Deep Bayesian Generative Models for Knowledge Transfer and MRI Processing

MLSS'19, Moscow

Evgeny Burnaev Evgenii Egorov Anna Kuzina

Skolkovo Institute of Science and Technology, Moscow, Russia
ADASE group
{e.burnaev, e.egorov, a.kuzina}@skoltech.ru

Outline

Overview

Probabilistic Machine Learning

- A probabilistic model considers the joint distribution over the

Probabilistic Machine Learning

- A probabilistic model considers the joint distribution over the
  - observed variables x (training data)

Probabilistic Machine Learning

- A probabilistic model considers the joint distribution over the
  - observed variables x (training data)
  - hidden variables θ (the parameters of interest)

Probabilistic Machine Learning

- A probabilistic model considers the joint distribution over the
  — observed variables x (training data)
  — hidden variables θ (the parameters of interest)
- The Bayesian Inference — estimate unknowns through the **posterior distribution**:

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{\int\limits_{\Theta} p(x|\theta)p(\theta)d\theta}$$

Probabilistic Machine Learning

- A probabilistic model considers the joint distribution over the
  — observed variables x (training data)
  — hidden variables θ (the parameters of interest)
- The Bayesian Inference — estimate unknowns through the **posterior distribution**:

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{\int\limits_{\Theta} p(x|\theta)p(\theta)d\theta}$$

- Here

$p(\theta)$ is the prior distribution,
$p(x|\theta)$ is the assumed model

Probabilistic Machine Learning: Challenge

Benefits

- Prior Knowledge/Structure Incorporation

Probabilistic Machine Learning: Challenge

Benefits

- Prior Knowledge/Structure Incorporation
- Ensembles and Uncertainty Estimation

Probabilistic Machine Learning: Challenge

Benefits

- Prior Knowledge/Structure Incorporation
- Ensembles and Uncertainty Estimation
- Coherent framework for the Sequential/Distributive Learning

Probabilistic Machine Learning: Challenge

Benefits

- Prior Knowledge/Structure Incorporation
- Ensembles and Uncertainty Estimation
- Coherent framework for the Sequential/Distributive Learning

Challenge

Evaluation of the posterior $p(\theta|x)$ is hard as require integration:

$$\int_{\Theta} p(x|\theta)p(\theta)d\theta$$

E.g. Θ — high-dimensional space, $p(x|\theta)$ — Deep Neural Network

Probabilistic Machine Learning: Challenge

Benefits

- Prior Knowledge/Structure Incorporation
- Ensembles and Uncertainty Estimation
- Coherent framework for the Sequential/Distributive Learning

Challenge

Evaluation of the posterior $p(\theta|x)$ is hard as require integration:

$$\int_{\Theta} p(x|\theta)p(\theta)d\theta$$

E.g. Θ — high-dimensional space, $p(x|\theta)$ — Deep Neural Network

Solution:

Approximate Inference

Approximate Inference: Approaches

MCMC

Variational Inference

1. Choose the **proposal** distribution $q_\phi(\theta)$ from the **tractable** family Q_ϕ

Approximate Inference: Approaches

MCMC

Variational Inference

1. Choose the **proposal** distribution $q_\phi(\theta)$ from the **tractable** family Q_ϕ
2. Draw samples from a Markov chain with the $p(\theta|x)$ invariant distribution

Approximate Inference: Approaches

MCMC

1. Choose the **proposal** distribution $q_\phi(\theta)$ from the **tractable** family Q_ϕ
2. Draw samples from a Markov chain with the $p(\theta|x)$ invariant distribution
3. Approximate expectations over $p(\theta|x)$ with averaging over the Markov chain samples

Variational Inference

Approximate Inference: Approaches

MCMC

1. Choose the **proposal** distribution $q_\phi(\theta)$ from the **tractable** family Q_ϕ
2. Draw samples from a Markov chain with the $p(\theta|x)$ invariant distribution
3. Approximate expectations over $p(\theta|x)$ with averaging over the Markov chain samples

Variational Inference

1. Choose the **surrogate** distribution $q_\phi(\theta)$ from the **tractable** family Q_ϕ

Approximate Inference: Approaches

MCMC

1. Choose the **proposal** distribution $q_\phi(\theta)$ from the **tractable** family Q_ϕ
2. Draw samples from a Markov chain with the $p(\theta|x)$ invariant distribution
3. Approximate expectations over $p(\theta|x)$ with averaging over the Markov chain samples

Variational Inference

1. Choose the **surrogate** distribution $q_\phi(\theta)$ from the **tractable** family Q_ϕ
2. Define the optimization problem by divergence minimization:

$$\mathcal{B}[p(x|\theta)p(\theta); q_\phi(\theta)]$$

Approximate Inference: Approaches

MCMC

1. Choose the **proposal** distribution $q_\phi(\theta)$ from the **tractable** family Q_ϕ
2. Draw samples from a Markov chain with the $p(\theta|x)$ invariant distribution
3. Approximate expectations over $p(\theta|x)$ with averaging over the Markov chain samples

Variational Inference

1. Choose the **surrogate** distribution $q_\phi(\theta)$ from the **tractable** family Q_ϕ
2. Define the optimization problem by divergence minimization:

$$\mathcal{B}[p(x|\theta)p(\theta); q_\phi(\theta)]$$

3. $q^*(\theta) = \arg\min_\phi \mathcal{B}[p(x|\theta)p(\theta); q_\phi(\theta)]$

Approximate Inference: Approaches

MCMC

1. Choose the **proposal** distribution $q_\phi(\theta)$ from the **tractable** family Q_ϕ
2. Draw samples from a Markov chain with the $p(\theta|x)$ invariant distribution
3. Approximate expectations over $p(\theta|x)$ with averaging over the Markov chain samples

Variational Inference

1. Choose the **surrogate** distribution $q_\phi(\theta)$ from the **tractable** family Q_ϕ
2. Define the optimization problem by divergence minimization:

$$\mathcal{B}[p(x|\theta)p(\theta); q_\phi(\theta)]$$

3. $q^*(\theta) = \arg\min_\phi \mathcal{B}[p(x|\theta)p(\theta); q_\phi(\theta)]$

"Tractable"

Approximate Inference: Approaches

MCMC

1. Choose the **proposal** distribution $q_\phi(\theta)$ from the **tractable** family Q_ϕ
2. Draw samples from a Markov chain with the $p(\theta|x)$ invariant distribution
3. Approximate expectations over $p(\theta|x)$ with averaging over the Markov chain samples

Variational Inference

1. Choose the **surrogate** distribution $q_\phi(\theta)$ from the **tractable** family Q_ϕ
2. Define the optimization problem by divergence minimization:
$$\mathcal{B}[p(x|\theta)p(\theta); q_\phi(\theta)]$$
3. $q^*(\theta) = \arg\min_\phi \mathcal{B}[p(x|\theta)p(\theta); q_\phi(\theta)]$

"Tractable"

