# 1 Hamiltonian Monte Carlo Solution to Toy Problems

The goal here is to show that the decomposition we obtained in the active learning examples is not a result of our approximation using black-box $\alpha$-divergence minimization but a property of BNN+LV themselves. To that end we will approximate using HMC. After a burn-in of $500,000$ samples we sample from the posterior $200,000$ samples. We thin out 90% by only keeping every tenth sample.

## 1.1 Heteroscedastic Problem

We define the stochastic function $y = 7\sin(x) + 3|\cos(x/2)|\epsilon$ with $\epsilon \sim \mathcal{N}(0, 1)$. The data availability is limited to specific regions of $x$. In particular, we sample 750 values of $x$ from a mixture of three Gaussians with mean parameters $\{\mu_1 = -4, \mu_2 = 0, \mu_3 = 4\}$, variance parameters $\{\sigma_1 = \frac{2}{5}, \sigma_2 = 0.9, \sigma_3 = \frac{2}{5}\}$ and with each Gaussian component having weight equal to $1/3$ in the mixture. Figure 1a shows the raw data. We have lots of points at both borders of the $x$ axis and in the center, but little data available in between.

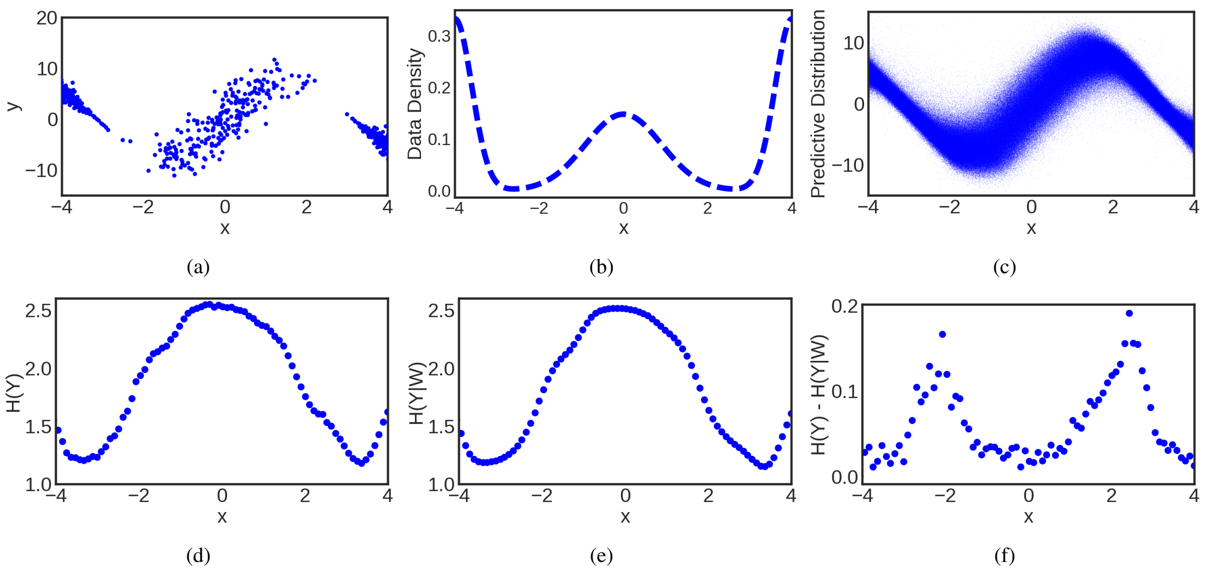

Figure 1: HMC results of active learning example using heteroscedastic data. (a): Raw data. (b): Density of $x$ in raw data. (c): Predictive distribution: $p(y|x)$ of BNN using HMC. (d): Entropy estimate $\mathrm{H}(y|x)$ of predictive distribution for each $x$. (e): Conditional Entropy estimate $\mathbf{E}_{\mathcal{W}}\mathrm{H}(y|x, \mathcal{W})$ of predictive distribution for each $x$. (f): Estimate of reduction in entropy for each $x$.

