Course Notes: Deep Learning for Visual Computing

Peter Wonka

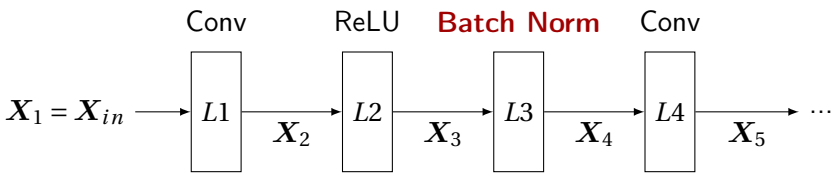
August 30, 2021

Contents

1 Network Normalization

1.1 Introduction to Normalization

- Idea: control statistics of the values in a tensor processed by the network, e.g. mean and variance
- Normalization is added as a layer (node) to the network
- Input and Output of the normalization layer are tensors of the same size
- Some normalization layers need access to a batch of samples (e.g. images) during training



1.2 Design Space for Designing Normalization for Vectors

- Input and output: rank 2 tensor, batch of vectors, $\boldsymbol{X} \in \mathbb{R}^{N \times D}$
  - $\boldsymbol{X}$ is the output of another layer
  - N - number of samples (e.g. images) in a batch / mini-batch
  - D - number of dimensions / number of values in a vector
  - $\boldsymbol{X}[i, j]$ is the jth entry of the vector decribing the ith sample

$$\boldsymbol{X} = \begin{bmatrix} — & \boldsymbol{X}[1,:] & — \\ — & \boldsymbol{X}[2,:] & — \\ & \vdots & \\ — & \boldsymbol{X}[N,:] & — \end{bmatrix} \in \mathbb{R}^{N \times D}$$

1	0	0	-9
1	-1	3	-5
3	1	1	7

- Multiple options to compute a statistic and then normalize, e.g. mean
- Option 1: Compute the mean for all features in a single sample
- Compute one value for each row of the data matrix
- E.g. compute the average pixel value per (vectorized) image

$$\mu = \begin{pmatrix} (1+0+0-9)/4 = -2 \\ -0.5 \\ 4 \end{pmatrix} \tag{1.1}$$

1	0	0	-9
1	-1	3	-5
3	1	1	7

- Option 2: Compute the mean for each feature across all samples
- Compute one value for each column of the data matrix (used in batch normalization)
- E.g. compute the average pixel value for each pixel position across the batch

$$\mu = \begin{pmatrix} (1+1+3)/4 = 5/3 \\ 0 \\ 4/3 \\ -7/3 \end{pmatrix} \tag{1.2}$$

1	0	0	-9
1	-1	3	-5
3	1	1	7

- Option 3: Compute a single mean for all pixels and all samples

$$\mu = (1+0+0-9+1-1+3-5+3+1+1+7)/12 = 1/6 \tag{1.3}$$

- Many other options, e.g.
  - Subsequent normalization like double centering in classical MDS
  - Normalization across groups of features
  - Many options for higher rank tensors that could be explored

1.3 Batch Normalization for Vectors

- Input: matrix $\boldsymbol{X} \in \mathbb{R}^{N \times D}$, N samples, each sample has D features
- Output: matrix $\boldsymbol{Y} \in \mathbb{R}^{N \times D}$
- Ideas:
  - First, normalize to 0 mean and 1 variance
  - Second, add a new mean and multiply with new standard deviation
  - Learnable parameters γ (new standard diviation) and β (new mean)
- Computation:
  - Per-feature mean, average N values, D values are computed, $\mu \in \mathbb{R}^D$:

$$\mu_j = \mu[j] = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{X}[i, j] \tag{1.4}$$

- Per-feature variance, D values are computed, $\sigma^2 \in \mathbb{R}^D$:

$$\sigma^2[j] = \frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{X}[i,j] - \mu[j]\right)^2 \tag{1.5}$$

- Normalize $\boldsymbol{X}$, $N \times D$ values are computed, small ϵ for robustness:

$$\hat{\boldsymbol{X}}[i,j] = \frac{\boldsymbol{X}[i,j] - \mu[j]}{\sqrt{\sigma^2[j] + \epsilon}} \tag{1.6}$$

- Shift and Rescale, $N \times D$ values are computed as matrix $\boldsymbol{Y}$:

$$\boldsymbol{Y}[i,j] = \gamma[j]\hat{\boldsymbol{X}}[i,j] + \beta[j] \tag{1.7}$$

- Requires one γ_j and one β_j per feature, $\gamma, \beta \in \mathbb{R}^D$
- Learning $\gamma = \sigma$ and $\beta = \mu$ will recover the identity function

1.4 Batch Normalization - Test Time

- Estimates for mean and variance depend on a mini-batch
- Cannot be computed at test time
- Computation during inference:
  - $\mu \in \mathbb{R}^D$ = (exponential moving) average of values seen during training
  - $\sigma^2 \in \mathbb{R}^D$ = (exponential moving) average of values seen during training
  - Normalize using Eq. 1.6
  - Shift and Rescale using Eq. 1.7
- Last two steps (Normalization, Shift and Rescale) are the same during test and training
- Note:
  - During testing batchnorm is a linear operator

