Nonparametric Bayesian Deep Networks with Local Competition

Konstantinos P. Panousis

National and Kapodistrian University of Athens

kpanousis@di.uoa.gr

06/13/2019

Joint work with
Sotirios Chatzis and Sergios Theodoridis

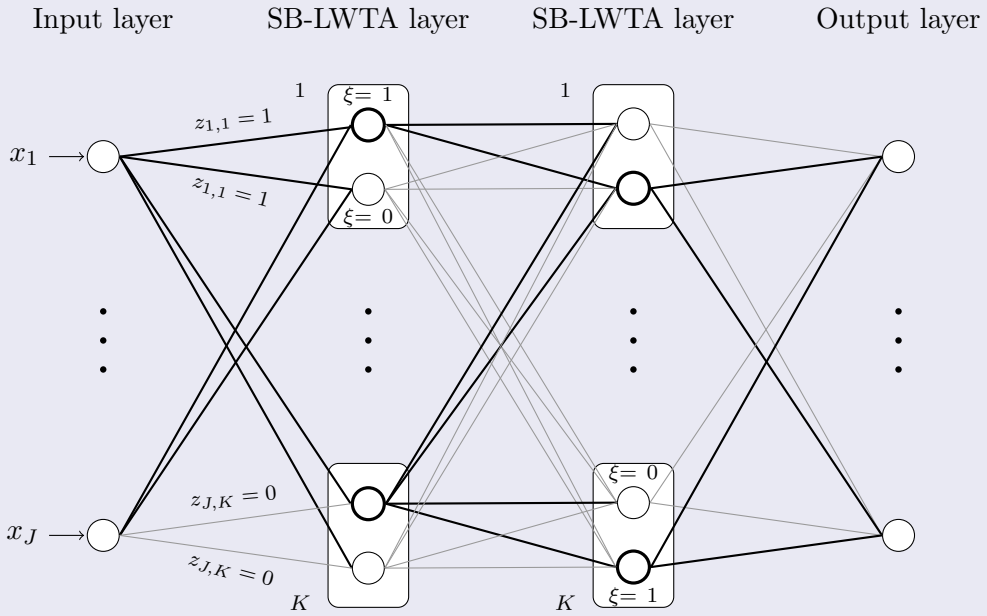
Overview

Approach

- Deep Neural Networks suffer from overparameterization:
  - Redundant parameters, overfitting tendencies and unnecessary computational costs.
- Goal: Attack the problem of redundancy.
- How: Infer the complexity of the architecture and the necessary bit precision, *concurrently with training*.
  - Use local competition between linear units as opposed to conventional non-linearities.
  - Let the data determine the needed set of network components: Non parametric Bayesian Inference.
- Indian Buffet Process + Local Competition + Bayesian inference + Bit precision reduction $\rightarrow$ Networks with reduced memory footprint.

Model Formulation

Feedforward Model



- Observations: $\{\boldsymbol{x}_n\}_{n=1}^{N} \in \mathbb{R}^J$
- The weights of all the synapses:
  $W:\ w_{j,k,u},\ j|_1^J, k|_1^K, u|_1^U,$
- The hidden variables $z_{j,k},\ j|_1^J,\ k|_1^K,$
  $Z:\ z_{j,k} \in \{0,1\}$, i.e., binary r.v.
- The latent variables $[\boldsymbol{\xi}_n]_k,\ n|_1^N,\ k|_1^K.$
  $[\boldsymbol{\xi}_n]_k \in$ one_hot(U).
- Layer Output:
  $[\boldsymbol{y}_n]_{ku} = [\boldsymbol{\xi}_n]_{ku} \sum_{j=1}^{J}(w_{j,k,u} \cdot z_{j,k}) \cdot [\boldsymbol{x}_n]_j$

Priors

- We impose suitable stick-breaking priors over the latent utility indicator variables $z_{j,k}$:

$$u_k \sim \text{Beta}(\alpha,1), \quad \pi_k = \prod_{i=1}^{k} u_i, \quad z_{j,k} \sim \text{Bernoulli}(\pi_k) \tag{1}$$

- A spherical prior over the weight matrices:

$$w_{j,k,u} \sim \mathcal{N}(0,1) \tag{2}$$

- A symmetric Discrete prior for the winning unit indicators:

$$[\boldsymbol{\xi}_n]_k \sim \text{Discrete}(1/U) \tag{3}$$

- Layer Output: $[\boldsymbol{y}_n]_k = [\boldsymbol{\xi}_n]_k \sum_{j=1}^{J} ([w_{j,k,u}]_{u=1}^{U} \cdot z_{j,k}) \cdot [\boldsymbol{x}_n]_j \in \mathbb{R}$

Posteriors

- We seek to infer the following posteriors:

