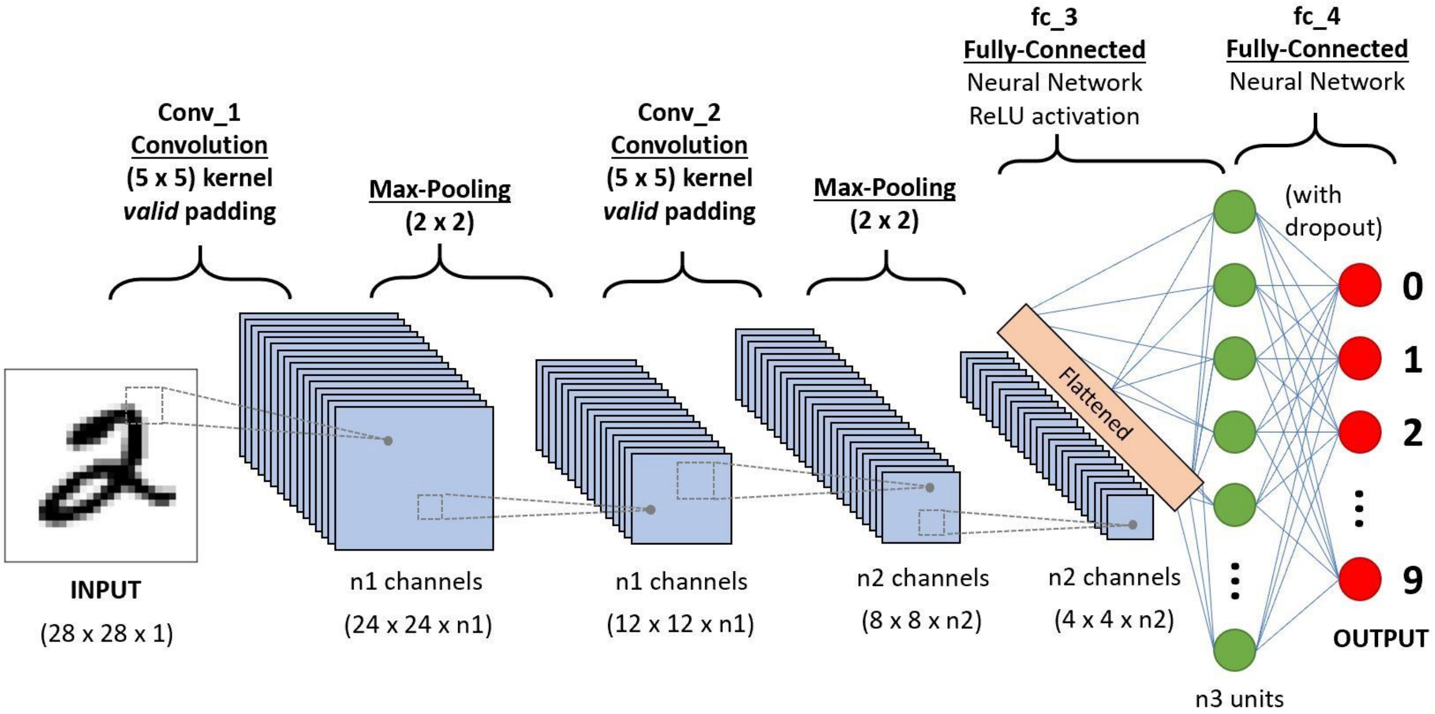
Restructuring Batch Normalization to Accelerate CNN training

By Jung, Wonkyung, Daejin Jung, Byeongho Kim, Sunjung Lee, Wonjong Rhee, and Jung Ho Ahn in *Proceedings of Machine Learning and Systems* 1 (2019): 14-26.

