High-Performance Large-Scale Image Recognition Without Normalization

Andrew Brock, Soham De, Samuel L. Smith, Karen Simonyan

Presented by Salvatore Romano
MSc in Data Science for Management
Neural Computing course by Prof. Sebastiano Battiato
University of Catania

NFNets actually is the best CNN configuration for **ImageNet**, with a score of 89.2%.

- **NFNets** (*Normalizer-FreeResNets*) family
- **Batch Normalization** problem and solution
- **Adaptive gradient clipping** concept
- **NFNets** architecture
- **Augmentation** used
- Results' evaluation

Batch Normalization

Definition and formula

Batch normalization is a technique for training very deep neural networks that standardizes the inputs to a layer for each mini-batch. How does it works?

1. Batch mean

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{1}$$

2. Batch variance

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_B)^2 \tag{2}$$

3. Normalization of the layer inputs

$$\overline{x_i} = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{3}$$

4. Scaling and shifting the normalized input

$$y_i = \gamma \overline{x_i} + \beta \tag{4}$$

Batch Normalization

Pros

- It downscales the residual branch
- It eliminates mean-shift
- It has a regularizing effect
- It allows efficient large-batch training

Cons

- it is a surprisingly expensive computational primitive
- it introduces a discrepancy between the behaviour of the model during training and at inference time
- it breaks the independence between training examples in the minibatch
- other limitations

Towards Removing Batch Normalization

Goal

In order to make ResNet normalizer-free, it is crucial to suppress the scale of the activations on the residual branch. To achieve this, the authors implemented the following solutions.

α and β as scalers

NFNets uses 2 scalers, α and β, to scale the activations at the start and end of the residual branch; α is set to a small constant of 0.2, while β for each block is defined as $\beta_i = \sqrt{\mathrm{Var}(h_i)}$, where $\mathrm{Var}(h_{i+1}) = \mathrm{Var}(h_i) + \alpha^2$.

Scaled Weight Standardization

NFNets uses **Scaled Weight Standardization** to prevent *mean-shift* in the hidden activations. This technique normalizes the weights of the convolutional layers such that:

$$\hat{W}_{ij} = \frac{W_{ij} - \mu_i}{\sqrt{N}\sigma_i} \tag{5}$$

Towards Removing Batch Normalization

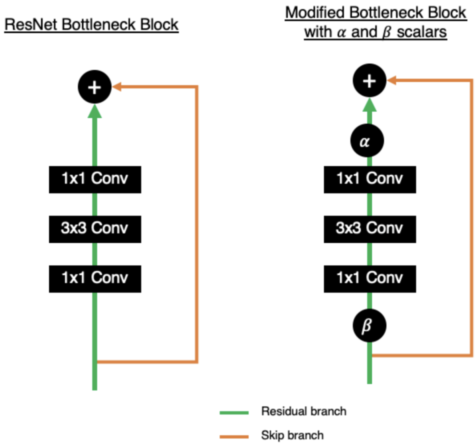

Figure: Introduction of α and β as scalers

Gradient Clipping

Definition and formula

Gradient clipping is a technique that tackles exploding gradients. If the gradient gets too large, we rescale it to keep it small. For the gradient vector $G = \partial L / \partial \theta$, the standard clipping algorithm clips the gradient before updating the parameter θ such that:

$$G \to \begin{cases} \lambda \frac{G}{\|G\|} & \text{if } \|G\| > \lambda \\ G & \text{otherwise} \end{cases} \tag{6}$$

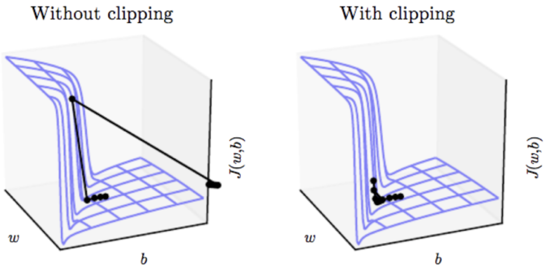

Figure: Parameters' behaviour without and with clipping

Adaptive Gradient Clipping

Problem of Gradient Clipping

The training stability was extremely sensitive to the choice of hyper-parameters λ, requiring fine-grained tuning when varying the model depth, the batch size, or the learning rate.