- Easy to sample from

Approximate Inference: Approaches

MCMC

1. Choose the **proposal** distribution $q_\phi(\theta)$ from the **tractable** family Q_ϕ
2. Draw samples from a Markov chain with the $p(\theta|x)$ invariant distribution
3. Approximate expectations over $p(\theta|x)$ with averaging over the Markov chain samples

Variational Inference

1. Choose the **surrogate** distribution $q_\phi(\theta)$ from the **tractable** family Q_ϕ
2. Define the optimization problem by divergence minimization:

$$\mathcal{B}[p(x|\theta)p(\theta); q_\phi(\theta)]$$

3. $q^*(\theta) = \arg\min_\phi \mathcal{B}[p(x|\theta)p(\theta); q_\phi(\theta)]$

"Tractable"

- Easy to sample from
- Easy to evaluate log-density

Approximate Inference: Approaches

MCMC

1. Choose the **proposal** distribution $q_\phi(\theta)$ from the **tractable** family Q_ϕ
2. Draw samples from a Markov chain with the $p(\theta|x)$ invariant distribution
3. Approximate expectations over $p(\theta|x)$ with averaging over the Markov chain samples

Variational Inference

1. Choose the **surrogate** distribution $q_\phi(\theta)$ from the **tractable** family Q_ϕ
2. Define the optimization problem by divergence minimization:
$$\mathcal{B}[p(x|\theta)p(\theta); q_\phi(\theta)]$$
3. $q^*(\theta) = \arg\min_\phi \mathcal{B}[p(x|\theta)p(\theta); q_\phi(\theta)]$

"Tractable"

- Easy to sample from
- Easy to evaluate log-density
- ...

Approximate Inference: Challenges

MCMC

Variational Inference

Pros

- Allow to trade computation time for increased accuracy
- Asymptotically unbiased
- Provide samples

Approximate Inference: Challenges

MCMC

Variational Inference

Pros

- Allow to trade computation time for increased accuracy
- Asymptotically unbiased
- Provide samples

Cons

- Sensitive to proposal selection
- Convergence diagnostic is hard
- Bad scalability (both on data and dimension)
- Provide only samples

Approximate Inference: Challenges

MCMC

Pros

- Allow to trade computation time for increased accuracy
- Asymptotically unbiased
- Provide samples

Cons

- Sensitive to proposal selection
- Convergence diagnostic is hard
- Bad scalability (both on data and dimension)
- Provide only samples

Variational Inference

Pros

- Scalability: Fine with stochastic optimization and amortization
- Easy to incorporate the structure of the problem to efficient optimization
- Flexible Q_λ families parametrized by DNN
- Provide approximations with density

Approximate Inference: Challenges

MCMC

Pros

- Allow to trade computation time for increased accuracy
- Asymptotically unbiased
- Provide samples

Cons

- Sensitive to proposal selection
- Convergence diagnostic is hard
- Bad scalability (both on data and dimension)
- Provide only samples

Variational Inference

Pros

- Scalability: Fine with stochastic optimization and amortization
- Easy to incorporate the structure of the problem to efficient optimization
- Flexible Q_λ families parametrized by DNN
- Provide approximations with density

Cons

- Biased (underestimating the posterior variances)
- Optimization is hard

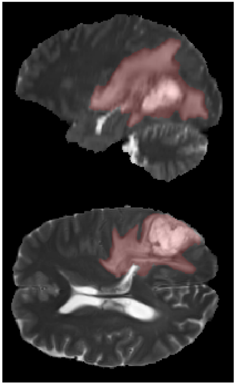
Bayesian generative models for knowledge transfer in DNN on 3D MRI data

A. Kuzina, E. Egorov, E. Burnaev. Bayesian generative models for knowledge transfer in MRI semantic segmentation problems. Journal: Frontiers in Neuroscience, section Brain Imaging Methods, 2019

https://arxiv.org/abs/1908.05480

Problem statement

Magnetic resonance imaging (MRI) — medical imaging technique used in radiology to form pictures of the body anatomy



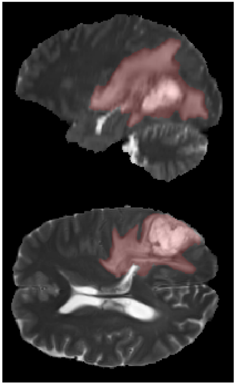
MRI with labelled brain tumor

Problem statement

Magnetic resonance imaging (MRI) — medical imaging technique used in radiology to form pictures of the body anatomy

MRI semantic segmentation applications in medicine:

- Tumors (e.g. brain, liver) analysis and monitoring



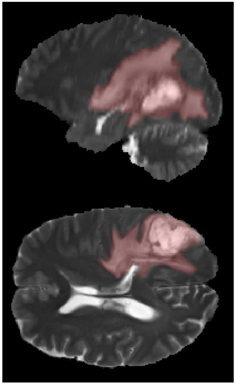
MRI with labelled brain tumor

Problem statement

Magnetic resonance imaging (MRI) — medical imaging technique used in radiology to form pictures of the body anatomy

MRI semantic segmentation applications in medicine:

- Tumors (e.g. brain, liver) analysis and monitoring
- Multiple sclerosis plaques detection



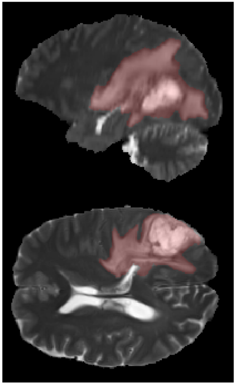
MRI with labelled brain tumor

Problem statement

Magnetic resonance imaging (MRI) — medical imaging technique used in radiology to form pictures of the body anatomy

MRI semantic segmentation applications in medicine:

- Tumors (e.g. brain, liver) analysis and monitoring
- Multiple sclerosis plaques detection
- White matter hyperintesities detection



MRI with labelled brain tumor

Main challenges of the area

1. **Scarce data**
   - Expensive annotation
   - Privacy concerns
   - Bad performance of transfer learning due to disease specificity

Main challenges of the area

1. Scarce data

- Expensive annotation
- Privacy concerns
- Bad performance of transfer learning due to disease specificity

2. High dimensionality

- 3D images
- Memory issues

Main challenges of the area

1. Scarce data

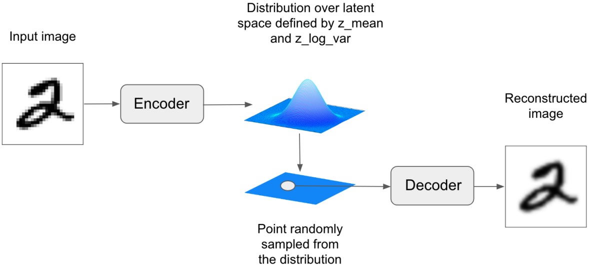
- Expensive annotation
- Privacy concerns
- Bad performance of transfer learning due to disease specificity