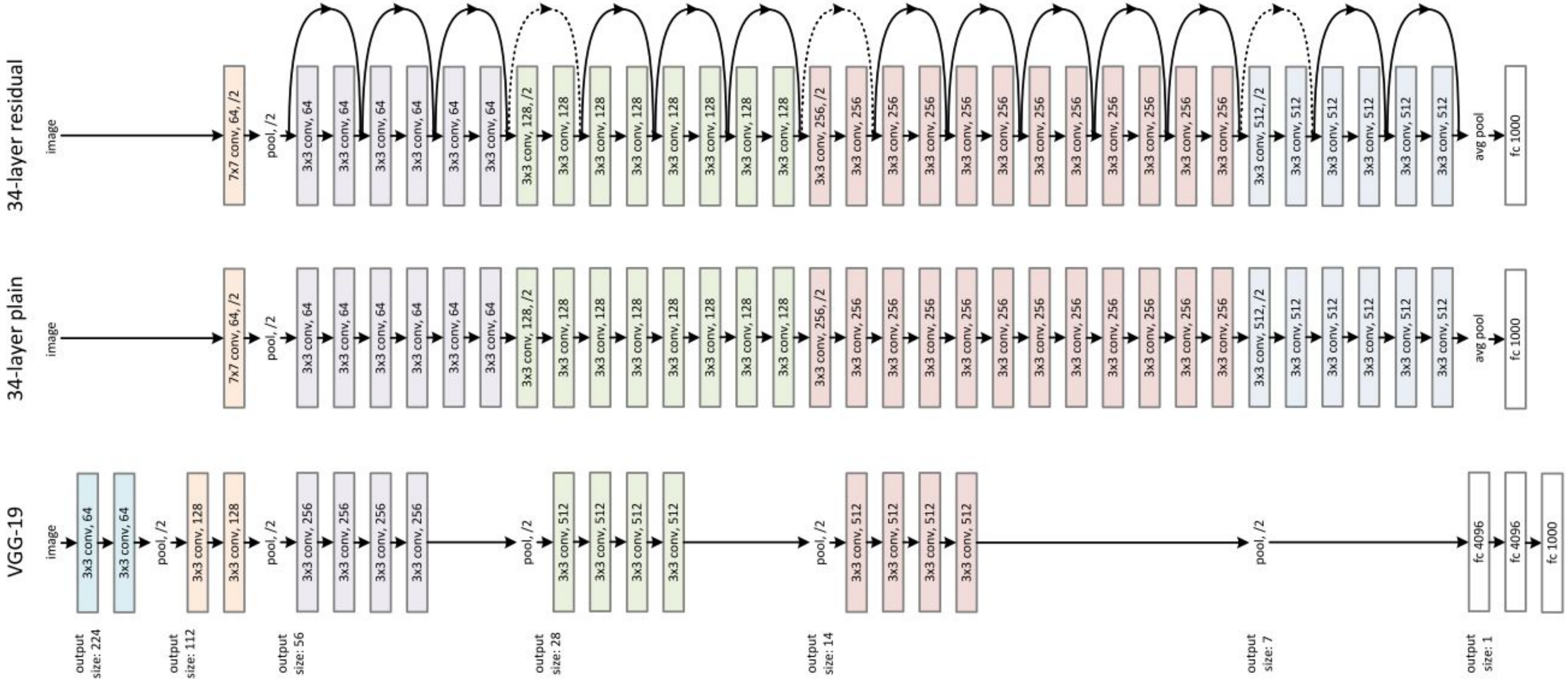
Presentation by: Saurav Sengupta

Core Principles: CNN



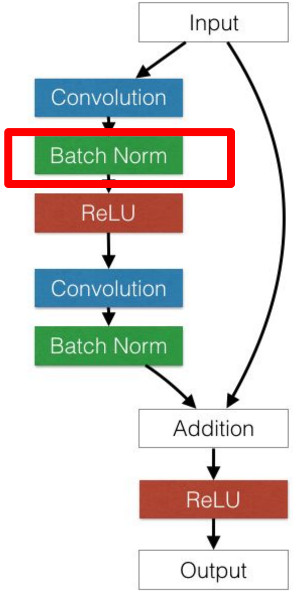
Ref: https://medium.com/analytics-vidhya/deep-residual-learning-for-image-recognition-resnet-94a9c71334c9

Residual Neural Networks (ResNets)



