Bayesian Deep Learning and Gumbel-Max Perturbation Models

Tamir Hazan
Technion

Yossi Adi, Nicolas Heess, Andreea Gane, Tommi Jaakkola, Guy Lorberbom, Chris Maddison, Alex Schwing, Danny Tarlow

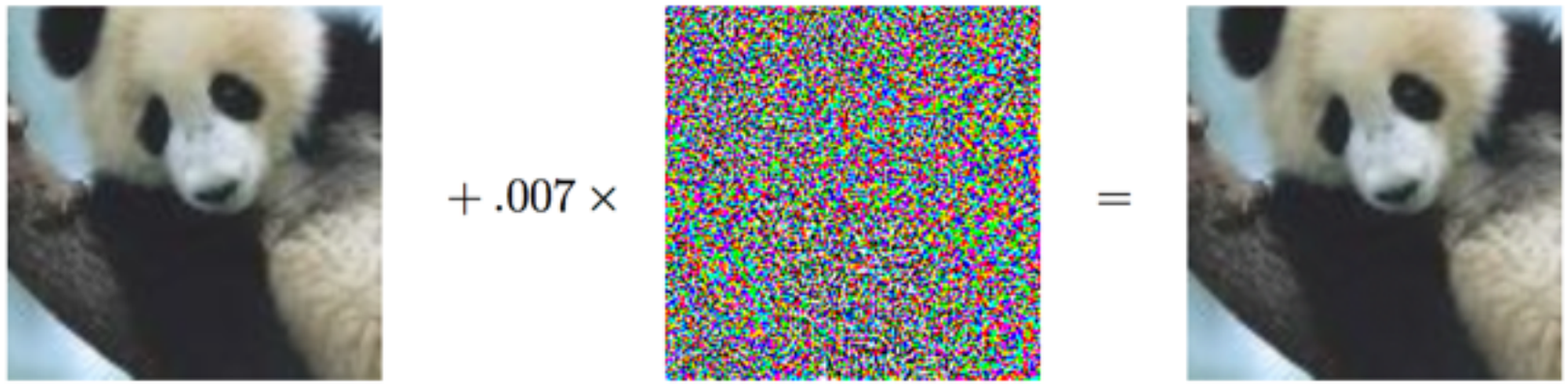
Why Bayesian deep learning

- AlexNet: 60M parameters
- VGG: 138M parameters
- GoogLeNet: 5M-23M parameters
- ResNet50: 25M parameters
- ResNet101: 44M parameters
- DenseNet190: 40M parameters

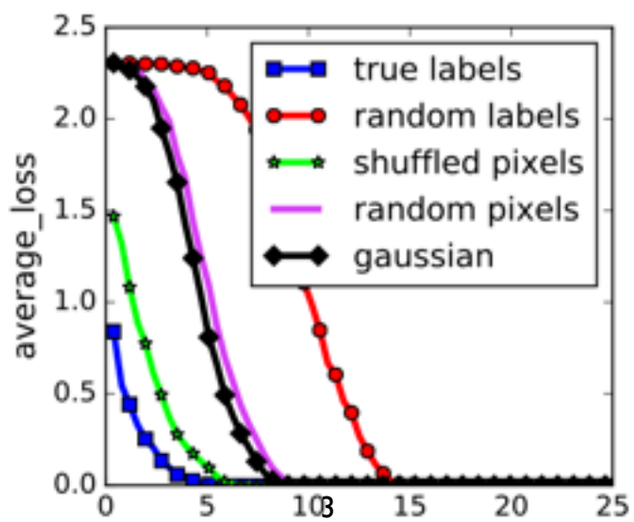
- Used to learn at most 1M data points.
- Overfitting
- Over confident predictions

Why Bayesian deep learning

- Over-confidence (Goodfellow et al. 2015)

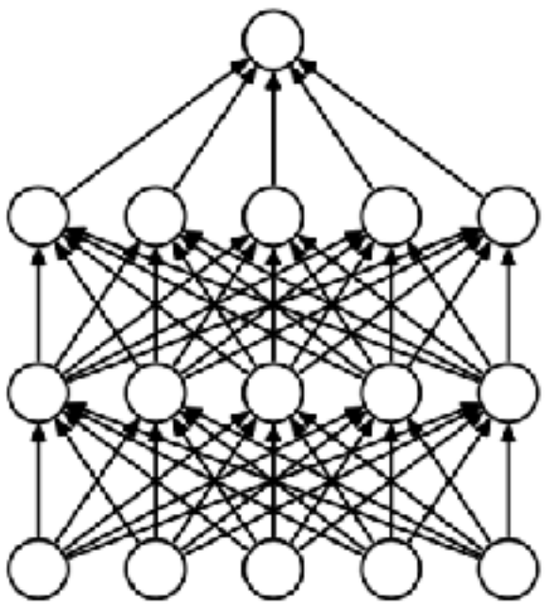


- “Deep nets easily fit random labels.” (Zhang et al. 2017)

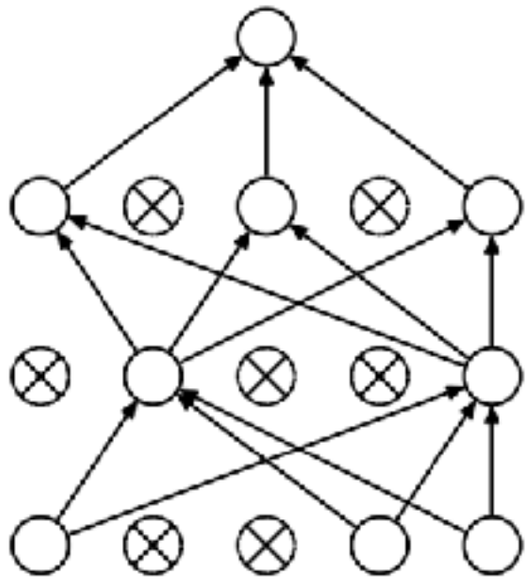


Why Bayesian deep learning

- To calibrate the predictions' confidence
- To avoid overfitting
- Dropout…

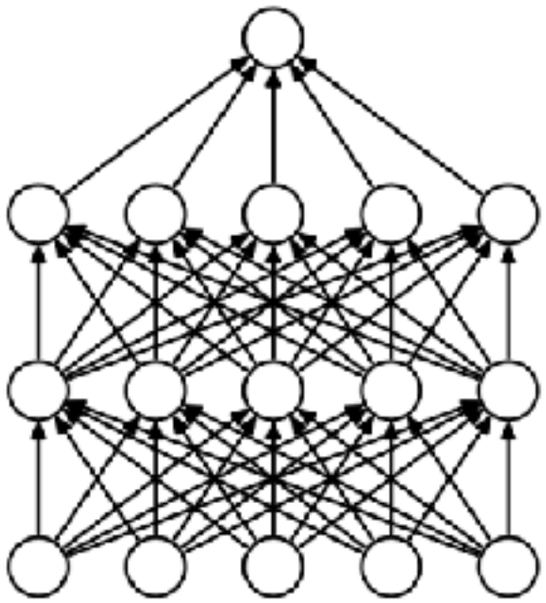


=



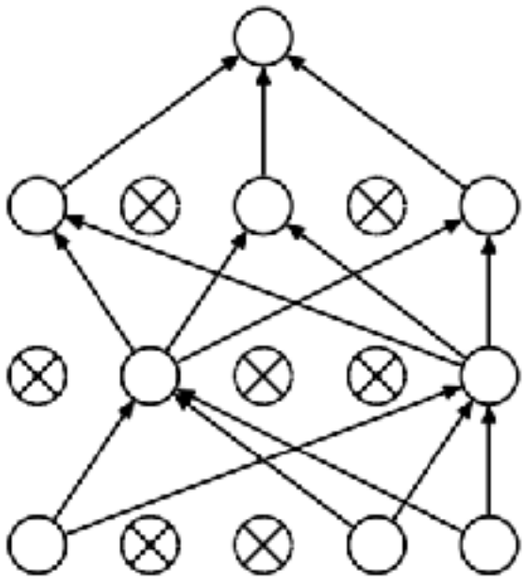
Dropout

- Dropout…



$$p(y|x, w)$$

w is determined by the training set S



=

$$p(y|x, S) = \int p(w|S)p(y|x, w, S)$$

posterior

$$\approx \int q(w|S)p(y|x, w, S)$$

approximated posterior

Dropout

- Bernoulli Dropout (Krishevsky et al. 2012)
  - multiplying a neuron with a Bernoulli random variable

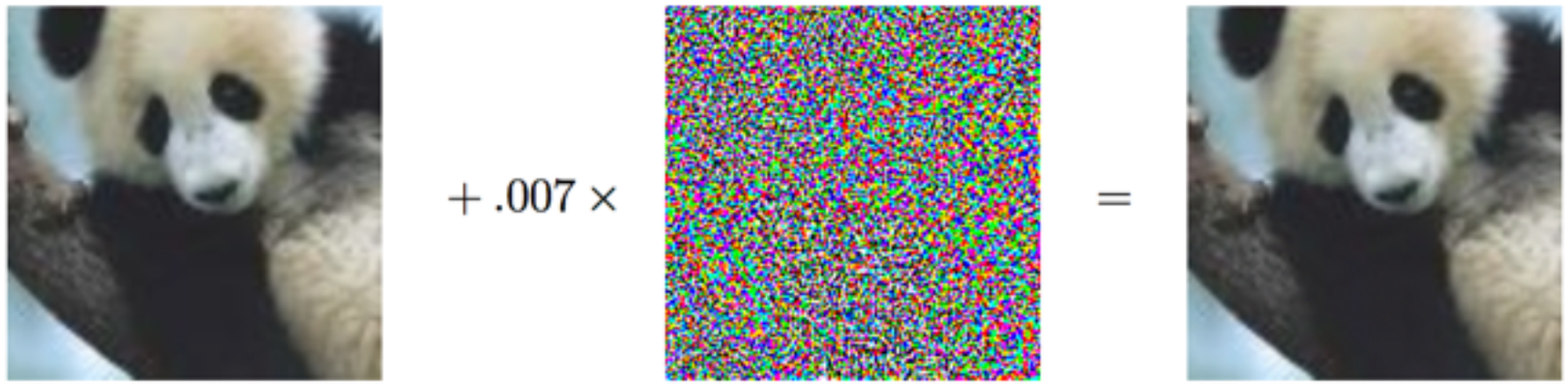
- Gaussian Dropout (Srivastava et al. 2014)
  - multiplying a neuron with a Gaussian random variable $\sim \mathcal{N}(1, 1)$

- Variational Dropout (Kingma et al. 2015, Blundell et al. 2015)
  - Learn by a Gaussian perturbation of the weights $\sim \mathcal{N}(w, \sigma)$