2. High dimensionality

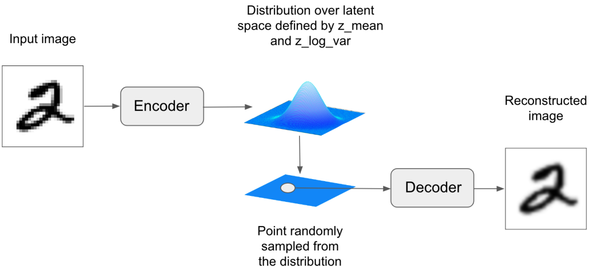
- 3D images
- Memory issues

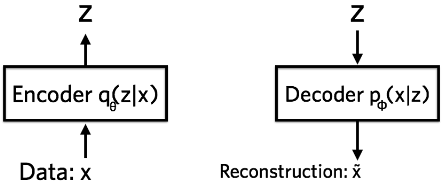
Solution for 1:
Transfer Learning under Bayesian Approach

Variational Autoencoder



Variational Autoencoder

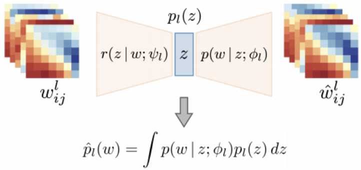




Deep Weight Prior

Main Idea

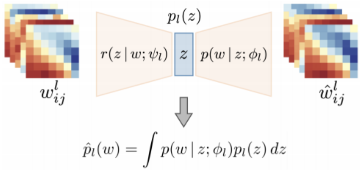
Use VAE model to learn implicit prior distribution over convolutional filters of each layer



Deep Weight Prior

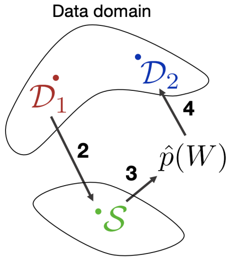
Main Idea

Use VAE model to learn implicit prior distribution over convolutional filters of each layer



Algorithm

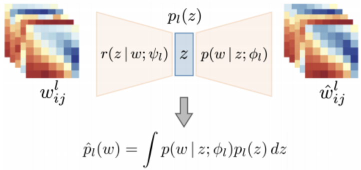
1. Train network on the bootstrapped source dataset ($\mathcal{D}_1$)



Deep Weight Prior

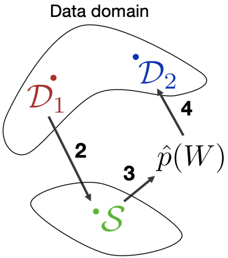
Main Idea

Use VAE model to learn implicit prior distribution over convolutional filters of each layer



Algorithm

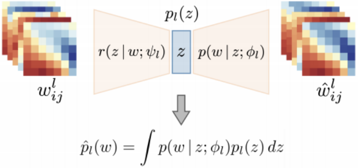
1. Train network on the bootstrapped source dataset ($\mathcal{D}_1$)
2. Collect learned filters



Deep Weight Prior

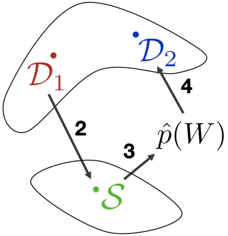
Main Idea

Use VAE model to learn implicit prior distribution over convolutional filters of each layer



Algorithm

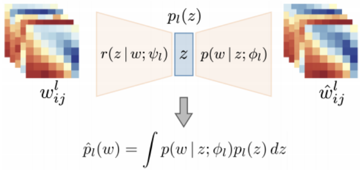
1. Train network on the bootstrapped source dataset ($\mathcal{D}_1$)
2. Collect learned filters
3. Train implicit prior distribution (VAE)



Deep Weight Prior

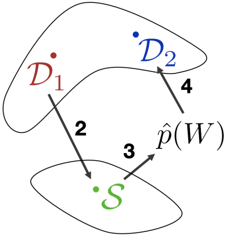
Main Idea

Use VAE model to learn implicit prior distribution over convolutional filters of each layer



Algorithm

1. Train network on the bootstrapped source dataset ($\mathcal{D}_1$)
2. Collect learned filters
3. Train implicit prior distribution (VAE)
4. Use trained prior for variational inference on the target dataset ($\mathcal{D}_2$)



Experimental Set-Up

Datasets

- 285 MRI of patients with brain tumor (BRATS18)
- 170 MRI of patients with multiple sclerosis (MS)

Prepocessing:

- Scaling
- Alignment
- Scull-stripping

Experimental Set-Up

Datasets

- 285 MRI of patients with brain tumor (BRATS18)
- 170 MRI of patients with multiple sclerosis (MS)

Prepocessing:

- Scaling
- Alignment
- Scull-stripping

Task:

Binary semantic segmentation

Experimental Set-Up

Datasets

- 285 MRI of patients with brain tumor (BRATS18)
- 170 MRI of patients with multiple sclerosis (MS)

Prepocessing:

- Scaling
- Alignment
- Scull-stripping

Task:
Binary semantic segmentation

Metrics:
Dice Similarity Coefficient:

$$\mathrm{DSC} = \frac{2TP}{2TP + FP + FN}$$

Intersection over Union:

$$\mathrm{IOU} = \frac{TP}{TP + FP + FN}$$

Example of MRI slices and ground truth segmentation

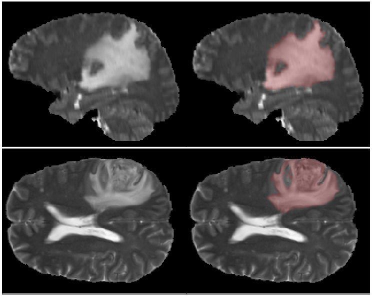


Figure: BRATS18 dataset

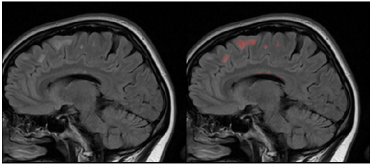


Figure: MS dataset

Experimental Set-Up [2]

- Train Unet models on the full MS dataset
  - Use bootstrapped sample from the initial dataset
  - Weights are randomly initialized

Experimental Set-Up [2]

- Train Unet models on the full MS dataset
  - Use bootstrapped sample from the initial dataset
  - Weights are randomly initialized
- Collect filters from the trained models
  - Use cycling learning rate to expand set of learned filters

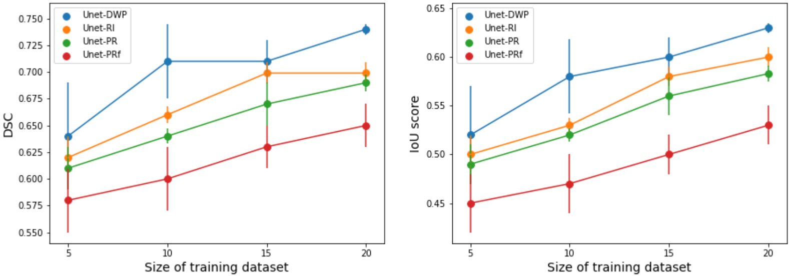
Experimental Set-Up [2]

- Train Unet models on the full MS dataset
  - Use bootstrapped sample from the initial dataset
  - Weights are randomly initialized
- Collect filters from the trained models
  - Use cycling learning rate to expand set of learned filters
- Train VAE for each layer / set of consecutive layers