## 1.2 Bimodal Problem

We consider a toy problem given by a regression task with bimodal data. We define $x \in [-0.5, 2]$ and $y = 10\sin(x) + \epsilon$ with probability $0.5$ and $y = 10\cos(x) + \epsilon$, otherwise, where $\epsilon \sim \mathcal{N}(0, 1)$ and $\epsilon$ is independent of $x$. The data availability is not uniform in $x$. In particular we sample 750 values of $x$ from an exponential distribution with $\lambda = 2$

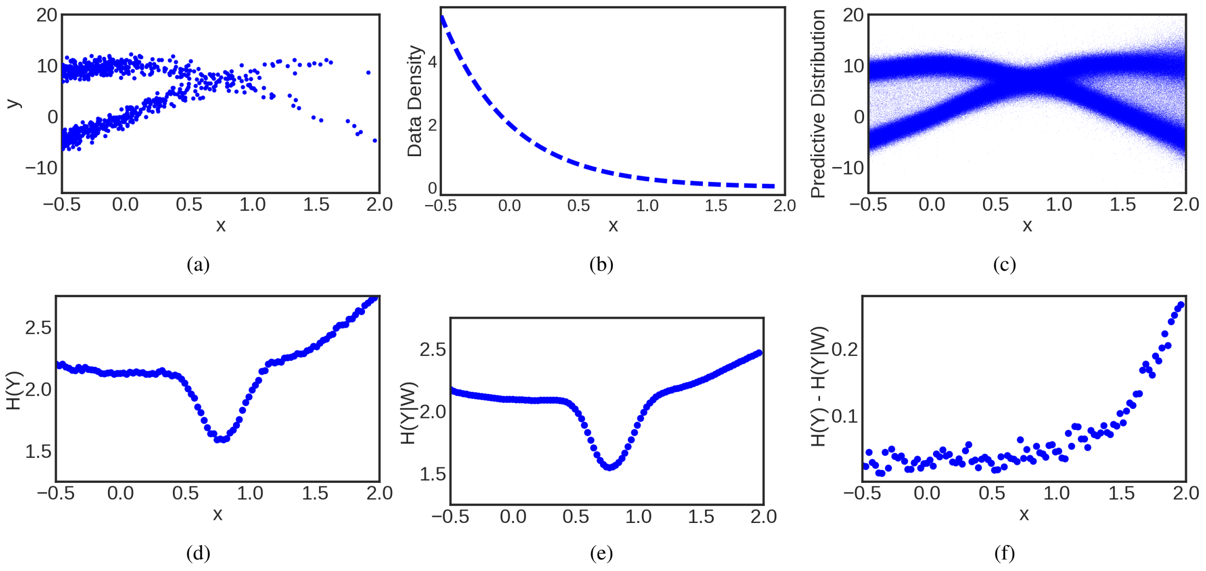

Figure 2: HMC results of active learning example using bimodal data. (a): Raw data. (b): Density of $x$ in raw data. (c): Predictive distribution: $p(y|x)$ of BNN using Hamilton monte carlo. (d): Entropy estimate $\mathrm{H}(y|x)$ of predictive distribution for each $x$. (e): Conditional Entropy estimate $\mathbf{E}_{\mathcal{W}}\mathrm{H}(y|x,\mathcal{W})$ of predictive distribution for each $x$. (f): Estimate of reduction in entropy for each $x$.

## 2 Solutions to Toy Problems for different values of $\alpha$

In the main document we pointed out that the decomposition of uncertainty does not work as good with other values of $\alpha$. We will see in the following, that lower values of $\alpha$ will put more and more emphasis on the latent variable $z$. We observe that the epistemic uncertainty will vanish as the $\alpha$-divergence minimization approaches variational Bayes.