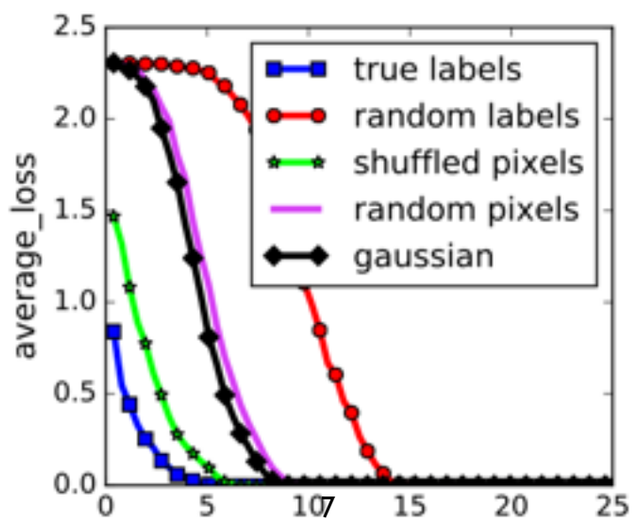
- MC Dropout (Gal and Ghahramani 2015)

Why Bayesian deep learning

- Over-confidence (Goodfellow et al. 2015)



- “Deep nets easily fit random labels.” (Zhang et al. 2017)



Approximated posterior

- Bayesian perspective $p(y|x, S) \approx \int q(w|S)p(y|x, w, S)dw$

q - approximated posterior

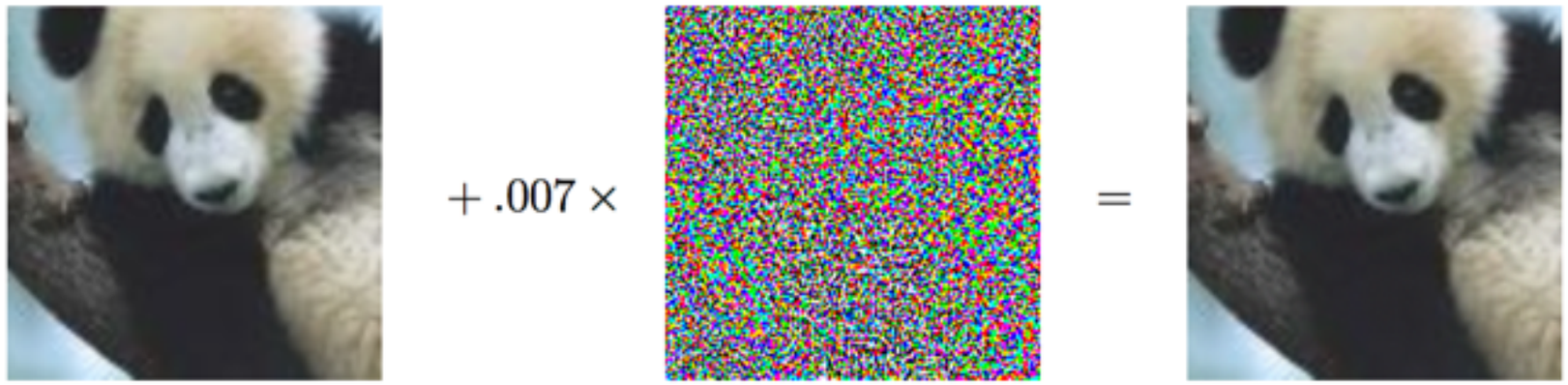
Model	In-dis.	Out-dis.	Train loss	Train acc.	Test loss	Test acc.	Train ent.	Test ent.
Softmax (0.1)	CIFAR-10	SVHN	3.78	0.123	3.86	0.115	1.51	1.54
Softmax (0.3)	CIFAR-10	SVHN	3.55	0.112	3.60	0.106	1.51	1.52
MC (0.1)	CIFAR-10	SVHN	3.79	0.123	3.87	0.115	1.50	1.53
MC (0.3)	CIFAR-10	SVHN	3.55	0.112	3.60	0.105	1.51	1.52
VARDO (0.1)	CIFAR-10	SVHN	2.47	0.112	2.49	0.099	**2.15**	**2.16**
VARDO (0.3)	CIFAR-10	SVHN	3.45	0.091	3.54	0.094	1.71	1.75

softmax: high loss & low entropy = over confidence

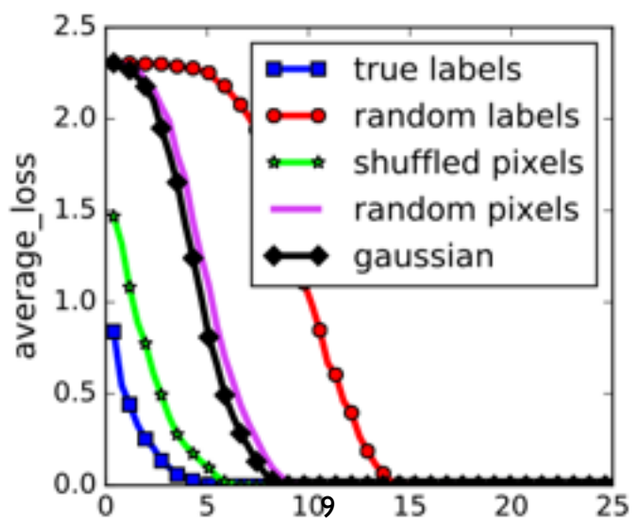
variational dropout : low loss & high entropy = calibrated confidence

Why Bayesian deep learning

- Over-confidence (Goodfellow et al. 2015)



- “Deep nets easily fit random labels.” (Zhang et al. 2017)



PAC-Bayesian bounds

generalization train error test error

$$G_S(w) = \frac{1}{m}\sum_{i=1}^{m} p(y_i|x_i, w) - \mathbb{E}_{(x,y)\sim D} \log p(y|x, w)$$

- Bayesian perspective $p(y|x, S) \approx \int q(w|S)p(y|x, w, S)$

$$\mathbb{E}_{w\sim q}[G_S(w)] \approx \log p(y|x, S) - \frac{1}{m}\sum_{i=1}^{m} \log p(y_i|x_i, S)$$

q - approximated posterior

PAC-Bayesian bounds

- PAC-Bayesian generalization bound (Donsker 1975, Shawe-Taylor and Williamson 1997, McAllester 1999)

$$\mathbb{E}_{w\sim q}[G_S(w)] \leq \frac{\log \mathbb{E}_{w\sim p, S\sim D^m}[e^{\lambda G_S(w)}] + KL(q||q_0)}{\lambda}$$

q - approximated posterior

q_0 - prior

- Proof sketch:

$$\mathbb{E}_{w\sim q_0}[e^{G_S(w)}] = \mathbb{E}_{w\sim q}\left[\frac{q_0(w)}{q(w)} e^{G_S(w)}\right] = \mathbb{E}_{w\sim q}\left[e^{G_S(w) - \log \frac{q(w)}{q_0(w)}}\right]$$

$$\mathbb{E}_{w\sim q}\left[e^{G_S(w) - \log \frac{q(w)}{q_0(w)}}\right] \overset{convexity}{\geq} e^{\mathbb{E}_{w\sim q}[G_S(w)] - KL(q||q_0)}$$

and taking the logarithm of both sides…

PAC-Bayesian bounds

- PAC-Bayesian generalization bound (Donsker 1975, Shawe-Taylor and Williamson 1997, McAllester 1999)

$$\mathbb{E}_{w\sim q}[G_S(w)] \leq \frac{\log \mathbb{E}_{w\sim p, S\sim D^m}[e^{\lambda G_S(w)}] + KL(q||q_0)}{\lambda}$$

q - approximated posterior

q_0 - prior

- "Deep nets easily fit random labels." ?!?

Model	Train loss	Train acc.	Test loss	Test acc.	KL
			MNIST		
VARDO w/o KL	0.04	0.986	13.77	0.11	54447932
VARDO w KL	2.32	0.112	2.31	0.12	5558
			Fashion-MNIST		
VARDO w/o KL	0.01	0.997	12.97	0.09	28423410
VARDO w KL	2.31	0.124	2.31	0.13	261

Dziugaite & Roy 2017, Molchanov et al. 2017

PAC-Bayesian bounds

learner complexity

learning complexity

$$\mathbb{E}_{w\sim q}[G_S(w)] \leq \frac{\log \mathbb{E}_{w\sim p, S\sim D^m}[e^{\lambda G_S(w)}] + KL(q||q_0)}{\lambda}$$

q - approximated posterior

q_0 - prior

- Learning complexity determined by the training data
- Learner complexity determined by the architecture

PAC-Bayesian bounds

learner complexity

learning complexity

$$\mathbb{E}_{w\sim q}[G_S(w)] \leq \frac{\log \mathbb{E}_{w\sim p, S\sim D^m}[e^{\lambda G_S(w)}] + KL(q||q_0)}{\lambda}$$

q - approximated posterior

q_0 - prior

- Learner complexity is determined by gradients flow

$$\mathbb{E}_{w\sim q_0, S\sim D^m}\left[e^{\lambda G_S(w)}\right] \leq$$

$$\mathbb{E}_{w\sim q_0} e^{\frac{\lambda^2}{m}\mathbb{E}_{(x,y)\sim D}\left[\|\nabla_x \log p(y|x;w)\|^2 \int_0^1 \frac{e^{\alpha \log p(y|x;w)}}{M(\alpha)} d\alpha\right]}$$

PAC-Bayesian bounds

- Learner complexity is determined by gradients flow

$$\mathbb{E}_{w\sim q_0, S\sim D^m}\left[e^{\lambda G_S(w)}\right] \leq$$

$$\mathbb{E}_{w\sim q_0} e^{\frac{\lambda^2}{m}\mathbb{E}_{(x,y)\sim D}\left[\|\nabla_x \log p(y|x;w)\|^2 \int_0^1 \frac{e^{\alpha \log p(y|x;w)}}{M(\alpha)} d\alpha\right]}$$

- Proof sketch:

$$M(\lambda) \stackrel{def}{=} \mathbb{E}_{w\sim q_0}[e^{\lambda \log p(y|x,w)}] \qquad H(\lambda) \stackrel{def}{=} \frac{1}{\lambda}\log M(\lambda)$$

$$H(\lambda) - H(0) = \int_0^\lambda H'(\hat{\alpha})d\hat{\alpha} \stackrel{\hat{\alpha}=\lambda\alpha}{=} \lambda \int_0^1 H'(\alpha)d\alpha$$

$$H'(\alpha) \stackrel{log-sobolev}{\leq} \frac{\mathbb{E}_{w\sim q_0}[e^{\alpha \log p(y|x,w)}\|\nabla \log p(y|x,w)\|^2]}{M(\alpha)}$$

PAC-Bayesian bounds

- shallow vs. deep

$$\mathbb{E}_{w\sim q_0} e^{\frac{\lambda^2}{m}\mathbb{E}_{(x,y)\sim D}\left[\|\nabla_x \log p(y|x;w)\|^2 \int_0^1 \frac{e^{\alpha \log p(y|x;w)}}{M(\alpha)} d\alpha\right]}$$

	MNIST	CIFAR10
Linear	66.23	2983.87
2 layers	1.48	1.00
3 layers	1.02	1.00
4 layers	1.00	1.00
5 layers	1.00	1.00