Experimental Set-Up [2]

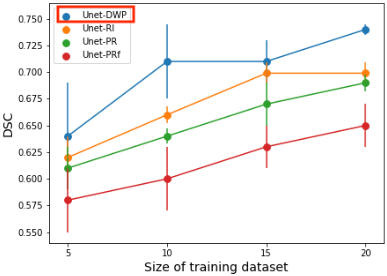
- Train Unet models on the full MS dataset
  - Use bootstrapped sample from the initial dataset
  - Weights are randomly initialized
- Collect filters from the trained models
  - Use cycling learning rate to expand set of learned filters
- Train VAE for each layer / set of consecutive layers
- Do variational inference with implicit prior (VAE) on subset of BRATS18 dataset (5-20 images)

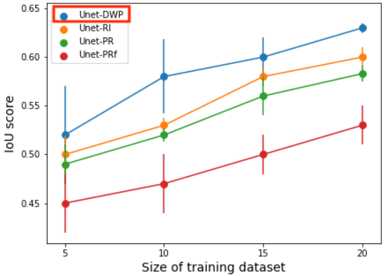
Results



- **Unet-RI** (orange): without transfer learning
- **Unet-PR** (green): fine-tuning of the whole network
- **Unet-PRf** (red): fine-tuning of the input and output block

Results





- **Unet-RI** (orange): without transfer learning
- **Unet-PR** (green): fine-tuning of the whole network
- **Unet-PRf** (red): fine-tuning of the input and output block

Additional Results

- Hypothesis: transfer learning between different parts of the human body is possible

Additional Results

- Hypothesis: transfer learning between different parts of the human body is possible
- Experiment: from CT scans of the liver to spleen on the CT scans for 41 patients

Additional Results

- Hypothesis: transfer learning between different parts of the human body is possible
- Experiment: from CT scans of the liver to spleen on the CT scans for 41 patients

	Task09_Spleen		
Train size	UNet-DWP (ours)	UNet-RI	UNet-PR
5	0.275	**0.284**	0.209
10	**0.328**	0.293	0.052
15	**0.389**	0.306	0.243
20	**0.353**	0.336	0.156

Table: Mean Dice Similarity Score for the subsets of `Task03_Liver` and `Task09_Spleen` datasets.

DWP Conclusion

Results

- DWP was successfully transfered to 3D
- We perform transfer learning by implicit prior training
- Proposed solutions outperforms other approaches

DWP Conclusion

Results

- DWP was successfully transfered to 3D
- We perform transfer learning by implicit prior training
- Proposed solutions outperforms other approaches

Next Step

How to update our prior with a new dataset, i.e. perform incremental learning for VAE?

Appendix: lower bound on KL-divergence for DWP

$$\mathrm{KL}(q_\theta(W)||p(W)) = \sum_{l,i,j} \mathrm{KL}(q_{\theta^l_{ij}}(w^l_{ij})||p(w^l_{ij})) \le$$

$$\le \sum_{l,i,j} \left(-\mathrm{H}(q_{\theta^l_{ij}}(w^l_{ij})) + \mathbb{E}_{q_{\theta^l_{ij}}(w^l_{ij})} \mathrm{KL}(r(z|w^l_{i,j})||p^l(z)) + \mathbb{E}_{r(z|w^l_{i,j})} \log p(w^l_{i,j}|z) \right) = \mathrm{KL}_{bound}$$

$$\mathcal{L}(\theta) = L_D - \mathrm{KL}(q_\theta(W)||p(W)) \ge L_D - \mathrm{KL}_{bound} \to \max$$

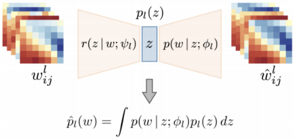
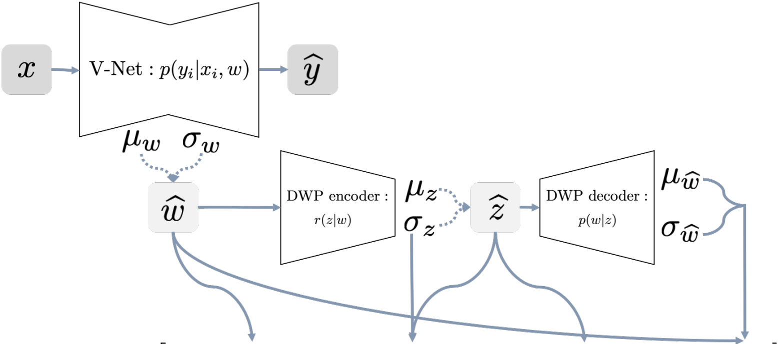


Figure: VAE for learning DWP

Appendix: DWP loss illustartion



$$\mathcal{L}^{\text{approx}} = \mathcal{L}_{\mathcal{D}^*} + \sum\nolimits_{p,k,i} \left[-\log q_{\theta^{(i)}_{pk}}(\widehat{w}^{(i)}_{p,k})) - \log r_{\psi^{(i)}}(\widehat{z}|\widehat{w}^{(i)}_{p,k}) + \log p^{(i)}(\widehat{z}) + \log p_{\phi^{(i)}}(\widehat{w}^{(i)}_{p,k})|\widehat{z}) \right]$$

Variational Inference via MaxEnt Pursuit

Evgenii Egorov, Kirill Neklyudov, Ruslan Kostoev, Evgeny Burnaev. MaxEntropy Pursuit Variational Inference. ISNN 2019: Advances in Neural Networks – ISNN, Springer, 2019 pp. 409-417

https://arxiv.org/abs/1905.07855

MPVI: General Idea

Solution Plan

1. Select family of simple “base learners” $h(\theta) \in Q_\lambda$, e.g. Normal Densities

MPVI: General Idea

Solution Plan

1. Select family of simple “base learners” $h(\theta) \in Q_\lambda$, e.g. Normal Densities
2. Iteratively improve the approximation by additive convex update $q_T(\theta) = (1 - \alpha)q_{T-1}(\theta) + \alpha h(\theta)$

MPVI: General Idea

Solution Plan

1. Select family of simple “base learners” $h(\theta) \in Q_\lambda$, e.g. Normal Densities
2. Iteratively improve the approximation by additive convex update $q_T(\theta) = (1-\alpha)q_{T-1}(\theta) + \alpha h(\theta)$
3. Perform functional gradient descent over KL-divergence to select each component

MPVI: General Idea

Solution Plan

1. Select family of simple "base learners" $h(\theta) \in Q_\lambda$, e.g. Normal Densities
2. Iteratively improve the approximation by additive convex update $q_T(\theta) = (1-\alpha)q_{T-1}(\theta) + \alpha h(\theta)$
3. Perform functional gradient descent over KL-divergence to select each component

Challenges

MPVI: General Idea

Solution Plan

1. Select family of simple "base learners" $h(\theta) \in Q_\lambda$, e.g. Normal Densities
2. Iteratively improve the approximation by additive convex update $q_T(\theta) = (1-\alpha)q_{T-1}(\theta) + \alpha h(\theta)$
3. Perform functional gradient descent over KL-divergence to select each component

