Modern CNN Architectures and Training Methods

Emrecan Ilik

***Abstract*—After the invention of convolutional neural networks (CNNs) many novelty of ideas on architecture and training methods of the CNNs came into effect. Improvements in architecture and training can enhance performance of the CNNs. This paper focuses on novelties in architecture and training methods which came into view over the last three years.**

I. Introduction

Zhuang Liu et al. 2022 propose a guideline to modernize a ConvNet (convolutional neural network). They introduce the idea of remodelling a ConvNet in a way that it possesses characteristics of a transformer, which improves performance. They build the architectural improvement on a residual neural network (ResNet) that gives way to the ConvNeXt. Figure 1 depicts the performance wise comparison of the ConvNext and transformers, namely Swin Transformer and DeiT.

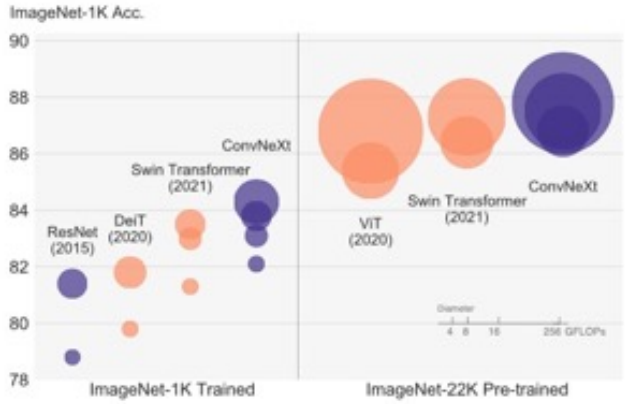


Fig. 1: ConvNeXt and Transformers Comparison - Zhuang Liu et al. 2022

As the figure shows, having improved the architecture, ConvNeXt can outperform transformers in image classification. Thus, Chapter 1 discusses architectural improvements. Furthermore, Brock et al. 2021 introduce normalizer-free residual neural networks (Normalizer-Free ResNets). They discuss batch normalization and its removal. In Chapter 2, normalizer-free ResNets are explained. Chapter 3 includes data augmentation techniques.

II. Chapter 1 - Architectural Improvements

Ze Liu et al. 2021 point out that Swin Transformers and Vision Transformers employ different strategies to build feature maps. Figure 2 represents those two strategies.

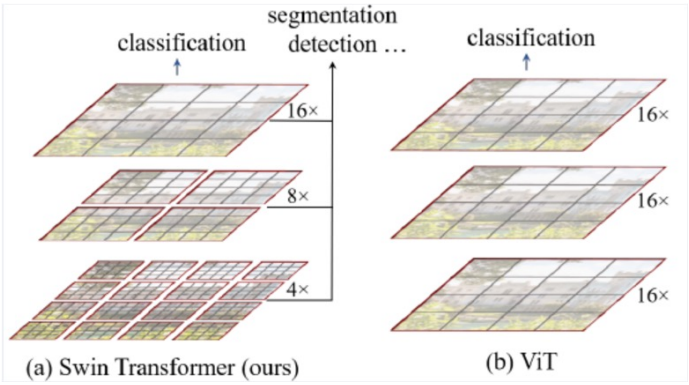


Fig. 2: "Patchify" Strategies of Swin and Vision Transformers - Ze Liu et al. 2021

Vision Transformers decompose an input image into 16x16 patches. Hence, each patch contains an information of size 16x16 of the input image. And, uniformity is preserved in every stage. That may create problems in detailed processing of the input image. By contrast, Swin Transformers use a different strategy to tackle this issue. Swin Transformers decompose the input image into smaller pixels and merge them into bigger pixels in deeper layers. First, the original image gets decomposed into smaller pixels of size 4x4. Then windows get merged in following stages (Ze Liu et al. 2021). If this characteristics of Swin Transformers is applied to ResNet with 4x4 patchify layer and stride 4, then accuracy is improved from 79.4 percent to 79.5 percent (Zhuang Liu et al. 2022).