### 2.1 $\alpha = 0.5$

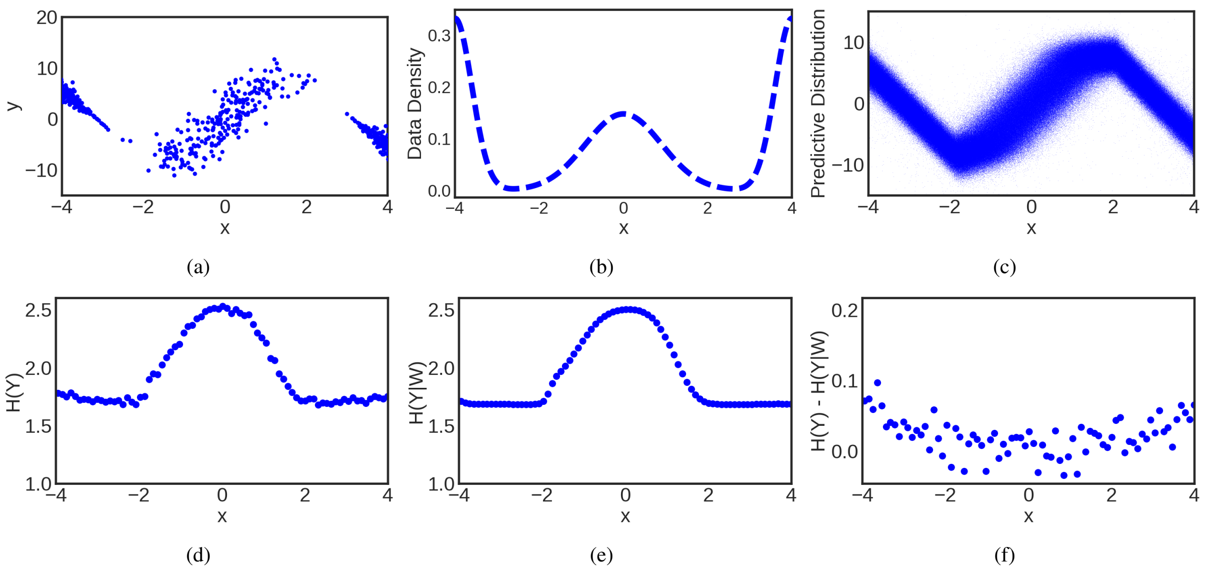

Figure 3: Active learning example using heteroscedastic data using a BNN optimized with bb-$\alpha$ with $\alpha = 0.5$. (a): Raw data. (b): Density of $x$ in raw data. (c): Predictive distribution: $p(y|x)$ of BNN. (d): Entropy estimate $\mathrm{H}(y|x)$ of predictive distribution for each $x$. (e): Conditional Entropy estimate $\mathbf{E}_{\mathcal{W}}\mathrm{H}(y|x,\mathcal{W})$ of predictive distribution for each $x$. (f): Estimate of reduction in entropy for each $x$.