Challenges

- Avoid degenerate solution (mixture of delta functions)

MPVI: General Idea

Solution Plan

1. Select family of simple "base learners" $h(\theta) \in Q_\lambda$, e.g. Normal Densities
2. Iteratively improve the approximation by additive convex update $q_T(\theta) = (1-\alpha)q_{T-1}(\theta) + \alpha h(\theta)$
3. Perform functional gradient descent over KL-divergence to select each component

Challenges

- Avoid degenerate solution (mixture of delta functions)
- Keep inference data scalable and computationally efficient

MPVI: General Idea

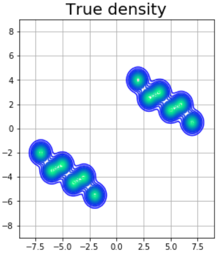
Solution Plan

1. Select family of simple "base learners" $h(\theta) \in Q_\lambda$, e.g. Normal Densities
2. Iteratively improve the approximation by additive convex update $q_T(\theta) = (1-\alpha)q_{T-1}(\theta) + \alpha h(\theta)$
3. Perform functional gradient descent over KL-divergence to select each component

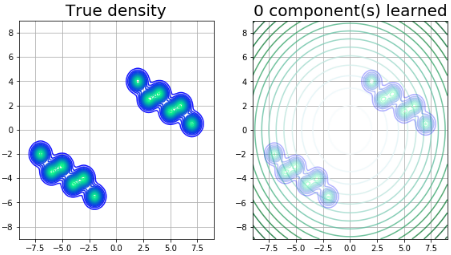
Challenges

- Avoid degenerate solution (mixture of delta functions)
- Keep inference data scalable and computationally efficient
- Avoid model specific work

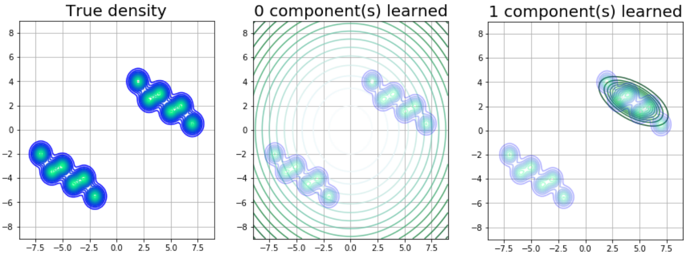
MPVI: Toy example



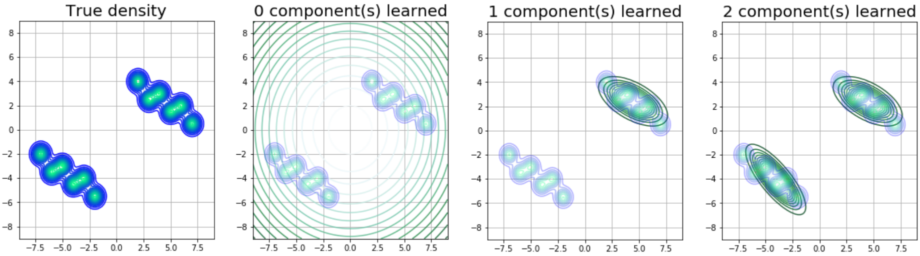
MPVI: Toy example



MPVI: Toy example



MPVI: Toy example



MPVI | Component Optimization: Problem

- Given some approximation of the posterior distribution q_t

MPVI | Component Optimization: Problem

- Given some approximation of the posterior distribution q_t
- Goal is to improve accuracy of the approximation in terms of the KL-divergence by using the additive mixture:

$$q_{t+1} = (1 - \alpha)q_t + \alpha h, \ \alpha \in (0; 1), \ h \in Q$$

MPVI | Component Optimization: Problem

- Given some approximation of the posterior distribution q_t
- Goal is to improve accuracy of the approximation in terms of the KL-divergence by using the additive mixture:

$$q_{t+1} = (1-\alpha)q_t + \alpha h,\ \alpha \in (0;1),\ h \in Q$$

- Hence, using Maximum Entropy Approach we state the following optimization problem:

$$\max_{h \in \mathcal{Q}} \mathcal{H}[h], s.t.$$
$$\mathcal{F}[q_{t+1}] - \mathcal{F}[q_t] > 0.$$

MPVI | Component Optimization: Problem

- Given some approximation of the posterior distribution q_t
- Goal is to improve accuracy of the approximation in terms of the KL-divergence by using the additive mixture:

$$q_{t+1} = (1-\alpha)q_t + \alpha h,\ \alpha \in (0;1),\ h \in Q$$

- Hence, using Maximum Entropy Approach we state the following optimization problem:

$$\max_{h \in \mathcal{Q}} \mathcal{H}[h], s.t.$$
$$\mathcal{F}[q_{t+1}] - \mathcal{F}[q_t] > 0.$$

- Here $\mathcal{F}[q]$ is ELBO (accuracy of approximation)

$$\mathcal{F}[q] = \mathbb{E}_q \left[\frac{p(x,\theta)}{q(\theta)} \right]$$

MPVI | Component Optimization: Problem

- Optimization problem:

$$\max_{h \in \mathcal{Q}} \mathcal{H}[h], s.t.$$
$$\mathcal{F}[q_{t+1}] - \mathcal{F}[q_t] > 0.$$

MPVI | Component Optimization: Problem

- Optimization problem:

$$\max_{h \in \mathcal{Q}} \mathcal{H}[h], s.t.$$
$$\mathcal{F}[q_{t+1}] - \mathcal{F}[q_t] > 0.$$

- Using Taylor expansion, we obtain the constraint in the following form:

$$\mathcal{F}[q_{t+1}] - \mathcal{F}[q_t] = \alpha \left\langle h - q_t, \log \frac{p(x,\theta)}{q_t} \right\rangle - \alpha^2 \int \frac{(h - q_t)^2}{q_t} d\theta + \dots$$

MPVI | Component Optimization: Problem

- Optimization problem:

$$\max_{h \in \mathcal{Q}} \mathcal{H}[h], s.t.$$
$$\mathcal{F}[q_{t+1}] - \mathcal{F}[q_t] > 0.$$

- Using Taylor expansion, we obtain the constraint in the following form:

$$\mathcal{F}[q_{t+1}] - \mathcal{F}[q_t] = \alpha \left\langle h - q_t, \log \frac{p(x,\theta)}{q_t} \right\rangle - \alpha^2 \int \frac{(h - q_t)^2}{q_t} d\theta + \dots$$

- Considering the first order terms, we get the following optimization problem:

$$\max_{h \in Q} \mathcal{H}[h] + \lambda \left\langle h, \log \frac{p(x,\theta)}{q_t} \right\rangle .$$

MPVI | Component Optimization: Solution

$$\max_{h \in Q} \mathcal{H}[h] + \lambda \left\langle h, \log \frac{p(x,\theta)}{q_t} \right\rangle$$