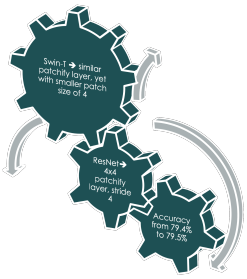


Fig. 3: Improvement in accuracy - Zhuang Liu et al. 2022

Dept-wise convolution helps to increase performance of the ResNet. The figure below illustrates the depth-wise convolution.

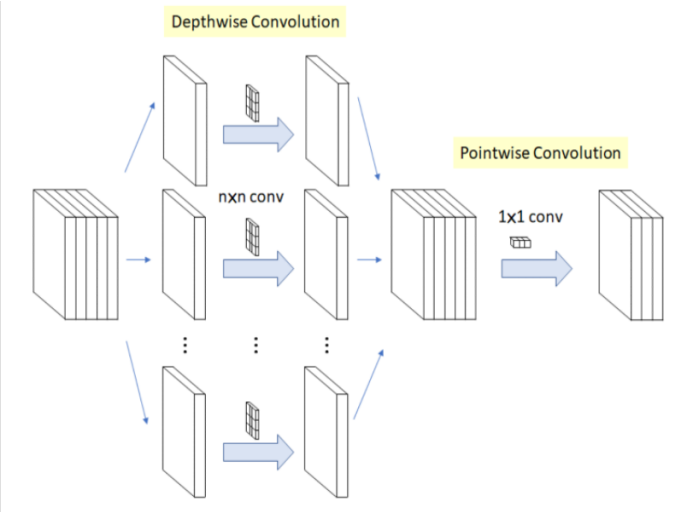


Fig. 4: Depth-wise Convolution. Source: Tsang 2018

In a standard convolution, width component of the filter and activation volume have the same length. Dept-wise convolution is a special type of convolution group convolution, in which number groups is equal to number of channels (Zhuang Liu et al. 2022. Whereas, in a depth-wise convolution a filter is applied to each channel. And, width component of the filter is a single dimension. If depth-wise convolution is applied to ResNet, then performance increases to 80.5 percent (Zhuang Liu et al. 2022).

Another important topic that should be explained is design of inverted bottleneck layer. Zhuang Liu et al. 2022 point out that transformer blocks create inverted bottleneck. The figure below depicts how inverted bottleneck design can be implement to ResNets.

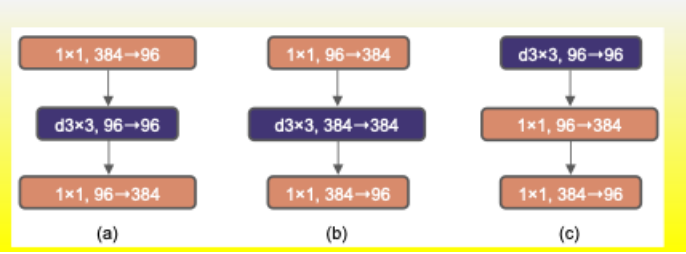


Fig. 5: Inverted Bottleneck Design- Zhuang Liu et al. 2022

Part(a) represents a ResNeXt block in which pixels get scaled down by 1x1 convolution, and then a 3x3 convolution is applied, in the next step pixels get mapped up by a 1x1 convolution. In part(b) an inverted bottleneck block is created. And, in part(c) position of the depth-wise convolution layer is moved up (Zhuang Liu et al. 2022). That design provides further improvement in performance.

Another issue that one should emphasize is large kernel sizes. Larger kernel sizes in ConvNets were previously discussed by Szmoegedy et al. 2014. The best size has however remained as 3x3. Zhuang Liu et al. 2022 state that Swin Transformers introduce a kernel size of at least 7x7. They also say that experiments with different kernel sizes of 3,5,7,9, and 11 were conducted. They observed that accuracy increases as the kernel size becomes wider. Yet, they state that no improvement beyond the kernel sizes of 7x7 was observed. The figure below summarizes implementation of larger kernel sizes on ResNeXt (Zhuang Liu et al. 2022).