Bayesian deep learning

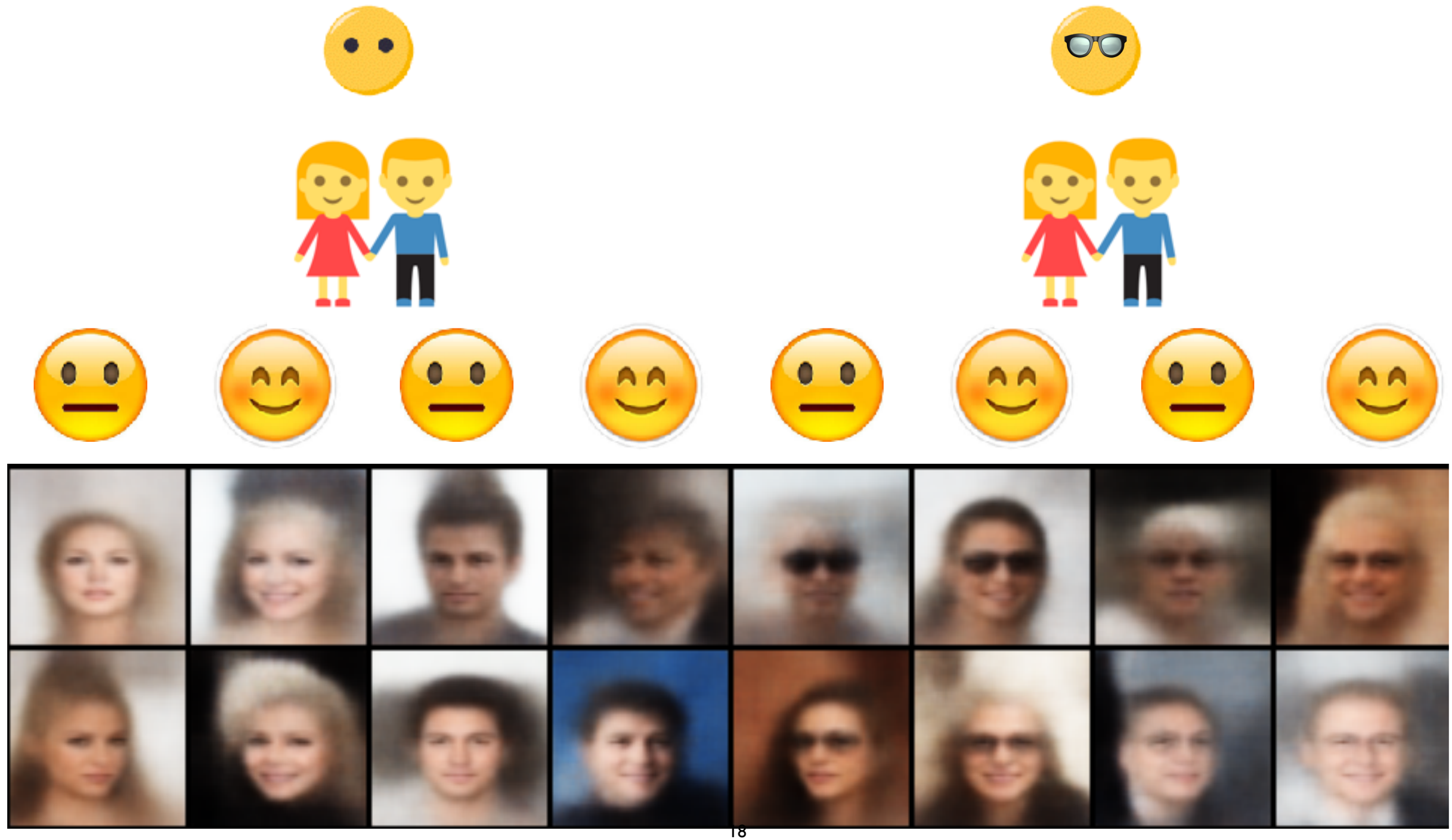
- Discriminative learning
  - approximated posterior: confidence measures
  - PAC-Bayesian bound: learner & learning complexity

- Generative learning
  - discrete variational auto-encoders
  - Gumbel-max perturbation models
  - structured latent spaces
  - semi supervised setting

- Reinforcement learning
  - approximated posterior: confidence measures
  - PAC-Bayesian bound: learner & learning complexity

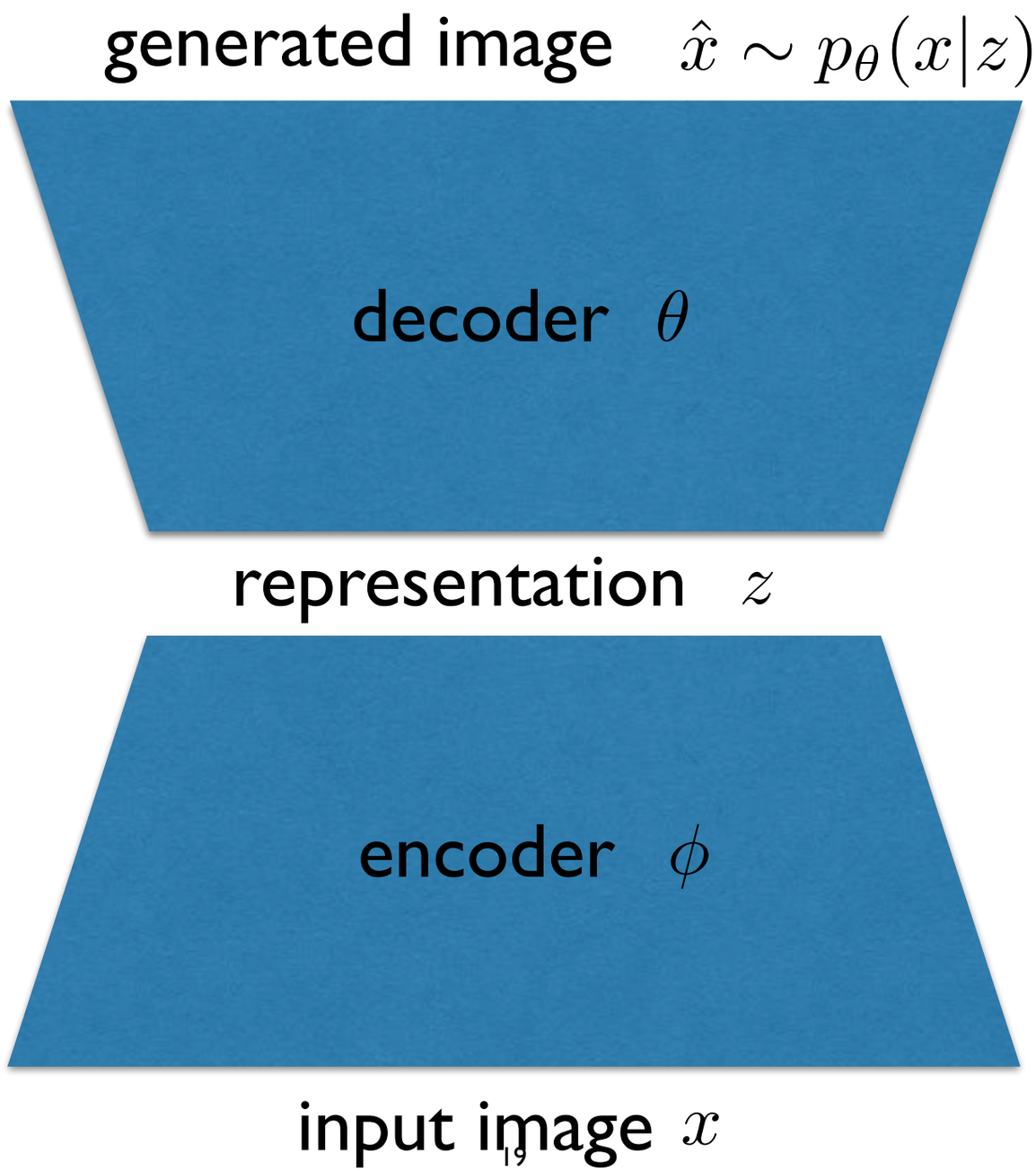
Generative Learning

- Discrete variational auto-encoders



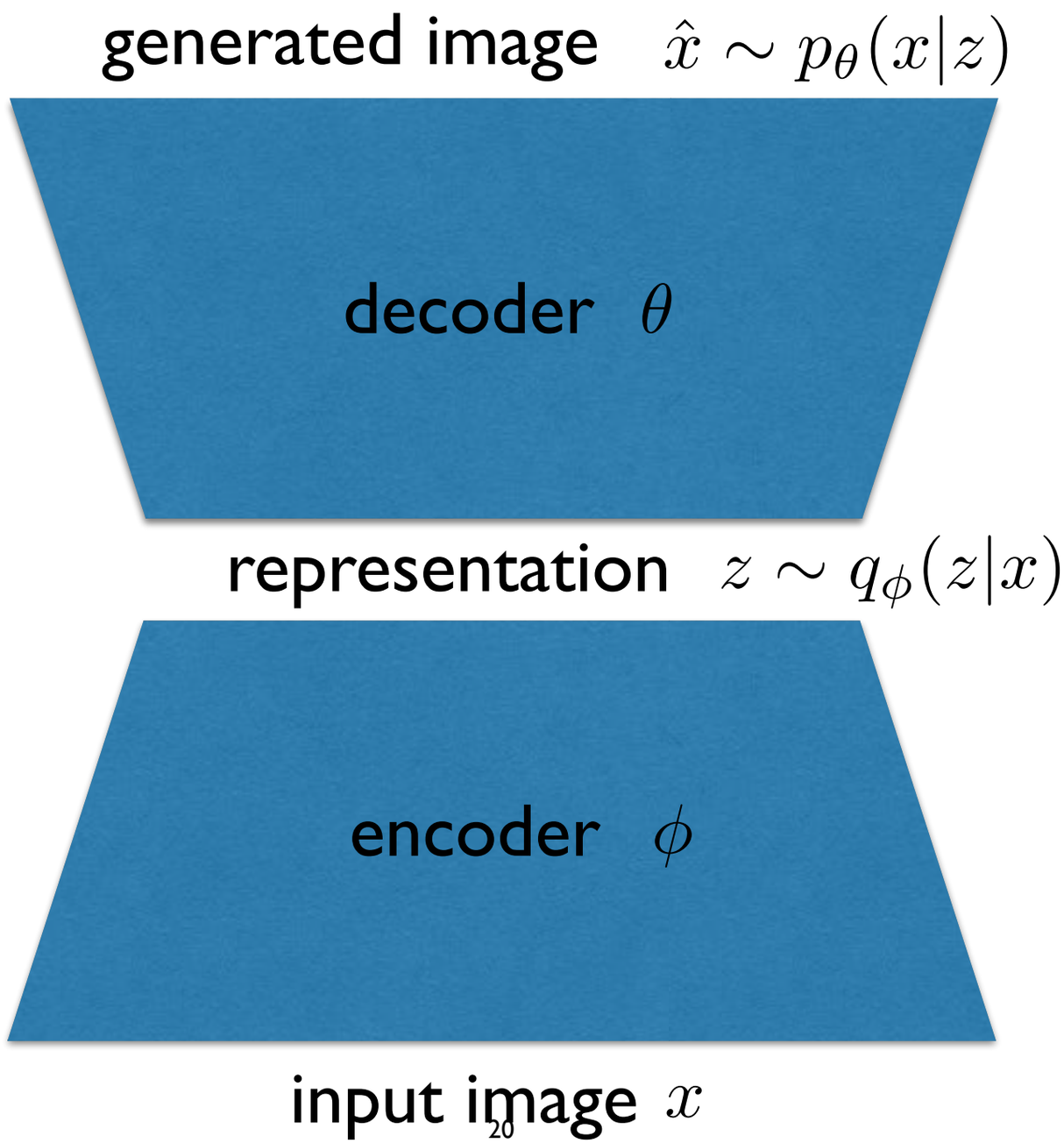
Variational Auto-Encoders

- Reznde et al. 2014, Kingma and Welling, 2014



Variational Auto-Encoders

$$\log \frac{1}{p_\theta(x)} \leq \mathbb{E}_{z \sim q_\phi} \log \frac{1}{p_\theta(x|z)} + KL(q_\phi(z|x)||p_\theta(z))$$