- Can be fused with the previous fully connected or conv layer

1.5 Spatial Batch Normalization

- Terms: **Spatial Batchnorm**, **Spatial Batch Normalization**
- Python:

```
torch.nn.BatchNorm2d(num_features, eps=1e-05, momentum=0.1, affine=True,
    track_running_stats=True)}
```

  - num_features: C from an expected input of size (N, C, H, W)
  - eps: added to the denominator for numerical stability
  - momentum m: update a statistic with $\text{stat}_{\text{new}} = \text{stat}_{\text{old}} \times (1 - m) + \text{stat}_{\text{new}} \times m$
    - computed during training and used during testing
  - affine: a boolean value that when set to True, this module has learnable affine parameters

1.6 Batch Normalization - Vectors vs. Rank 3 Tensors

- Batch normalization for a **batch of vectors**
- Input: $\boldsymbol{X} : N \times D$
  - Aggregate over all samples N: $N \to 1$

$$\mu, \sigma : 1 \times D \tag{1.8}$$

$$\gamma, \beta : 1 \times D \tag{1.9}$$

$$\tag{1.10}$$

- Batch normalization for a **batch of rank 3 tensors** (e.g. images)
- Input: $\boldsymbol{X} : N \times C \times H \times W$
  - Aggregate over all samples N: $N \to 1$
  - Aggregate over all pixels per channel: $H, W \to 1$

$$\mu, \sigma : 1 \times C \times 1 \times 1 \tag{1.11}$$

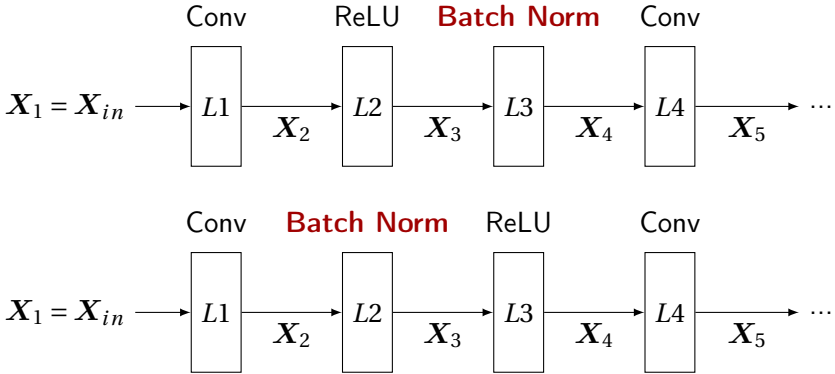
$$\gamma, \beta : 1 \times C \times 1 \times 1 \tag{1.12}$$

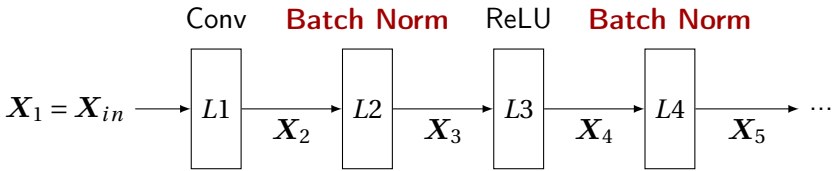
$$\tag{1.13}$$

1.7 Batch Normalization and Bias Terms

- Batch normalization makes the bias terms unnecessary

1.8 Where to put Batch Normalization?





- Originally:
  - batch normalization was inserted after the fully connected or convolutional layers
  - before the non-linearity
- Now:
  - different versions exist
  - batchnorm before and after the non-linear layers, typically not both

1.9 Batch Normalization Discussion

- Advantages:
  - Makes deep networks much easier to train!
  - Improves gradient flow
  - Allows higher learning rates, faster convergence
  - Network becomes more robust to initialization
  - Acts as regularization during training
  - Zero overhead at test-time: can be fused with conv
- Disadvantages:
  - Behaves differently during training and testing: common source of bugs!
  - Hard to parallelize
  - Not good for RNNs (recursive neural networks)

1.10 Layer Normalization

- Literature: Ba, Kiros, Hinton, Layer Normalization
- Idea:
  - Normalize over all features (dimensions) of a single input sample
  - Same behaviour at train and test time
  - Does not need a batch of samples
  - Useful for RNNs

- **Batch normalization** for batches of vectors, Input: $\boldsymbol{X} : N \times D$
  - Aggregate over all samples N: $N \to 1$

$$\mu, \sigma : 1 \times D \tag{1.14}$$

$$\gamma, \beta : 1 \times D \tag{1.15}$$

$$\tag{1.16}$$

- **Layer normalization** for batches of vectors, Input: $\boldsymbol{X} : N \times D$
  - Aggregate over all D input feature dimensions: $D \rightarrow 1$

$$\mu, \sigma : N \times 1 \tag{1.17}$$

$$\gamma, \beta : 1 \times D \tag{1.18}$$

$$\tag{1.19}$$

- Note: parameters γ and β are still learned per feature dimension!

