}$ and the variational parameters $\boldsymbol{\phi}$ (see appendix C).

Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes

Variational Auto-Encoders

- ELBO 함수의 구체적 예시

As explained in section 2.4, we sample from the posterior $\mathbf{z}^{(i,l)} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})$ using $\mathbf{z}^{(i,l)} = g_{\boldsymbol{\phi}}(\mathbf{x}^{(i)}, \boldsymbol{\epsilon}^{(l)}) = \boldsymbol{\mu}^{(i)} + \boldsymbol{\sigma}^{(i)} \odot \boldsymbol{\epsilon}^{(l)}$ where $\boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. With $\odot$ we signify an element-wise product. In this model both $p_{\boldsymbol{\theta}}(\mathbf{z})$ (the prior) and $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ are Gaussian; in this case, we can use the estimator of eq. (7) where the KL divergence can be computed and differentiated without estimation (see appendix B). The resulting estimator for this model and datapoint $\mathbf{x}^{(i)}$ is:

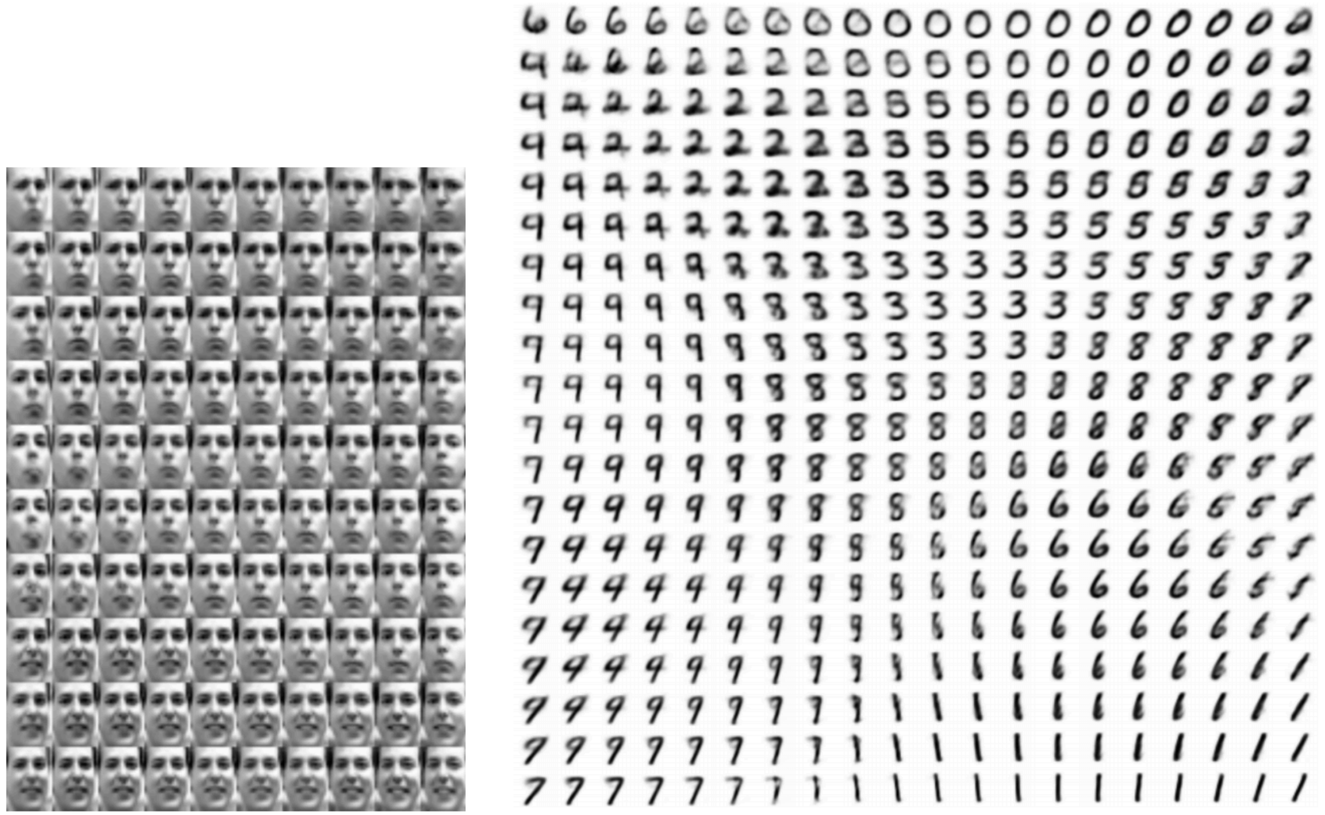
$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) \simeq \frac{1}{2} \sum_{j=1}^{J} \left(1 + \log((\sigma_j^{(i)})^2) - (\mu_j^{(i)})^2 - (\sigma_j^{(i)})^2\right) + \frac{1}{L} \sum_{l=1}^{L} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}^{(i,l)})$$

$$\text{where} \quad \mathbf{z}^{(i,l)} = \boldsymbol{\mu}^{(i)} + \boldsymbol{\sigma}^{(i)} \odot \boldsymbol{\epsilon}^{(l)} \quad \text{and} \quad \boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(0, \mathbf{I}) \tag{10}$$

As explained above and in appendix C, the decoding term $\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}^{(i,l)})$ is a Bernoulli or Gaussian MLP, depending on the type of data we are modelling.

Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes

Variational Auto-Encoders



(a) Learned Frey Face manifold

(b) Learned MNIST manifold

Figure 4: Visualisations of learned data manifold for generative models with two-dimensional latent space, learned with AEVB. Since the prior of the latent space is Gaussian, linearly spaced coordinates on the unit square were transformed through the inverse CDF of the Gaussian to produce values of the latent variables $\mathbf{z}$. For each of these values $\mathbf{z}$, we plotted the corresponding generative $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ with the learned parameters $\boldsymbol{\theta}$.

Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes

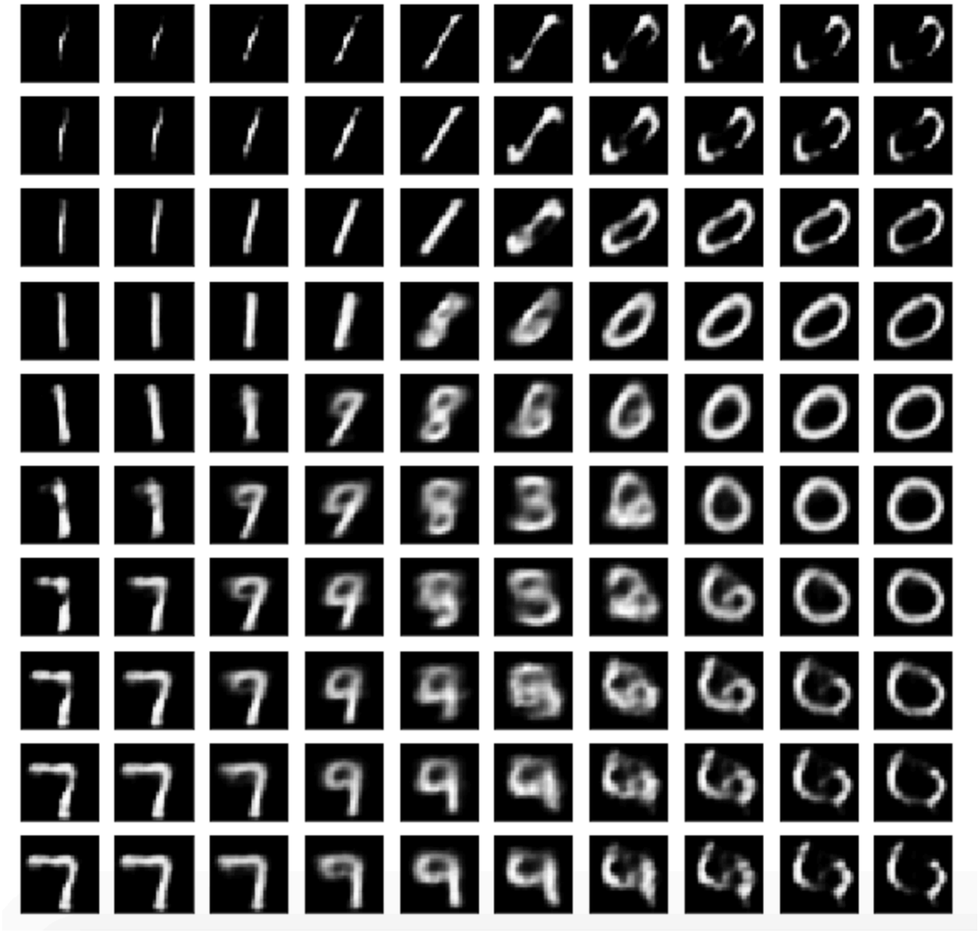
Variational Auto-Encoders

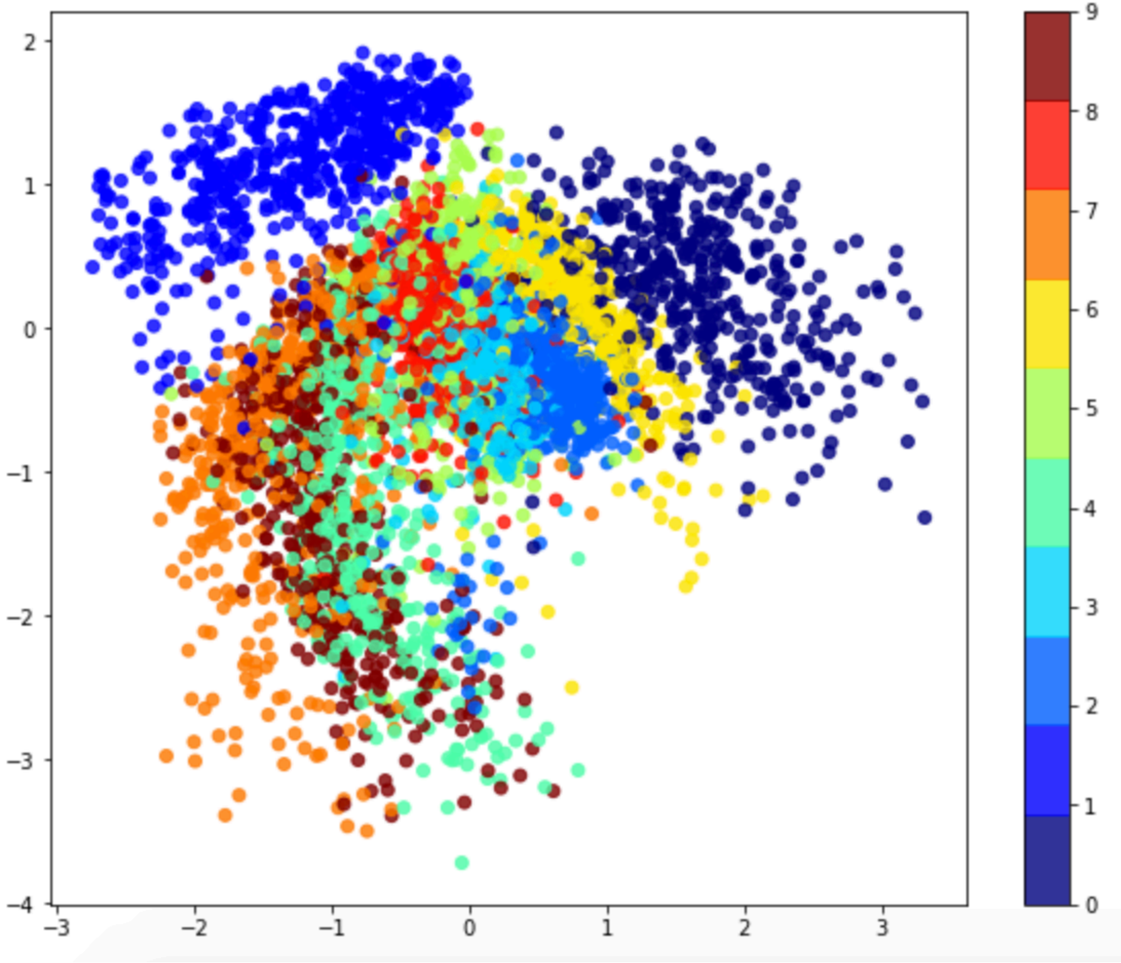
(a) 2-D latent space (b) 5-D latent space (c) 10-D latent space (d) 20-D latent space

Figure 5: Random samples from learned generative models of MNIST for different dimensionalities of latent space.

Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes

Variational Auto-Encoders





Bayes By Backprop.

Bayes By Backprop.

- VAE에서 latent variables Z의 posterior를 구하기 위해 variational inference를 사용했듯이 Bayes by Backprop.에서는 weight의 posterior를 구하기 위해 variational inference를 사용합니다.

- VAE에서와 마찬가지로 ELBO를 최대화하는 방식으로 트레이닝이 진행되며, neural networks에서 사용하는 standard backpropagation을 이용하기 위해 reparameterization trick을 사용합니다.

- weight의 posterior를 얻음으로써 predictive distribution을 얻을 수 있다는 장점이 있습니다. 하지만 Bayesian linear regression과 달리 analytical한 형태의 distribution을 얻을 수는 없고 많은 샘플링을 통해 mean과 variance를 측정할 수 있습니다.

- Predictive distribution을 얻음으로써 예측에 대한 uncertainty를 얻어낼 수 있습니다.

Bayes By Backprop.

Abstract

We introduce a new, efficient, principled and backpropagation-compatible algorithm for learning a probability distribution on the weights of a neural network, called *Bayes by Backprop*. It regularises the weights by minimising a compression cost, known as the variational free energy or the expected lower bound on the marginal likelihood. We show that this principled kind of regularisation yields comparable performance to dropout on MNIST classification. We then demonstrate how the learnt uncertainty in the weights can be used to improve generalisation in non-linear regression problems, and how this weight uncertainty can be used to drive the exploration-exploitation trade-off in reinforcement learning.

Charles Blundell et al., Weight Uncertainty in Neural Networks

Bayes By Backprop.