Variational Auto-Encoders

$$\log \frac{1}{p_\theta(x)} \leq \mathbb{E}_{z \sim q_\phi} \log \frac{1}{p_\theta(x|z)} + KL(q_\phi(z|x)||p_\theta(z))$$

Variational Auto-Encoders

$$\log \frac{1}{p_\theta(x)} \leq \mathbb{E}_{z \sim q_\phi} \log \frac{1}{p_\theta(x|z)} + KL(q_\phi(z|x)||p_\theta(z))$$

$$p_\theta(x|z) = e^{\theta(x,z)}$$

$$q_\phi(z|x) = e^{\phi(x,z)}$$

Variational Auto-Encoders

$$\log \frac{1}{p_\theta(x)} \leq \mathbb{E}_{z\sim q_\phi} \log \frac{1}{p_\theta(x|z)} + KL(q_\phi(z|x)||p_\theta(z))$$

$$p_\theta(x|z) = e^{\theta(x,z)}$$

$$q_\phi(z|x) = e^{\phi(x,z)}$$

- Learning using gradient descent

$$\nabla_\phi \mathbb{E}_{z\sim q_\phi} \log p_\theta(x|z) = \mathbb{E}_{z\sim q_\phi} \nabla\phi(x,z)\theta(x,z)$$

Variational Auto-Encoders

$$\log \frac{1}{p_\theta(x)} \leq \mathbb{E}_{z\sim q_\phi} \log \frac{1}{p_\theta(x|z)} + KL(q_\phi(z|x)||p_\theta(z))$$

$$p_\theta(x|z) = e^{\theta(x,z)}$$

$$q_\phi(z|x) = e^{\phi(x,z)}$$

- Learning using gradient descent

$$\nabla_\phi \mathbb{E}_{z\sim q_\phi} \log p_\theta(x|z) = \mathbb{E}_{z\sim q_\phi} \nabla\phi(x,z)\theta(x,z)$$

  - the gradient $\nabla\phi(x,z)$ is stretched by $\theta(x,z)$ - high variance
  - reducing the variance using Gumbel perturbations

Discrete variational auto encoders

$$q_\phi(z|x) = e^{\phi(x,z)}$$

- Theorem: (Fisher 1928, Gumbel 1953, McFadden 1973)

  Let $\gamma(z)$ be i.i.d. with Gumbel distribution with zero mean

$$G(t) \stackrel{def}{=} \mathbb{P}[\gamma(z) \leq t] = e^{-e^{-t+c}}$$

Discrete variational auto encoders

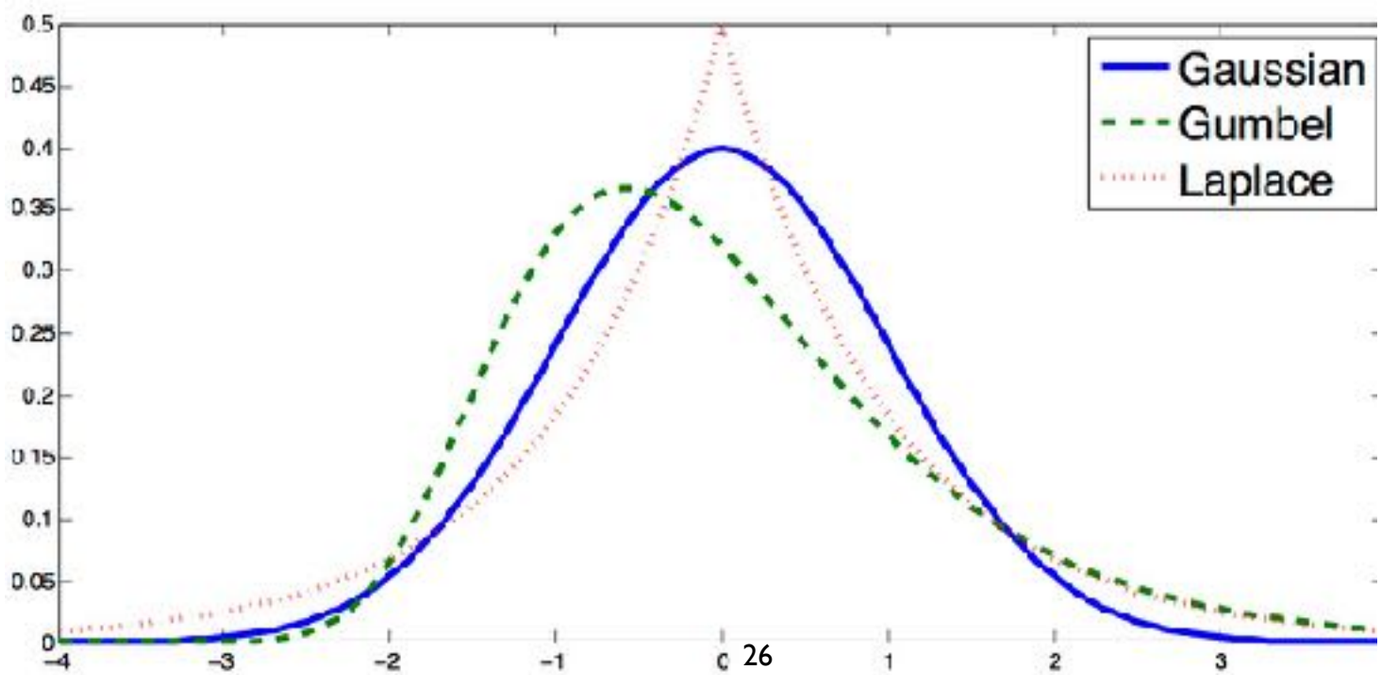
$$q_\phi(z|x) = e^{\phi(x,z)}$$

- Theorem: (Fisher 1928, Gumbel 1953, McFadden 1973)

  Let $\gamma(z)$ be i.i.d. with Gumbel distribution with zero mean

$$G(t) \stackrel{def}{=} \mathbb{P}[\gamma(z) \leq t] = e^{-e^{-t+c}}$$

$$g(t) = G'(t)$$



Discrete variational auto encoders

$$q_\phi(z|x) = e^{\phi(x,z)}$$

- Theorem: (Fisher 1928, Gumbel 1953, McFadden 1973)

  Let $\gamma(z)$ be i.i.d. with Gumbel distribution with zero mean

$$G(t) \stackrel{def}{=} \mathbb{P}[\gamma(z) \leq t] = e^{-e^{-t+c}}$$

then

$$e^{\phi(x,z)} = \mathbb{P}_{\gamma \sim g}[z^{\phi+\gamma} = z]$$

$$z^{\phi+\gamma} = \arg\max_{\hat{z}} \{\phi(x, \hat{z}) + \gamma(\hat{z})\}]$$

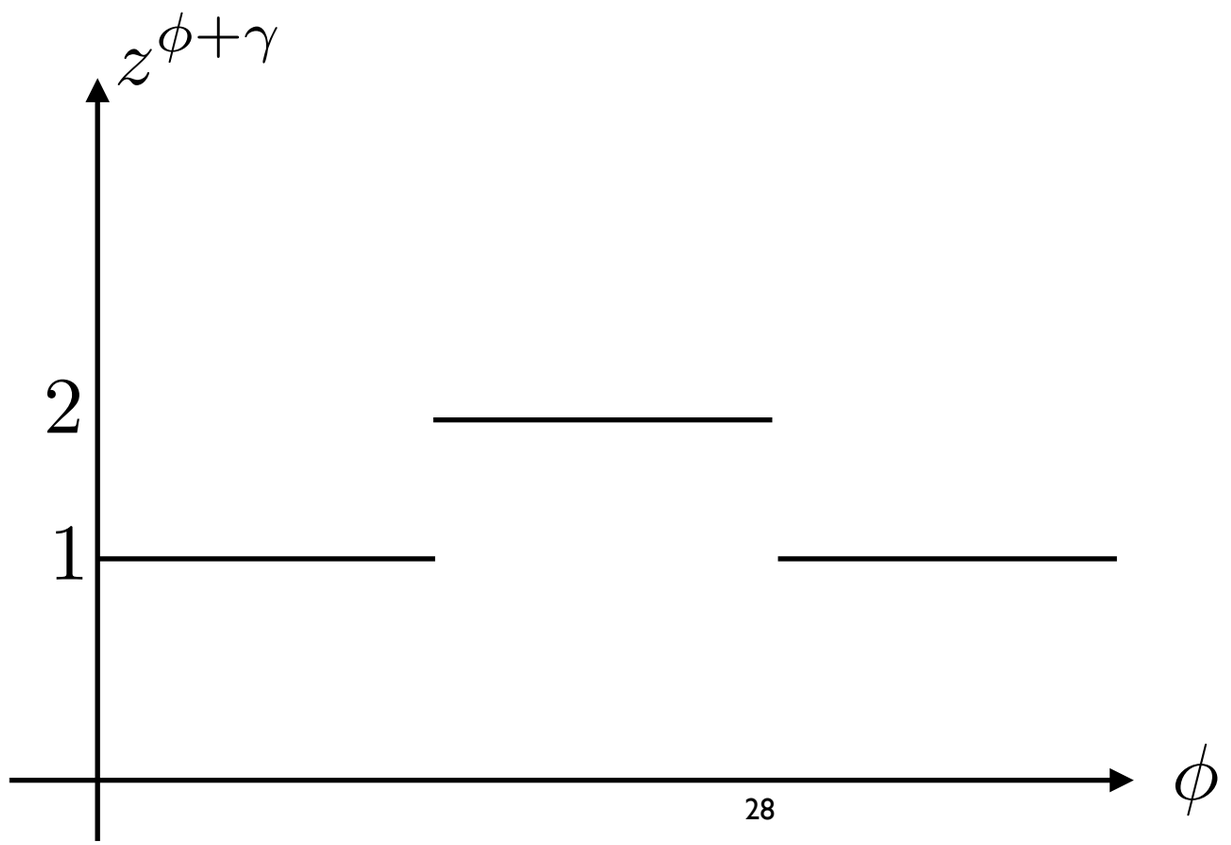
Discrete variational auto encoders

- Gumbel perturbation models

$$z^{\phi+\gamma} = \arg\max_{\hat{z}}\{\phi(x,\hat{z}) + \gamma(\hat{z})\}]$$

$$\mathbb{E}_{z\sim q_\phi} \log p_\theta(x|z) = \mathbb{E}_{\gamma\sim g}[\theta(x, z^{\phi+\gamma})]$$