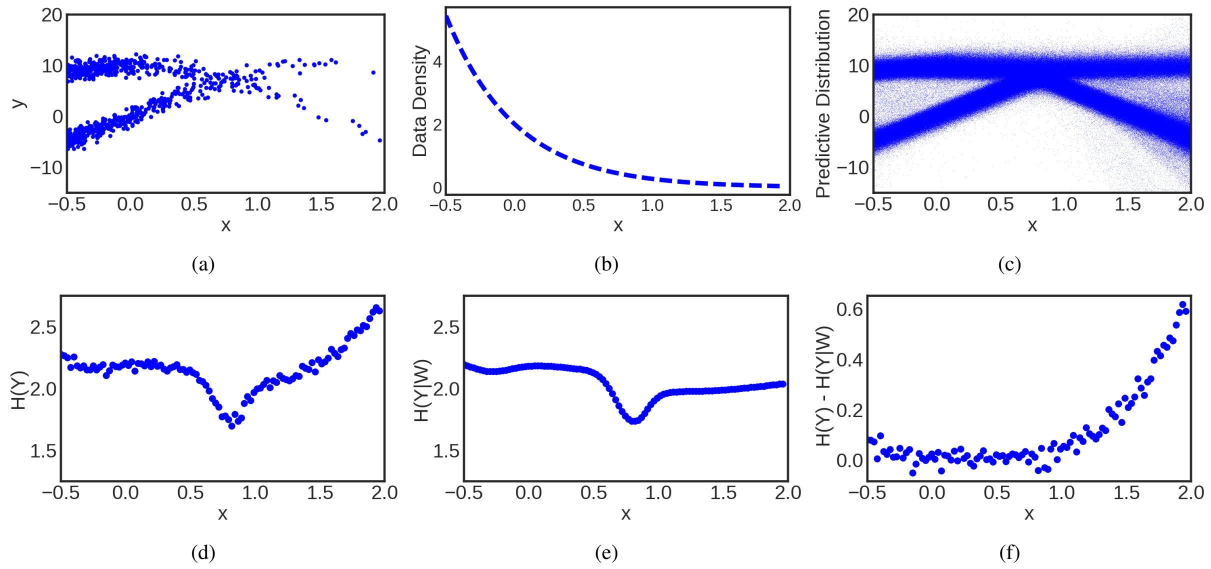

Figure 4: Active learning example using bimodal data using a BNN optimized with bb-$\alpha$ with $\alpha = 0.5$. (a): Raw data. (b): Density of $x$ in raw data. (c): Predictive distribution: $p(y|x)$ of BNN. (d): Entropy estimate $\mathrm{H}(y|x)$ of predictive distribution for each $x$. (e): Conditional Entropy estimate $\mathbf{E}_{\mathcal{W}}\mathrm{H}(y|x,\mathcal{W})$ of predictive distribution for each $x$. (f): Estimate of reduction in entropy for each $x$.

## 2.2 $VB$ solutions to Toy Problems

We approximate the method variational Bayes by setting $\alpha$ to $10^{-6}$.

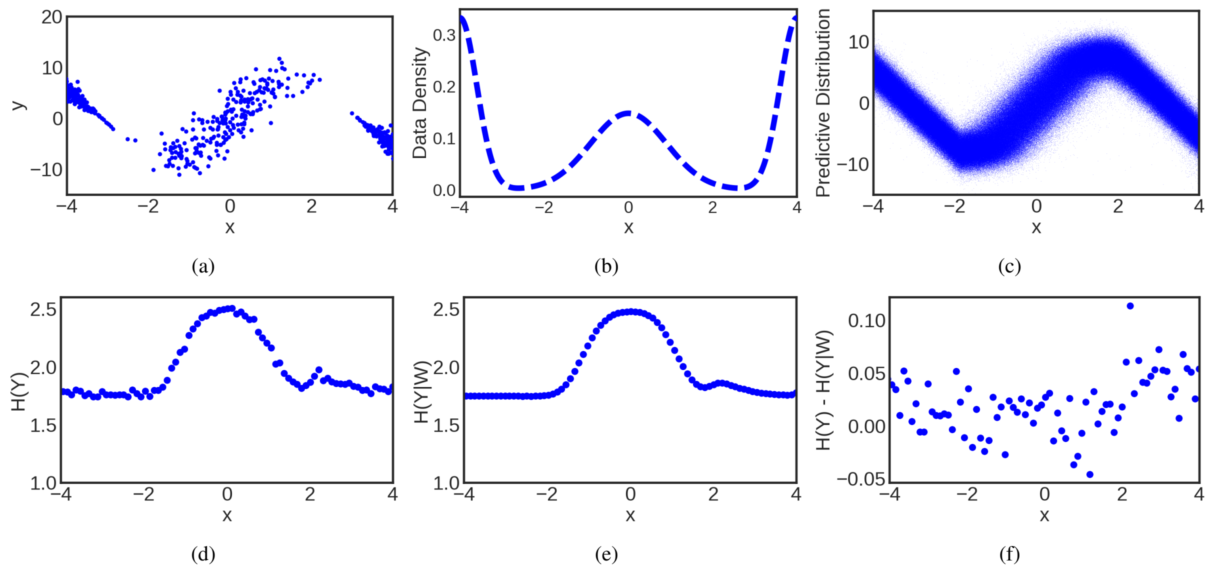

Figure 5: Active learning example using heteroscedastic data using a BNN optimized by variational Bayes. (a): Raw data. (b): Density of $x$ in raw data. (c): Predictive distribution: $p(y|x)$ of BNN. (d): Entropy estimate $\mathrm{H}(y|x)$ of predictive distribution for each $x$. (e): Conditional Entropy estimate $\mathbf{E}_{\mathcal{W}}\mathrm{H}(y|x,\mathcal{W})$ of predictive distribution for each $x$. (f): Estimate of reduction in entropy for each $x$.