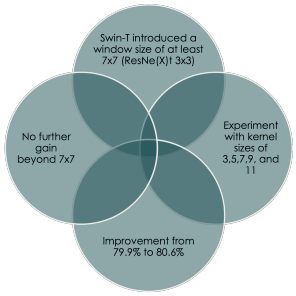


Fig. 6: Implementation of Larger Kernel Sizes - Zhuang Liu et al. 2022

According to Zhuang Liu et al. 2022, another architectural difference between a Swin Transformer block and a ResNet block is that a Swin Transformer block has fewer normalization layers and activation functions. Figure 7 compares architectures of Swin Transformers and ResNet blocks regarding number of normalization layers and activation functions.

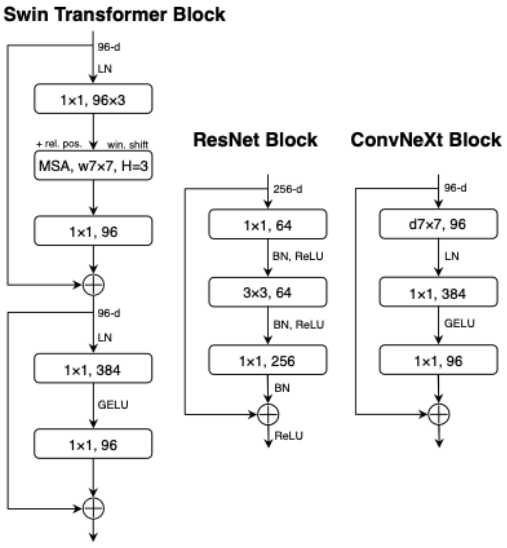


Fig. 7: Block Design of ConvNeXt Regarding Number Normalization Layers and Activation Functions - Zhuang Liu et al. 2022

The figure indicates that a Swin Transformer block employs two layer normalization layers, and an activation function GELU. Whereas a ResNet block possesses 3 batch normalization layers and it uses the ReLU activation function three times. If the design logic of the Swin Transformers blocks is applied to the ConvNeXt, then we end up with eliminating ReLU activation functions and batch normalization layers. Instead, we employ one layer normalization layer and one GELU activation function. It is worth mentioning that using GELU instead of ReLU doesn't provide increase in performance (Zhuang Liu et al. 2022). As a consequence, performance of the ConvNeXt block increases up to 81.4 percent.
In the next chapter benefits, and disadvantages of batch normalization and its removal are discussed.

III. Chapter 2 - Training Methods

Brock et al. 2021 state that batch normalization is an important element in designing image classification models, however it also has negative characteristics due to its reliance on batch size. Similarly, Wu and He 2018 say that beyond providing benefits, making inadequate choices on how to implement batch normalization can affect performance negatively. Therefore, batch normalization has been seen as a necessary evil.

According to Luo et al. 2018 batch normalization improves convergence and generalization in neural networks, and it has regularization effect. Hoffer et al. (2017) indicate that test accuracy of networks can be enhanced through tuning the batch size. Furthermore, Brock et al. 2021 hold that batch normalization enables efficient large-batch training, and eliminates mean-shift.

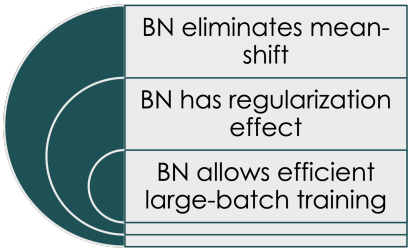


Fig. 8: Benefits of Batch Normalization - Brock et al. 2021

Batch normalization induces also a set of disadvantages. Brock et. al (2021) point out 3 important disadvantages of batch normalization:

1) Bulo, Porzi, and Kontschieder 2018 hold that batch normalization is computationally expensive, since it occupies memory
2) Batch normalization can cause imbalances on the model during training time and inference time (Summers and Dinneen 2019;Singh and Shrivastava 2019)
3) Batch normalization cause implementation error during distributed training (Pham et al. 2019)

Fig. 9: Disadvantages of Batch Normalization - Brock et al. 2021

To summarize, we can say that batch normalization provides benefits to train the CNNs. However, it also represents some disadvantages which affect the model performance. Yet, it has been an important component in architecture of the CNNs. Having discussed pros and cons of batch normalization, we can start explaining different approaches how to replace or remove it. Zhuang Liu et al. 2022 and Brock et al. 2021 have different approaches to cope with the disadvantages of batch normalization. Those ideas are discussed in the next page.

Wu and He 2018 state that batch normalization is a very important method in terms of development of deep learning, since it allows various many networks to train. However, they also point out that error significantly increases as batch size decreases. According to Wu and He 2018 this limits batch normalization's implementation to train bigger models. Thus, they offer group normalization instead of batch normalization.

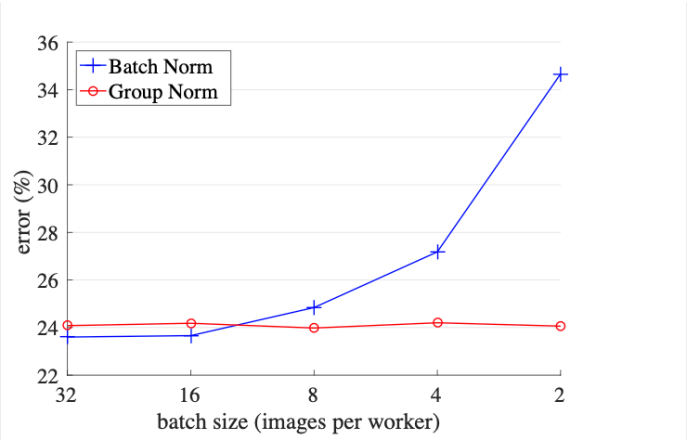


Fig. 10: Batch size and Error - Wu and He 2018

Moreover, Wu and He 2018 express that directly changing batch normalization with layer normalization can affect performance negatively. Yet, it is reported by Ba, Kiros, and Hinton 2016 that layer normalization has been used by transformers in different scenarios and it provided good performance. Moving from that point, Zhuang Liu et al. 2022 suggests the idea of substituting batch normalization with layer normalization. They state that having conducted all the architectural improvements that we have discussed so far, the model encountered no difficulties due to direct substituting of batch normalization with layer normalization. On the contrary, performance increased from 81.4 percent to 81.5 percent.

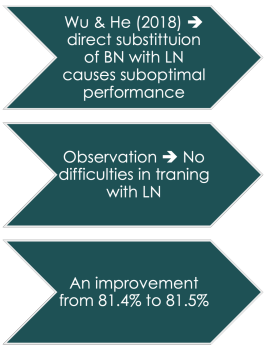


Fig. 11: Substituting Batch Normalization with Layer Normalization - Zhuang Liu et al. 2022

Brock et al. 2021 introduces normalizer-free ResNets with adaptive gradient clipping (AGC). It is held by Brock et al. 2021 that gradient clipping is used to cope with the exploding gradients problems. Because if we do not use normalization, then we end up with having large gradients. The figure below illustrates the gradient clipping.

$$G \rightarrow \begin{cases} \lambda \frac{G}{\|G\|} & \text{if } \|G\| > \lambda, \\ G & \text{otherwise.} \end{cases}$$

Fig. 12: Gradient Clipping - Brock et al. 2021

Consequently, the figure above shows that if the norm of our gradient vector is greater than the clipping threshold lambda, then we calculate it through multiplication of lambda and division of gradient vector and its norm. If the gradient vector is not greater then lambda, then we are okay with it, we don't reshape the gradient vector. On top of that, Brock et al. 2021 hold that the parameter lambda must be tune, furthermore training stability is easily affected by lambda. In order to overcome this problem they introduce the adaptive gradient clipping.