- 일반적인 deep learning model과의 차이점

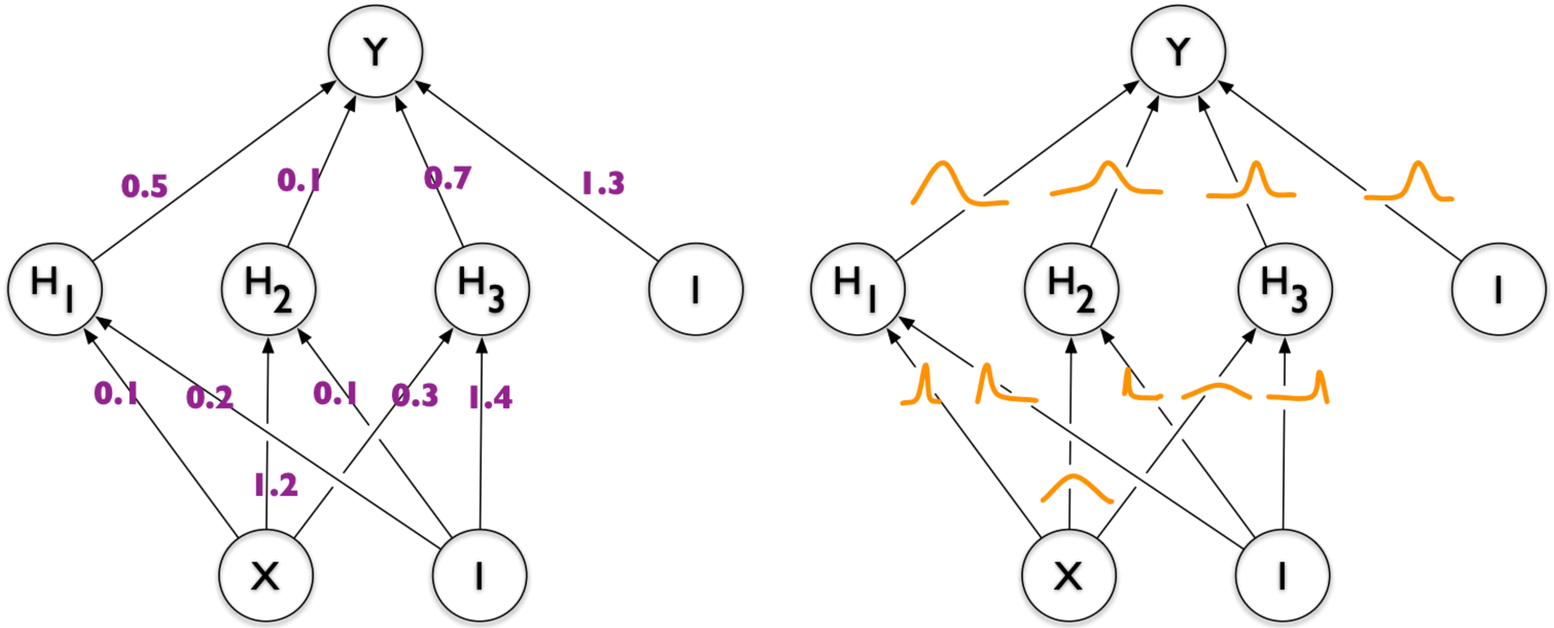


Figure 1. Left: each weight has a fixed value, as provided by classical backpropagation. Right: each weight is assigned a distribution, as provided by Bayes by Backprop.

Charles Blundell et al., Weight Uncertainty in Neural Networks

Bayes By Backprop.

- Posterior, Predictive Distrubution, Ensemble

3. Being Bayesian by Backpropagation

Bayesian inference for neural networks calculates the posterior distribution of the weights given the training data, $P(\mathbf{w}|\mathcal{D})$. This distribution answers predictive queries about unseen data by taking expectations: the predictive distribution of an unknown label $\hat{\mathbf{y}}$ of a test data item $\hat{\mathbf{x}}$, is given by $P(\hat{\mathbf{y}}|\hat{\mathbf{x}}) = \mathbb{E}_{P(\mathbf{w}|\mathcal{D})}[P(\hat{\mathbf{y}}|\hat{\mathbf{x}}, \mathbf{w})]$. Each possible configuration of the weights, weighted according to the posterior distribution, makes a prediction about the unknown label given the test data item $\hat{\mathbf{x}}$. Thus taking an expectation under the posterior distribution on weights is equivalent to using an ensemble of an uncountably infinite number of neural networks. Unfortunately, this is intractable for neural networks of any practical size.

Charles Blundell et al., Weight Uncertainty in Neural Networks

Bayes By Backprop.

- Optimal Paramaters

Previously Hinton and Van Camp (1993) and Graves (2011) suggested finding a variational approximation to the Bayesian posterior distribution on the weights. Variational learning finds the parameters θ of a distribution on the weights $q(\mathbf{w}|\theta)$ that minimises the Kullback-Leibler (KL)

divergence with the true Bayesian posterior on the weights:

variational posterior, true posterior

$$\theta^\star = \arg\min_\theta \mathrm{KL}[q(\mathbf{w}|\theta)||P(\mathbf{w}|\mathcal{D})]$$

$$= \arg\min_\theta \int q(\mathbf{w}|\theta) \log \frac{q(\mathbf{w}|\theta)}{P(\mathbf{w})P(\mathcal{D}|\mathbf{w})} \mathbf{dw}$$

Negative ELBO

$$= \arg\min_\theta \mathrm{KL}\left[q(\mathbf{w}|\theta) \,||\, P(\mathbf{w})\right] - \mathbb{E}_{q(\mathbf{w}|\theta)}\left[\log P(\mathcal{D}|\mathbf{w})\right].$$

variational posterior, prior, log-likelihood

$$P(D) = ELBO + KL(q||p)$$

$$ELBO = \mathbb{E}_{q(\mathbf{w}|\theta)}\left[\log \frac{P(\mathbf{w})P(D\,|\,\mathbf{w})}{q(\mathbf{w}\,|\,\theta)}\right]$$

Charles Blundell et al., Weight Uncertainty in Neural Networks

Bayes By Backprop.

- ELBO (Evidence Lower BOund)

The resulting cost function is variously known as the variational free energy (Neal and Hinton, 1998; Yedidia et al., 2000; Friston et al., 2007) or the expected lower bound (Saul et al., 1996; Neal and Hinton, 1998; Jaakkola and Jordan, 2000). For simplicity we shall denote it as

$$\mathcal{F}(\mathcal{D}, \theta) = \mathrm{KL}\left[q(\mathbf{w}|\theta) \,||\, P(\mathbf{w})\right] - \mathbb{E}_{q(\mathbf{w}|\theta)}\left[\log P(\mathcal{D}|\mathbf{w})\right]. \quad (1)$$

Negative ELBO

The cost function of (1) is a sum of a data-dependent part, which we shall refer to as the likelihood cost, and a prior-dependent part, which we shall refer to as the complexity cost. The cost function embodies a trade-off between satisfying the complexity of the data $\mathcal{D}$ and satisfying the simplicity prior $P(\mathbf{w})$. (1) is also readily given an information theoretic interpretation as a minimum description length cost (Hinton and Van Camp, 1993; Graves, 2011). Exactly minimising this cost naïvely is computationally prohibitive. Instead gradient descent and various approximations are used.

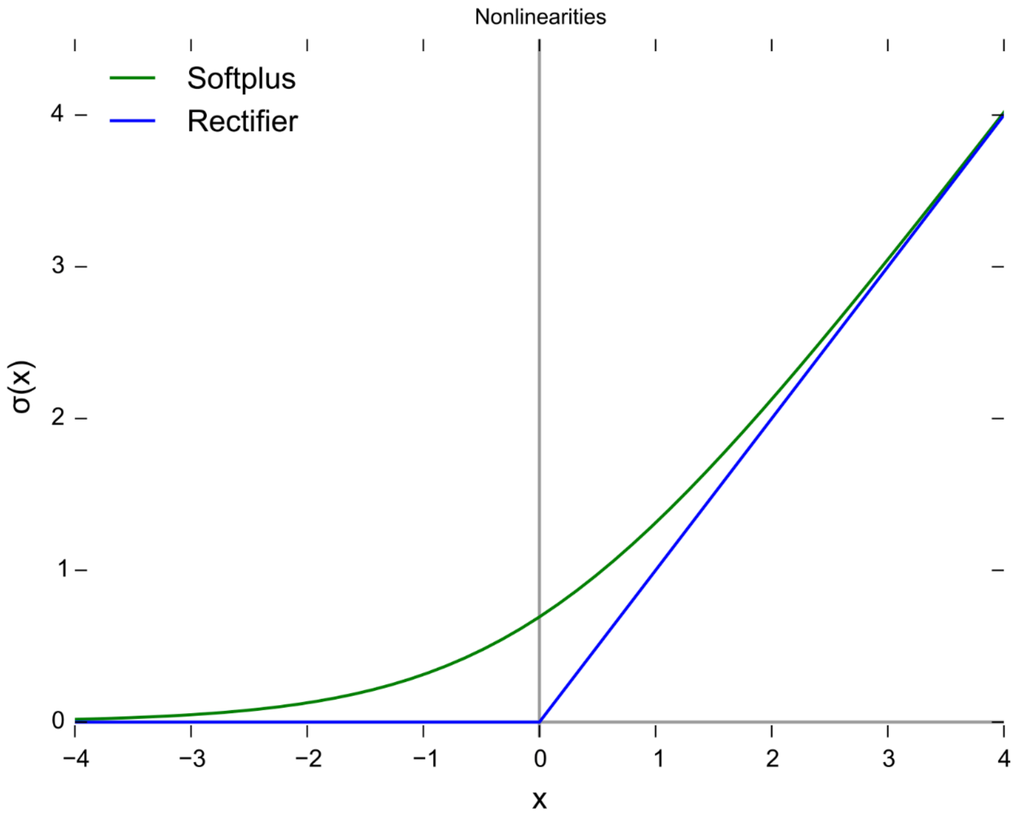
Charles Blundell et al., Weight Uncertainty in Neural Networks

Bayes By Backprop.

- Reparameterization Trick

3.2. Gaussian variational posterior

Suppose that the variational posterior is a diagonal Gaussian distribution, then a sample of the weights $\mathbf{w}$ can be obtained by sampling a unit Gaussian, shifting it by a mean μ and scaling by a standard deviation σ. We parameterise the standard deviation pointwise as $\sigma = \log(1 + \exp(\rho))$ and so σ is always non-negative. The variational posterior parameters are $\theta = (\mu, \rho)$. Thus the transform from a sample of parameter-free noise and the variational posterior parameters that yields a posterior sample of the weights $\mathbf{w}$ is: $\mathbf{w} = t(\theta, \epsilon) = \mu + \log(1 + \exp(\rho)) \circ \epsilon$ where $\circ$ is pointwise multiplication. Each step of optimisation proceeds as follows:



https://en.wikipedia.org/wiki/Rectifier_(neural_networks)

Charles Blundell et al., Weight Uncertainty in Neural Networks

Bayes By Backprop.

- Training Steps

1. Sample $\epsilon \sim \mathcal{N}(0, I)$.
2. Let $\mathbf{w} = \mu + \log(1 + \exp(\rho)) \circ \epsilon$.
3. Let $\theta = (\mu, \rho)$.
4. Let $f(\mathbf{w}, \theta) = \log q(\mathbf{w}|\theta) - \log P(\mathbf{w})P(\mathcal{D}|\mathbf{w})$.
5. Calculate the gradient with respect to the mean

$$\Delta_\mu = \frac{\partial f(\mathbf{w}, \theta)}{\partial \mathbf{w}} + \frac{\partial f(\mathbf{w}, \theta)}{\partial \mu}. \tag{3}$$

6. Calculate the gradient with respect to the standard deviation parameter ρ

$$\Delta_\rho = \frac{\partial f(\mathbf{w}, \theta)}{\partial \mathbf{w}} \frac{\epsilon}{1 + \exp(-\rho)} + \frac{\partial f(\mathbf{w}, \theta)}{\partial \rho}. \tag{4}$$

7. Update the variational parameters:

$$\mu \leftarrow \mu - \alpha\Delta_\mu \tag{5}$$
$$\rho \leftarrow \rho - \alpha\Delta_\rho. \tag{6}$$

Charles Blundell et al., Weight Uncertainty in Neural Networks

Bayes By Backprop.

- Scale mixture prior

We propose using a scale mixture of two Gaussian densities as the prior. Each density is zero mean, but differing variances:

$$P(\mathbf{w}) = \prod_j \pi \mathcal{N}(\mathbf{w}_j|0,\sigma_1^2) + (1-\pi)\mathcal{N}(\mathbf{w}_j|0,\sigma_2^2), \quad (7)$$

where $\mathbf{w}_j$ is the jth weight of the network, $\mathcal{N}(x|\mu,\sigma^2)$ is the Gaussian density evaluated at x with mean μ and variance σ^2 and σ_1^2 and σ_2^2 are the variances of the mixture components. The first mixture component of the prior is given a larger variance than the second, $\sigma_1 > \sigma_2$, providing a heavier tail in the prior density than a plain Gaussian prior. The second mixture component has a small variance $\sigma_2 \ll 1$ causing many of the weights to *a priori* tightly concentrate around zero. Our prior resembles a spike-and-slab prior (Mitchell and Beauchamp, 1988; George and McCul-

Charles Blundell et al., Weight Uncertainty in Neural Networks

Bayes By Backprop.

Table 1. Classification Error Rates on MNIST. ⋆ indicates result used an ensemble of 5 networks.

Method	# Units/Layer	# Weights	Test Error
SGD, no regularisation (Simard et al., 2003)	800	1.3m	1.6%
SGD, dropout (Hinton et al., 2012)			≈ 1.3%
SGD, dropconnect (Wan et al., 2013)	800	1.3m	**1.2%**⋆
SGD	400	500k	1.83%
	800	1.3m	1.84%
	1200	2.4m	1.88%
SGD, dropout	400	500k	1.51%
	800	1.3m	1.33%
	1200	2.4m	1.36%
Bayes by Backprop, Gaussian	400	500k	1.82%
	800	1.3m	1.99%
	1200	2.4m	2.04%
Bayes by Backprop, Scale mixture	400	500k	1.36%
	800	1.3m	1.34%
	1200	2.4m	**1.32%**

Charles Blundell et al., Weight Uncertainty in Neural Networks

Bayes By Backprop.

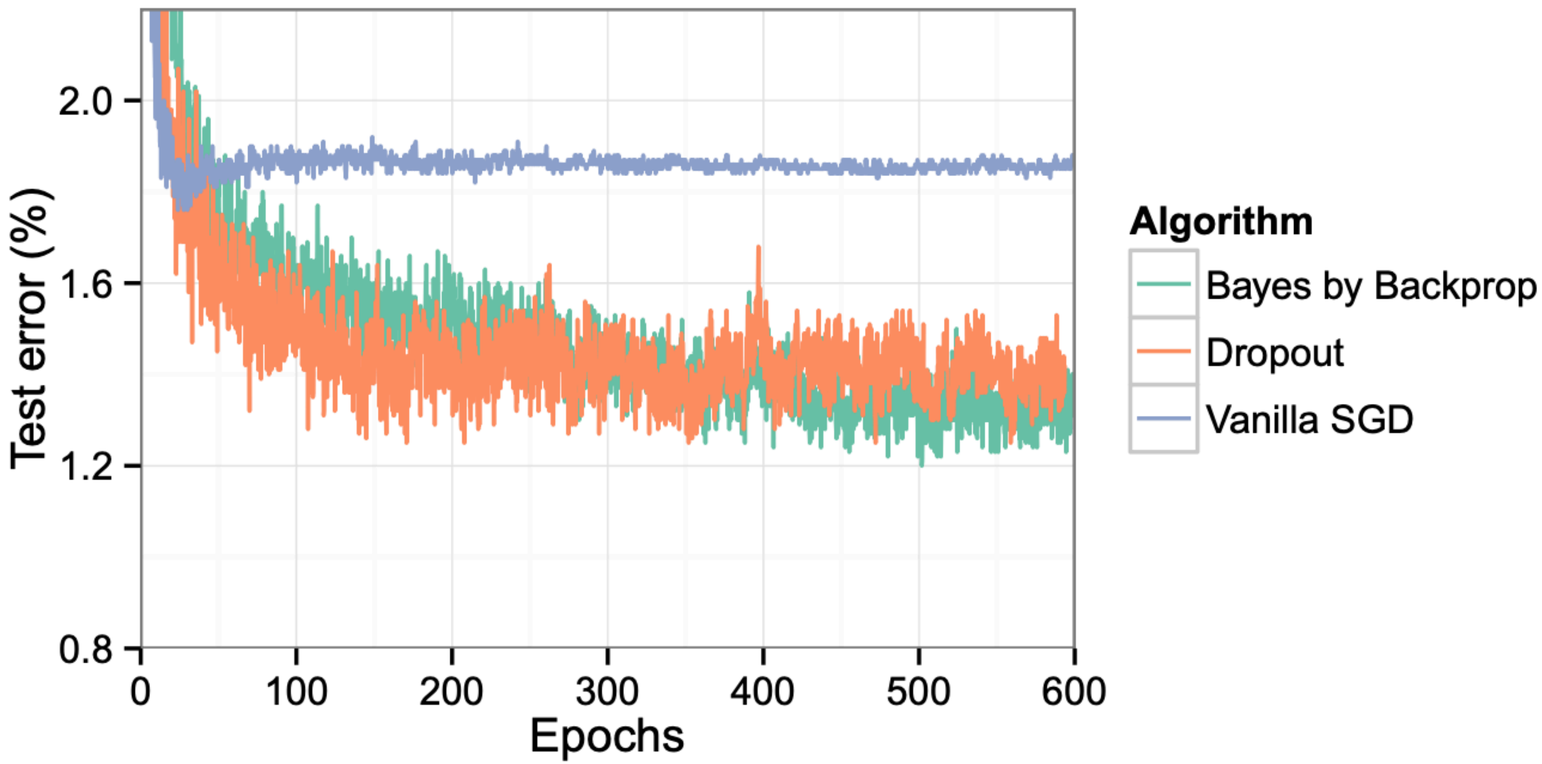


Figure 2. Test error on MNIST as training progresses.