Definition and formula

To overcome this issue, has been introduced in the model the **AGC** (*Adaptive Gradient Clipping*).The *AGC* is given such that each unit i of the gradient of the ℓ-th layer G_i^ℓ (defined as the i^{th} row of matrix G^ℓ) is clipped as:

$$G_i^\ell \rightarrow \begin{cases} \lambda \frac{\|W_i^\ell\|_F^*}{\|G_i^\ell\|_F} G_i^\ell & \text{if } \frac{\|G_i^\ell\|_F}{\|W_i^\ell\|_F^\star} > \lambda \\ G_i^\ell & \textit{otherwise.} \end{cases} \tag{7}$$

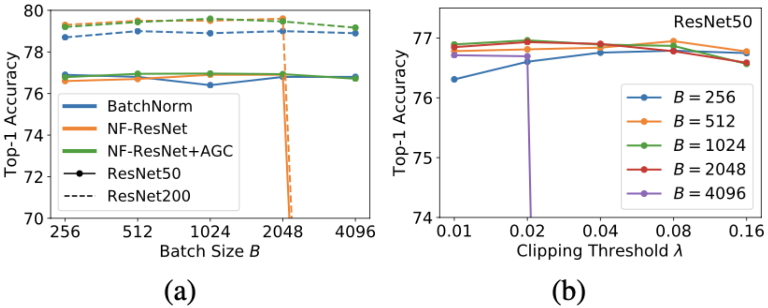

Figure: (a) AGC efficiently scales NF-ResNets to larger batch sizes. (b) The performance across different clipping thresholds λ.

NF-Net is a modified version of **SE-ResNeXt-D**

Stage	SE-ResNeXt-50	NFNet-F0
Stem	$\begin{bmatrix} \text{conv, 7x7, 64} \\ \text{max pool, 3x3} \end{bmatrix}$	$\begin{bmatrix} \text{conv, 3x3, 16} \\ \text{conv, 3x3, 32} \\ \text{conv, 3x3, 64} \\ \text{conv, 3x3, 128} \end{bmatrix}$
Conv Blocks 1	$\begin{bmatrix} \text{conv, 1x1, 128} & \\ \text{conv, 3x3, 128} & C = 32 \\ \text{conv, 1x1, 256} & \\ \text{SE} & \end{bmatrix} \times 3$	$\begin{bmatrix} \text{conv, 1x1, 128} & \\ \text{conv, 3x3, 128} & \\ \text{conv, 3x3, 128} & C = 1 \\ \text{conv, 1x1, 256} & \\ \text{SE} & \end{bmatrix} \times 1$
Conv Blocks 2	$\begin{bmatrix} \text{conv, 1x1, 256} & \\ \text{conv, 3x3, 256} & C = 32 \\ \text{conv, 1x1, 512} & \\ \text{SE} & \end{bmatrix} \times 4$	$\begin{bmatrix} \text{conv, 1x1, 256} & \\ \text{conv, 3x3, 256} & \\ \text{conv, 3x3, 256} & C = 2 \\ \text{conv, 1x1, 512} & \\ \text{SE} & \end{bmatrix} \times 2$
Conv Blocks 3	$\begin{bmatrix} \text{conv, 1x1, 512} & \\ \text{conv, 3x3, 512} & C = 32 \\ \text{conv, 1x1, 1024} & \\ \text{SE} & \end{bmatrix} \times 6$	$\begin{bmatrix} \text{conv, 1x1, 768} & \\ \text{conv, 3x3, 768} & \\ \text{conv, 3x3, 768} & C = 6 \\ \text{conv, 1x1, 1536} & \\ \text{SE} & \end{bmatrix} \times 6$
Conv Blocks 4	$\begin{bmatrix} \text{conv, 1x1, 1024} & \\ \text{conv, 3x3, 1024} & C = 32 \\ \text{conv, 1x1, 2048} & \\ \text{SE} & \end{bmatrix} \times 3$	$\begin{bmatrix} \text{conv, 1x1, 768} & \\ \text{conv, 3x3, 768} & \\ \text{conv, 3x3, 768} & C = 6 \\ \text{conv, 1x1, 1536} & \\ \text{SE} & \end{bmatrix} \times 3$
Fully Connected	Average pool, 100-d fc, softmax	