Problem Proprieties

- Strictly concave over h

MPVI | Component Optimization: Solution

$$\max_{h \in Q} \mathcal{H}[h] + \lambda \left\langle h, \log \frac{p(x,\theta)}{q_t} \right\rangle$$

Problem Proprieties

- Strictly concave over h
- Could be solved by stochastic gradient optimization, i.e. scalable over dataset size

MPVI | Component Optimization: Solution

$$\max_{h \in Q} \mathcal{H}[h] + \lambda \left\langle h, \log \frac{p(x,\theta)}{q_t} \right\rangle$$

Problem Proprieties

- Strictly concave over h
- Could be solved by stochastic gradient optimization, i.e. scalable over dataset size
- Exact solution is $h^* = \frac{1}{Z(\lambda)} \left[\frac{p(x,\theta)}{q_t} \right]^\lambda$

MPVI | Component Optimization: Solution

$$\max_{h \in Q} \mathcal{H}[h] + \lambda \left\langle h, \log \frac{p(x,\theta)}{q_t} \right\rangle$$

Problem Proprieties

- Strictly concave over h
- Could be solved by stochastic gradient optimization, i.e. scalable over dataset size
- Exact solution is $h^* = \frac{1}{Z(\lambda)} \left[\frac{p(x,\theta)}{q_t}\right]^\lambda$

Since h^* is intractable, we can find it by optimizing

$$h^* = \arg\min_{h \in Q} KL \left(h \middle\| \frac{1}{Z(\lambda)} \left[\frac{p(x,\theta)}{q_t}\right]^\lambda \right).$$

MPVI | Connection with Variational Inference

For $\lambda = 1$ **MPVI** optimization problem:

$$\arg\max_{h \in Q} \mathcal{H}[h] + \left\langle h, \log \frac{p(x, \theta)}{q_t} \right\rangle =$$

MPVI | Connection with Variational Inference

For $\lambda = 1$ **MPVI** optimization problem:

$$\arg\max_{h \in Q} \mathcal{H}[h] + \left\langle h, \log \frac{p(x,\theta)}{q_t} \right\rangle = \arg\max_{h \in Q} \underbrace{\int h \log \frac{p(x,\theta)}{h} d\theta}_{\text{term (1)}} - \underbrace{\int h \log q_t d\theta}_{\text{term (2)}}$$

MPVI | Connection with Variational Inference

For $\lambda = 1$ **MPVI** optimization problem:

$$\arg\max_{h \in Q} \mathcal{H}[h] + \left\langle h, \log \frac{p(x,\theta)}{q_t} \right\rangle = \arg\max_{h \in Q} \underbrace{\int h \log \frac{p(x,\theta)}{h} d\theta}_{\text{term (1)}} - \underbrace{\int h \log q_t d\theta}_{\text{term (2)}}$$

We can note than:

- Term (1) corresponds to the standard **Variational Inference** objective (ELBO)

MPVI | Connection with Variational Inference

For $\lambda = 1$ **MPVI** optimization problem:

$$\arg\max_{h \in Q} \mathcal{H}[h] + \left\langle h, \log \frac{p(x,\theta)}{q_t} \right\rangle = \arg\max_{h \in Q} \underbrace{\int h \log \frac{p(x,\theta)}{h} d\theta}_{\text{term (1)}} - \underbrace{\int h \log q_t d\theta}_{\text{term (2)}}$$

We can note than:

- Term (1) corresponds to the standard **Variational Inference** objective (ELBO)
- Term (2) plays the role of **similarity penalty** with the current solution q_t

MPVI | Weight Optimization

After getting h for given q_t, we should select α in a

$$q_{t+1}(\theta) = (1-\alpha)q_t(\theta) + \alpha h(\theta)$$

MPVI | Weight Optimization

After getting h for given q_t, we should select α in a

$$q_{t+1}(\theta) = (1-\alpha)q_t(\theta) + \alpha h(\theta)$$

For that we solve

$$\min_{\alpha \in (0;1)} KL((1-\alpha)q_t(\theta) + \alpha h(\theta) || p(\theta|x))$$

Theoretical solution

- Convex problem
- $\alpha^* = -\frac{\int \frac{1}{p(\theta|x)} q_t(h - q_t) d\theta}{\int \frac{1}{p(\theta|x)} (h-q_t)^2 d\theta} = -\frac{\int \frac{1}{p(x,\theta)} q_t(h - q_t) d\theta}{\int \frac{1}{p(x,\theta)} (h-q_t)^2 d\theta}$

MPVI | Weight Optimization

After getting h for given q_t, we should select α in a

$$q_{t+1}(\theta) = (1-\alpha)q_t(\theta) + \alpha h(\theta)$$

For that we solve

$$\min_{\alpha\in(0;1)} KL((1-\alpha)q_t(\theta) + \alpha h(\theta)||p(\theta|x))$$

Theoretical solution

- Convex problem
- $\alpha^* = -\dfrac{\int \frac{1}{p(\theta|x)} q_t(h-q_t)d\theta}{\int \frac{1}{p(\theta|x)}(h-q_t)^2 d\theta} = -\dfrac{\int \frac{1}{p(x,\theta)} q_t(h-q_t)d\theta}{\int \frac{1}{p(x,\theta)}(h-q_t)^2 d\theta}$

Implementation

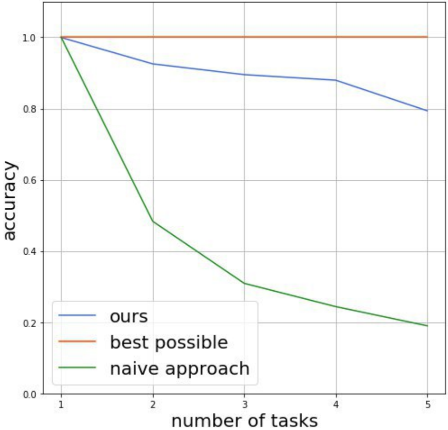
In practice we use stochastic gradient descent over α

MPVI Incremental Learning

Problem: Neural Networks suffer from Catastrophic Forgetting
Solution: $p(\theta|x, x^{\text{new}}) \approx (1-\alpha)q(\theta|x) + \alpha q(\theta|x^{\text{new}})$

Experiment

- Dataset: MNIST, 10 classes classification
- Incremental setting: pair classes arrive: 0 vs 1, 2 vs 3, etc.
- Neural Network: LeNet-5
- Prior: Factorized Normal
- Metric: Accuracy



BooVAE: incremental learning for VAE

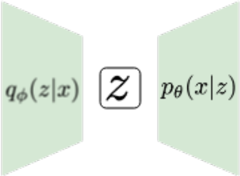
Anna Kuzina, Evgenii Egorov, Evgeny Burnaev. BooVAE: A scalable framework for continual VAE learning under boosting approach

https://arxiv.org/abs/1908.11853

Boosting for Incremental Learning: VAE

Optimal Prior for VAE[1]

$$p^*(z) = \arg\max_{p(\cdot)} \mathcal{L}(Data, p) = \frac{1}{N}\sum_{n=1}^{N} q_\phi(z|x_n)$$