- But argmax derivative is not informative



Discrete variational auto encoders

- Gumbel perturbation models

$$z^{\phi+\gamma} = \arg\max_{\hat{z}}\{\phi(x,\hat{z}) + \gamma(\hat{z})\}]$$

$$\mathbb{E}_{z\sim q_\phi} \log p_\theta(x|z) = \mathbb{E}_{\gamma\sim g}[\theta(x, z^{\phi+\gamma})]$$

- Gumbel-Softmax (Maddison et al. 2016, Jang et al. 2016)

$$\mathbb{P}_{\gamma\sim g}[z^{\phi+\gamma} = z] = \mathbb{E}_{\gamma\sim g}[1_{\mathrm{z}^{\phi+\gamma}=\mathrm{z}}] \approx \mathbb{E}_{\gamma\sim g} \frac{e^{\phi(x,z)+\gamma(z)}}{\sum_{\hat{z}} e^{\phi(x,\hat{z})+\gamma(\hat{z})}}$$

Direct optimization

- Theorem

$$\nabla_w \mathbb{E}_\gamma[\theta(x, z^{\phi+\gamma})] = \lim_{\epsilon \to 0} \frac{1}{\epsilon} \Big(\mathbb{E}_\gamma[\nabla_w \phi(x, z^{\epsilon\theta+\phi+\gamma}; w) - \nabla_w \phi(x, z^{\phi+\gamma}; w)] \Big)$$

Direct optimization

- Theorem

$$\nabla_w \mathbb{E}_\gamma[\theta(x, z^{\phi+\gamma})] = \lim_{\epsilon \to 0} \frac{1}{\epsilon}\Big(\mathbb{E}_\gamma[\nabla_w \phi(x, z^{\epsilon\theta+\phi+\gamma}; w) - \nabla_w \phi(x, z^{\phi+\gamma}; w)]\Big)$$

- Proof sketch

$G(w, \epsilon) = \mathbb{E}_\gamma[\max_{\hat{z}}\{\epsilon\theta(x, \hat{z}) + \phi(x, \hat{z}; w) + \gamma(\hat{z})\}]$ **is smooth**

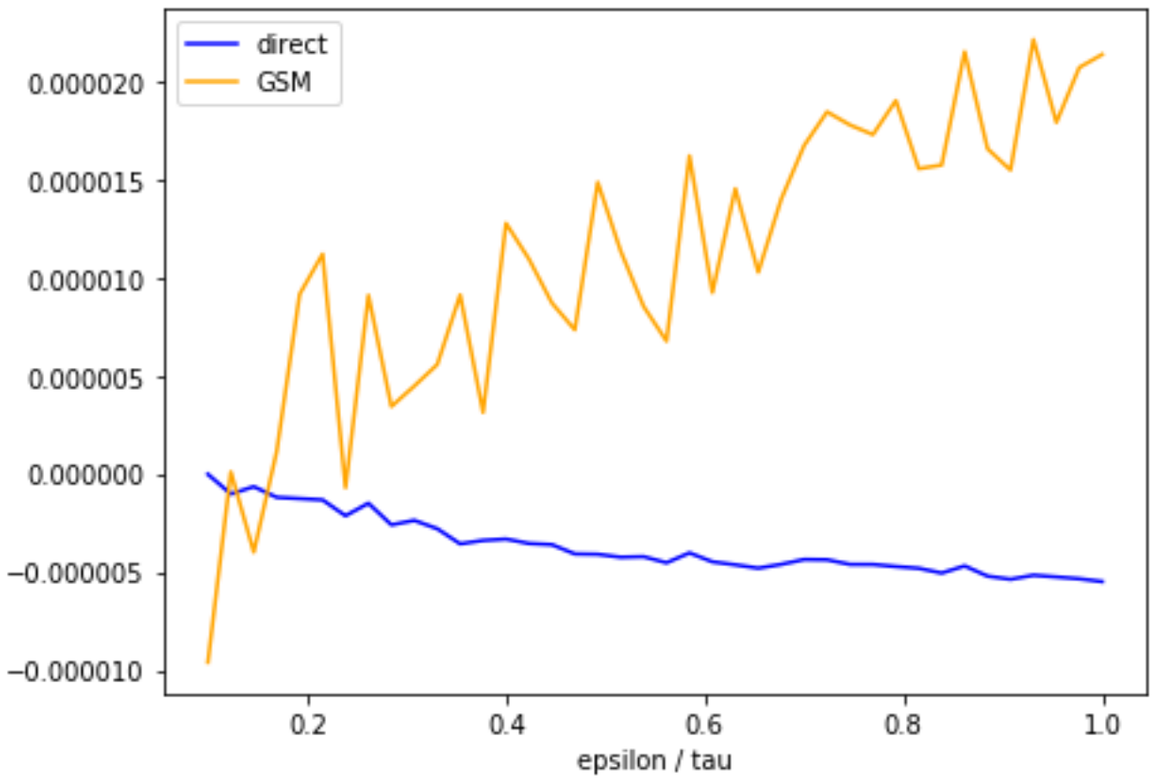
$$\partial_w \partial_\epsilon G(w, 0) = \nabla_w \mathbb{E}_\gamma[\theta(x, z^{\phi+\gamma})]$$

$$\partial_\epsilon \partial_w G(w, 0) = \lim_{\epsilon \to 0} \frac{1}{\epsilon}(E_\gamma[\nabla_w \phi(x, z^{\epsilon\theta+\phi+\gamma}; w) - \nabla_w \phi(x, z^{\phi+\gamma}; w)])$$

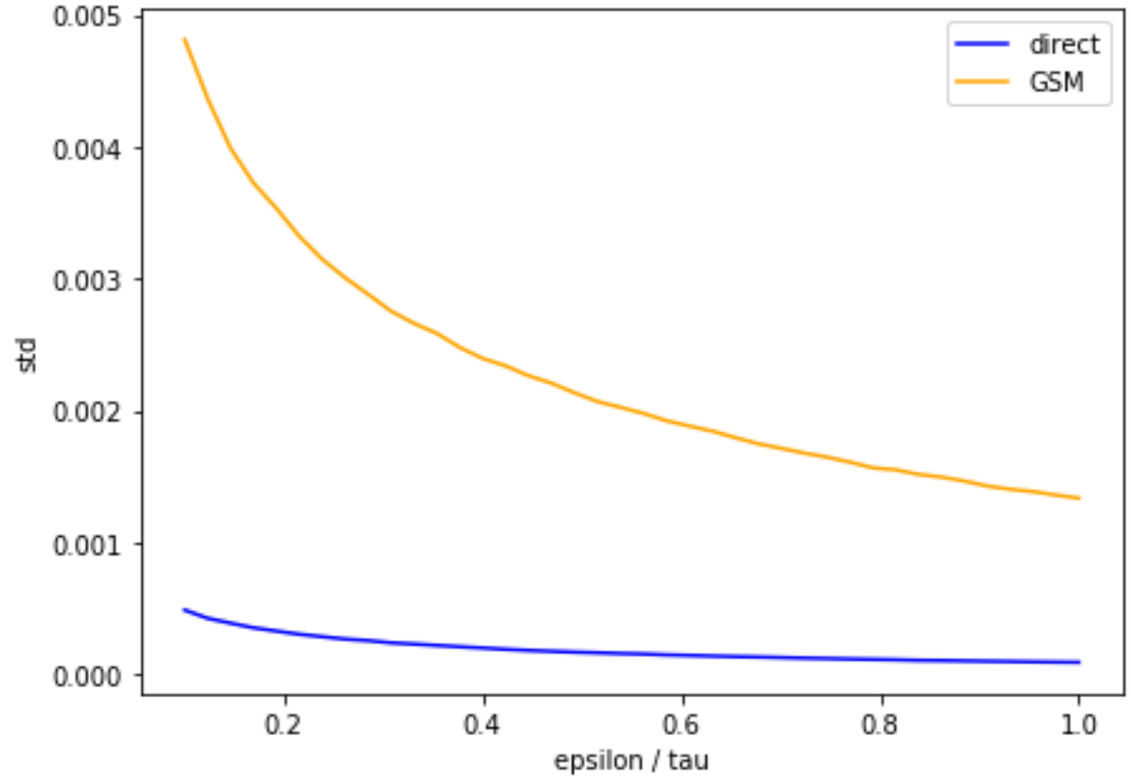
$$\partial_w \partial_\epsilon G(w, \epsilon) = \partial_\epsilon \partial_w G(w, \epsilon)$$

Bias/Variance tradeoff

bias



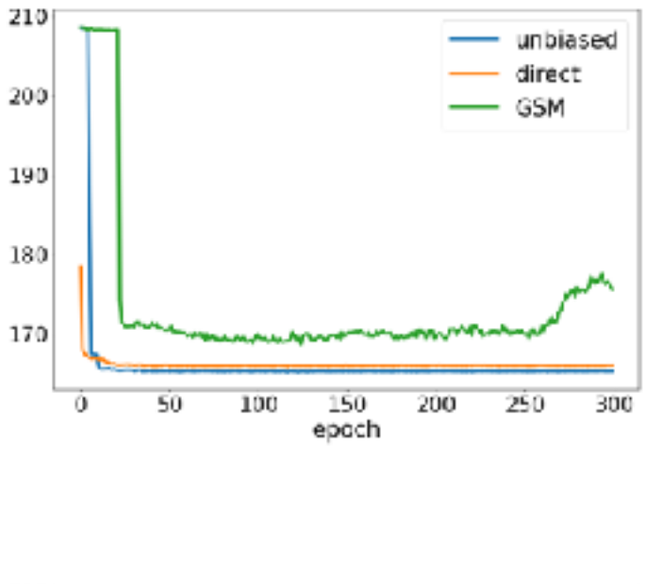
variance



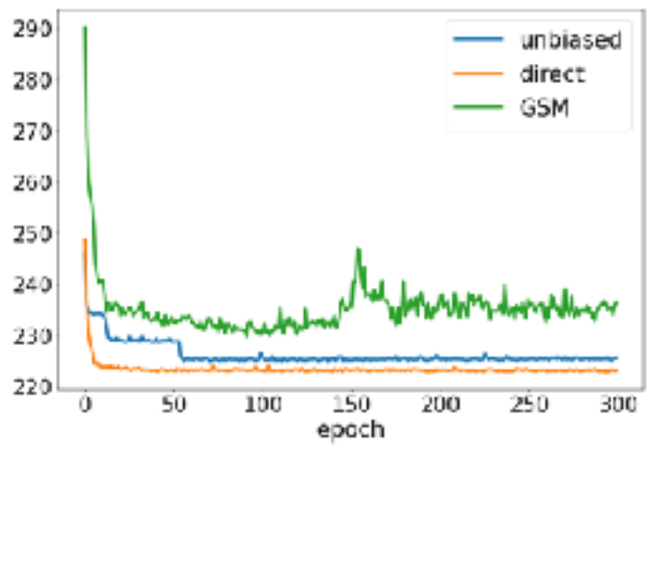
- Blue: Gumbel-Max perturbation model
- Yellow: Gumbel-Softmax approximation