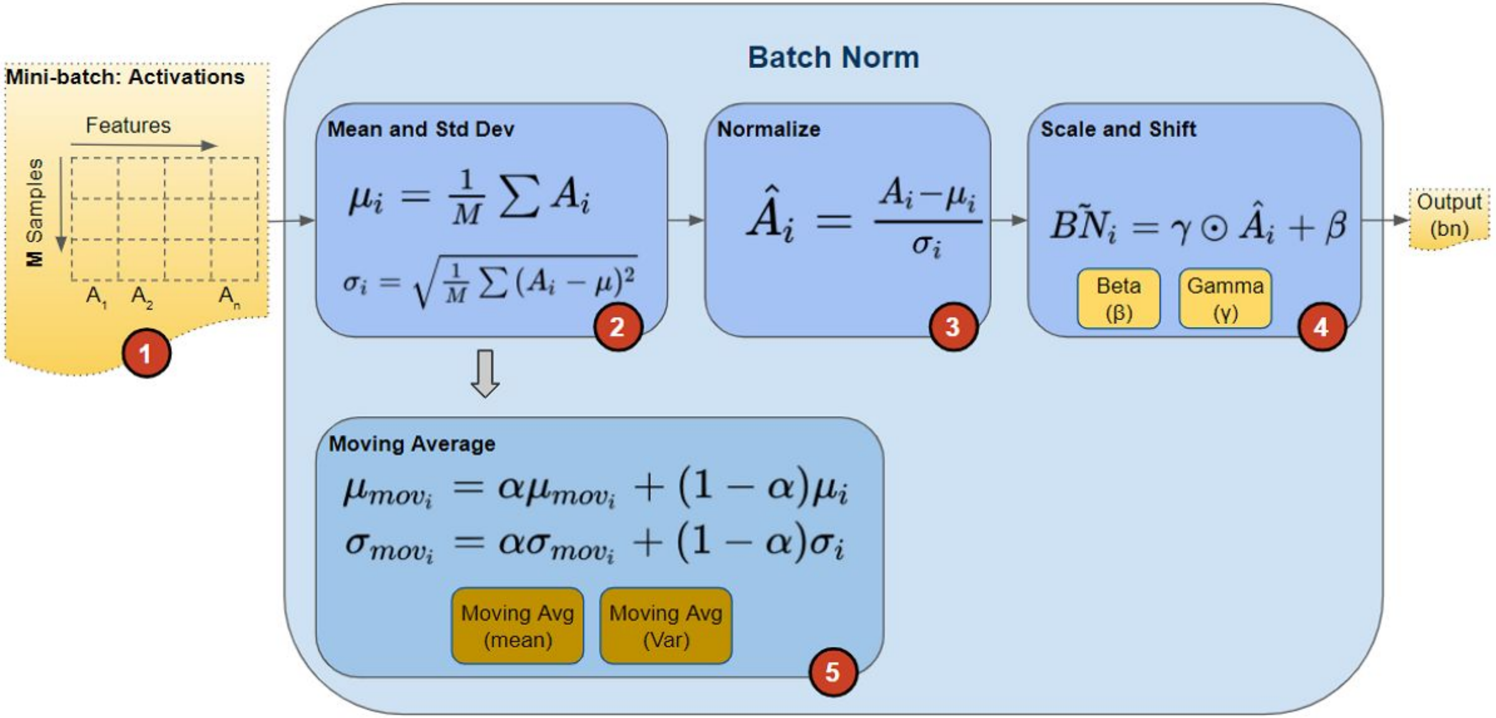
Ref: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

ResNets with Batch Norm



Ref: http://torch.ch/blog/2016/02/04/resnets.html

Batch Normalization



Mini-batch: Activations

Features

M Samples

A_1 A_2 A_n

1

Batch Norm

Mean and Std Dev

$$\mu_i = \frac{1}{M}\sum A_i$$

$$\sigma_i = \sqrt{\frac{1}{M}\sum(A_i - \mu)^2}$$

2

Normalize

$$\hat{A}_i = \frac{A_i - \mu_i}{\sigma_i}$$

3

Scale and Shift

$$B\tilde{N}_i = \gamma \odot \hat{A}_i + \beta$$

Beta (β)

Gamma (γ)

4

Output (bn)

Moving Average

$$\mu_{mov_i} = \alpha\mu_{mov_i} + (1-\alpha)\mu_i$$

$$\sigma_{mov_i} = \alpha\sigma_{mov_i} + (1-\alpha)\sigma_i$$

Moving Avg (mean)

Moving Avg (Var)

5

Ref: https://towardsdatascience.com/batch-norm-explained-visually-how-it-works-and-why-neural-networks-need-it-b18919692739

Why does batch norm work?

Nobody knows for sure.

But there are conjectures as to why it might work:

1. BN makes the optimization landscape significantly smoother
2. Batch normalization also helps avoid large gradient updates that can result in diverging loss and activations growing uncontrollably.