Brock et al. 2021 explains the adaptive gradient clipping as follows: $W^l \epsilon R^{NxM}$ represents that W is the weight vector in the l^{th} layer. $G^l \epsilon R^{NxM}$ means that G is gradient of loss with respect to W^l. And then we take the Frobenius norm of W^l which yields the equation below. The equation below shows the Frobenius norm. It means here that we sum all the elements of the vector and then take the square of it:

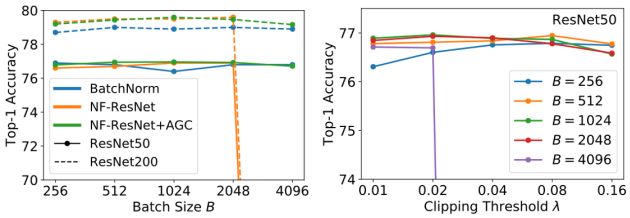
$$\| W^l \|_F^2 \sum_{i=1}^{M} \sum_{j=1}^{N} (W_{ij}^2)$$

$\| W^l \|_F$ / $\| G^l \|_F$ provides a simple of how much a single gradient step will change the weights. Let's explain why this is a good measure. $\Delta W^l = -h \cdot G^l$ shows how the parameters will change from one iteration to another one in which h represents the learning rate. Here, the weights are descending towards the negatively in the direction of the gradients. . Consequently, we end up with having the following equation if we take the Frobenius norm of the equation above: $\| \Delta W^l \|_F$ / $\| W^l \|_F$ = $h \cdot \| G^l \|$ / $\| W^l \|_F$. As a result, Brock et al. 2021 defines the adaptive gradients clipping as follows:

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

Fig. 13: Adaptive Gradient Clipping - Brock et al. 2021

Brock et al. 2021 hold that adaptive gradient clipping can scale NF-ResNets successfully. The figure below shows performance of the AGC.



(a) (b)

Fig. 14: a) AGC scales NF-ResNets to larger batch sizes b) Performance based on different clipping constants
Brock et al. 2021

Having touched upon the adaptive gradient clipping, another topic Brock et al. 2021 mention is modification of residual block. The figure blow represents the design of the residual block.

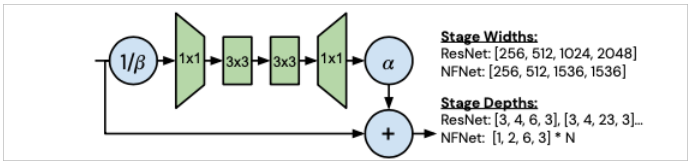


Fig. 15: NFNet Bottleneck Block - Brock et al. 2021

The residual block can be formalized as $h^{l+1} = h^l + \alpha f^l(h^l/\beta^l)$. The figure above and the formula indicate that first, input data from the previous layer is divided by β. Then it is convoluted by 1x1, 3x3, 3x3, and 1x1 convolutions respectively. In the next step, it is multiplied by α to cope with non-linearities. And then, h^l is added, which is the input of the l^{th} residual block. The function f^l is determined by the l^{th} block to control its variance. Hence, variance of the input should be equal to variance of the output. Thus, we have the following equation $Var(f^l(z)) = Var(z), \forall l$. In this case, variance of h^l is controlled through $\beta^l = \sqrt{Var(h^l)}$. In order to safeguard that $Var(f^l(z) = Var(z)$ holds, Scaled Weight Standardization is used. It is showed in the figure below (Brock et al., 2021).