Charles Blundell et al., Weight Uncertainty in Neural Networks

Bayes By Backprop.

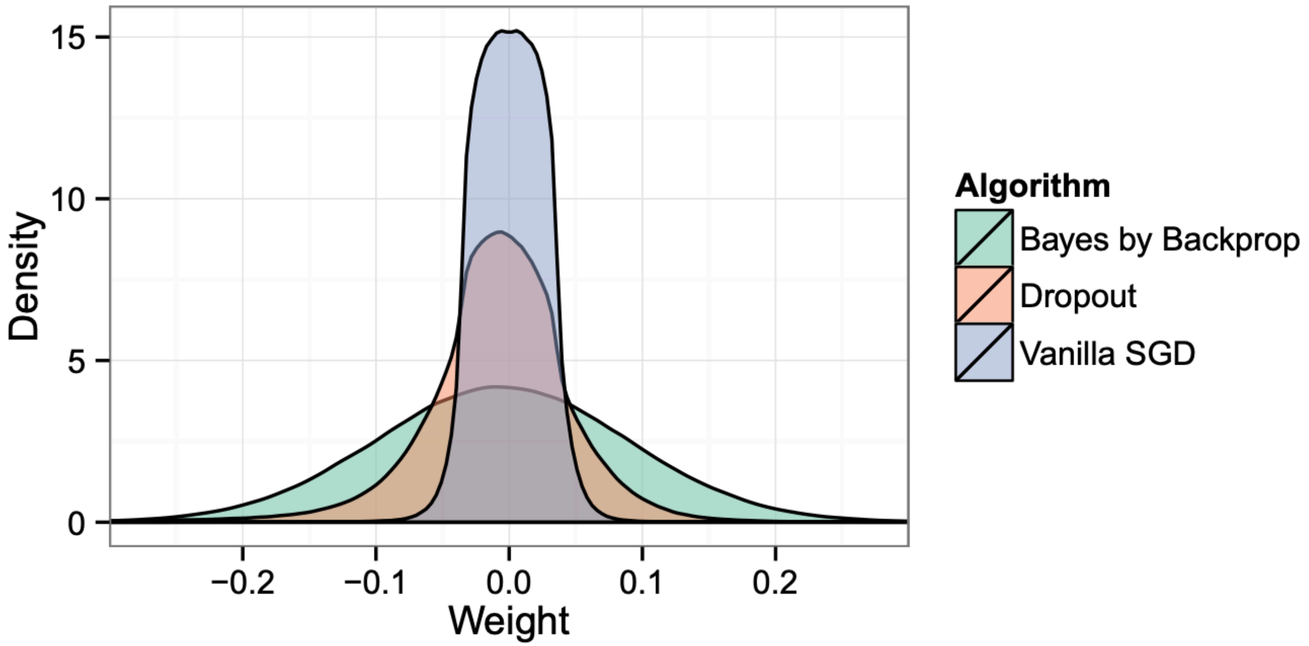


Figure 3. Histogram of the trained weights of the neural network, for Dropout, plain SGD, and samples from Bayes by Backprop.

Charles Blundell et al., Weight Uncertainty in Neural Networks

Bayes By Backprop.

"We took a Bayes by Backprop trained network with two layers of 1200 units and ordered the weights by their signal–to–noise ratio ($|\mu_i|/\sigma_i$). We removed the weights with the lowest signal to noise ratio."

Table 2. Classification Errors after Weight pruning

Proportion removed	# Weights	Test Error
0%	2.4m	1.24%
50%	1.2m	1.24%
75%	600k	1.24%
95%	120k	1.29%
98%	48k	1.39%

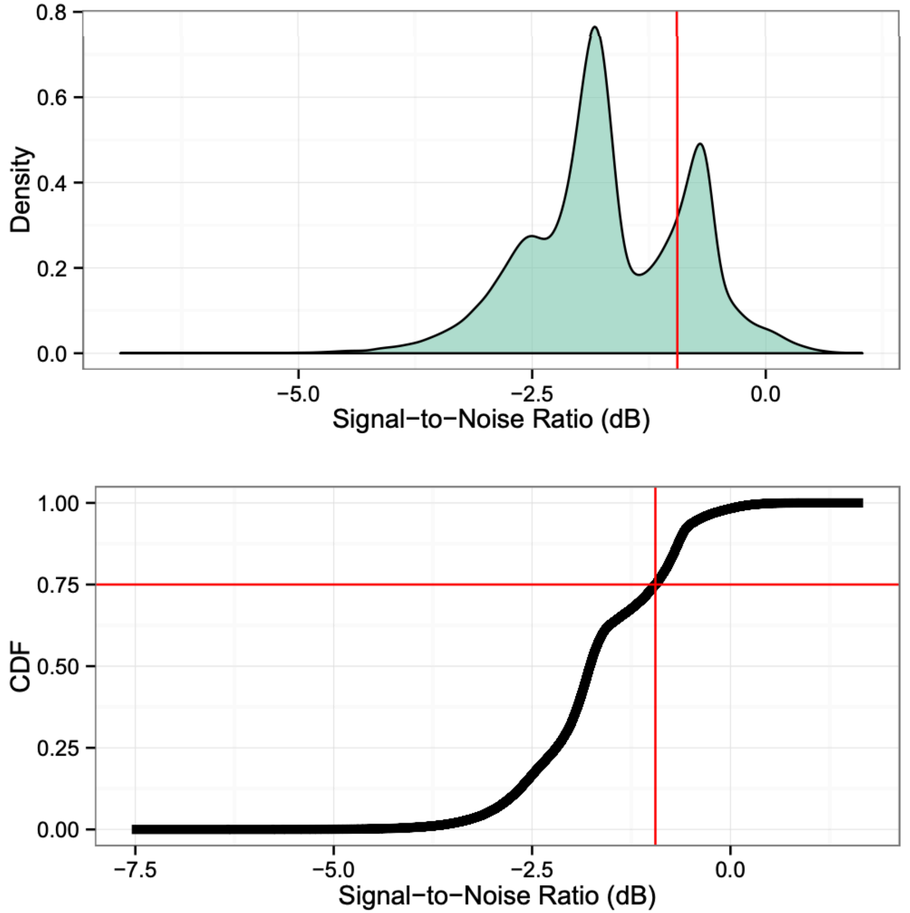


Figure 4. Density and CDF of the Signal-to-Noise ratio over all weights in the network. The red line denotes the 75% cut-off.

Charles Blundell et al., Weight Uncertainty in Neural Networks

Bayes By Backprop.

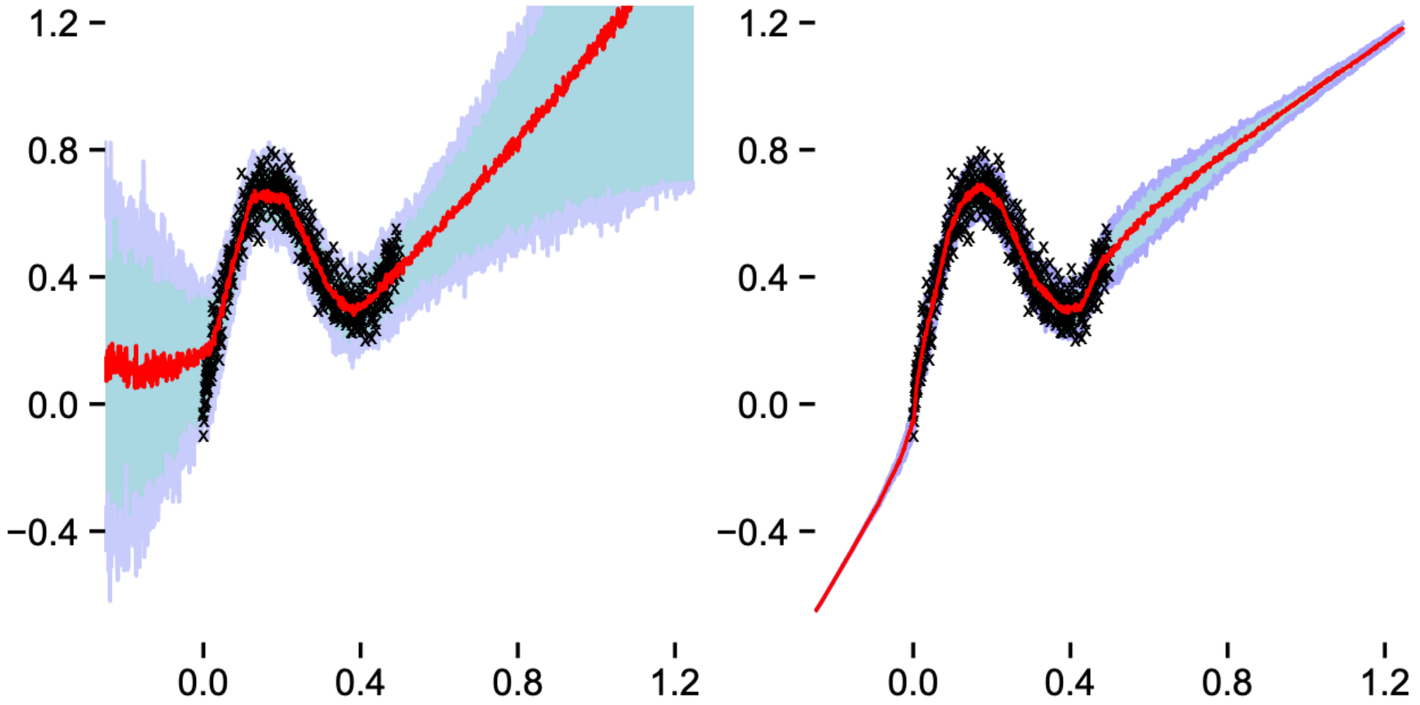
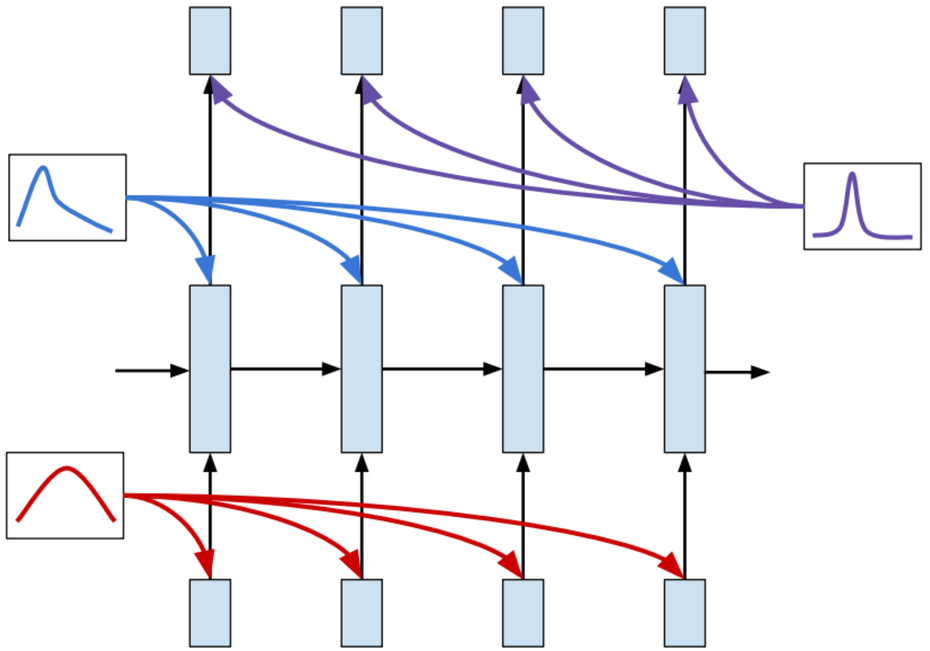


Figure 5. Regression of noisy data with interquatile ranges. Black crosses are training samples. Red lines are median predictions. Blue/purple region is interquartile range. Left: Bayes by Backprop neural network, Right: standard neural network.

Charles Blundell et al., Weight Uncertainty in Neural Networks

Bayesian RNN



Algorithm: Bayes by Backprop for RNNs

Sample $\epsilon \sim \mathcal{N}(0, I)$, $\epsilon \in \mathbb{R}^d$, and set network parameters to $\theta = \mu + \sigma\epsilon$.
Sample a minibatch of truncated sequences (x, y).
Do forward and backward propagation as normal, and let g be the gradient w.r.t θ.
Let $g_\theta^{KL}, g_\mu^{KL}, g_\sigma^{KL}$ be the gradients of $\log \mathcal{N}(\theta|\mu, \sigma^2) - \log p(\theta)$ w.r.t. θ, μ and σ respectively.
Update μ using the gradient $\frac{g + \frac{1}{C} g_\theta^{KL}}{B} + \frac{g_\mu^{KL}}{BC}$.
Update σ using the gradient $\left(\frac{g + \frac{1}{C} g_\theta^{KL}}{B}\right)\epsilon + \frac{g_\sigma^{KL}}{BC}$.

Figure 1: Illustration (left) and Algorithm (right) of Bayes by Backprop applied to an RNN.

