Fast Matrix Normalization for bilinear pooling using Rank-1 Update

Skoltech

NLA Final Project

A*

the team

Alsu Vakhitova

Skoltech, DS
Alsu.Vakhitova@skoltech.ru

Andreea Dogaru

Skoltech, DS
Andreea.Dogaru@skoltech.ru

Timotei Ardelean

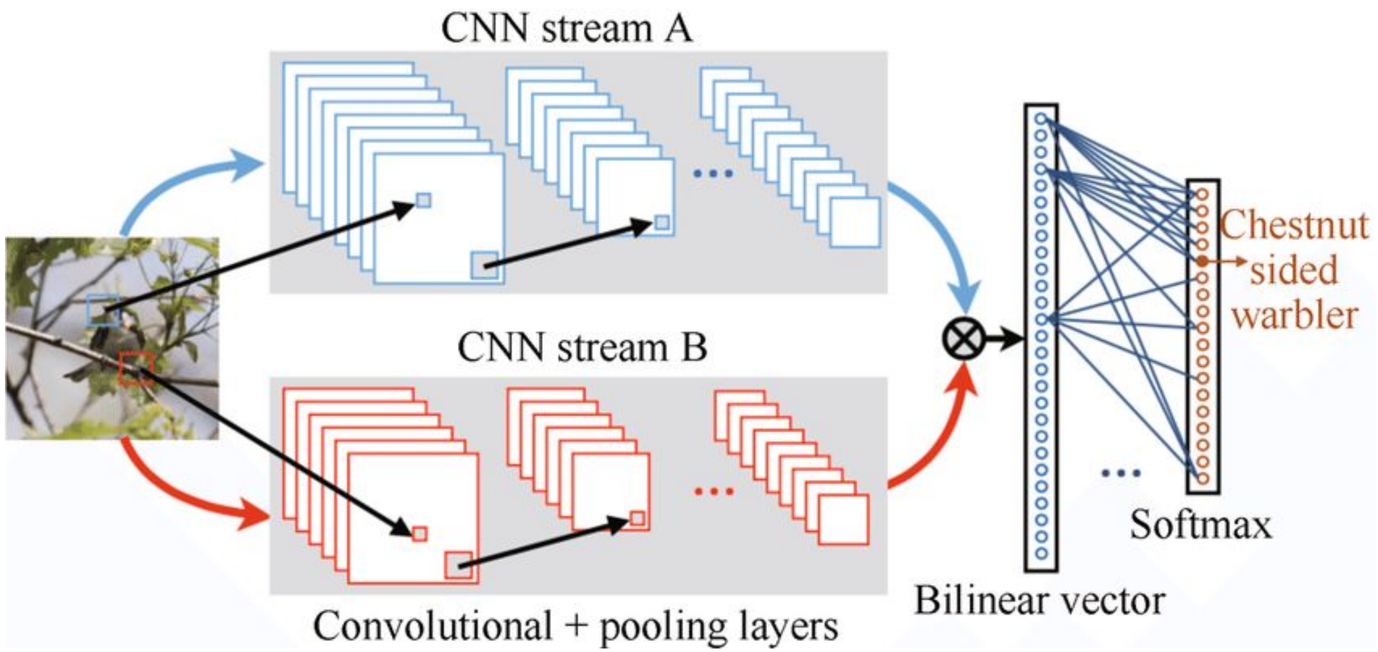
Skoltech, DS
Timotei.Ardelean@skoltech.ru

Skoltech

Overview

- Background
- Problem statement
- Rank-1 Update Normalization (RUN)
- Experiments & Results
- Conclusion

Background: Bilinear CNN



- Used in ***Fine-Grained Visual Recognition***
- FGVR is more challenging because the intra-category differences are small and other factors can be overwhelming

Skoltech

Background: Matrix Normalization

- Output of Bilinear Pooling - Bilinear Matrix
- Singular values of BM must be normalized
- Previous normalization methods:
  - SVD - bad on GPU, $O(D^3)$, no CBP support
  - Newton-Schulz (NS) iteration - good on GPU, $O(D^3)$, no CBP support

Skoltech

Background: Compact Bilinear Pooling

- Bilinear Matrix is vectorized before the classification layer
- The vector is very large - memory-intensive
- Efficient alternative is Compact Bilinear Pooling:
  - Approximates the Bilinear Matrix
  - Reduces the dimensionality by two orders of magnitude
  - Little-to-no performance loss
  - Tensor Sketch and Random Maclaurin

Skoltech

Problem statement

- Implement a better method (RUN) for normalizing the bilinear matrix, featuring:
    - Good complexity and efficient computation on GPU
    - Compatibility with Compact Bilinear Pooling
    - Based on power method

Skoltech

Rank-1 Update Normalization (RUN)

Algorithm 1: Rank-1 Update Normalization (RUN).