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

Fig. 16: Scaled Weight Standardization - Brock et al. 2021

Moreover, the weights are updated via the following equation $\mu_i = (1 \div N) \sum_j W_{ij}$. As a consequence variance is computed through $\sigma_i^2 = (1/N) \sum_j (W_{ij} - \mu_i)^2$. In order to eliminate non-linear effects the activation function ReLU is modified as $\gamma = \sqrt{2/(1 - (1/\pi}$. Once all these structural modifications are successfully built, normalization is no more needed. A whole NF-ResNet block is shown in the figure below in detail. (Source: https://www.youtube.com/watch?v=lqGBLaTt6-o, https://www.youtube.com/watch?v=Q6j2IVXaqP4)

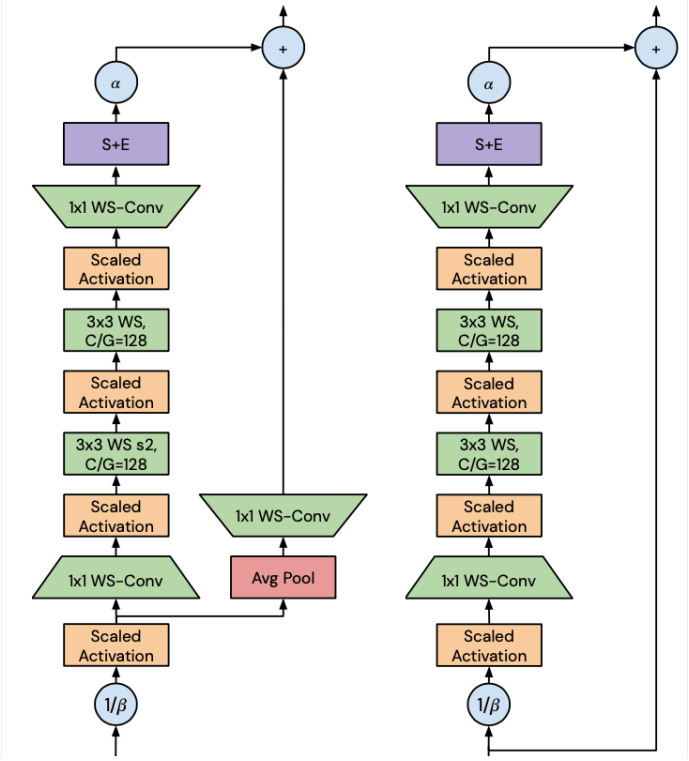


Fig. 17: NF-ResNet block - Brock et al. 2021

Thus far, architectural improvements and new training methods have been discussed. Chapter 3 refers briefly to data augmentation techniques.

IV. Chapter 3 - Data Augmentation Techniques

This chapter briefly discusses data augmentation techniques like Mixup, Cutout, and CutMix. In Mixup technique, different images are mixed. Cutout removes a random part of the image while in CutMix a random part of the image is replaced with another image (Yun et al. 2019). The figure below depicts those data augmentation techniques. It is better to use different data augmentation techniques all together rather than using only one technique.

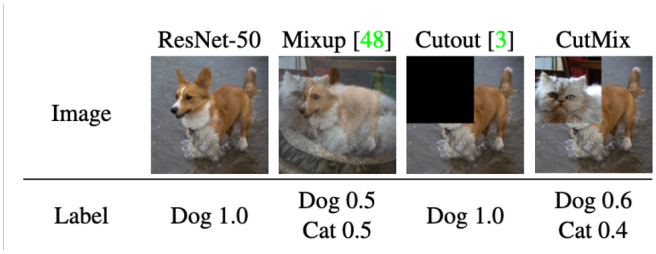


Fig. 18: Data augmentation techniques - Yun et al. 2019

V. Summary

As a conclusion, we can say that performance of CNNs can be enhanced through improving architecture of the network. Zhuang Liu et al. 2022 implements properties of transformers to ResNets, which results in better performance. The figure below shows how performance of a ResNet gets better across every architectural improvement.