Meire Fortunato et al., Bayesian Recurrent Neural Networks

Bayesian CNN

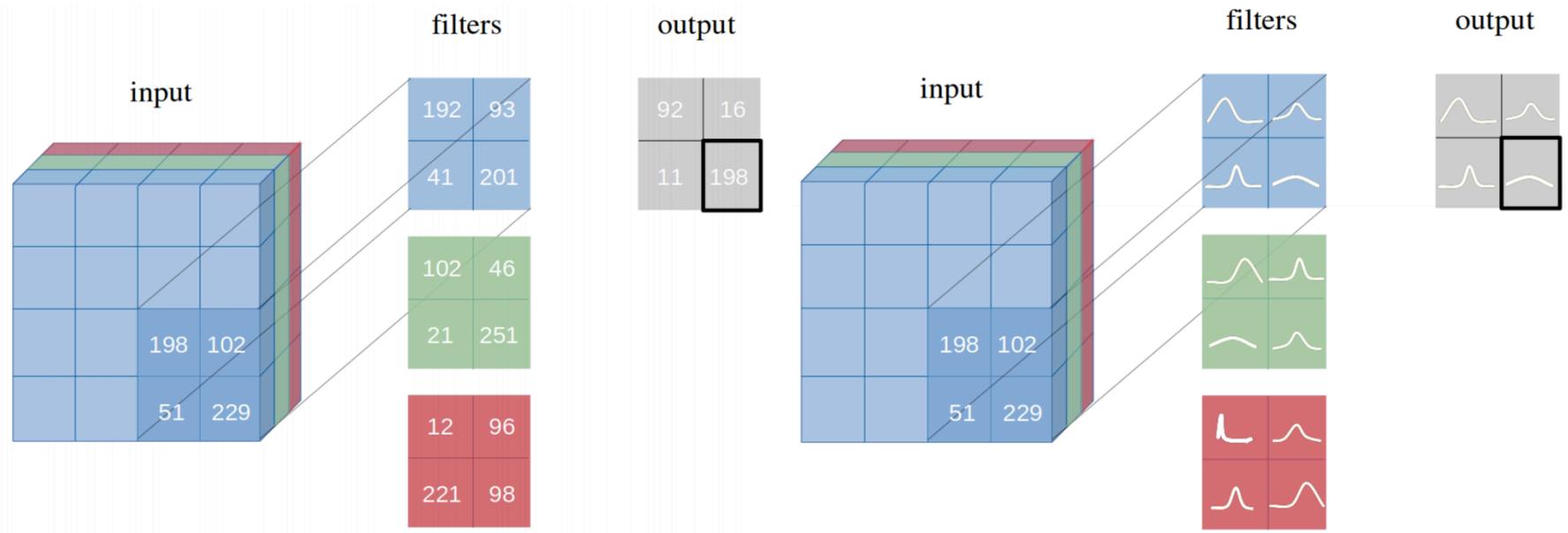
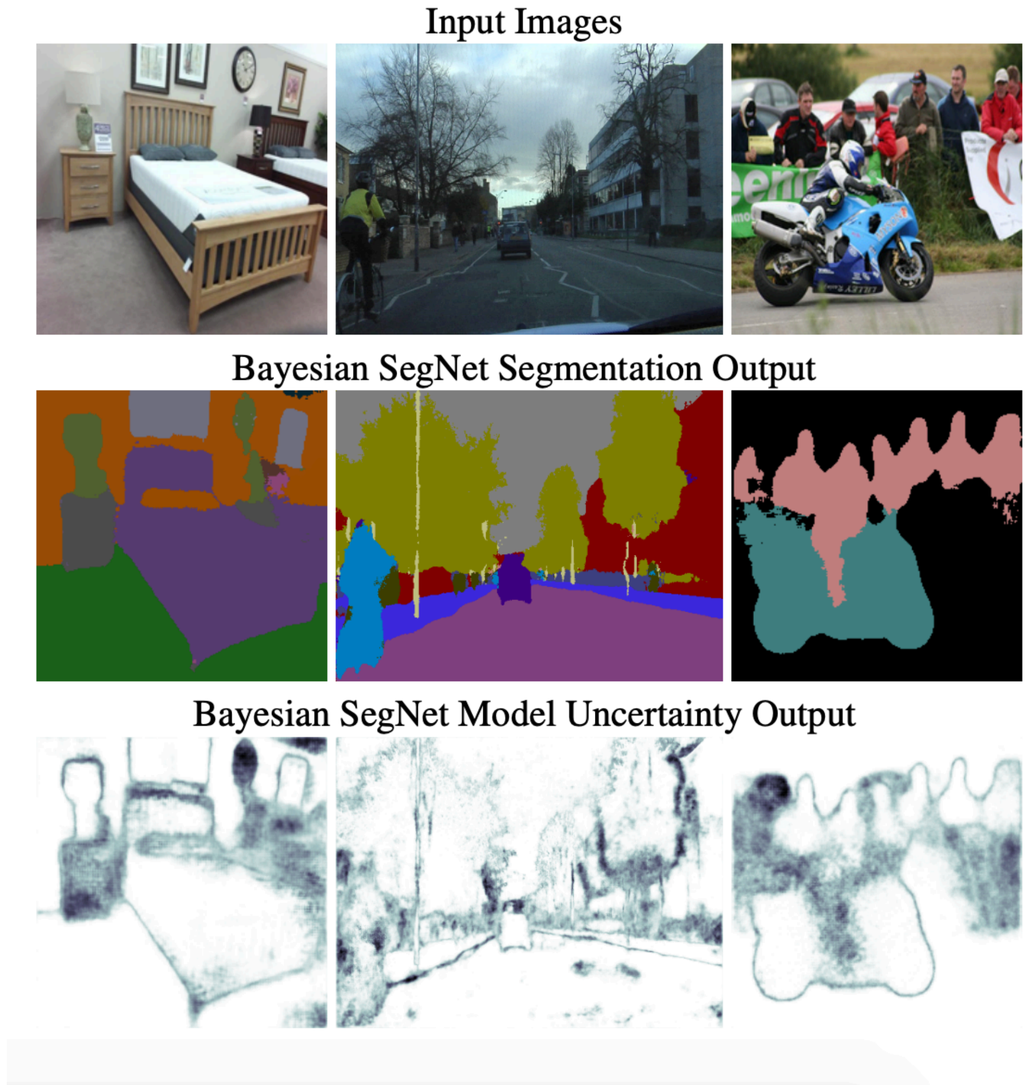


Figure 4: Input image with exemplary pixel values, filters, and corresponding output with point-estimates (top) and probability distributions (bottom) over weights.[55]

Kumar Shridhar et al., A Comprehensive guide to Bayesian Convolutional Neural Network with Variational Inference

Bayesian SegNet



Alex Kendall et al., Bayesian SegNet: Model Uncertainty in Deep Convolutional Encoder-Decoder Architectures for Scene Understanding

Bayesian SegNet

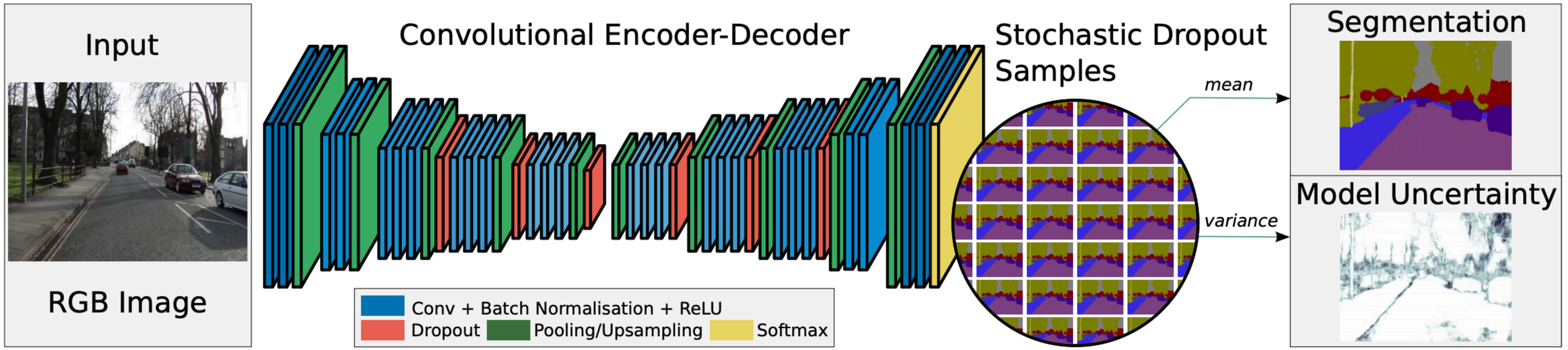
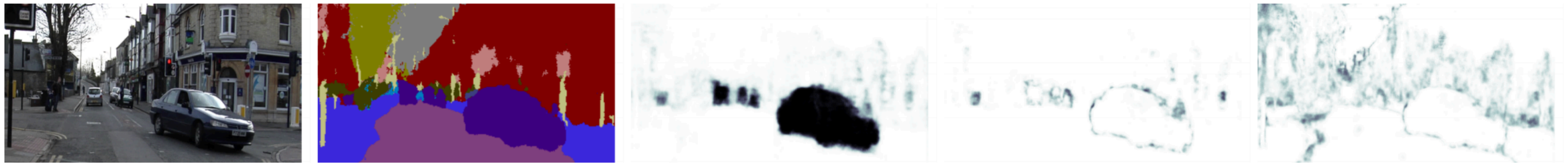


Figure 2: **A schematic of the Bayesian SegNet architecture.** This diagram shows the entire pipeline for the system which is trained end-to-end in one step with stochastic gradient descent. The encoders are based on the 13 convolutional layers of the VGG-16 network [34], with the decoder placing them in reverse. The probabilistic output is obtained from Monte Carlo samples of the model with dropout at test time. We take the variance of these softmax samples as the model uncertainty for each class.

Alex Kendall et al., Bayesian SegNet: Model Uncertainty in Deep Convolutional Encoder-Decoder Architectures for Scene Understanding

Bayesian SegNet



(a) Input Image (b) Semantic Segmentation (c) Softmax Uncertainty Car Class (d) Dropout Uncertainty Car Class (e) Dropout Uncertainty All Classes

Figure 3: **Comparison of uncertainty with Monte Carlo dropout and uncertainty from softmax regression (c-e: darker colour represents larger value).** This figure shows that softmax regression is only capable of inferring relative probabilities between classes. In contrast, dropout uncertainty can produce an estimate of absolute model uncertainty.

Alex Kendall et al., Bayesian SegNet: Model Uncertainty in Deep Convolutional Encoder-Decoder Architectures for Scene Understanding

MC Dropout

Dropout as a Bayesian Approximation

MCDropout

Abstract

Deep learning tools have gained tremendous attention in applied machine learning. However such tools for regression and classification do not capture model uncertainty. In comparison, Bayesian models offer a mathematically grounded framework to reason about model uncertainty, but usually come with a prohibitive computational cost. In this paper we develop a new theoretical framework casting dropout training in deep neural networks (NNs) as approximate Bayesian inference in deep Gaussian processes. A direct result of this theory gives us tools to model uncertainty with dropout NNs – extracting information from existing models that has been thrown away so far. This mitigates the problem of representing uncertainty in deep learning without sacrificing either computational complexity or test accuracy. We perform an extensive study of the properties of dropout's uncertainty. Various network architectures and nonlinearities are assessed on tasks of regression and classification, using MNIST as an example. We show a considerable improvement in predictive log-likelihood and RMSE compared to existing state-of-the-art methods, and finish by using dropout's uncertainty in deep reinforcement learning.

Gal, Yarin, and Zoubin Ghahramani. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning." international conference on machine learning. 2016.

MCDropout

- Variational inference를 이용한 Bayesian neural networks는 구현하는데 번거로운 점이 있습니다.
- MCDropout (Monte Carlo Dropout)은 dropout을 이용하여 Bayesian neural networks와 같은 효과를 내도록합니다.
- Posterior에서 sampling하는 작업을 dropout으로 대신하고, Prior를 설정하던 것을 L2 regularisation으로 대체합니다.
- Loss function은 다음과 같이 Error term과 regularization term으로 구성됩니다.

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

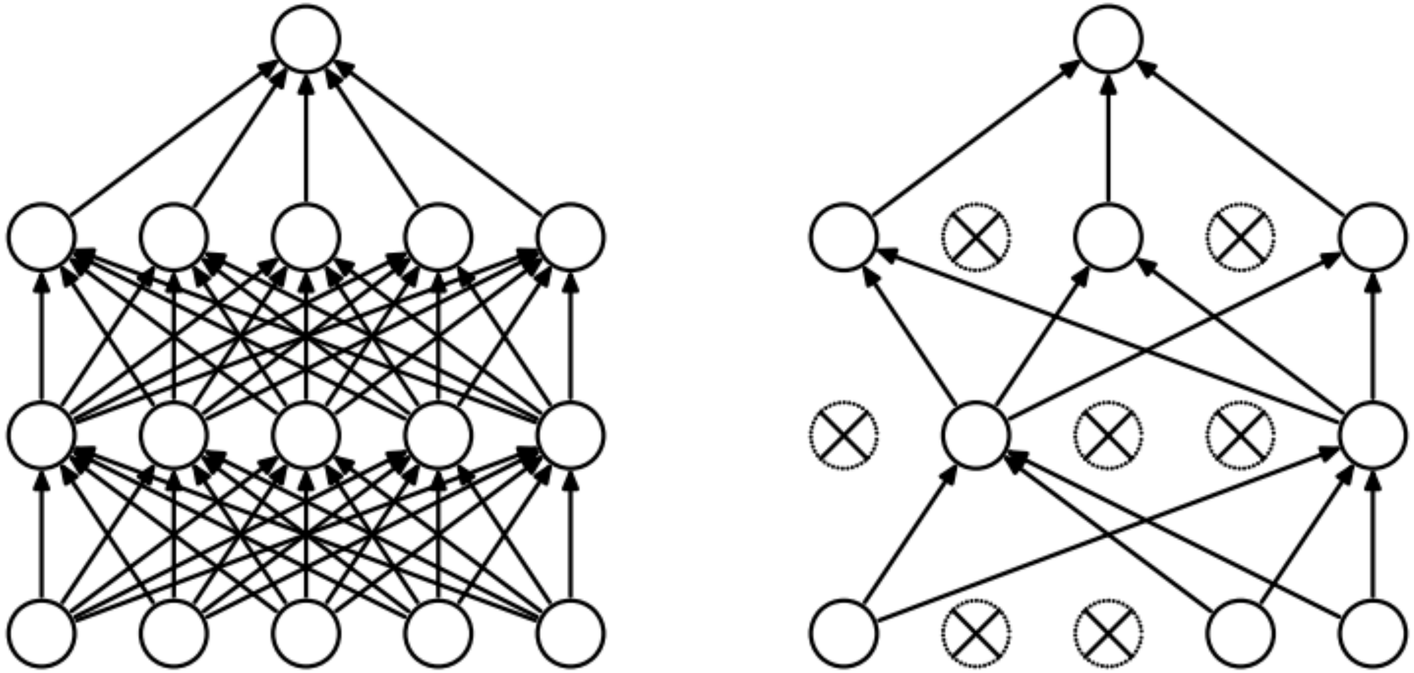
N: 데이터 갯수, $E(\cdot, \cdot)$: softmax loss or square loss

$\mathbf{y}_i$: target, $\hat{\mathbf{y}}_\mathbf{i}$: network output, λ: weight decay, L: # of layers

$\mathbf{W}$: weight, $\mathbf{b}$:bias

Gal, Yarin, and Zoubin Ghahramani. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning." international conference on machine learning. 2016.