DenseNet

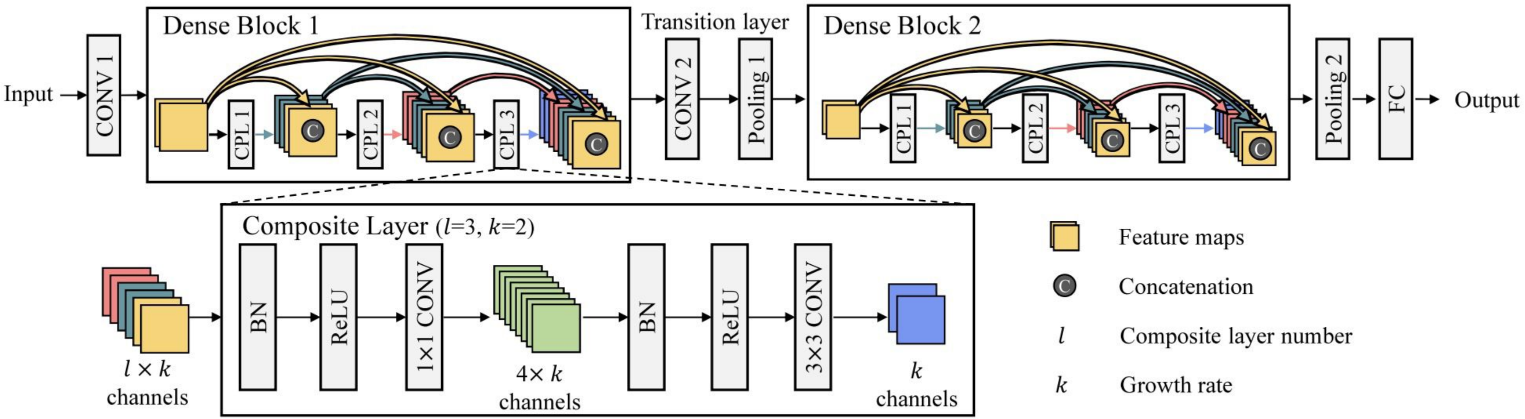


Figure 2. An exemplar DensetNet structure with two Dense Blocks, connected through a transition layer which changes the number and size of the input feature maps to a Dense Block. Each Dense Block has multiple Composite Layers (CPLs), each of which is connected to every other CPL within a Dense Block in a feed-forward fashion. A CPL consists of six layers (BN, ReLU, 1×1 CONV, BN, ReLU, and 3×3 CONV). The 1×1 CONV layer in a CPL, called bottleneck layer, limits the number of input feature maps to 4×k while the second CONV outputs k channels that are concatenated to the input feature maps. Growth rate (k) is the variable for how many feature maps are concatenated per CPL; feature maps stack up as they go through CPLs.

How is it different from ResNet?

- ResNet uses Element Wise Sum (EWS).
- DenseNets use concatenated feature maps.
- Less number of parameters to be optimized.
- Much more dense connectivity.

Different Optimization Techniques for accelerators

Two choices:

1. Optimize Convolution Layers
2. Optimize the non-Convolution layers

Optimize Convolution layers by data reuse techniques for weights and input and output feature maps (ifmap and ofmaps), or pruning weights or adapting new memory technologies.

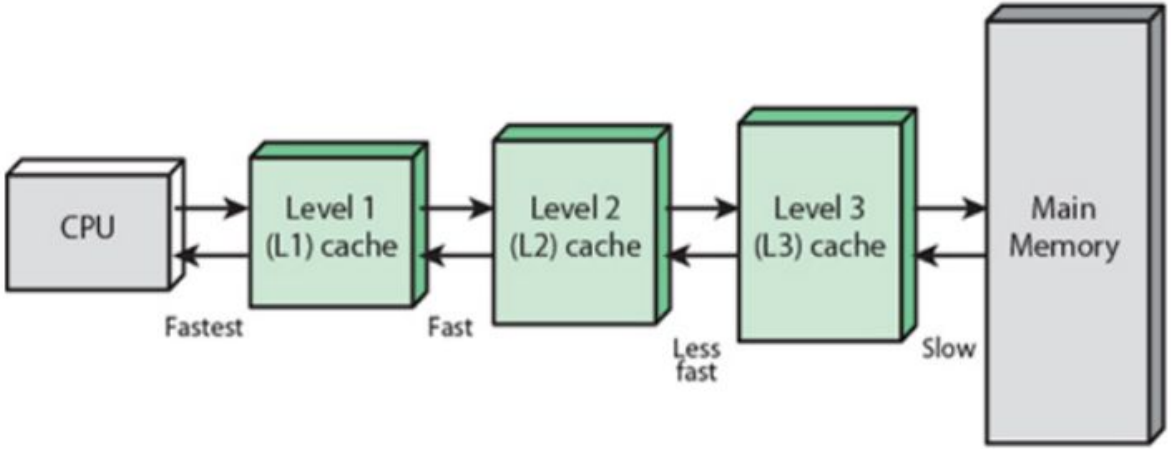
Most research has not focused on the non-CONV layers that most of the time outnumber the CONV layers.