Input: Local features $\mathbf{F} \in \mathbb{R}^{N \times D}, \eta, K$.

Output: Normalized local features $\mathbf{F}_K$.

Generate $\mathbf{v_0} = [v_1, \ldots, v_D] \in \mathbb{R}^D$, where $\{v_i\}_{i=1}^D$ are i.i.d. normal distribution.

for $k \in [1, K]$ **do**

$\quad \mathbf{v}_k = \mathbf{F}^\top \mathbf{F} \mathbf{v}_{k-1}$

$\mathbf{F}_K = \mathbf{F} - \eta \frac{\mathbf{F}\mathbf{v}_K \mathbf{v}_K^\top}{\|\mathbf{v}_K\|_2^2}$

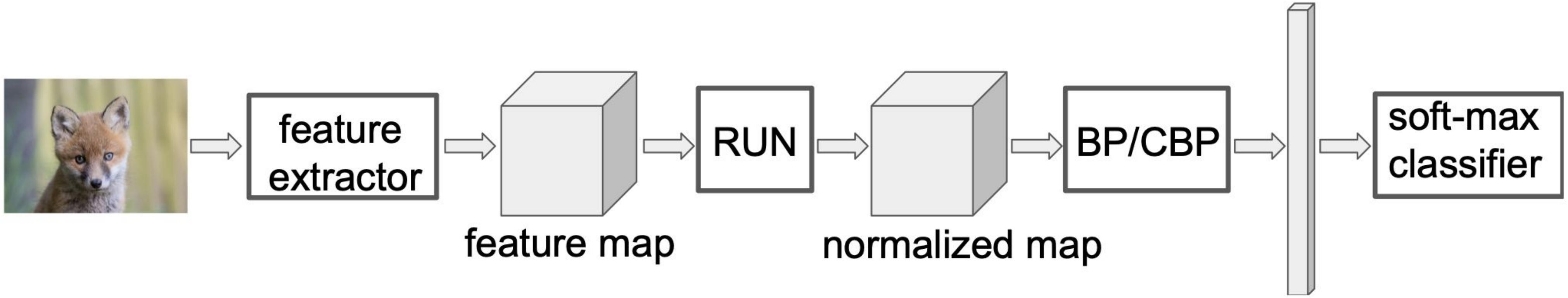
return $\boldsymbol{F}_K$

Rank-1 Update Normalization (RUN)

- Only two matrix-by-vector multiplications per iteration → O(KDN)
  - K - # iterations, D - depth
  - N=WH, W - width, H - height
  - All predecessors had complexity $O(D^3)$
    - D>>W,H,K → O(KDN) < $O(D^3)$
- Applied directly on feature map → compatible with CBP

Skoltech

Experiments: Setup



- Scaled input: 448×448×3
- Feature extractor: second-to-last layer of a pretrained VGG16
- Last convolutional feature map: 28 × 28 × 512

Skoltech

Experiments: Tasks and Datasets

Fine-grained Recognition

Caltech-UCSD Birds 200

200 bird species

Texture Recognition

Describable Textures Dataset

47 texture attributes

Scene Recognition

MIT Indoor Scenes

67 indoor scenarios

Results: Accuracy

Dataset	BCNN	iBCNN (NS)	RUN
CUB-200	84.45	84.92	85.47
DTD	70.85	72.45	71.01
MIT	77.76	78.88	80.30

Skoltech

Results: Time

Dataset	BCNN	iBCNN (NS)	RUN
CUB-200	5.678	100.605	9.579
DTD	2.42	143.62	5.693
MIT	4.613	100.640	9.707

- Time for matrix normalization, ms
- Measured during inference on the test dataset

Skoltech

Discussion of results

- We managed to reproduce the results leading to and including the development of RUN.
- Proper normalization of the Bilinear matrix proved to be indeed important in FGVR.
- RUN is significantly faster than iBCNN with NS while leading to similar or even superior accuracy.
- We found out that RUN works <u>better with uniform distribution initialization</u>, rather than normal

Skoltech

Conclusions

- Investigated BCNN and various matrix normalization approaches
- Implemented RUN algorithm as well as all other parts of the NN
- Reproduced most important parts of the paper
- Obtained expected results

Skoltech

thx.

Questions?

Skoltech