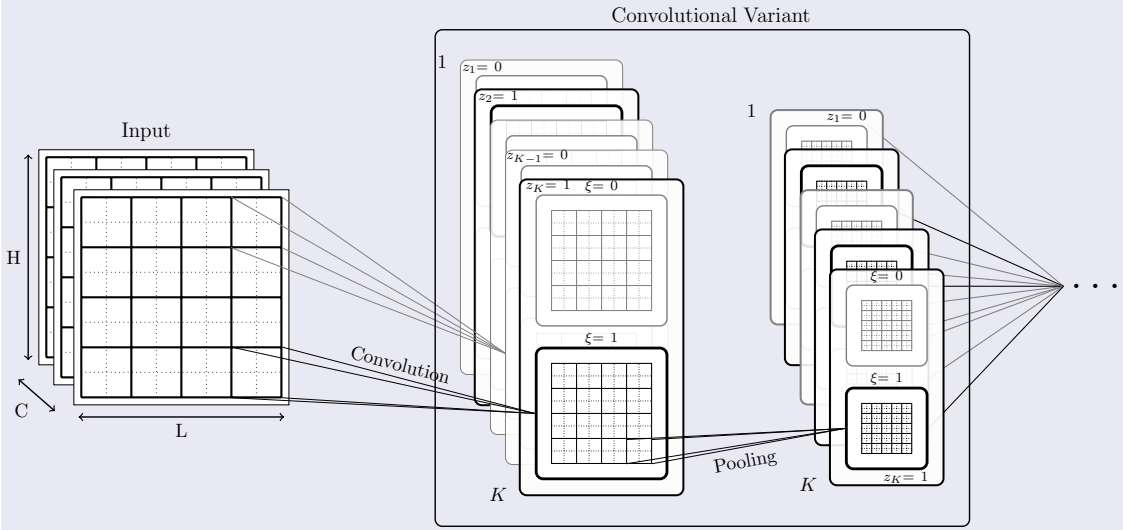
$$q(u_k) = \text{Beta}(u_k|a_k, b_k) \tag{4}$$

$$q(z_{j,k}) = \text{Bernoulli}(z_{j,k}|\tilde{\pi}_{j,k}) \tag{5}$$

$$q(\boldsymbol{W}) = \prod_{j,k,u} \mathcal{N}(w_{j,k,u}|\mu_{j,k,u},\ \sigma^2_{j,k,u}) \tag{6}$$

$$q([\boldsymbol{\xi}_n]_k) = \text{Discrete}\left([\boldsymbol{\xi}_n]_k \middle| \text{softmax}(\textstyle\sum_{j=1}^{J} [w_{j,k,u}]_{u=1}^{U} \cdot z_{j,k} \cdot [\boldsymbol{x}_n]_j)\right) \tag{7}$$

Convolutional Variant



- Input volumes (tensors):
  $\{\boldsymbol{X}\}_{n=1}^{N} \in \mathbb{R}^{H \times L \times C}$
- The kernel weights:
  $\boldsymbol{W}_k \in \mathbb{R}^{h \times l \times C \times U},\ k|_1^K.$
- The utility hidden variables:
  $z_k \in \{0, 1\},\ k|_1^K.$
- The latent indicator variables:
  $[\boldsymbol{\xi}_n]_k,\ n|_1^N,\ k|_1^K \in \text{one\_hot(U)}.$
- The output volumes per layer:
  $[\boldsymbol{Y}_n]_k = [\boldsymbol{\xi}_n]_k \Big((\boldsymbol{W}_k \cdot z_k) \star \boldsymbol{X}_n \Big),$
  $[\boldsymbol{Y}_n]_k \in \mathbb{R}^{H \times L \times U},\ k|_1^K.$

Convolutional Variant: Priors

- The corresponding priors are:
  - $u_k \sim \text{Beta}(\alpha, 1), \quad \pi_k = \prod_{i=1}^{k} u_i, \quad z_k \sim \text{Bernoulli}(\pi_k)$
  - analogous spherical priors and a symmetric Discrete priors are imposed on the networks weights and winning latent indicators respectively.

Convolutional Variant: Posteriors

$$q(u_k) = \text{Beta}(u_k|a_k, b_k) \tag{8}$$

$$q(z_k) = \text{Bernoulli}(z_k|\tilde{\pi}_k) \tag{9}$$

$$q(\boldsymbol{W}) = \prod_{h,l,c,k,u} \mathcal{N}(w_{h,l,c,k,u}|\mu_{h,l,c,k,u}, \sigma^2_{h,l,c,k,u}) \tag{10}$$

$$q([\boldsymbol{\xi}_n]_k) = \text{Discrete}\left([\boldsymbol{\xi}_n]_k \middle| \text{softmax}(\textstyle\sum_{h',l'} [z_k \boldsymbol{W}_k \star \boldsymbol{X}_n]_{h',l',u})\right) \tag{11}$$

Training

Variational Inference with the Reparameterization trick (Concrete Relaxations and the Kumaraswamy distribution).

Inference

- Component Omission:
  - We exploit the component utility indicators z.
  - We introduce a threshold τ.
  - Any component with inferred posterior $q(z)$ below the threshold is omitted from computation, i.e., when $\tilde{\pi}_{j,k} < \tau$.
- Weight Compression:
  - Use the mean of the weights posterior to assess which bits are significant.
  - This way, we remove bits that fluctuate too much under posterior uncertainty.

Experiments

- Considered Architectures: Feedforward LeNet-300-100, Convolutional LeNet-5, VGG-like.
- Outperformed state-of-the-art approaches providing lighter computational footprint with no compromise in accuracy.

Learned Architecture LeNet-300-100.

Method	Error (%)	# Weights	Bit precision
Original	1.6	$235K/30K/1K$	23/23/23
StructuredBP	1.7	$23,664/6,120/450$	23/23/23
Sparse-VD	1.92	$58,368/8,208/720$	8/11/14
BC-GHS	1.8	$26,746/1,204/140$	13/11/10
SB-ReLU	1.75	13.698/6.510/730	3/4/11
SB-LWTA (2 units)	**1.7**	$\mathbf{12,522/6,114/534}$	**2/3/11**
SB-LWTA (4 units)	1.75	$23,328/9,348/618$	2/3/12

Details in the poster session...