Evaluating discrete VAEs

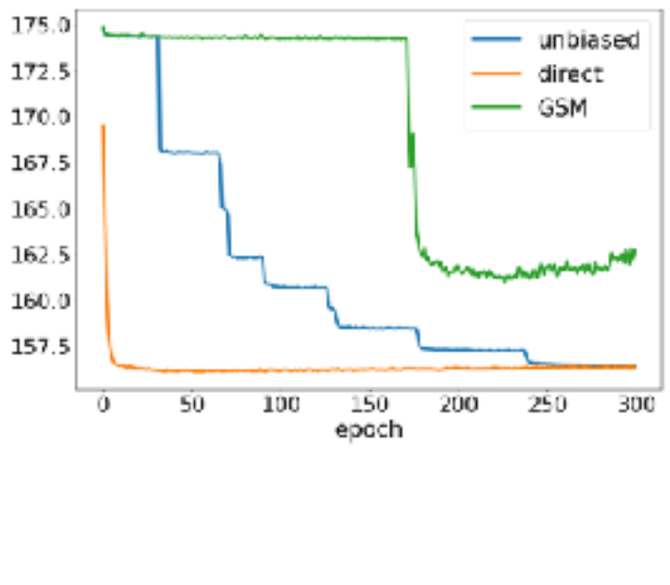
MNIST



Fashion MNIST

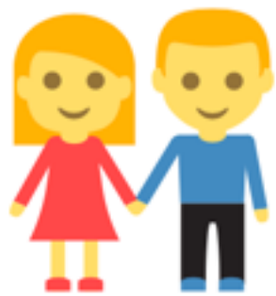


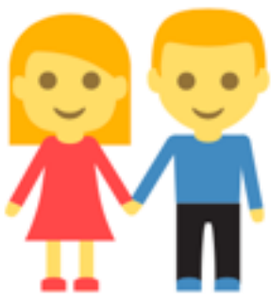
Omniglot

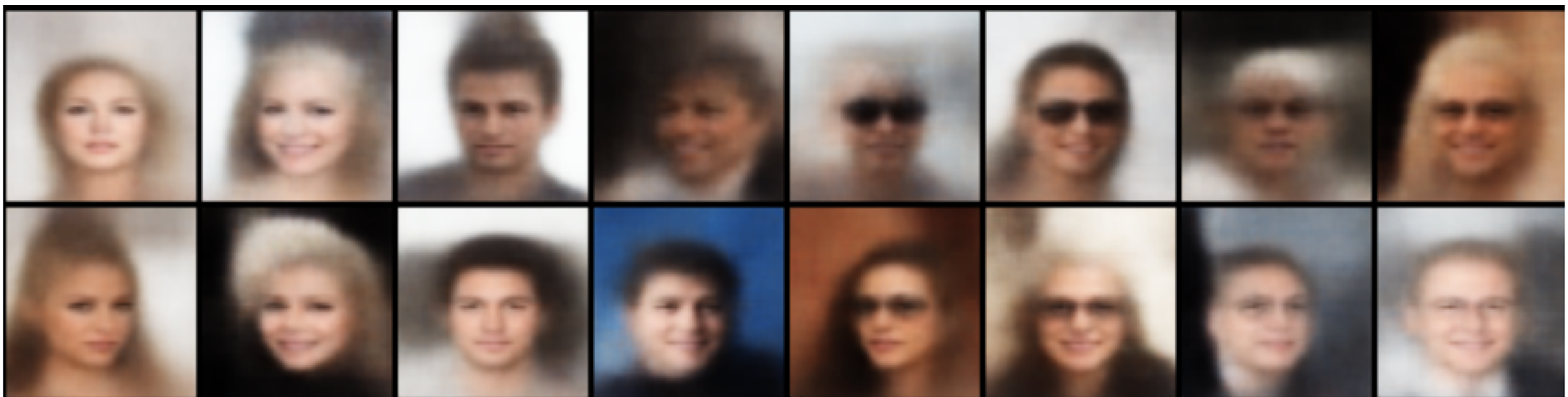


Structured VAEs

$$\phi(x, z) = \sum_i \phi_i(x) z_i + \sum_{i,j} \phi_{i,j}(x) z_i z_j$$

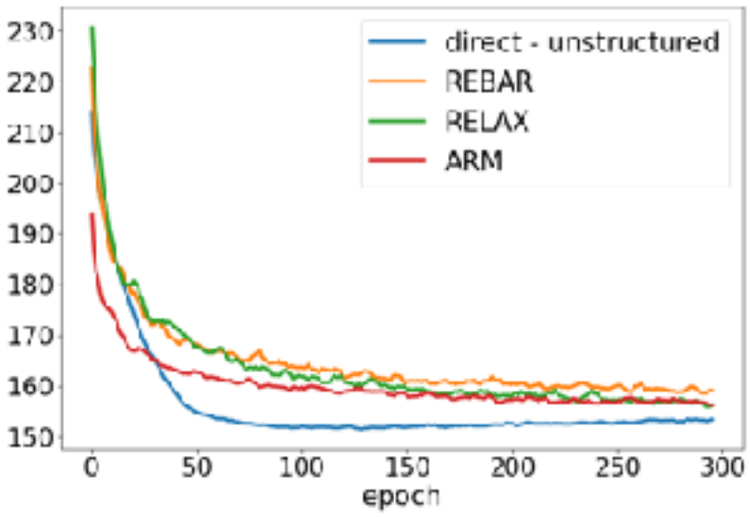


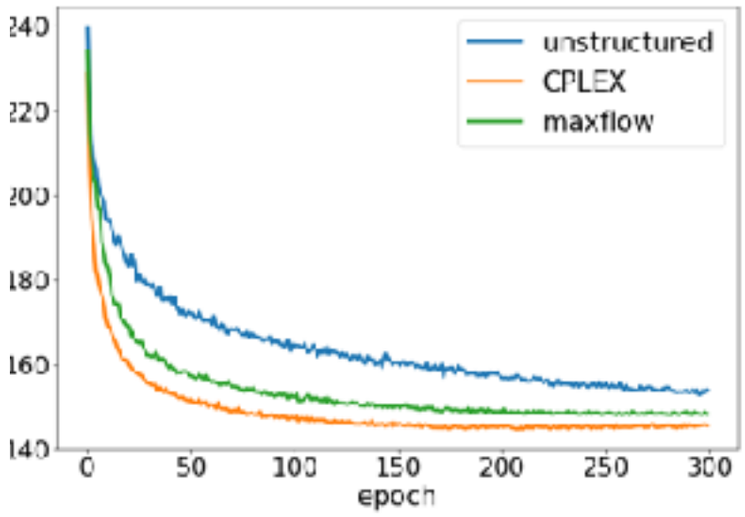


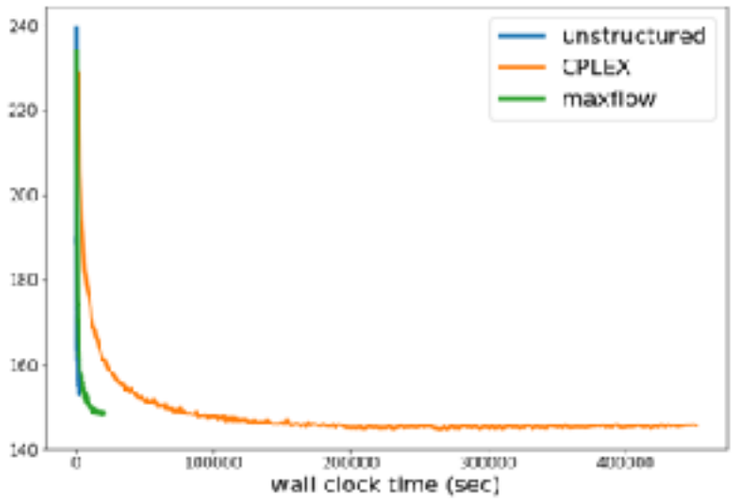


Structured VAEs

$$\phi(x, z) = \sum_i \phi_i(x) z_i + \sum_{i,j} \phi_{i,j}(x) z_i z_j$$

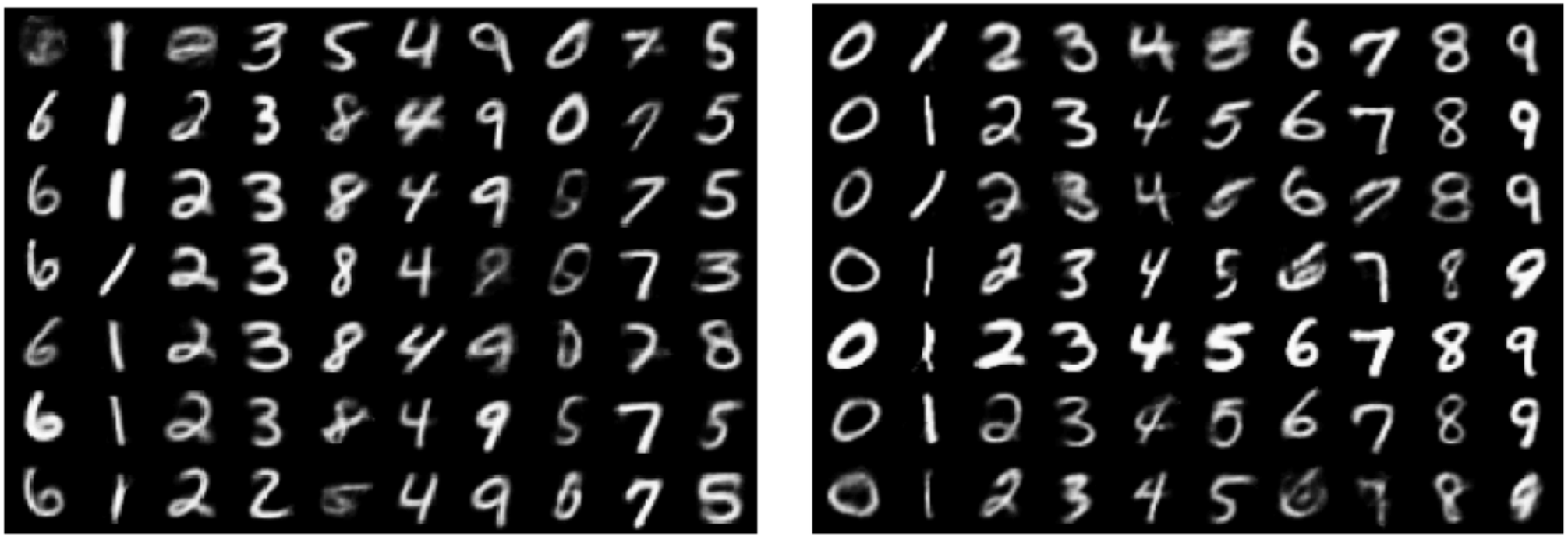






Semi-supervised VAEs

$$\sum_{x \in S} \mathbb{E}_\gamma[\theta(x, z^{\phi+\gamma})] + \sum_{(x,z) \in S_1} \mathbb{E}_\gamma[\ell(z, z^{\phi+\gamma})] + \sum_{x \in S} KL(q_\phi(z|x)||p_\theta(z))$$



unsupervised

semisupervised

	MNIST				Fashion-MNIST			
	accuracy		bound		accuracy		bound	
#labels	direct	GSM	direct	GSM	direct	GSM	direct	GSM
50	92.6%	84.7%	90.24	91.23	63.3%	61.2%	129.66	129.813
100	95.4%	88.4%	90.93	90.64	67.2%	64.2%	130.822	129.054
300	96.4%	91.7%	90.39	90.01	70.0%	69.3%	130.653	130.371
600	96.7%	92.3%	90.78	89.77	72.1%	71.6%	130.81	129.973
1200	96.8%	92.7%	90.45	90.37	73.7%	73.2%	130.921	130.063