DenseNet-121 (DenseNet with 120 CONV layers plus one FC layer) spends more than half of the execution time on the non-CONV layers.

Key Problems

- CONV layers with smaller filter sizes have relatively high memory access rates compared to the total amount of computation. DenseNet has high number of these small filter sized CONV layers.
- non-CONV layers of DenseNet-121 are mostly bottlenecked by the peak main memory bandwidth of the system we use.
- CONV layers underutilize the available bandwidth.
- Memory bandwidths are not high enough to completely utilize all the computational resources.

Interlude: How memory works

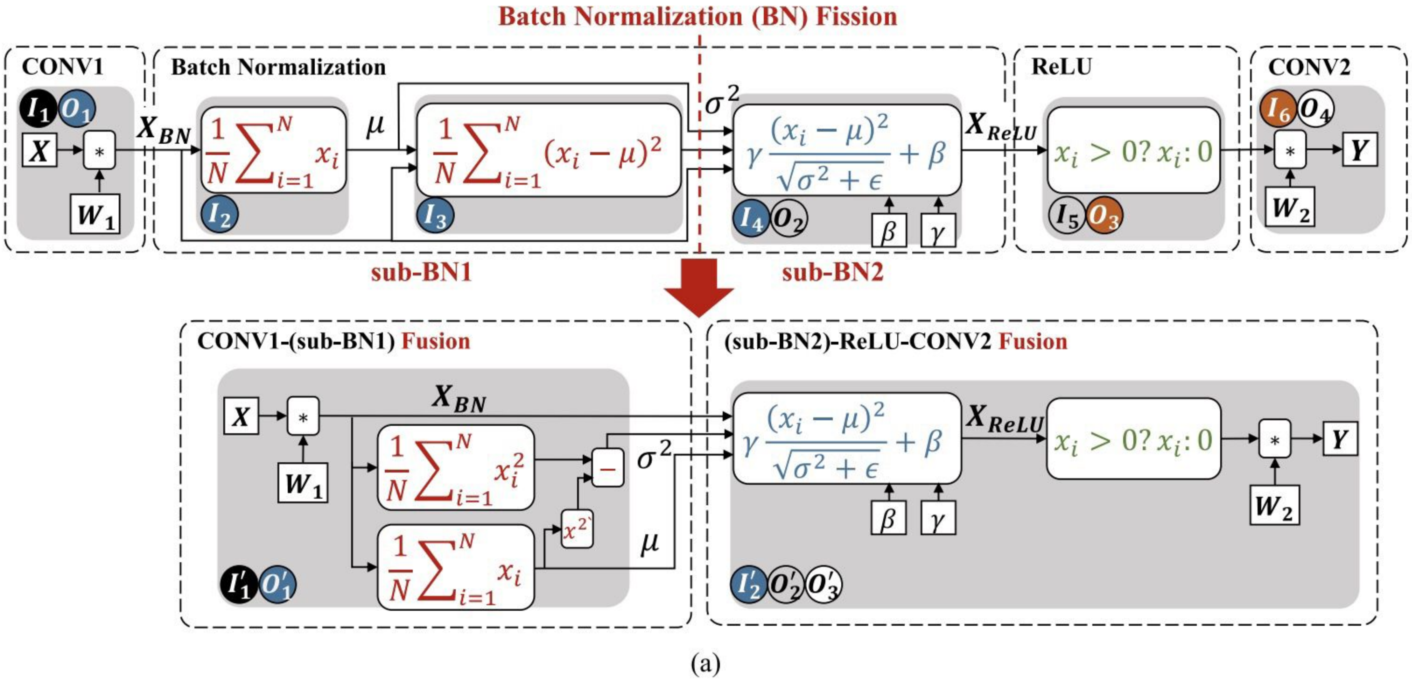


Memory bandwidth: Rate at which data can be read from memory by the processor.