MCDropout

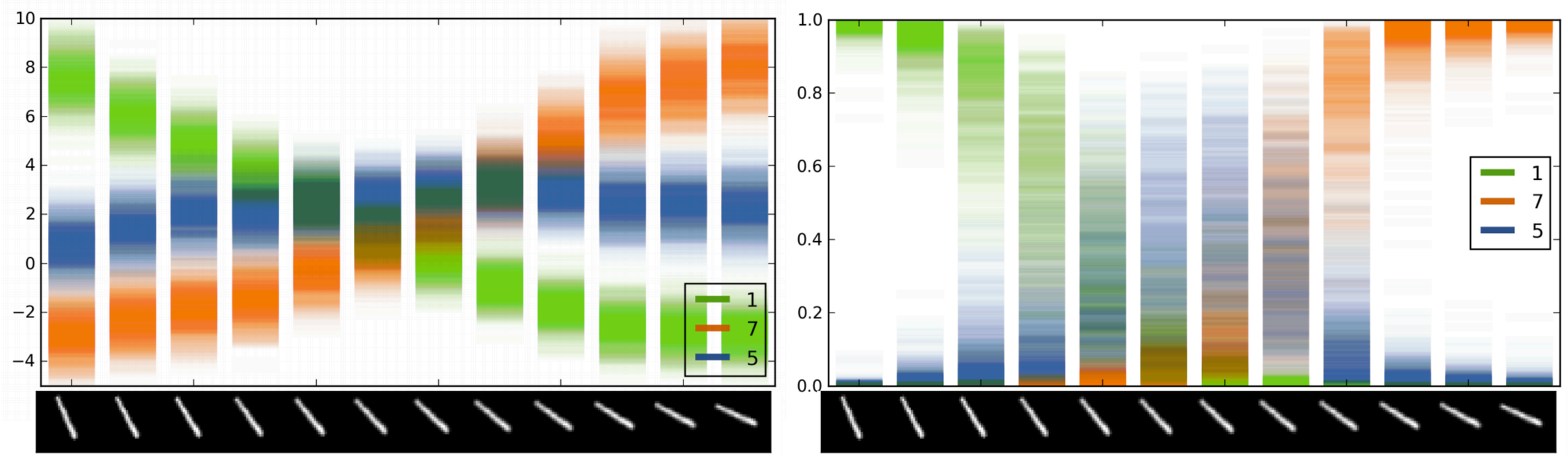


(a) Standard Neural Net (b) After applying dropout.

Figure 1: Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." The journal of machine learning research 15.1 (2014): 1929-1958.

MCDropout



(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).

Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." The journal of machine learning research 15.1 (2014): 1929-1958.

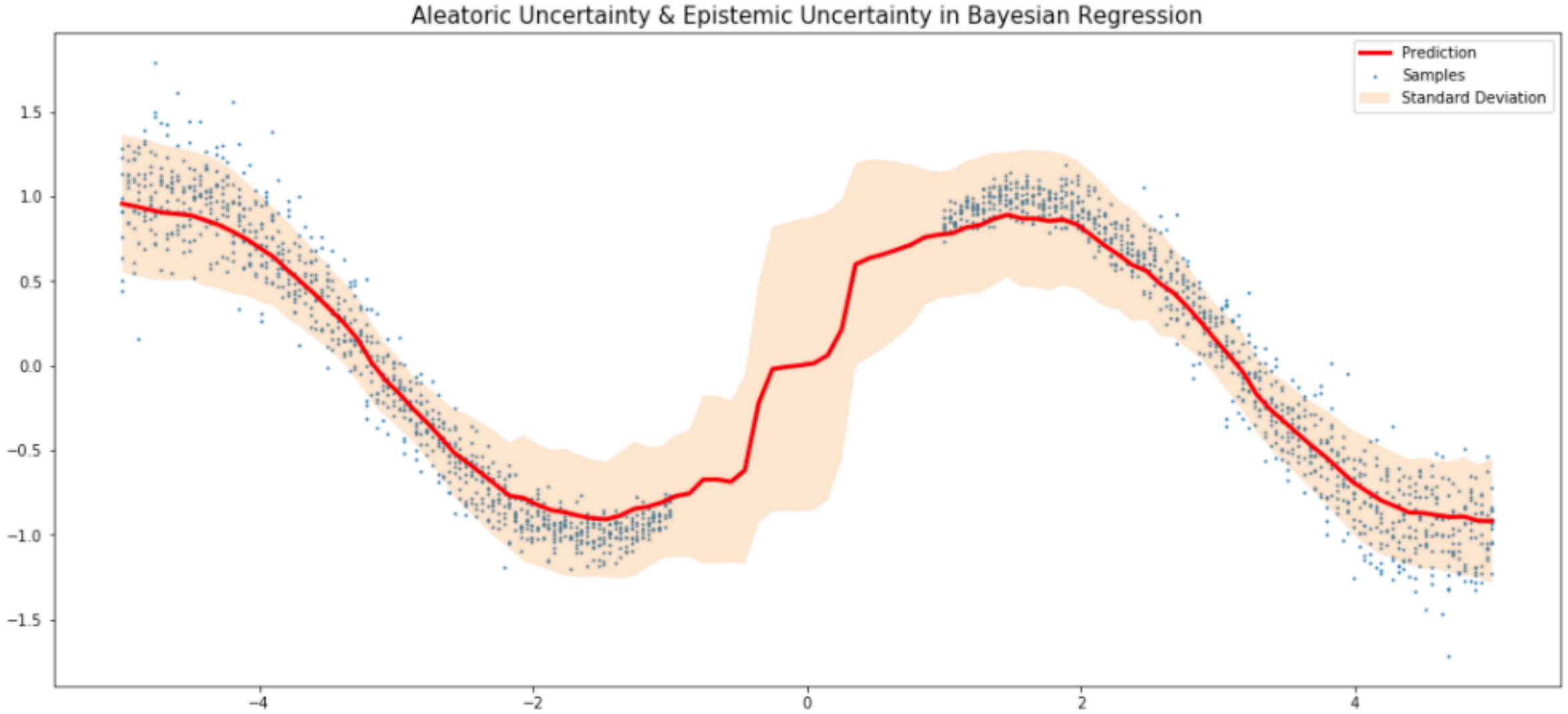
Uncertainty

What uncertainties do we need in bayesian deep learning for computer vision?

Uncertainty

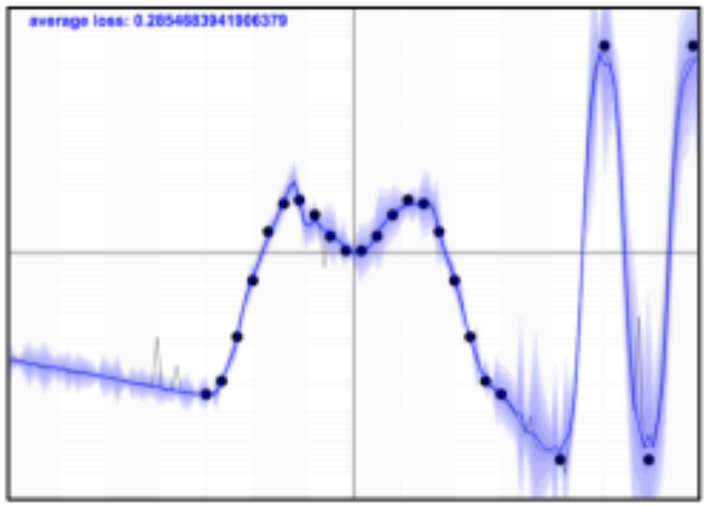
- Uncertainty는 크게 epistemic uncertainty와 aleatoric uncertainty로 구분합니다.
- Epistemic uncertainty는 model uncertainty라고도 불리며, 모델의 파라메터의 불확실성에 의해서 일어나는 불확실성을 뜻합니다. 충분히 많은 데이터를 통해 모델의 파라메터를 트레이닝하여 극복할 수 있는 요소입니다.
- Aleatoric uncertainty는 data uncertainty라고도 불리며, 데이터가 가지는 불확실성을 의미합니다. 센서 데이터와 같이 데이터에 노이즈가 있는 경우 많은 많은 데이터가 공급되어도 불확실성을 줄일 수 없습니다.
- Aleatoric uncertainty는 homoscedastic, heteroscedastic 두종류로 분류할 수 있습니다. homoscedastic uncertainty는 서로 다른 입력 데이터에 대한 각각의 출력 데이터의 분산이 같은 경우이고, heteroscedastic uncertainty은 서로 다른 입력 데이터에 대한 각각의 출력 데이터의 분산이 다를 수 있는 경우입니다.
- GPS 장치 같이 지도의 어떤 지점을 인식하든 오차 범위가 일정하게 정해져 있는 경우 homoscedastic uncertainty를 갖는다 할 수 있습니다. 어떤 물체인식 모델이 특정 물체에 대해 정확하게 인식하고 다른 물체에 대해 부정확하게 인식한다면 heteroscedastic uncertainty를 갖는다 할 수 있습니다.

Uncertainty

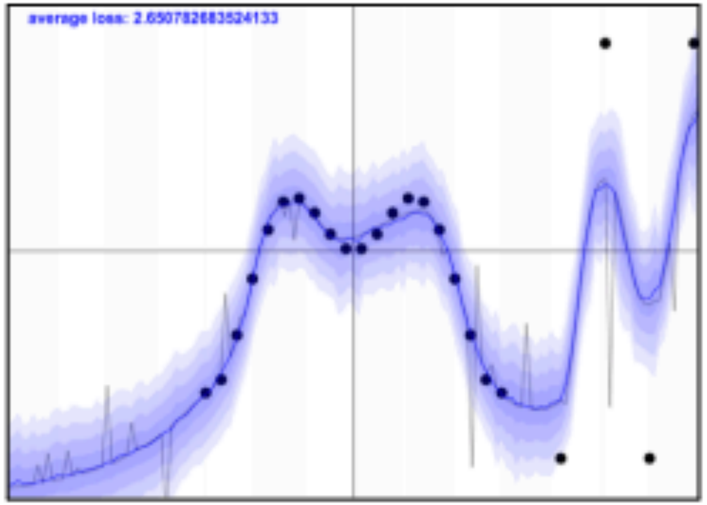


src/uncertainty.ipynb

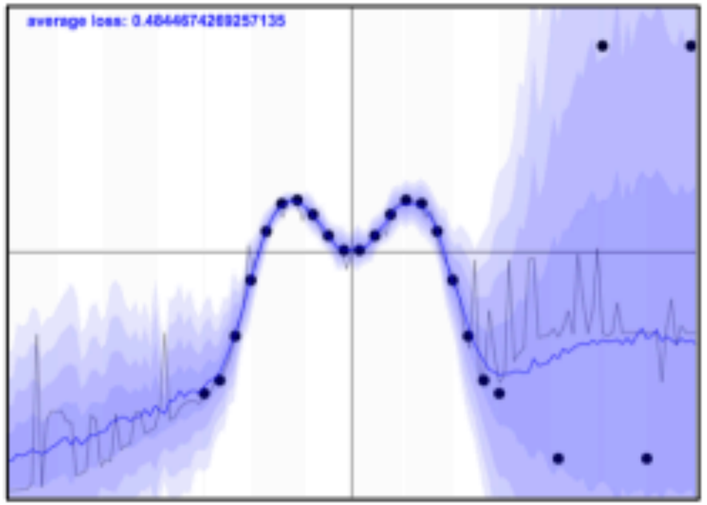
Uncertainty



(a) Homoscedastic model with small observation noise.



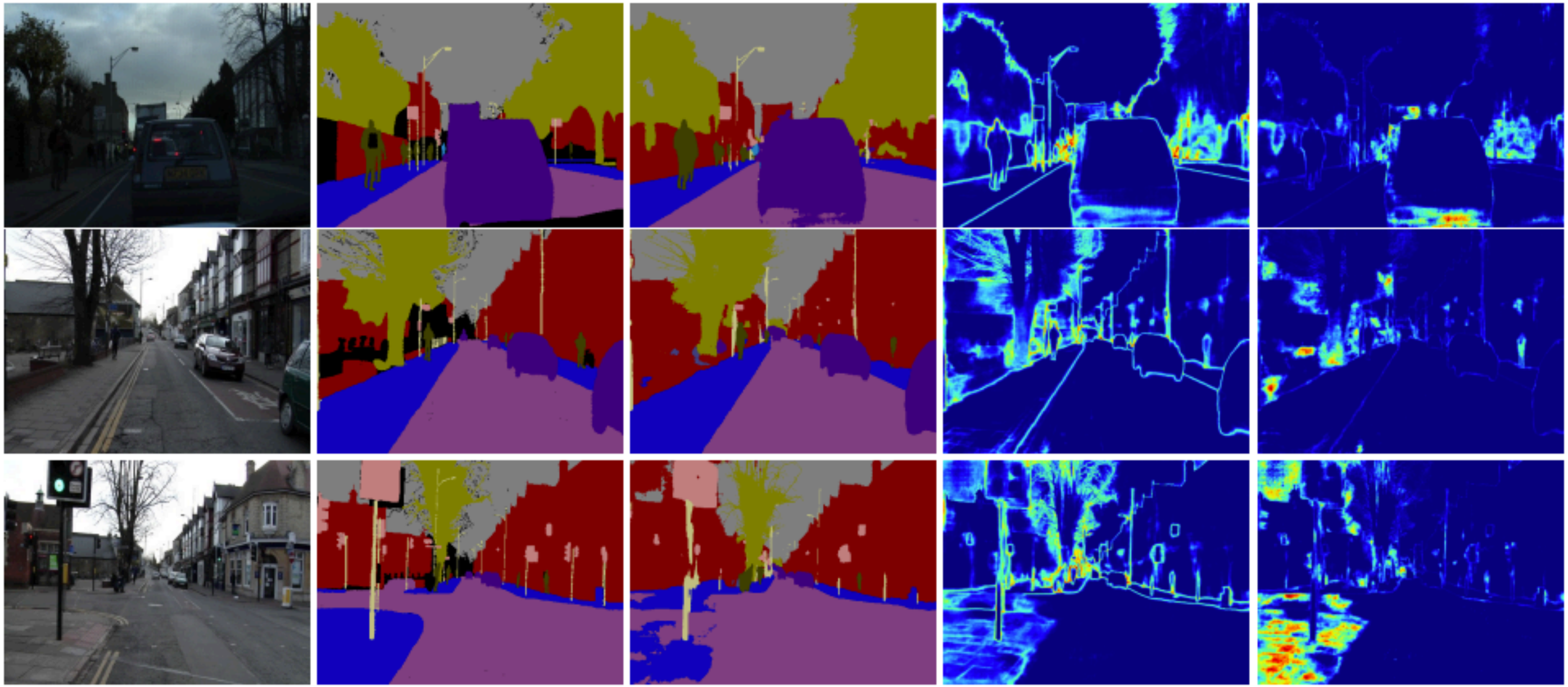
(b) Homoscedastic model with large observation noise.



(c) Heteroscedastic model with data-dependent observation noise.

Gal, Yarin. "Uncertainty in deep learning." University of Cambridge 1.3 (2016). p.85

Uncertainty



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

Kendall, Alex, and Yarin Gal. "What uncertainties do we need in bayesian deep learning for computer vision?." Advances in neural information processing systems. 2017.

Aleatoric Uncertainty In Bayesian Regression

- Bayesian Neural Networks에서 regression의 경우 추가적으로 variance를 나타내는 값 σ^2를 출력값으로 설정하여 data에 대한 uncentainty를 나타내는데 사용할 수 있습니다.

- 이 때, σ^2를 모든 데이터에 대해 동일하도록 설정하면 homoschedastic model이 되고, 각각의 입력 데이터에 대한 함수 $\sigma^2(\mathbf{x}_i)$로 만들면 heteroschedastic model이 됩니다.

- Predictive distribution을 Gaussian으로 설정했을 때, loss function을 다음과 같이 만들 수 있습니다.

$$\mathcal{L}_{\mathrm{NN}}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2\sigma(\mathbf{x}_i)^2}\left\|\mathbf{y}_i - \mathbf{f}(\mathbf{x}_i)\right\|^2 + \frac{1}{2}\log\sigma(\mathbf{x}_i)^2$$

N: # of the data points

$\mathbf{y}_i$: i-th target

$\mathbf{f}(\mathbf{x}_i)$: mean of output

$\sigma^2(\mathbf{x}_i)$: variance of output

Kendall, Alex, and Yarin Gal. "What uncertainties do we need in bayesian deep learning for computer vision?." Advances in neural information processing systems. 2017.