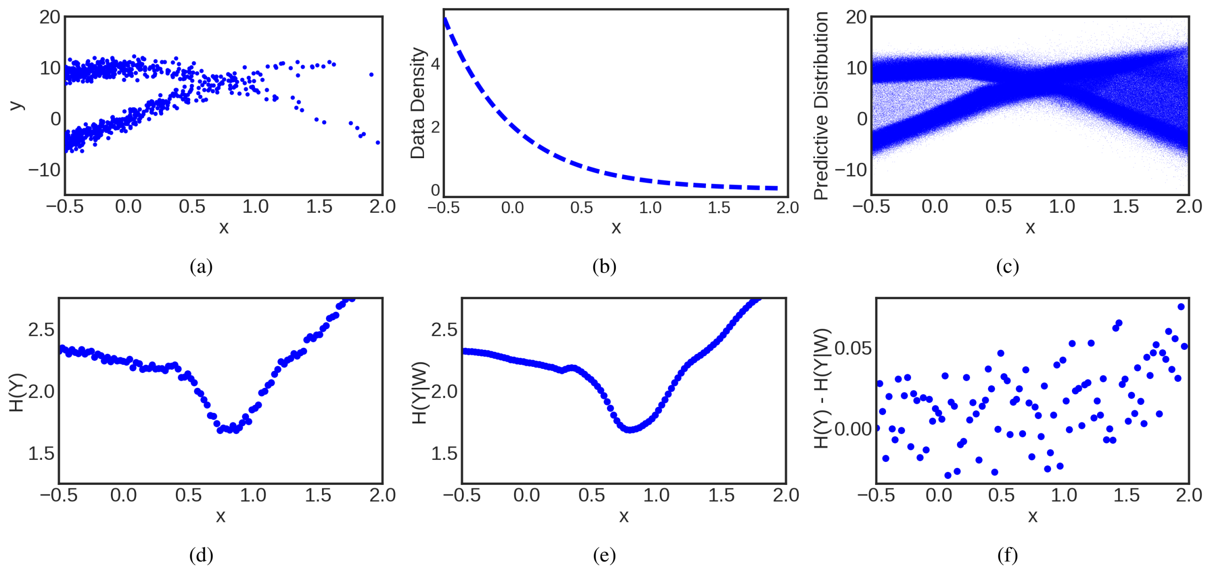

Figure 6: Active learning example using bimodal data using a BNN optimized by variational Bayes. (a): Raw data. (b): Density of $x$ in raw data. (c): Predictive distribution: $p(y|x)$ of BNN. (d): Entropy estimate $\mathrm{H}(y|x)$ of predictive distribution for each $x$. (e): Conditional Entropy estimate $\mathbf{E}_{\mathcal{W}}\mathrm{H}(y|x,\mathcal{W})$ of predictive distribution for each $x$. (f): Estimate of reduction in entropy for each $x$.

## 3 Experiments Specification

### 3.1 Active Learning

All models start with the available described in the respective paragraphs. We train for $5000$ epochs a BNN+LV with two-hidden layer and 20 hidden units per layer. We use Adam as optimizer with a learning rate of $0.001$. For Gaussian processes(GPs) we use the standard RBF kernel using the python GPy implementation. For the entropy estimation we use a nearest-neighbor approach as explained in the main document with $k = 25$ and $500$ samples of $q(W)$ and $500$ samples of $p(z)$.

For active learning we evaluate performance using a held-out test set of size $500$ for the bimodal and heteroscedastic problem and $2500$ for the wet-chicken problem. The test data is sampled uniformly in state (and action) space. In each of the $n = 150$ iterations we sample a pool set of size $50$ uniformly in input space. In each iteration a method can decide to include $5$ data points into the training set. After that, the models are re-trained (from scratch) and the performance is evaluated on the test set.

### 3.2 Wet-chicken

We use the continuous two-dimensional version of the problem. A canoeist is paddling on a two-dimensional river. The canoeist's position at time $t$ is $(x_t, y_t)$. The river has width $w = 5$ and length $l = 5$ with a waterfall at the end, that is, at $y_t = l$. The canoeist wants to move as close to the waterfall as possible because at time $t$ he gets reward $r_t = -(l - y_t)$. However, going beyond the waterfall boundary makes the canoeist fall down, having to start back again at the origin $(0, 0)$. At time $t$ the canoeist can choose an action $(a_{t,x}, a_{t,y}) \in [-1, 1]^2$ that represents the direction and magnitude of his paddling. The river dynamics have stochastic turbulences $s_t$ and drift $v_t$ that depend on the canoeist's position on the $x$ axis. The larger $x_t$, the larger the drift and the smaller $x_t$, the larger the turbulences.