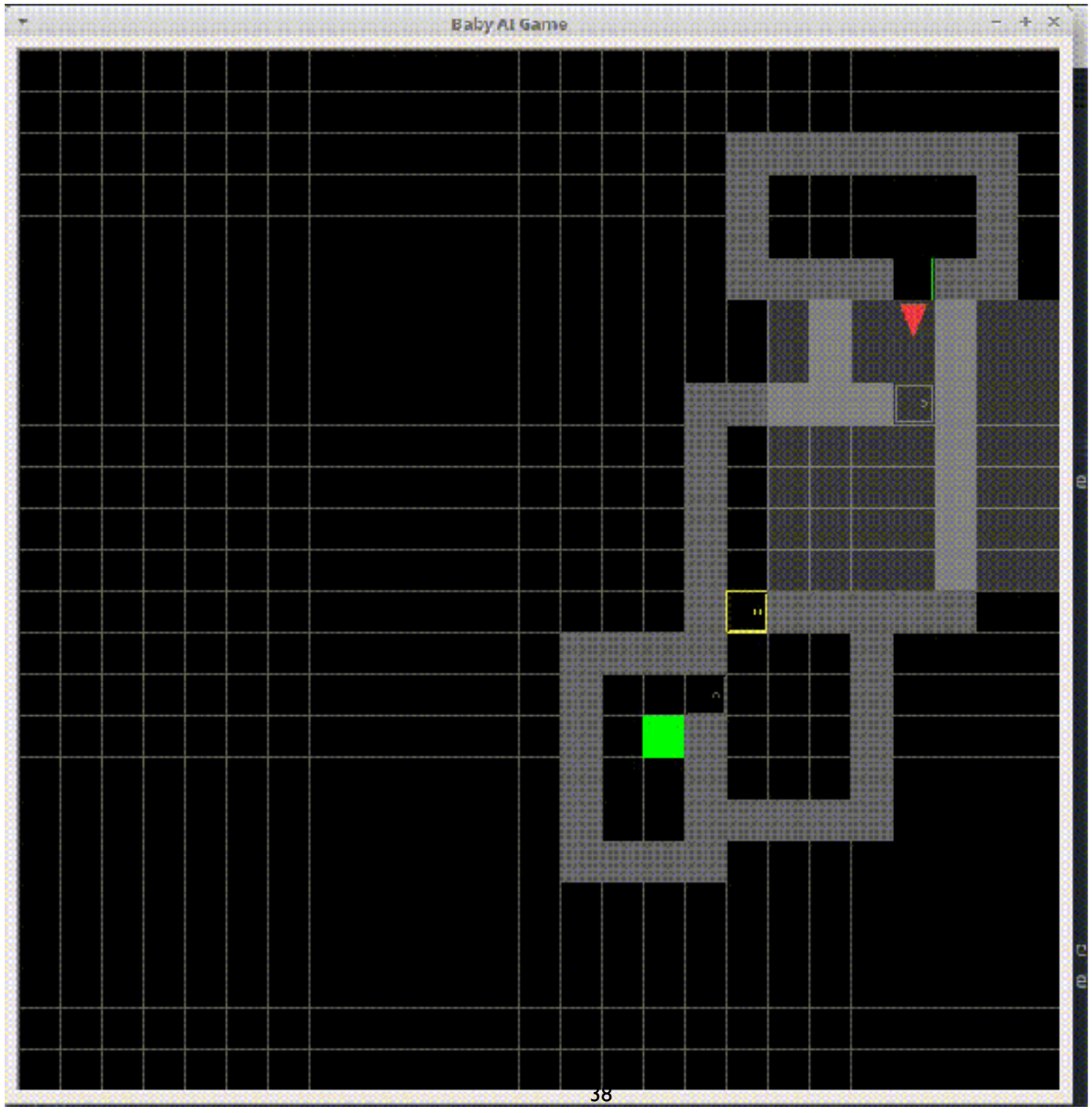
Bayesian deep learning

- Discriminative learning
  - approximated posterior: confidence measures
  - PAC-Bayesian bound: learner & learning complexity

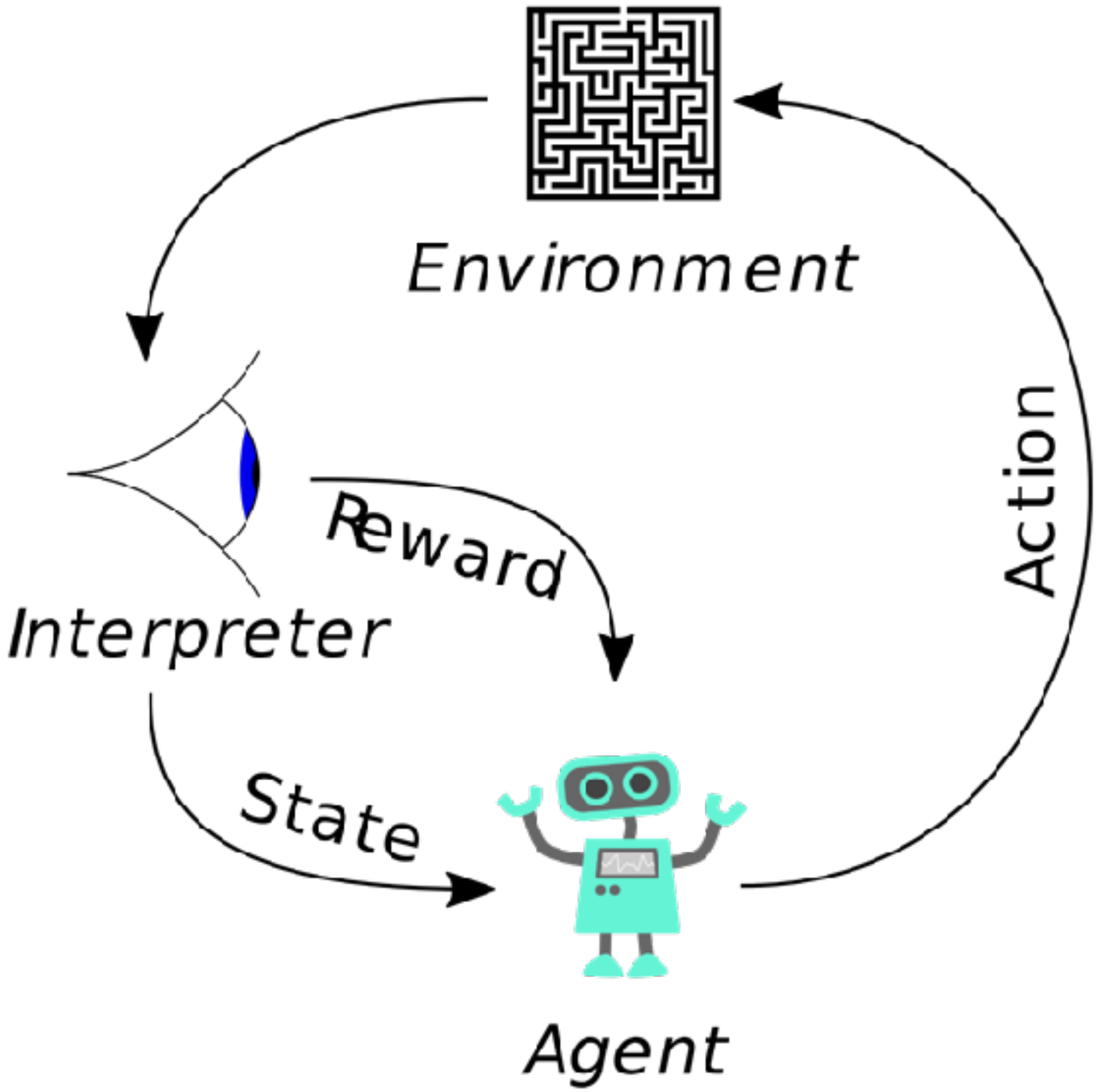
- Generative learning
  - discrete variational auto-encoders
  - Gumbel-max perturbation models
  - structured latent spaces
  - semi supervised setting

- Reinforcement learning
  - approximated posterior: confidence measures
  - PAC-Bayesian bound: learner & learning complexity

Reinforcement learning



Reinforcement learning



Reinforcement learning

$$a_t \sim \pi_\theta(\cdot \mid s_t) \text{ for } t \in \{0, \ldots, T-1\}$$

$$\Pi_\theta(a \mid S) = \prod_{t=0}^{T-1} \pi_\theta(a_t \mid s_{(a_0 \ldots a_{t-1})})$$

$$r_t, s_{t+1} \sim p(\cdot, \cdot \mid a_t, s_t) \text{ for } t \in \{0, \ldots, T-1\}$$

$$p_\theta(a, s, r) = \prod_{t=0}^{T-1} \pi_\theta(a_t \mid s_t) p(r_t, s_{t+1} \mid s_t, a_t)$$

$$\mathbb{E}_{a,s,r \sim p_\theta} \left[\sum_{t=0}^{T-1} r_t \right] = \mathbb{E}_{S \sim P, \Gamma} \left[R(a^{\log \Pi_\theta + \Gamma}, S) \right]$$

A* sampling

- Theorem: (Luce 1957, Maddison et al. 2014)

  Let $\gamma(z)$ be i.i.d. with Gumbel distribution with zero mean

$$G(t) \stackrel{def}{=} \mathbb{P}[\gamma(z) \leq t] = e^{-e^{-t+c}}$$

then

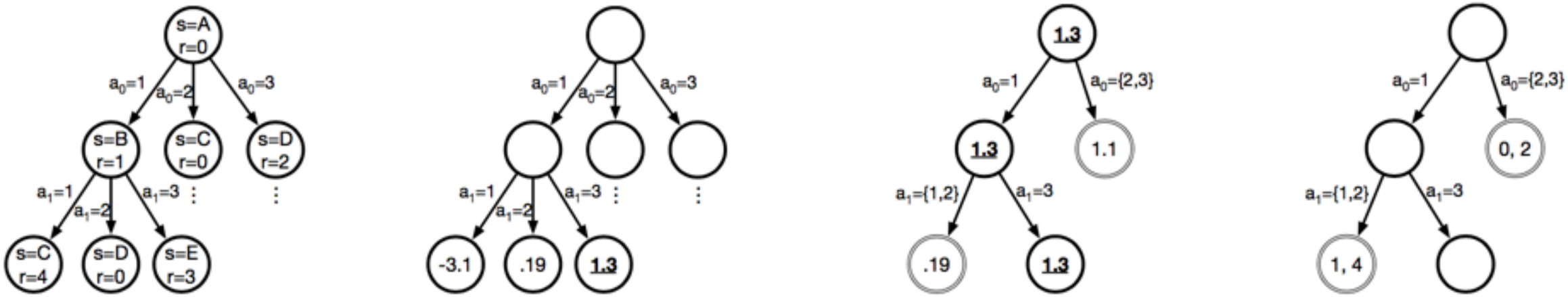
$$\max_{\hat{z}}\{\phi(x, \hat{z}) + \gamma(\hat{z})\}]$$

$$z^{\phi+\gamma} = \arg\max_{\hat{z}}\{\phi(x, \hat{z}) + \gamma(\hat{z})\}]$$

are conditionally independent

Direct optimization for policy gradient

- A* sample of trajectories



(a) State-reward tree (b) Gumbels for trajectories (c) Gumbels for regions (d) Returns for regions