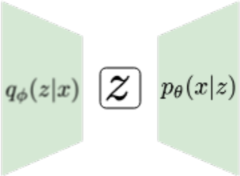


[1] Tomczak, J. M., Welling, M. (2017). VAE with a VampPrior

Boosting for Incremental Learning: VAE

Optimal Prior for VAE[1]

$$p^*(z) = \arg\max_{p(\cdot)} \mathcal{L}(Data, p) = \frac{1}{N}\sum_{n=1}^{N} q_\phi(z|x_n)$$



- We approximate $p^*(z)$ via boosting
- Given p_T from the previous task, learn a new component h to update the learned prior for the new task T+1:

$$KL\left(\alpha h + (1-\alpha)p_T || p^*\right) \to \min_{h,\alpha}$$

[1] Tomczak, J. M., Welling, M. (2017). VAE with a VampPrior

BooVAE algorithm

Require: Dataset $\{(x_i)\}_{i=1}^N$
Require: λ, Maximal number of components K

Choose random subset $\mathcal{M} \subset \mathcal{D}$
Initialize prior $p_0 = q_{\phi^*}(z|u_0)$
$\{\theta^*, \phi^*, u_0\} = \arg\max \mathcal{L}(p_0, \theta, \phi)$
$k = 1$

$k = k + 1$
end if
end while
return p_K, θ^*, ϕ^*

BooVAE algorithm

Require: Dataset $\{(x_i)\}_{i=1}^N$
Require: λ, Maximal number of components K
Choose random subset $\mathcal{M} \subset \mathcal{D}$
Initialize prior $p_0 = q_{\phi^*}(z|u_0)$
$\{\theta^*, \phi^*, u_0\} = \arg\max \mathcal{L}(p_0, \theta, \phi)$
$k = 1$
while not converged **do**
  Update network parameters $\theta^*, \phi^* = \arg\max \mathcal{L}(p_{k-1}, \theta, \phi)$
  if $k < K$ **then**
    $k = k + 1$
  end if
end while
return p_K, θ^*, ϕ^*

BooVAE algorithm

Require: Dataset $\{(x_i)\}_{i=1}^N$
Require: λ, Maximal number of components K

Choose random subset $\mathcal{M} \subset \mathcal{D}$
Initialize prior $p_0 = q_{\phi^*}(z|u_0)$
$\{\theta^*, \phi^*, u_0\} = \arg\max \mathcal{L}(p_0, \theta, \phi)$
$k = 1$
while not converged **do**
  Update network parameters $\theta^*, \phi^* = \arg\max \mathcal{L}(p_{k-1}, \theta, \phi)$
  if $k < K$ **then**
    Update optimal prior $p^*(z) = \frac{1}{n} \sum_{x \in \mathcal{M}} q_{\phi^*}(z|x)$
    $k = k + 1$
  end if
end while
return p_K, θ^*, ϕ^*

BooVAE algorithm

Require: Dataset $\{(x_i)\}_{i=1}^N$
Require: λ, Maximal number of components K
Choose random subset $\mathcal{M} \subset \mathcal{D}$
Initialize prior $p_0 = q_{\phi^*}(z|u_0)$
$\{\theta^*, \phi^*, u_0\} = \arg\max \mathcal{L}(p_0, \theta, \phi)$
$k = 1$
while not converged **do**
Update network parameters $\theta^*, \phi^* = \arg\max \mathcal{L}(p_{k-1}, \theta, \phi)$
if $k < K$ **then**
Update optimal prior $p^*(z) = \frac{1}{n}\sum_{x \in \mathcal{M}} q_{\phi^*}(z|x)$

Add new component $p_k = \alpha^* h^* + (1 - \alpha^*) p_{k-1}$

$k = k + 1$
end if
end while
return p_K, θ^*, ϕ^*

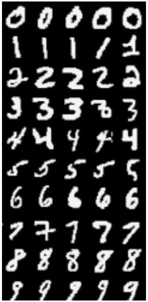
BooVAE algorithm

Require: Dataset $\{(x_i)\}_{i=1}^N$
Require: λ, Maximal number of components K
Choose random subset $\mathcal{M} \subset \mathcal{D}$
Initialize prior $p_0 = q_{\phi^*}(z|u_0)$
$\{\theta^*, \phi^*, u_0\} = \arg\max \mathcal{L}(p_0, \theta, \phi)$
$k = 1$
while not converged **do**
  Update network parameters $\theta^*, \phi^* = \arg\max \mathcal{L}(p_{k-1}, \theta, \phi)$
  if $k < K$ **then**
    Update optimal prior $p^*(z) = \frac{1}{n}\sum_{x \in \mathcal{M}} q_{\phi^*}(z|x)$

    Add new component $p_k = \alpha^* h^* + (1 - \alpha^*) p_{k-1}$
      $h^* = \arg\min KL\left(h \middle\| \left[\frac{p^*}{p_{k-1}}\right]^\lambda\right)$
      $\alpha^* = \arg\min KL(\alpha h + (1-\alpha) p_{k-1} \| p^*)$
    $k = k + 1$
  end if
end while
return p_K, θ^*, ϕ^*

Boosting: experimental set-up [1]

- Tasks are arriving sequentially, one at a time
- **Datsets**: MNIST, fashion MNIST (10 classes each)



MNIST

Fashion MNIST

Boosting: experimental set-up [2]

- For task in $\{1 \ldots K\}$:
  - Train VAE on the images from the current task

Boosting: experimental set-up [2]

- For task in $\{1 \dots K\}$:
  - Train VAE on the images from the current task
- NLL metric on the whole test set:

$$\log p(x) \approx \log \frac{1}{K} \sum_{i=1}^{K} \frac{p_\theta(x|z_i)p(z_i)}{q_\phi(z_i|x)}, \quad z_i \sim q_\phi(z|x)$$

Boosting: experimental set-up [2]

- For task in $\{1 \dots K\}$:
  - Train VAE on the images from the current task
- NLL metric on the whole test set:

$$\log p(x) \approx \log \frac{1}{K} \sum_{i=1}^{K} \frac{p_\theta(x|z_i)p(z_i)}{q_\phi(z_i|x)}, \quad z_i \sim q_\phi(z|x)$$

- Diversity of generated images

$$\sum_{k=1}^{K} \mathrm{KL}\left(u || \widehat{x}\right), \; u \sim \mathrm{Be}\left(\frac{1}{K}\right), \; \widehat{x} \sim \mathrm{Be}\left(\frac{N_k}{N}\right)$$

N_k — number of images from class k among N generated.