NFNet Architercure

- Activation function used is **GELU** (*Gaussian Error Linear Units*)
- All blocks employ the pre-activation **ResNe(X)t** bottleneck pattern with an added $3 \cdot 3$ grouped convolution inside the bottleneck
- All convolutions employ *Scaled Weight Standardization* to prevent the emergence of a mean-shift

Variant	Depth	Dropout	Train	Test
F0	$[1, 2, 6, 3]$	0.2	192px	256px
F1	$[2, 4, 12, 6]$	0.3	224px	320px
F2	$[3, 6, 18, 9]$	0.4	256px	352px
F3	$[4, 8, 24, 12]$	0.4	320px	416px
F4	$[5, 10, 30, 15]$	0.5	384px	512px
F5	$[6, 12, 36, 18]$	0.5	416px	544px
F6	$[7, 14, 42, 21]$	0.5	448px	576px

Table: NFNet family depths, drop rates, and input resolutions

Training Details

- *Softmax cross-entropy loss* with label smoothing of 0.1
- *Stochastic gradient descent* with **Nesterov's momentum** 0.9
- *Weight decay coefficient* of $2 \cdot 10^{-5}$
- *Learning rate* warms up from 0 to its maximal value that is chosen as $0.1 \cdot B/256$
- $\lambda = 0.01$ and $\epsilon = 10^{-3}$ for every parameter except the FC weight of the linear classifier layer

Methods' application

The application of *RandAugment* is after applying *MixUp* or *CutMix* and it is applied to 4 layers; the combination of these methods results in an intense level of augmentation which progressively benefits **NFNets**.

- *RandAugment* is used for all the images in a batch
- *MixUp* is applied to half the images in a batch with $\alpha = 0.2$
- *CutMix* is applied to the other half of the images in the batch

	F0	F1	F2	F3
Baseline	80.4	81.7	82.0	82.3
+ Modified Width	80.9	81.8	82.0	82.3
+ Second Conv	81.3	82.2	82.4	82.7
+ MixUp	82.2	82.9	83.1	83.5
+ RandAugment	83.2	84.6	84.8	85.0
+ CutMix	83.6	84.7	85.1	85.7
Default Width + Augs	83.1	84.5	85.0	85.5

Table: The effect of architectural modifications and data augmentation on ImageNet Top-1 accuracy

Results

Model	#FLOPS	#Params	ImageNet Top-1	TPUv3-core-days
NFNet-F4+ (ours)	367B	527M	89.2	1.86k
NFNet-F4 (ours)	215B	316M	89.2	3.7k
EffNet-L2 + Meta Pseudo Labels	-	480M	**90.2**	22.5k
EffNet-L2 + NoisyStudent + SAM	-	480M	88.6	12.3k
ViT-H/14	-	632M	88.55 ± 0.04	2.5k
ViT-L/16	-	307M	87.76 ± 0.03	0.68k
BiT-L ResNet152x4	-	928M	87.54 ± 0.02	9.9k
ResNeXt-101 32x48d (IG-940M)	-	829M	86.4	-

Example

- It is possible to run an example of pre-trained NFNet classifier, at the following link:

https://colab.research.google.com/github/deepmind/deepmind-research/blob/master/nfnets/nfnet_demo_colab.ipynb#scrollTo=qeotZfkBYrIg

References

References

High-Performance Large-Scale Image Recognition Without Normalization, by Andrew Brock, Soham De, Samuel L. Smith and Karen Simonyan.

THANKS FOR YOUR ATTENTION!