Key Problems: Contd.

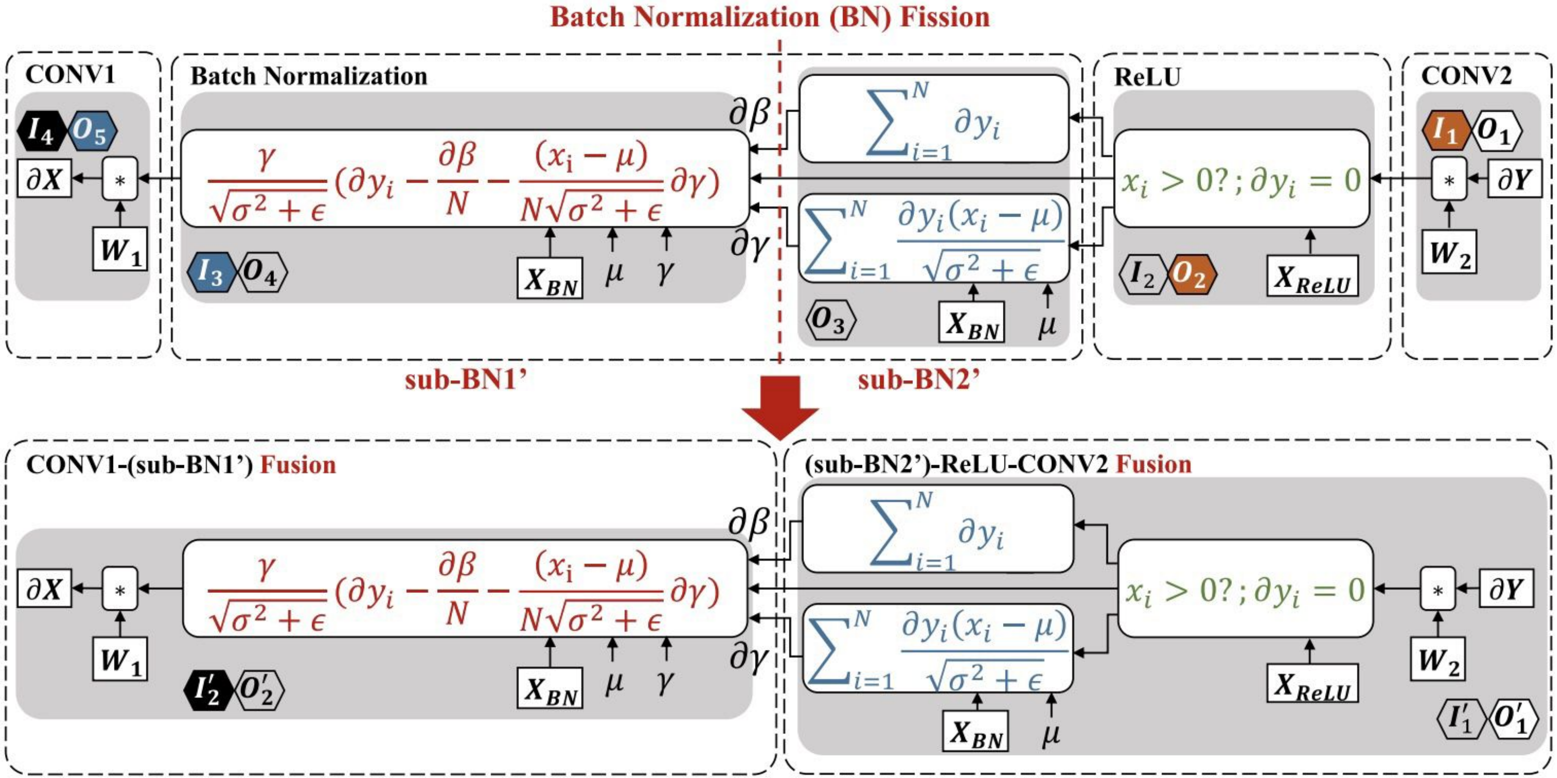
- Data reuse techniques do not work with Batch Norm layers.
- BN layers have strict data dependency, cross-layer data reuse is prohibited.