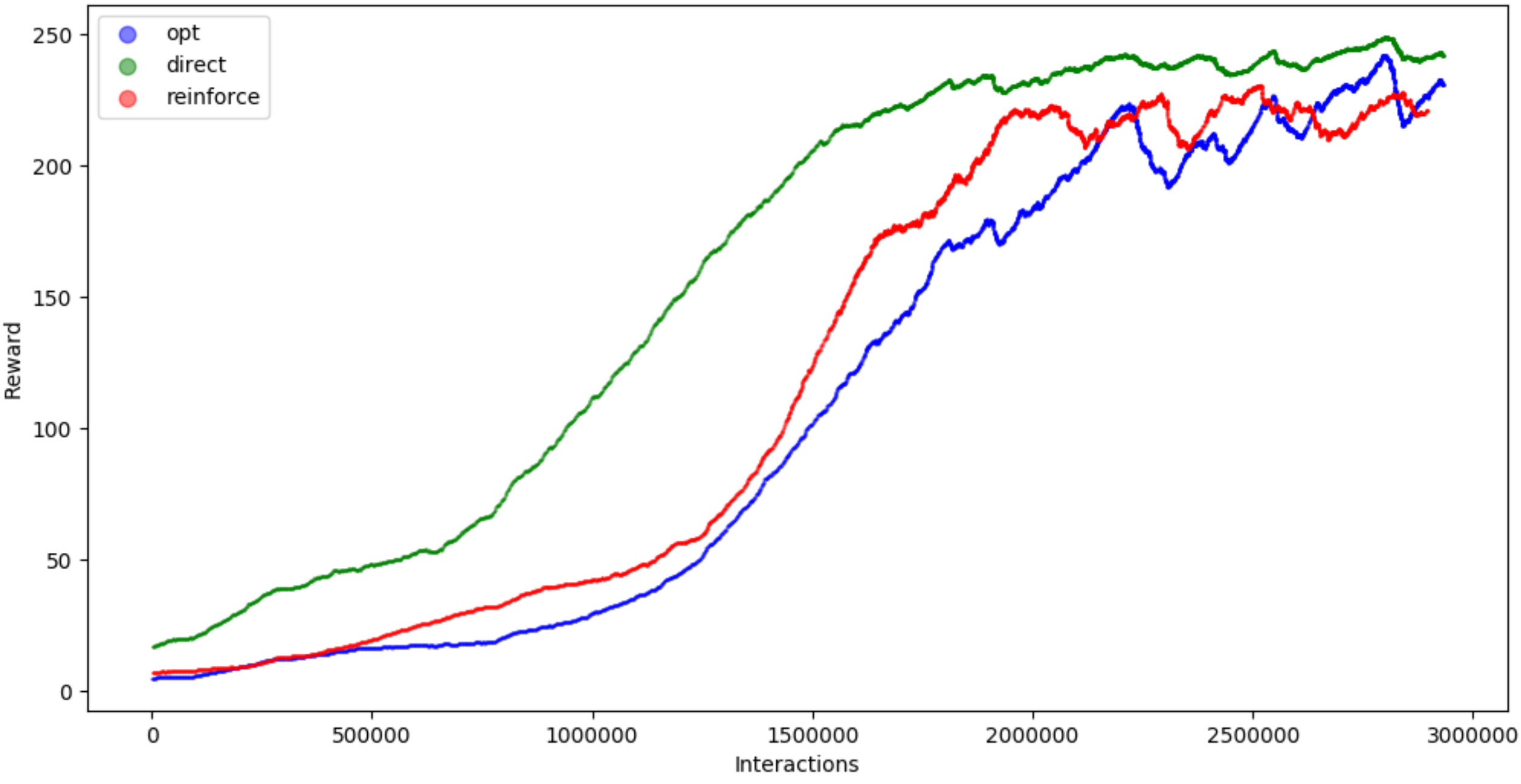
$$\nabla_\theta \mathbb{E}_{S\sim P,\Gamma}\left[R(a^{\log \Pi_\theta + \Gamma}, S)\right] =$$

$$\frac{1}{\epsilon}\mathbb{E}_{S\sim P,\Gamma}\left[\nabla_\theta \log \Pi_\theta(a^{\log \Pi_\theta + \Gamma + \epsilon R} \mid S) - \nabla_\theta \log \Pi_\theta(a^{\log \Pi_\theta + \Gamma} \mid S)\right].$$

- Prioritizing trajectories with high perturbed reward

Direct optimization for policy gradient

- Over the mini grid environment



Risk Sensitivity

- The gradient computation

$$\frac{1}{\epsilon}\mathbb{E}_{S\sim P,\Gamma}\left[\nabla_\theta \log \Pi_\theta(a^{\log \Pi_\theta+\Gamma+\epsilon R} \mid S) - \nabla_\theta \log \Pi_\theta(a^{\log \Pi_\theta+\Gamma} \mid S)\right].$$

is realized from the inference function

$$\hat{G}(\theta,\epsilon) = \mathbb{E}_{S\sim P}\left[\frac{1}{\epsilon}\log\left(\mathbb{E}_{a\sim\Pi_\theta(\cdot|S)}\left[\exp(\epsilon R(a,S))\right]\right)\right]$$

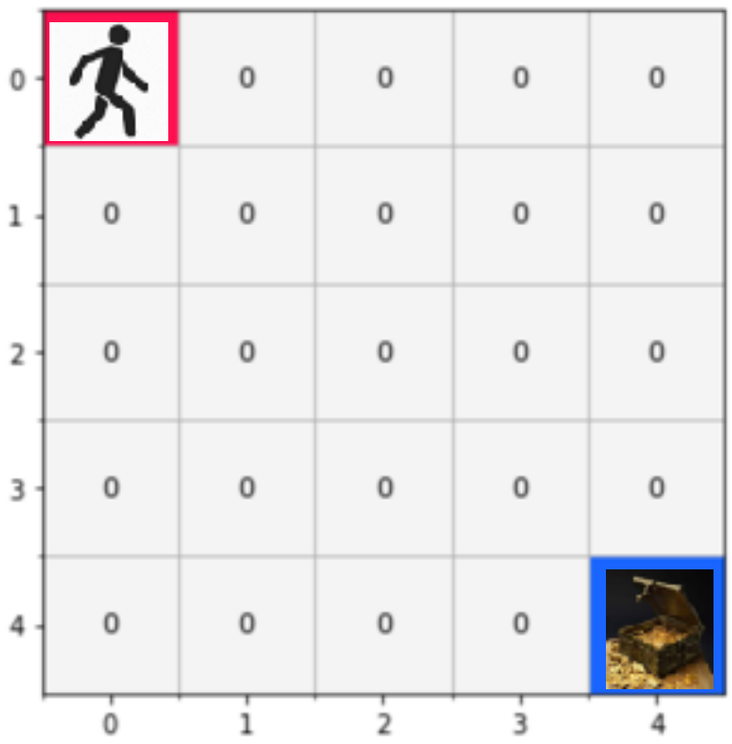
- This function encapsulates the risk/reward tradeoff:

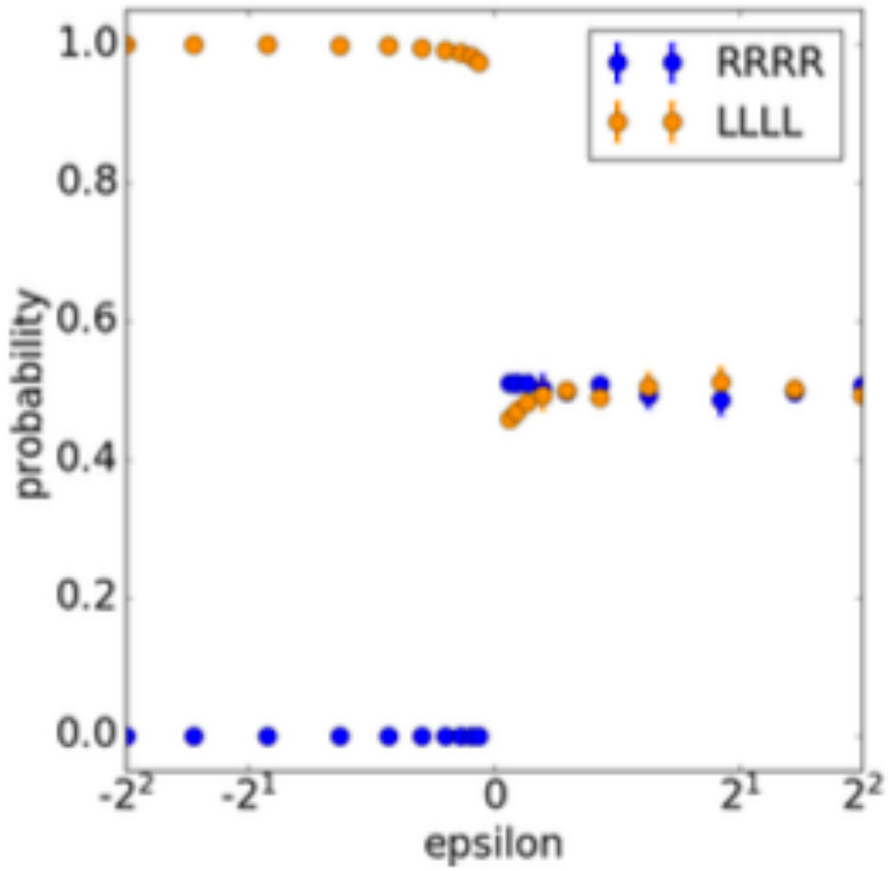
$$\mathbb{E}_{S\sim P,a\sim\Pi_\theta(\cdot|S)}[R(a,S)] + \frac{\epsilon}{2}\mathbb{E}_{S\sim P}[\mathrm{var}_{a\sim\Pi_\theta(\cdot|S)}(R(a,S))] + \mathcal{O}(\epsilon^2)$$

- positive ϵ — risk seeking, negative ϵ — risk avoiding

Risk Sensitivity

deep sea treasure
(Nguyen 2018)





Bayesian deep learning

- Discriminative learning
  - approximated posterior: confidence measures
  - PAC-Bayesian bound: learner & learning complexity

- Generative learning
  - discrete variational auto-encoders
  - Gumbel-max perturbation models
  - structured latent spaces
  - semi supervised setting

- Reinforcement learning
  - approximated posterior: confidence measures
  - PAC-Bayesian bound: learner & learning complexity

Thank you