The underlying dynamics are given by the following system of equations. The drift and the turbulence magnitude are given by $v_t = 3x_tw^{-1}$ and $s_t = 3.5 - v_t$, respectively. The new location $(x_{t+1}, y_{t+1})$ is given by the current location $(x_t, y_t)$ and current action $(a_{t,x}, a_{t,y})$ using

$$x_{t+1} = \begin{cases} 0 & \text{if} \quad x_t + a_{t,x} < 0 \\ 0 & \text{if} \quad \hat{y}_{t+1} > l \\ w & \text{if} \quad x_t + a_{t,x} > w \\ x_t + a_{t,x} & \text{otherwise} \end{cases}, \qquad y_{t+1} = \begin{cases} 0 & \text{if} \quad \hat{y}_{t+1} < 0 \\ 0 & \text{if} \quad \hat{y}_{t+1} > l \\ \hat{y}_{t+1} & \text{otherwise} \end{cases}, \tag{1}$$

where $\hat{y}_{t+1} = y_t + (a_{t,y} - 1) + v_t + s_t\tau_t$ and $\tau_t \sim \text{Unif}([-1, 1])$ is a random variable that represents the current turbulence. As the canoeist moves closer to the waterfall, the distribution for the next state becomes increasingly bi-modal because when he is close to the waterfall, the change in the current location can be large if the canoeist falls down the waterfall and starts again at $(0, 0)$. The distribution may also be truncated uniform for states close to the borders. Furthermore the system has heteroscedastic noise, the smaller the value of $x_t$ the higher the noise variance.

## 3.3 Reinforcement Learning

### 3.3.1 Industrial benchmark

Policies in the industrial benchmark specify changes $\Delta_v$, $\Delta_g$ and $\Delta_s$ in three steering variables $v$ (velocity), $g$ (gain) and $s$ (shift) as a function of $\mathbf{s}_t$. In the behavior policy these changes are stochastic and sampled according to

$$\Delta_v = \begin{cases} z_v\,, & \text{if } v(t) < 40 \\ -z_v\,, & \text{if } v(t) > 60 \\ u_v\,, & \text{otherwise} \end{cases} \tag{2}$$

$$\Delta_g = \begin{cases} z_g\,, & \text{if } g(t) < 40 \\ -z_g\,, & \text{if } g(t) > 60 \\ u_g\,, & \text{otherwise} \end{cases} \tag{3}$$

$$\Delta_s = u_s\,, \tag{4}$$

where $z_v, z_g \sim \mathcal{N}(0.5, \frac{1}{\sqrt{3}})$ and $u_v, u_g, u_s \sim \mathcal{U}(-1, 1)$. The velocity $v(t)$ and gain $g(t)$ can take values in $[0, 100]$. Therefore, the data collection policy will try to keep these values only in the medium range given by the interval $[40, 60]$. Because of this, large parts of the state space will be unobserved. After collecting the data, the $30,000$ state transitions are used to train a BNN with latent variables with the same hyperparameters as in Depeweg et al. (2016). Finally, we train different policies using the Monte Carlo approximation and we set the horizon of $T = 100$ steps, with $M = 50$ and $N = 25$ and a minibatch size of 1 for 750 epochs. The total training time on a single CPU is around 18 hours.

### 3.3.2 Wind Turbine Simulator

In this problem we observe the turbine state $s(t)$, with features such as current wind speed and currently produced power. Our actions $a(t)$ adjust the turbine's behavior, with known upper and lower bounds. The goal is to maximize energy output over a $T$-step horizon.

We are given a batch of around 5,000 state transitions generated by a behavior policy $\pi_b$. The policy does limited exploration around the neutral action $a(t) = 0$.

The system is expected to be highly stochastic due to the unpredictability of future wind dynamics. Furthermore the dimensionality of state observation is much higher than the action dimensionality, so, with the limited dataset that we have, we expect it to be very challenging to accurately learn the influence of the action on the reward.

First we train a BNN with two hidden layer and 50 hidden units per layer on the available batch using $\alpha$-divergence minimization with $\alpha = 1.0$. In the second step, using the model, we train a policy with 20 hidden units on each of the two layers in the usual way, using the Monte Carlo estimate. The total training time on a single CPU is around 8 hours.

## References

Depeweg, S., Hernández-Lobato, J. M., Doshi-Velez, F., and Udluft, S. Learning and policy search in stochastic dynamical systems with bayesian neural networks. *International Conference on Learning Representations (ICLR)*, 2016.