Epistemic Uncertainty In Bayesian Regression

- Bayesian neural networks를 이용한 regression의 경우 epistemic uncentainty는 출력값의 분산을 구하여 측정할 수 있습니다.

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

σ_t^2: t번째 시행시의 variance 결과값

$\hat{\mathbf{y}}_t$: t번째 시행시의 mean 결과값

분산 = 제곱의 평균 - 평균의 제곱

$$\begin{aligned}\mathrm{Var}(X) &= \mathrm{E}\left[(X - \mathrm{E}[X])^2\right] \\ &= \mathrm{E}\left[X^2 - 2X\,\mathrm{E}[X] + \mathrm{E}[X]^2\right] \\ &= \mathrm{E}\left[X^2\right] - 2\,\mathrm{E}[X]\,\mathrm{E}[X] + \mathrm{E}[X]^2 \\ &= \mathrm{E}\left[X^2\right] - \mathrm{E}[X]^2\end{aligned}$$

Kendall, Alex, and Yarin Gal. "What uncertainties do we need in bayesian deep learning for computer vision?." Advances in neural information processing systems. 2017.

Aleatoric Uncertainty In Bayesian Classification

- Regression에서 aleatoric uncertainty를 구한 방법처럼 classification에서도 variance를 나타내는 출력값 $\sigma^2(\mathbf{x}_i)$을 두어 aleatoric uncertainty를 구할 수 있습니다.

- Regression의 경우 결과값의 범위가 실수값 $(-\infty, \infty)$으로 주어졌으나, classfication의 경우 출력값이 0보다 크고 각 클래스들의 확률의 합이 1이 되어야 합니다.

- 이와 같은 조건을 만족 시키기 위해 softmax layer를 지나기 전, logit space상에서 variance를 둡니다.

$$\hat{\mathbf{x}}_i \mid \mathbf{W} \sim \mathcal{N}\left(\mathbf{f}_i^{\mathbf{W}}, \left(\sigma_i^{\mathbf{W}}\right)^2\right)$$

$$\hat{\mathbf{p}}_i = \mathrm{Softmax}(\hat{\mathbf{x}}_i)$$

Aleatoric Uncertainty In Bayesian Classification

- Classification의 경우 aleatoric uncertainty를 측정하기 위한 loss function은 다음과 같이 나타낼 수 있습니다.

$$\hat{\mathbf{x}}_{i,t} = \mathbf{f}_i^{\mathbf{W}} + \sigma_i^{\mathbf{W}} \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, I)$$

$$\mathcal{L}_x = \sum_i \log \frac{1}{T} \sum_t \exp\left(\hat{x}_{i,t,c} - \log \sum_{c'} \exp \hat{x}_{i,t,c'} \right)$$

$Softmax(\hat{\mathbf{x}}_{i,t})$

i: data point의 index

T: ϵ_t의 샘플링 횟수

$\hat{x}_{i,t,c}$: legit vector $\mathbf{x}_{i,t}$의 c번째 element

Epistemic Uncertainty In Bayesian Classification

- Bayesian Neural Networks에서 classification의 경우 predictive distribution은 다음과 같이 모델의 출력값을 평균내어 근사할 수 있습니다.

$$p(y = c \mid \mathbf{x}, \mathbf{X}, \mathbf{Y}) \approx \frac{1}{T} \sum_{t=1}^{T} \text{Softmax}\left(\mathbf{f}^{\widehat{\mathbf{W}}_t}(\mathbf{x})\right)$$

$\mathbf{x}$: test시의 입력, y: test시의 출력, c: 출력 클래스
$\mathbf{X}$: 트레이닝시의 입력 데이터셋, $\mathbf{Y}$: 트레이닝시의 출력 데이터셋
T: test 횟수, $\hat{W}_t$: t회 test때 sampling된 weight, $\mathbf{f}^{\hat{W}_t}$: t회 test때의 neural network

- 이렇게 구한 categorical distribution의 entropy를 측정하여 uncertainty를 나타내는 대표값으로 둘 수 있습니다.

$$H(\mathbf{p}) = -\sum_{c=1}^{C} p_c \log p_c$$

$\mathbf{p}$: 출력 클래스들에 대한 확률값을 가진 벡터

Kendall, Alex, and Yarin Gal. "What uncertainties do we need in bayesian deep learning for computer vision?." Advances in neural information processing systems. 2017.

GAN

Generative Adversarial Nets

- GAN (Generative Adversarial Nets)은 generator와 discriminator로 구분되어 있습니다.
- Generator는 discriminator가 구별을 하지 못하도록 트레이닝 됩니다. 따라서 생성된 결과가 노이즈에서 점차 트레이닝 데이터와 가깝게 변해갑니다.
- Discriminator는 트레이닝 데이터와 generator가 만든 결과를 구분하도록 트레이닝 됩니다. Generator가 만든 결과를 입력으로 받으면 출력을 0으로, 트레이닝 데이터를 입력으로 받으면 출력을 1로 내보내도록 학습합니다. 그러나 generator가 생성하는 이미지가 점차 트레이닝 데이터와 가까워짐에 따라 discriminator가 제대로 구별하지 못하게 됩니다.
- GAN은 VAE와 더불어 generative model로써 대표적인 모델입니다. VAE가 blurry한 결과를 내보내는데 반해 GAN는 매우 선명한 결과를 내보내는 장점을 가지고 있습니다. 하지만 GAN은 model collapse 현상이나 트레이닝이 매우 어렵다는 점 등 단점을 가지고 있습니다.

Generative Adversarial Nets

Abstract

We propose a new framework for estimating generative models via an adversarial process, in which we simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game. In the space of arbitrary functions G and D, a unique solution exists, with G recovering the training data distribution and D equal to $\frac{1}{2}$ everywhere. In the case where G and D are defined by multilayer perceptrons, the entire system can be trained with backpropagation. There is no need for any Markov chains or unrolled approximate inference networks during either training or generation of samples. Experiments demonstrate the potential of the framework through qualitative and quantitative evaluation of the generated samples.

I. Goodfellow et al., Generative adversarial nets

Generative Adversarial Nets

- GAN, Value function $V(G, D)$ 정의

3 Adversarial nets

The adversarial modeling framework is most straightforward to apply when the models are both multilayer perceptrons. To learn the generator's distribution p_g over data $\boldsymbol{x}$, we define a prior on input noise variables $p_{\boldsymbol{z}}(\boldsymbol{z})$, then represent a mapping to data space as $G(\boldsymbol{z}; \theta_g)$, where G is a differentiable function represented by a multilayer perceptron with parameters θ_g. We also define a second multilayer perceptron $D(\boldsymbol{x}; \theta_d)$ that outputs a single scalar. $D(\boldsymbol{x})$ represents the probability that $\boldsymbol{x}$ came from the data rather than p_g. We train D to maximize the probability of assigning the correct label to both training examples and samples from G. We simultaneously train G to minimize $\log(1 - D(G(\boldsymbol{z})))$. In other words, D and G play the following two-player minimax game with value function $V(G, D)$:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]. \quad (1)$$

I. Goodfellow et al., Generative adversarial nets

Generative Adversarial Nets

- Discriminator의 트레이닝 방법, Generator의 loss saturation 문제

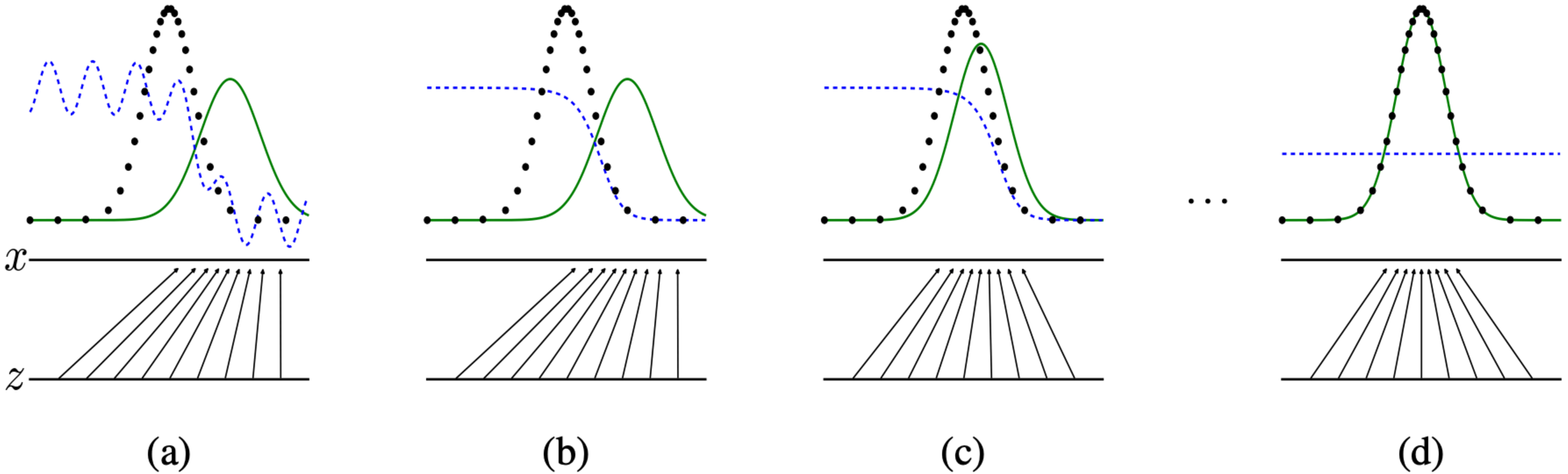
In the next section, we present a theoretical analysis of adversarial nets, essentially showing that the training criterion allows one to recover the data generating distribution as G and D are given enough capacity, i.e., in the non-parametric limit. See Figure 1 for a less formal, more pedagogical explanation of the approach. In practice, we must implement the game using an iterative, numerical approach. Optimizing D to completion in the inner loop of training is computationally prohibitive, and on finite datasets would result in overfitting. Instead, we alternate between k steps of optimizing D and one step of optimizing G. This results in D being maintained near its optimal solution, so long as G changes slowly enough. The procedure is formally presented in Algorithm 1.

In practice, equation 1 may not provide sufficient gradient for G to learn well. Early in learning, when G is poor, D can reject samples with high confidence because they are clearly different from the training data. In this case, $\log(1 - D(G(\boldsymbol{z})))$ saturates. Rather than training G to minimize $\log(1 - D(G(\boldsymbol{z})))$ we can train G to maximize $\log D(G(\boldsymbol{z}))$. This objective function results in the same fixed point of the dynamics of G and D but provides much stronger gradients early in learning.

I. Goodfellow et al., Generative adversarial nets

Generative Adversarial Nets

- 트레이닝 과정 도식화



I. Goodfellow et al., Generative adversarial nets

Generative Adversarial Nets

- 트레이닝 Pseudo Code

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

I. Goodfellow et al., Generative adversarial nets

Generative Adversarial Nets

- GAN의 optimal solution이 $p_g = p_{data}$임을 증명, discriminator의 optimal solution 구함

4.1 Global Optimality of $p_g = p_{\text{data}}$

We first consider the optimal discriminator D for any given generator G.

Proposition 1. *For G fixed, the optimal discriminator D is*

$$D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})} \tag{2}$$

Proof. The training criterion for the discriminator D, given any generator G, is to maximize the quantity $V(G, D)$

$$\begin{aligned} V(G,D) &= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x}))dx + \int_z p_{\boldsymbol{z}}(\boldsymbol{z}) \log(1 - D(g(\boldsymbol{z})))dz \\ &= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x})) + p_g(\boldsymbol{x}) \log(1 - D(\boldsymbol{x}))dx \end{aligned} \tag{3}$$

For any $(a, b) \in \mathbb{R}^2 \setminus \{0, 0\}$, the function $y \to a\log(y) + b\log(1 - y)$ achieves its maximum in $[0, 1]$ at $\frac{a}{a+b}$. The discriminator does not need to be defined outside of $Supp(p_{\text{data}}) \cup Supp(p_g)$, concluding the proof. □

I. Goodfellow et al., Generative adversarial nets

Generative Adversarial Nets

- Optimal discriminator에 대한 generator의 value function

Note that the training objective for D can be interpreted as maximizing the log-likelihood for estimating the conditional probability $P(Y = y|\boldsymbol{x})$, where Y indicates whether $\boldsymbol{x}$ comes from p_{data} (with $y = 1$) or from p_g (with $y = 0$). The minimax game in Eq. 1 can now be reformulated as:

$$
\begin{aligned}
C(G) &= \max_D V(G, D) \\
&= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}}[\log(1 - D_G^*(G(\boldsymbol{z})))] \qquad (4) \\
&= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x} \sim p_g}[\log(1 - D_G^*(\boldsymbol{x}))] \\
&= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] + \mathbb{E}_{\boldsymbol{x} \sim p_g}\left[\log \frac{p_g(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right]
\end{aligned}
$$

I. Goodfellow et al., Generative adversarial nets

Generative Adversarial Nets

- Generator의 criterion은 JSD (Jensen-Shannon divergence)와 같음을 보임

Theorem 1. *The global minimum of the virtual training criterion $C(G)$ is achieved if and only if $p_g = p_{data}$. At that point, $C(G)$ achieves the value $-\log 4$.*

Proof. For $p_g = p_{\text{data}}$, $D_G^*(\boldsymbol{x}) = \frac{1}{2}$, (consider Eq. 2). Hence, by inspecting Eq. 4 at $D_G^*(\boldsymbol{x}) = \frac{1}{2}$, we find $C(G) = \log \frac{1}{2} + \log \frac{1}{2} = -\log 4$. To see that this is the best possible value of $C(G)$, reached only for $p_g = p_{\text{data}}$, observe that

$$\mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} [-\log 2] + \mathbb{E}_{\boldsymbol{x} \sim p_g} [-\log 2] = -\log 4$$

and that by subtracting this expression from $C(G) = V(D_G^*, G)$, we obtain:

$$C(G) = -\log(4) + KL\left(p_{\text{data}} \middle\| \frac{p_{\text{data}} + p_g}{2}\right) + KL\left(p_g \middle\| \frac{p_{\text{data}} + p_g}{2}\right) \tag{5}$$

where KL is the Kullback–Leibler divergence. We recognize in the previous expression the Jensen–Shannon divergence between the model's distribution and the data generating process:

$$C(G) = -\log(4) + 2 \cdot JSD\left(p_{\text{data}} \| p_g\right) \tag{6}$$

Since the Jensen–Shannon divergence between two distributions is always non-negative, and zero iff they are equal, we have shown that $C^* = -\log(4)$ is the global minimum of $C(G)$ and that the only solution is $p_g = p_{\text{data}}$, i.e., the generative model perfectly replicating the data distribution. □