1.11 Instance Normalization

- Literature: Ulyanov et al., Improved Texture Networks: Maximizing Quality and Diversity in Feed-forward Stylization and Texture Synthesis
- Idea:
  - Normalize over all values in a channel
  - Same behaviour at train and test time
  - No complication with parallelization

- **Batch normalization** for batches of rank 3 tensors, Input: $\boldsymbol{X} : N \times C \times H \times W$
  - Aggregate over all samples N: $N \to 1$
  - Aggregate over all pixels per channel: $H, W \to 1$

$$\mu, \sigma : 1 \times C \times 1 \times 1 \tag{1.20}$$

$$\gamma, \beta : 1 \times C \times 1 \times 1 \tag{1.21}$$

$$\tag{1.22}$$

- **Instance normalization** for batches of rank 3 tensors. Input: $\boldsymbol{X} : N \times C \times H \times W$
  - Aggregate over all pixels per channel: $H, W \rightarrow 1$

$$\mu, \sigma : N \times C \times 1 \times 1 \tag{1.23}$$

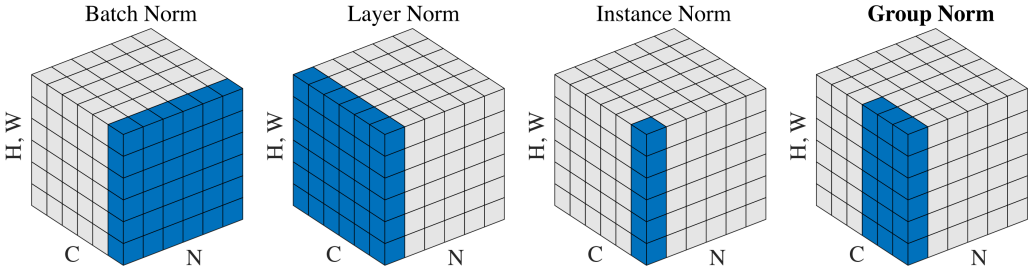
$$\gamma, \beta : 1 \times C \times 1 \times 1 \tag{1.24}$$

$$\tag{1.25}$$

- Do **not** aggregate over all samples N
- Parameters γ and β are learned per channel

1.12 Group Normalization and Comparison

- Literature: Wu and He, Group Normalization
- Visualization:
  - Each image shows a rank 4 tensor input to a normalization layer
  - N: samples in the mini-batch
  - C: channels in the layer axis
  - (H, W): pixels (only one spatial axis for W and H jointly)
  - Pixels in blue are used to estimate one mean and one variance value in the first part of the normalization.



- Needs a parameter number of groups (num_groups)
- Special Cases:
  - If $num\_groups = 1$, group norm is layer norm
  - If $num\_groups = C$, group norm is instance norm

1.13 Per-pixel normalization

- Literature: Karras et al., Progressive Growing of GANs for Improved Quality, Stability, and Variation
- Input: rank-3 tensor (image) $\boldsymbol{X} : C \times H \times W$
  - Compute the Euclidian length of each pixel and normalize the length to one
  - The rank-3 tensor has $H \times W$ pixels, each pixel has C dimensions / values
  - All $H \times W$ vectors $\boldsymbol{X}[:, i, j]$ have length 1 after per-pixel normalization

1.14 Spectral Normalization

- Literature: Miyato, Spectral Normalization for Generative Adversarial Networks

1.15 Batch Normalization and Distributed Computing

- Problem:
  - Some architectures require a large amount of memory for training, e.g. semantic image segmentation
  - Only a few images can be processed on a single GPU
  - **Batch size** on a single GPU is **small**
- Solution:
  - Requires a distributed implementation of batch normalization
  - **Synchronized Batch Normalization**
  - Behavior is not very intuitive
- Example result:

Batch size	BN size	# of GPUs	mmAP	Time(h)
16-base	0	8	36.2	33.2
2	2	2	31.5	131.2
4	4	4	34.9	91.4
8	8	8	35.9	71.5
16	2	8	31.0	45.6
16	16	8	37.0	39.5
32	32	8	37.3	45.5
64	64	8	35.3	40.9
64	32	16	37.1	19.6
64	16	32	37.1	11.2
128	32	32	37.1	11.3
128	16	64	37.0	6.5
256	32	64	37.1	7.2
256	16	128	37.1	**4.1**

1.16 Discussion

- Literature: Lipton and Steinhardt, Troubling Trends in Machine Learning Scholarship
  - Failure to distinguish between explanation and speculation
  - Failure to identify the sources of empirical gains, e.g. emphasizing unnecessary modifications to neural architectures when gains actually stem from hyperparameter tuning
  - Mathiness: the use of mathematics that obfuscates or impresses rather than clarifies, e.g. by confusing technical and non-technical concepts
  - Misuse of language, e.g. by choosing terms of art with colloquial connotations or by overloading established technical term
- The batch normalization paper is given as example for blurring the lines between **explanation and speculation** regarding the concept of **internal-covariate-shift**