MNIST results

Below: the smaller the better

height# tasks	Standard	Standard + EWC	MoG	Boo (ours)	Boo (ours) + EWC
2	343.54 (26.38)	256.55 (8.38)	**96.50 (1.95)**	100.11 (1.39)	97.49 (0.40)
3	122.05 (2.31)	121.91 (1.31)	107.78 (3.59)	104.33 (1.34)	**102.90 (0.95)**
4	146.06 (0.32)	142.00 (2.28)	123.95 (5.05)	118.78 (1.45)	**117.07 (0.46)**
5	197.02 (5.68)	192.84 (0.12)	143.44 (8.05)	132.08 (0.64)	**130.80 (0.87)**
6	164.29 (3.78)	159.80 (3.14)	143.33 (2.49)	135.42 (1.64)	**131.83 (1.24)**
7	205.21 (5.58)	187.43 (5.20)	163.14 (9.02)	142.21 (1.85)	**137.38 (1.57)**
8	213.25 (9.22)	189.06 (4.72)	172.00 (12.93)	140.80 (2.42)	**138.47 (2.50)**
9	171.04 (3.64)	160.47 (2.53)	164.18 (9.49)	141.70 (0.97)	**140.13 (2.67)**
10	186.79 (2.32)	170.26 (2.20)	181.53 (29.02)	142.92 (1.99)	**140.68 (1.86)**

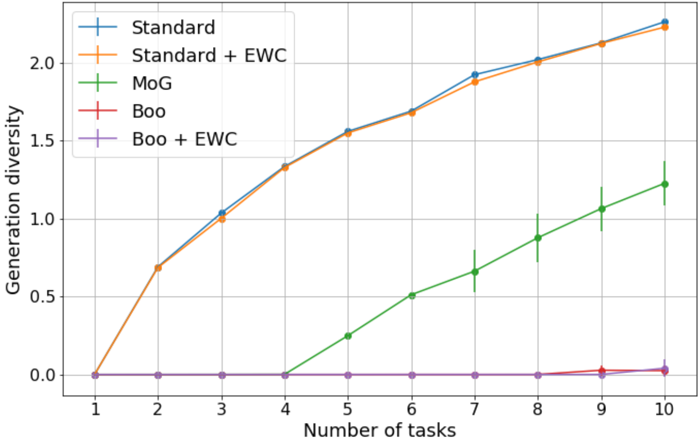
Table: NLL Results on MNIST.

Fashion MNIST results

height# tasks	Standard	Standard + EWC	MoG	Boo (ours)	Boo (ours) + EWC
2	262.22 (2.92)	271.14 (6.05)	239.43 (2.76)	**227.83 (3.34)**	229.81 (2.31)
3	289.45 (2.72)	287.45 (3.87)	266.18 (1.87)	**255.85 (1.61)**	256.47 (2.16)
4	274.08 (2.42)	272.82 (1.02)	264.35 (3.16)	**248.96 (0.85)**	249.08 (1.40)
5	272.87 (1.80)	270.44 (0.98)	264.51 (1.93)	**253.12 (1.43)**	**253.26 (1.20)**
6	487.05 (43.78)	417.81 (7.44)	282.00 (4.06)	**250.87 (2.12)**	**250.64 (1.18)**
7	274.72 (2.93)	272.09 (6.25)	292.93 (9.16)	**250.87 (0.69)**	253.50 (2.59)
8	1827.62 (489.47)	565.81 (22.94)	448.55 (103.92)	260.05 (5.25)	**250.30 (0.48)**
9	321.49 (17.36)	289.17 (2.43)	321.72 (14.11)	**256.42 (1.00)**	**256.33 (0.78)**
10	964.90 (237.27)	427.83 (21.19)	440.96 (49.75)	284.86 (21.21)	**256.58 (1.27)**

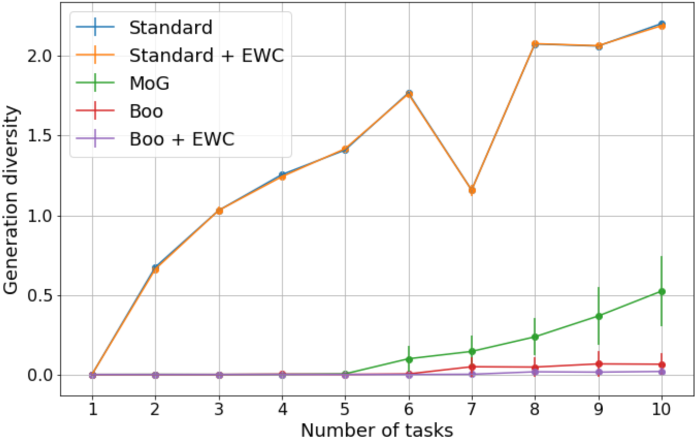
Table: NLL Results on fashion MNIST.

Diversity of generated images



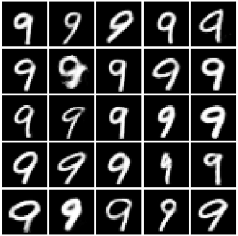
MNIST: KL divergence between uniform and generated distribution

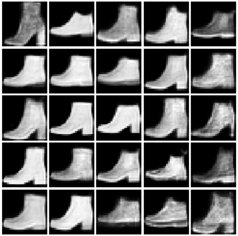
Diversity of generated images



fashion MNIST: KL divergence between uniform and generated distr.

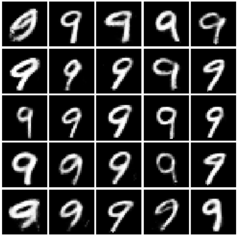
Diversity of generated images

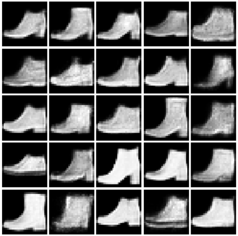




Standard prior: Samples after training on 10 tasks incrementally

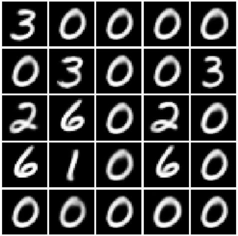
Diversity of generated images

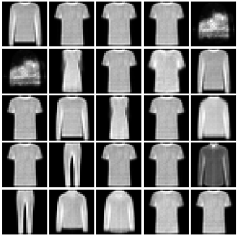




Standard + EWC: Samples after training on 10 tasks incrementally

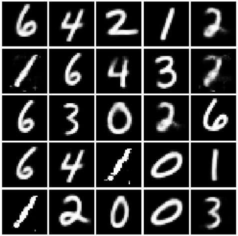
Diversity of generated images

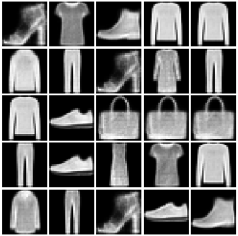




MoG: Samples after training on 10 tasks incrementally

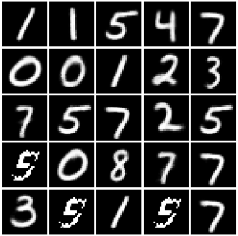
Diversity of generated images

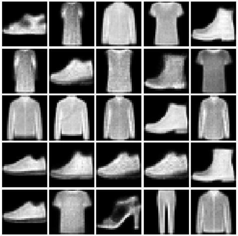




Boo: Samples after training on 10 tasks incrementally

Diversity of generated images





Boo + EWC: Samples after training on 10 tasks incrementally