Solution: Fission and Fusion (Forward)



$$V(X) = E((X - E(X))^2) = E(X^2) - E(X)^2$$

Solution: Fission and Fusion (Backward)


Batch Normalization (BN) Fission
CONV1
I_4 O_5
∂X
*
W_1
Batch Normalization
$\frac{\gamma}{\sqrt{\sigma^2+\epsilon}}(\partial y_i - \frac{\partial \beta}{N} - \frac{(x_i-\mu)}{N\sqrt{\sigma^2+\epsilon}}\partial\gamma)$
I_3 O_4
X_{BN} μ γ
$\partial\beta$
$\partial\gamma$
$\sum_{i=1}^{N}\partial y_i$
$\sum_{i=1}^{N}\frac{\partial y_i(x_i-\mu)}{\sqrt{\sigma^2+\epsilon}}$
O_3
X_{BN} μ
ReLU
$x_i > 0?;\partial y_i = 0$
I_2 O_2
X_{ReLU}
CONV2
I_1 O_1
*
∂Y
W_2
sub-BN1'
sub-BN2'
CONV1-(sub-BN1') Fusion
∂X
*
W_1
$\frac{\gamma}{\sqrt{\sigma^2+\epsilon}}(\partial y_i - \frac{\partial \beta}{N} - \frac{(x_i-\mu)}{N\sqrt{\sigma^2+\epsilon}}\partial\gamma)$
X_{BN} μ γ
I'_2 O'_2
$\partial\beta$
$\partial\gamma$
(sub-BN2')-ReLU-CONV2 Fusion
$\sum_{i=1}^{N}\partial y_i$
$\sum_{i=1}^{N}\frac{\partial y_i(x_i-\mu)}{\sqrt{\sigma^2+\epsilon}}$
X_{BN} μ
$x_i > 0?;\partial y_i = 0$
X_{ReLU}
*
∂Y
W_2
I'_1 O'_1

Results

- BNFF removes five memory sweeps per BN layer.

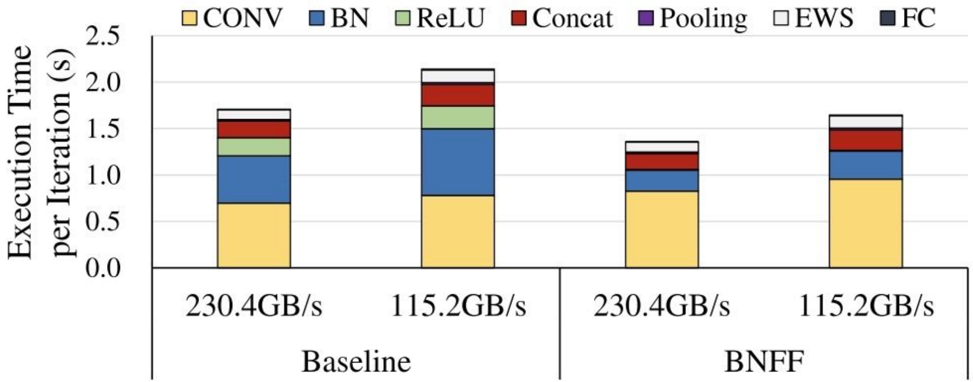


Figure 8. Execution time comparison of baseline and BNFF for 230.4GB/s and 115.2GB/s memory bandwidth using the Xeon processor with DenseNet-121.

Results (Contd.)

“Experiments on a latest chip multiprocessor showed that the proposed BN restructuring can improve the training performance of DenseNet-121 by 47.9% for forward pass and by 15.4% for backward pass, leading to overall **25.7%** improvement. Applying the BN restructuring to GPU with an open-source linear algebra library also showed **17.4%** of performance improvement. The large improvement suggests that non-CONV layers are important candidates for acceleration and that future research should pay keen attention to how CNN models evolve”

Limitations

- Only works on the various combinations of Conv+BatchNorm and BatchNorm+ReLU+Conv so works on DenseNets or ResNet but might not work on other implementations that do not have those sequences.
- Does not reduce the timing of backpropagation significantly because the number of reads and writes remain mostly the same.