Jensen Shannon Divergence

$$\text{JSD}(P\|Q) = \frac{1}{2}D(P\|M) + \frac{1}{2}D(Q\|M)$$

where $M = \frac{1}{2}(P + Q)$

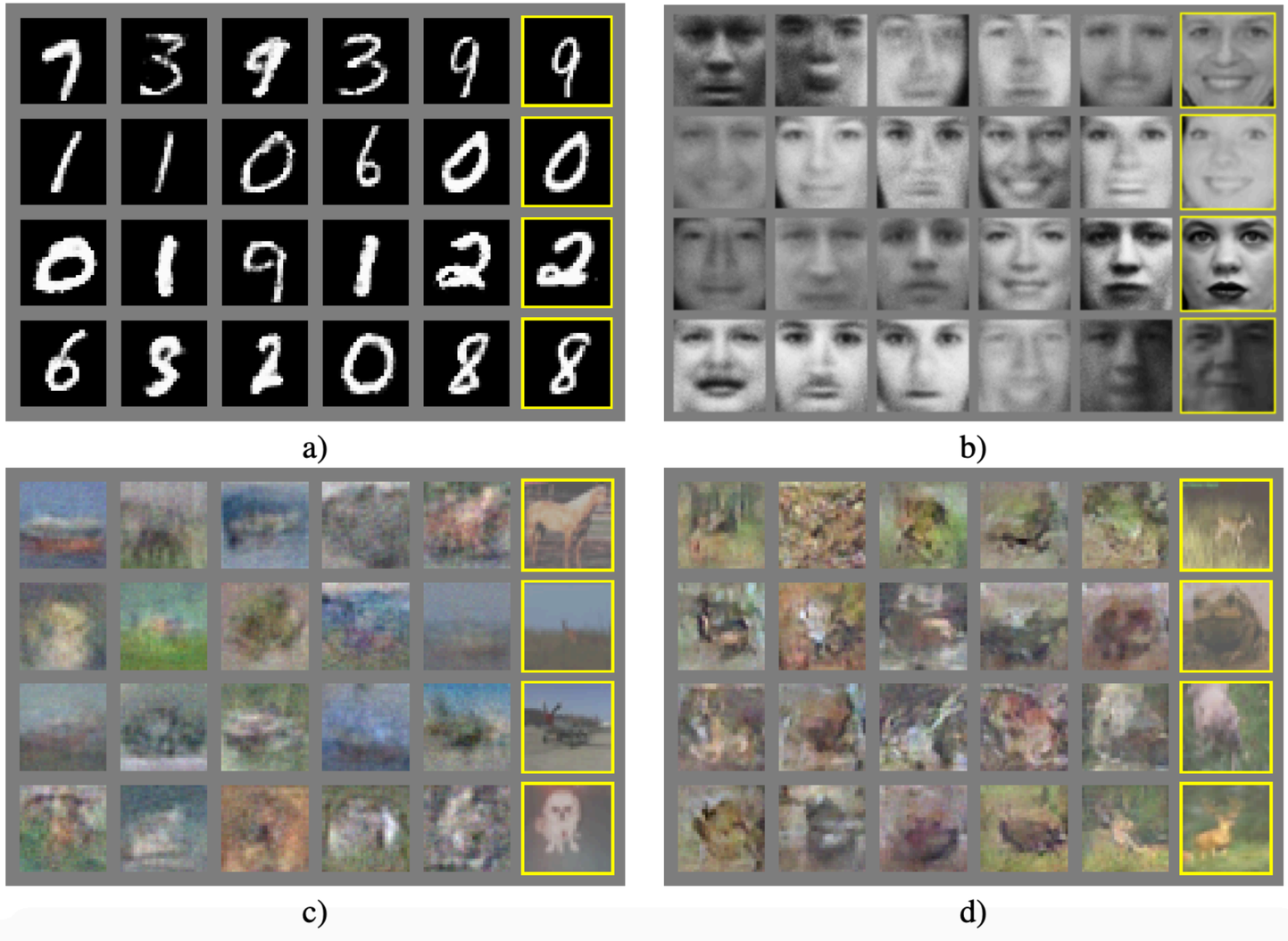
I. Goodfellow et al., Generative adversarial nets

Generative Adversarial Nets

$$\min_G \max_D V(D,G) = \min_G V(D^*,G) \qquad \text{for fixed optimal } \mathcal{D}$$

$$\begin{aligned}
V(D^*,G) &= \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D^*(\mathbf{x})] + \mathbb{E}_{\mathbf{x}\sim p_g}[\log(1-D^*(\mathbf{x}))] \\
&= \int_{\mathbf{x}} d\mathbf{x}\; p_{\text{data}}(\mathbf{x}) \log \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x})+p_g(\mathbf{x})} + \int_{\mathbf{x}} d\mathbf{x}\; p_g(\mathbf{x}) \log \frac{p_g(\mathbf{x})}{p_{\text{data}}(\mathbf{x})+p_g(\mathbf{x})} \\
&= -2\log 2 + 2\log 2 + \int_{\mathbf{x}} d\mathbf{x}\; p_{\text{data}}(\mathbf{x}) \log \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x})+p_g(\mathbf{x})} + \int_{\mathbf{x}} d\mathbf{x}\; p_g(\mathbf{x}) \log \frac{p_g(\mathbf{x})}{p_{\text{data}}(\mathbf{x})+p_g(\mathbf{x})} \\
&= -\log 4 + \int_{\mathbf{x}} d\mathbf{x}\; p_{\text{data}}(\mathbf{x}) \log \frac{2\cdot p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x})+p_g(\mathbf{x})} + \int_{\mathbf{x}} d\mathbf{x}\; p_g(\mathbf{x}) \log \frac{2\cdot p_g(\mathbf{x})}{p_{\text{data}}(\mathbf{x})+p_g(\mathbf{x})} \\
&= -\log 4 + KL\left(p_{\text{data}} \middle|\middle| \frac{p_{\text{data}}+p_g}{2}\right) + KL\left(p_g \middle|\middle| \frac{p_{\text{data}}+p_g}{2}\right) \\
&= -\log 4 + 2\cdot JSD(p_{\text{data}}||p_g)
\end{aligned}$$

Generative Adversarial Nets



I. Goodfellow et al., Generative adversarial nets

Deep Convolutional GAN

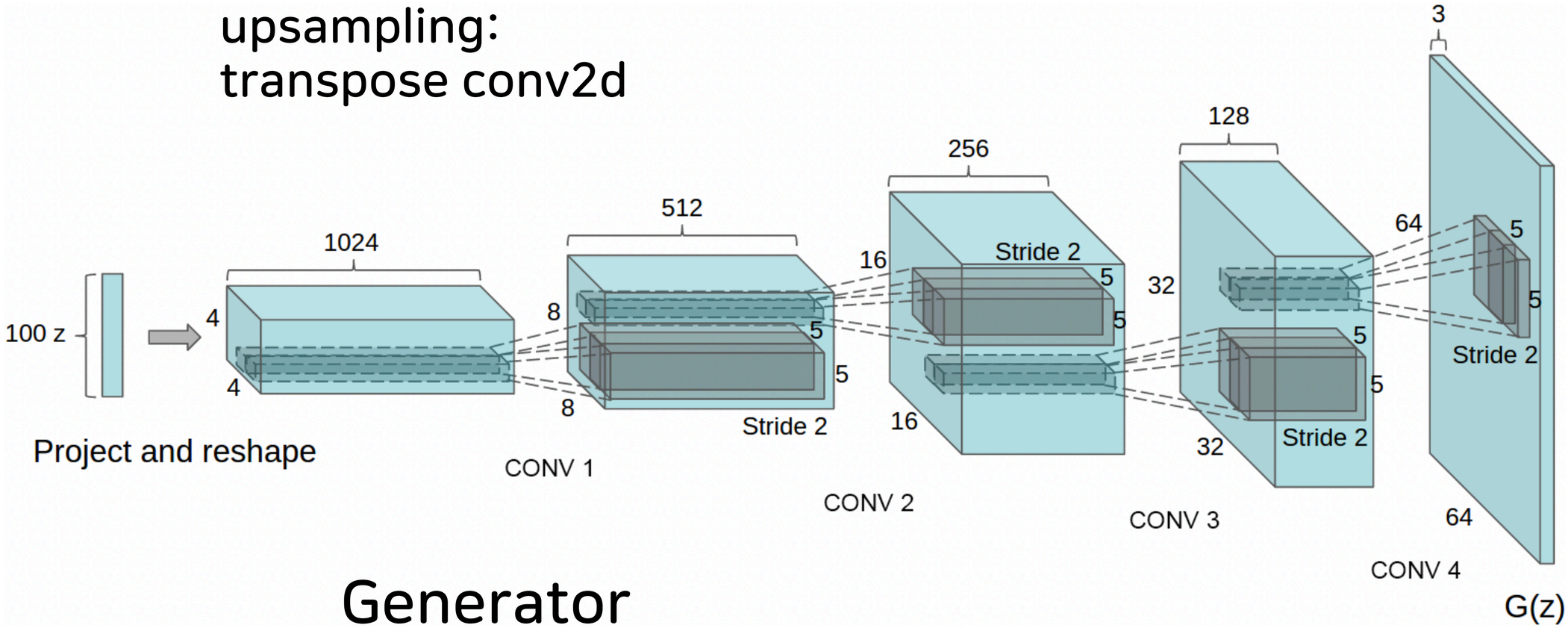
- Guidelines for Stable DCGAN

Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

A. Radford, et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

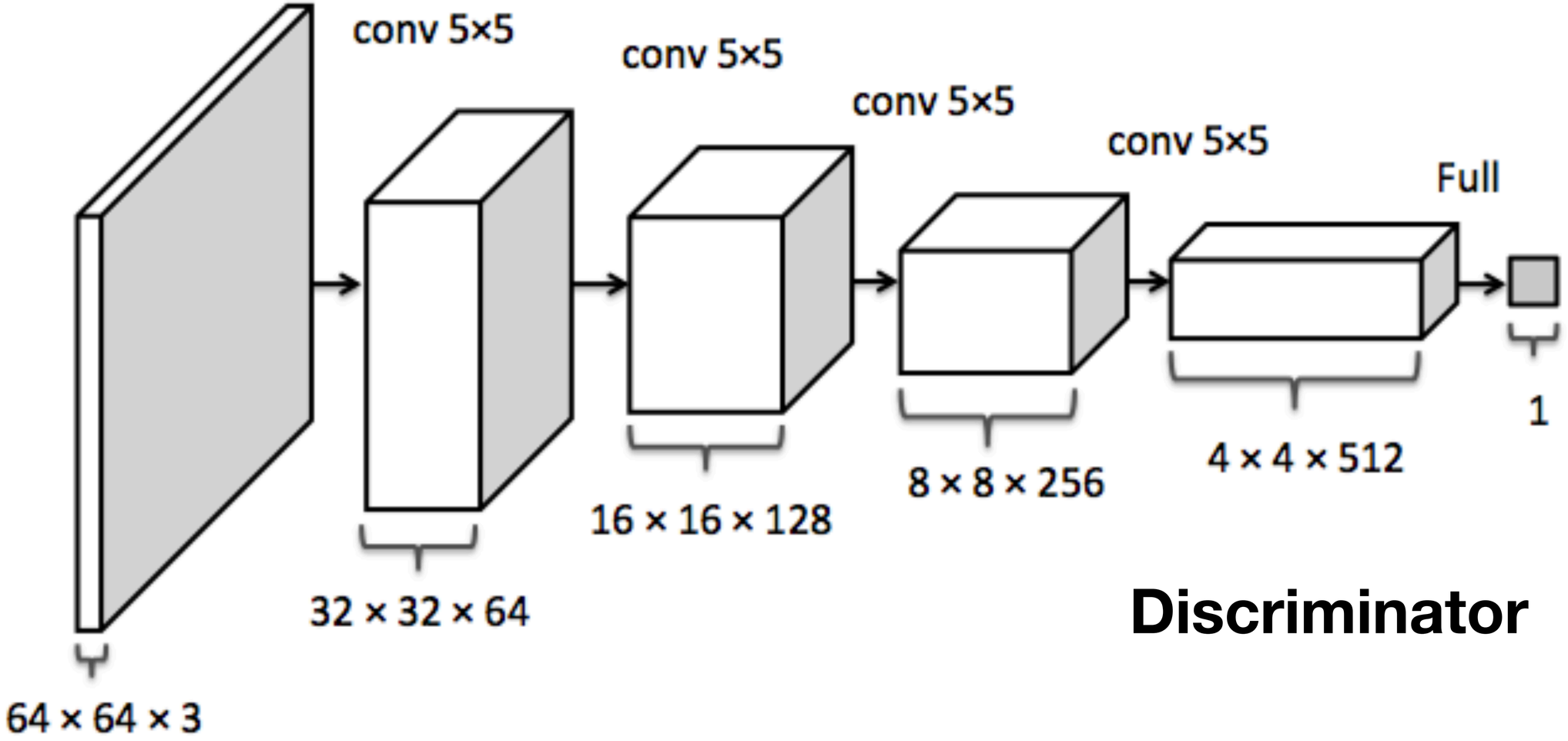
Deep Convolutional GAN



Generator

A. Radford, et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

Deep Convolutional GAN



R. Yeh, et al., Semantic Image Inpainting with Perceptual and Contextual Losses

Deep Convolutional GAN

- DCGAN

A. Radford, et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

InfoGAN

InfoGAN

- InfoGAN은 unsupervised 방식으로 disentagled representation을 학습합니다. 따라서 VAE의 GAN버전이라고 할 수 있습니다.
- 이를 위해 latent code와 generator가 만들어낸 결과의 mutual information을 최대화합니다.
- Generator의 입력으로 주어지는 노이즈 데이터 추가적으로 discrete latent code와 continuous latent code를 덧붙입니다.
- Discriminator의 중간 레이어의 출력을 가져다 Q-network라 부르는 posterior를 근사하는 모듈에 입력합니다. 그리고 이에 대한 출력으로 본래 discrete latent code와 continuous latent code을 regression합니다.
- Bayesian이 적용된 모델이라고는 볼 수 없지만 mutual information과 variational inference의 아이디어를 차용해 만든 모델로써 의미가 있습니다.

InfoGAN

Abstract

This paper describes InfoGAN, an information-theoretic extension to the Generative Adversarial Network that is able to learn disentangled representations in a completely unsupervised manner. InfoGAN is a generative adversarial network that also maximizes the mutual information between a small subset of the latent variables and the observation. We derive a lower bound of the mutual information objective that can be optimized efficiently. Specifically, InfoGAN successfully disentangles writing styles from digit shapes on the MNIST dataset, pose from lighting of 3D rendered images, and background digits from the central digit on the SVHN dataset. It also discovers visual concepts that include hair styles, presence/absence of eyeglasses, and emotions on the CelebA face dataset. Experiments show that InfoGAN learns interpretable representations that are competitive with representations learned by existing supervised methods.

X. Chen, et al., InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets

InfoGAN

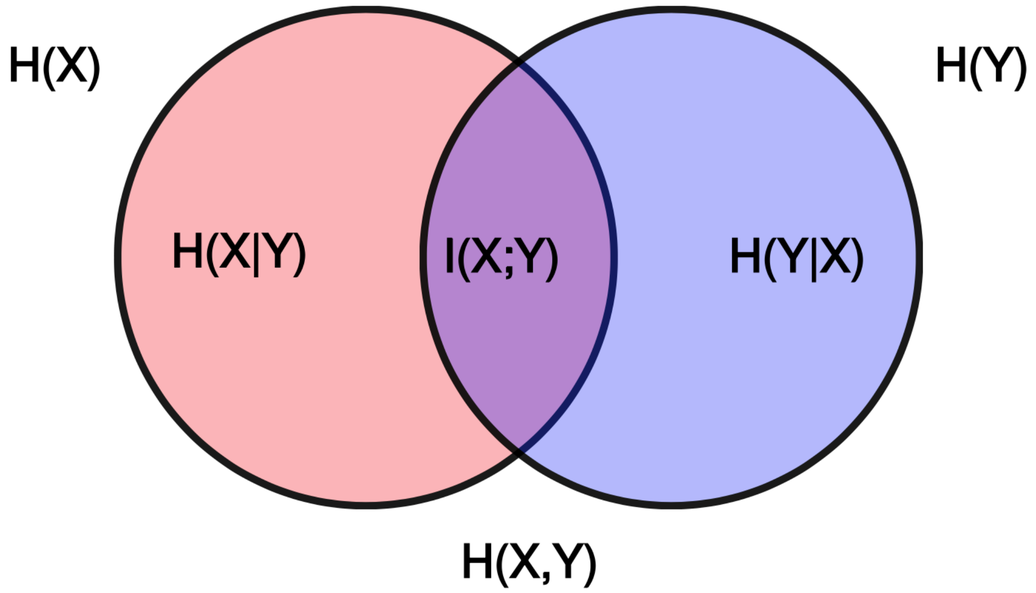
We now propose a method for discovering these latent factors in an unsupervised way: we provide the generator network with both the incompressible noise z and the latent code c, so the form of the generator becomes $G(z, c)$. However, in standard GAN, the generator is free to ignore the additional latent code c by finding a solution satisfying $P_G(x|c) = P_G(x)$. To cope with the problem of trivial codes, we propose an information-theoretic regularization: there should be high mutual information between latent codes c and generator distribution $G(z, c)$. Thus $I(c; G(z, c))$ should be high.

In information theory, mutual information between X and Y, $I(X; Y)$, measures the "amount of information" learned from knowledge of random variable Y about the other random variable X. The mutual information can be expressed as the difference of two entropy terms:

$$I(X; Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) \tag{2}$$

X. Chen, et al., InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets

InfoGAN



$$
\begin{aligned}
I(X;Y) &= \int_Y \int_X p(x,y) \log \left(\frac{p(x,y)}{p(x)p(y)} \right) \mathrm{d}x \, \mathrm{d}y \\
&= H(X) - H(X|Y) \\
&= H(Y) - H(Y|X)
\end{aligned}
$$

Entropy: Uncertainty of Random Variables

Mutual Information:
Uncertainty of X - Uncertainty of X given Y
= Removed Uncertainty of X given Y
= Information of X given Y

en.wikipedia.org/wiki/Mutual_information

InfoGAN

This definition has an intuitive interpretation: $I(X;Y)$ is the reduction of uncertainty in X when Y is observed. If X and Y are independent, then $I(X;Y) = 0$, because knowing one variable reveals nothing about the other; by contrast, if X and Y are related by a deterministic, invertible function, then maximal mutual information is attained. This interpretation makes it easy to formulate a cost: given any $x \sim P_G(x)$, we want $P_G(c|x)$ to have a small entropy. In other words, the information in the latent code c should not be lost in the generation process. Similar mutual information inspired objectives have been considered before in the context of clustering [26–28]. Therefore, we propose to solve the following information-regularized minimax game:

$$\min_G \max_D V_I(D,G) = V(D,G) - \lambda I(c; G(z,c)) \tag{3}$$

X. Chen, et al., InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets

InfoGAN

5 Variational Mutual Information Maximization

In practice, the mutual information term $I(c; G(z, c))$ is hard to maximize directly as it requires access to the posterior $P(c|x)$. Fortunately we can obtain a lower bound of it by defining an auxiliary distribution $Q(c|x)$ to approximate $P(c|x)$:

$$
\begin{aligned}
I(c; G(z, c)) &= H(c) - H(c|G(z, c)) \\
&= \mathbb{E}_{x \sim G(z,c)}[\mathbb{E}_{c' \sim P(c|x)}[\log P(c'|x)]] + H(c) \\
&= \mathbb{E}_{x \sim G(z,c)}[\underbrace{D_{\mathrm{KL}}(P(\cdot|x) \parallel Q(\cdot|x))}_{\geq 0} + \mathbb{E}_{c' \sim P(c|x)}[\log Q(c'|x)]] + H(c) \qquad (4) \\
&\geq \mathbb{E}_{x \sim G(z,c)}[\mathbb{E}_{c' \sim P(c|x)}[\log Q(c'|x)]] + H(c)
\end{aligned}
$$

This technique of lower bounding mutual information is known as Variational Information Maximization [29]. We note in addition that the entropy of latent codes $H(c)$ can be optimized over as well since for common distributions it has a simple analytical form. However, in this paper we opt for simplicity by fixing the latent code distribution and we will treat $H(c)$ as a constant. So far we have bypassed the problem of having to compute the posterior $P(c|x)$ explicitly via this lower bound but we still need to be able to sample from the posterior in the inner expectation. Next we state a simple lemma, with its proof deferred to Appendix, that removes the need to sample from the posterior.

X. Chen, et al., InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets

InfoGAN

Lemma 5.1 *For random variables X, Y and function $f(x, y)$ under suitable regularity conditions:* $\mathbb{E}_{x\sim X, y\sim Y|x}[f(x, y)] = \mathbb{E}_{x\sim X, y\sim Y|x, x'\sim X|y}[f(x', y)]$.

By using Lemma A.1, we can define a variational lower bound, $L_I(G, Q)$, of the mutual information, $I(c; G(z, c))$:

$$\begin{aligned} L_I(G, Q) &= E_{c\sim P(c), x\sim G(z,c)}[\log Q(c|x)] + H(c) \\ &= E_{x\sim G(z,c)}[\mathbb{E}_{c'\sim P(c|x)}[\log Q(c'|x)]] + H(c) \\ &\leq I(c; G(z, c)) \end{aligned} \tag{5}$$

We note that $L_I(G, Q)$ is easy to approximate with Monte Carlo simulation. In particular, L_I can be maximized w.r.t. Q directly and w.r.t. G via the reparametrization trick. Hence $L_I(G, Q)$ can be added to GAN's objectives with no change to GAN's training procedure and we call the resulting algorithm Information Maximizing Generative Adversarial Networks (InfoGAN).

X. Chen, et al., InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets

InfoGAN

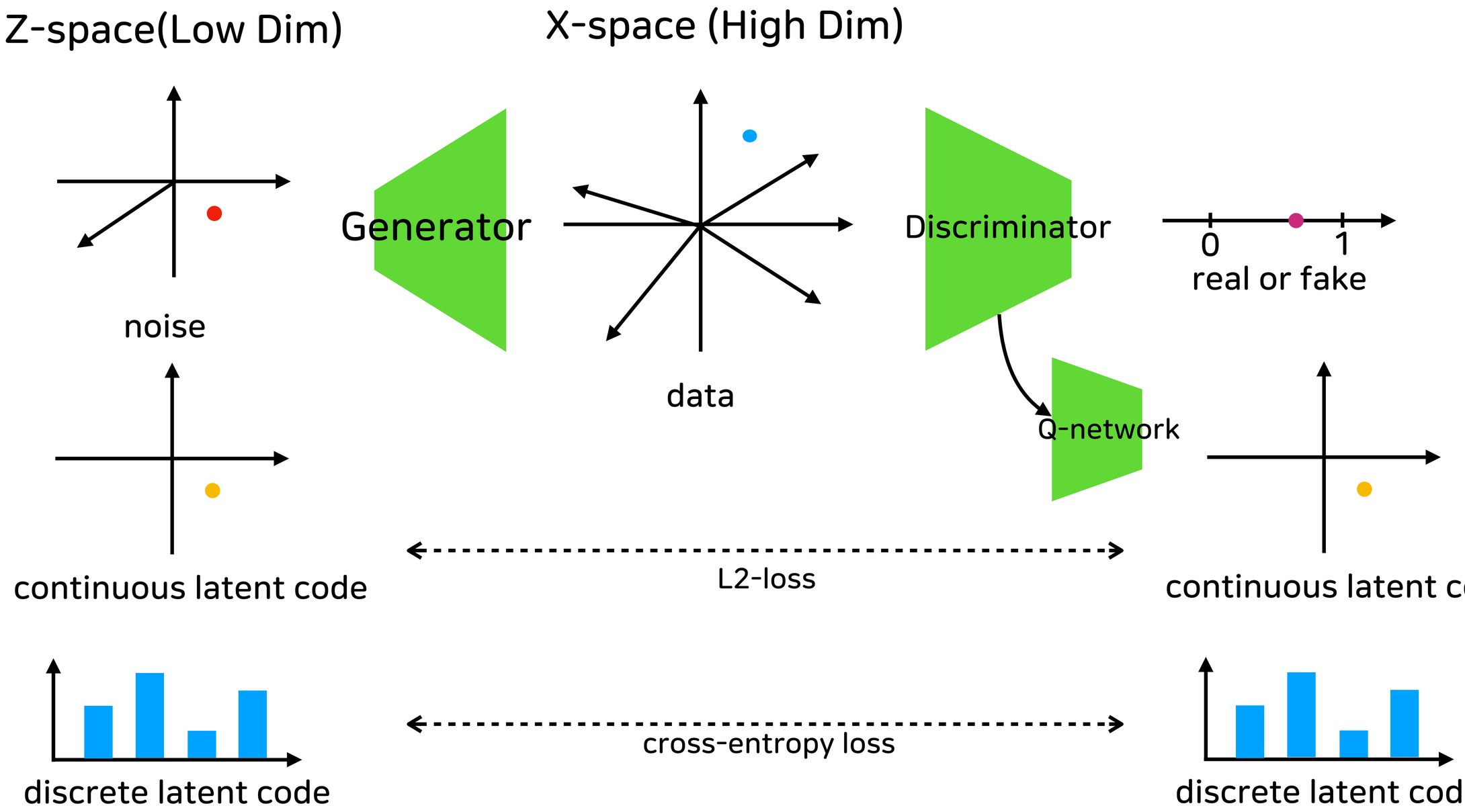
Eq (4) shows that the lower bound becomes tight as the auxiliary distribution Q approaches the true posterior distribution: $\mathbb{E}_x[D_{\mathrm{KL}}(P(\cdot|x) \parallel Q(\cdot|x))] \to 0$. In addition, we know that when the variational lower bound attains its maximum $L_I(G, Q) = H(c)$ for discrete latent codes, the bound becomes tight and the maximal mutual information is achieved. In Appendix, we note how InfoGAN can be connected to the Wake-Sleep algorithm [30] to provide an alternative interpretation.

Hence, InfoGAN is defined as the following minimax game with a variational regularization of mutual information and a hyperparameter λ:

$$\min_{G,Q} \max_{D} V_{\mathrm{InfoGAN}}(D, G, Q) = V(D, G) - \lambda L_I(G, Q) \tag{6}$$

X. Chen, et al., InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets

InfoGAN


Z-space(Low Dim)
X-space (High Dim)
Generator
Discriminator
0
1
real or fake
noise
data
Q-network
continuous latent code
L2-loss
continuous latent code
discrete latent code
cross-entropy loss
discrete latent code

InfoGAN

6 Implementation

In practice, we parametrize the auxiliary distribution Q as a neural network. In most experiments, Q and D share all convolutional layers and there is one final fully connected layer to output parameters for the conditional distribution $Q(c|x)$, which means InfoGAN only adds a negligible computation cost to GAN. We have also observed that $L_I(G, Q)$ always converges faster than normal GAN objectives and hence InfoGAN essentially comes for *free* with GAN.

For categorical latent code c_i, we use the natural choice of softmax nonlinearity to represent $Q(c_i|x)$. For continuous latent code c_j, there are more options depending on what is the true posterior $P(c_j|x)$. In our experiments, we have found that simply treating $Q(c_j|x)$ as a factored Gaussian is sufficient.

Even though InfoGAN introduces an extra hyperparameter λ, it's easy to tune and simply setting to 1 is sufficient for discrete latent codes. When the latent code contains continuous variables, a smaller λ is typically used to ensure that $\lambda L_I(G, Q)$, which now involves differential entropy, is on the same scale as GAN objectives.

Since GAN is known to be difficult to train, we design our experiments based on existing techniques introduced by DC-GAN [19], which are enough to stabilize InfoGAN training and we did not have to introduce new trick. Detailed experimental setup is described in Appendix.

X. Chen, et al., InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets

InfoGAN

(a) Azimuth (pose)

(b) Elevation

(c) Lighting

(d) Wide or Narrow

X. Chen, et al., InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets

InfoGAN

(a) Rotation (b) Width

X. Chen, et al., InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets

InfoGAN

(a) Continuous variation: Lighting

(b) Discrete variation: Plate Context

X. Chen, et al., InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets

InfoGAN

(a) Azimuth (pose)

(b) Presence or absence of glasses

(c) Hair style

(d) Emotion

X. Chen, et al., InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets

Generative Adversarial Nets

- WGAN

Figure 5: Algorithms trained with a DCGAN generator. Left: WGAN algorithm. Right: standard GAN formulation. Both algorithms produce high quality samples.

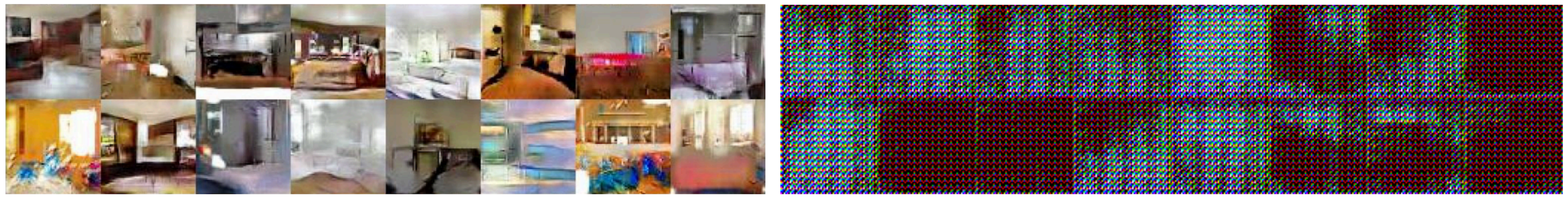


Figure 6: Algorithms trained with a generator without batch normalization and constant number of filters at every layer (as opposed to duplicating them every time as in [18]).

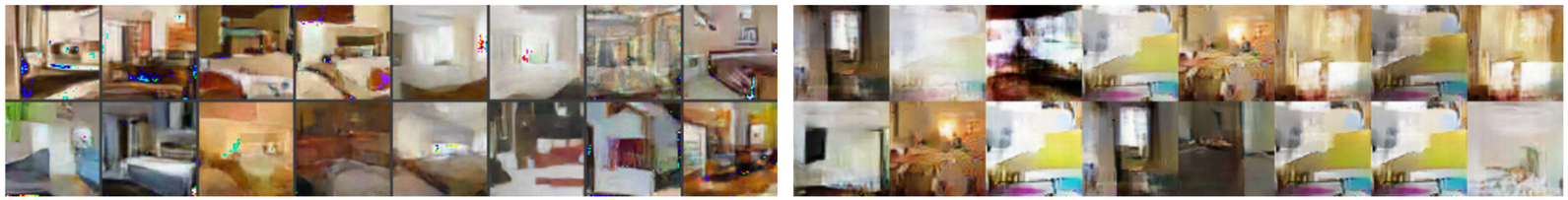


Figure 7: Algorithms trained with an MLP generator with 4 layers and 512 units with ReLU nonlinearities. The number of parameters is similar to that of a DCGAN, but it lacks a

Martin Arjovsky, et al., Wasserstein GAN

Generative Adversarial Nets

- Progressive GAN

Tero Karras, et al., Progressive Growing of GANs for Improved Quality, Stability, and Variation

Generative Adversarial Nets

- Style GAN

Tero Karras, et al., A Style-Based Generator Architecture for Generative Adversarial Networks