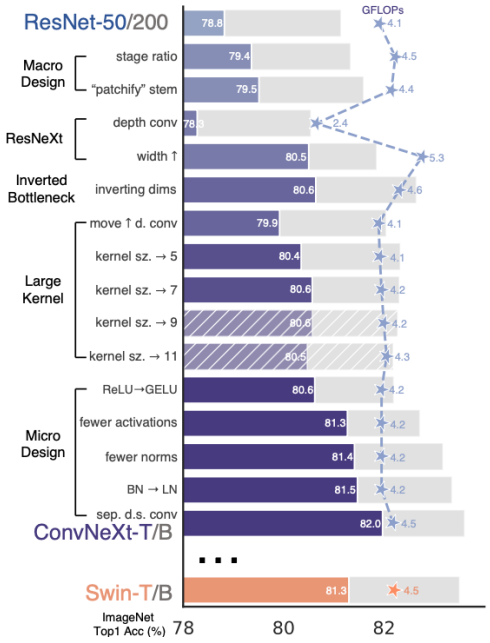


Fig. 19: Performance-wise Improvement of ConvNeXt - Zhuang Liu et al. 2022

The figure shows that ConvNeXt reaches out an accuracy of 81.4 percent when all architectural changes from stage ratio to usage of fewer normalization layers. At that point ConvNeXt starts performing better than a Swin Transformer.

Training techniques like removing batch normalization, implementation of adaptive gradient clipping and modifying the residual block provides also increase in performance.

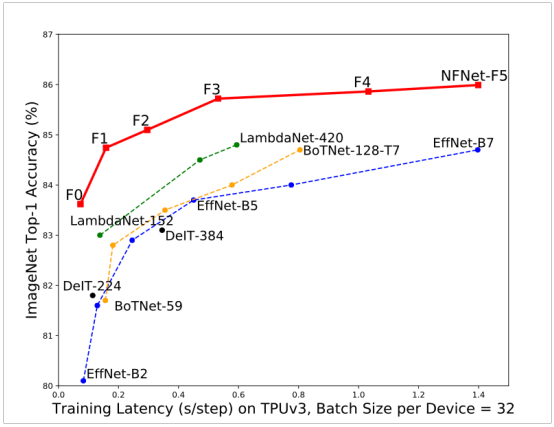


Fig. 20: Performance of NF-Net - Brock et al. 2021

References

Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E Hinton (2016). "Layer normalization". In: *arXiv preprint arXiv:1607.06450*.

Brock, Andy et al. (2021). "High-performance large-scale image recognition without normalization". In: *International Conference on Machine Learning*. PMLR, pp. 1059–1071.

Bulo, Samuel Rota, Lorenzo Porzi, and Peter Kontschieder (2018). "In-place activated batchnorm for memory-optimized training of dnns". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5639–5647.

Liu, Ze et al. (2021). "Swin transformer: Hierarchical vision transformer using shifted windows". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10012–10022.

Liu, Zhuang et al. (2022). "A convnet for the 2020s". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11976–11986.

Luo, Ping et al. (2018). "Towards understanding regularization in batch normalization". In: *arXiv preprint arXiv:1809.00846*.

Pham, Hung Viet et al. (2019). "CRADLE: cross-backend validation to detect and localize bugs in deep learning libraries". In: *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, pp. 1027–1038.

Singh, Saurabh and Abhinav Shrivastava (2019). "Evalnorm: Estimating batch normalization statistics for evaluation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3633–3641.

Summers, Cecilia and Michael J Dinneen (2019). "Four things everyone should know to improve batch normalization". In: *arXiv preprint arXiv:1906.03548*.

Szmoegedy, C et al. (2014). "Going deeper with convolutions". In: *CVPR*.

Tsang, Sik-Ho (2018). *Review: Xception — With Depthwise Separable Convolution, Better Than Inception-v3 (Image Classification)*. URL: https://towardsdatascience.com/review-xception-with-depthwise-separable-convolution-better-than-inception-v3-image-dc967dd42568 (visited on 09/19/2022).

Wu, Yuxin and Kaiming He (2018). "Group normalization". In: *Proceedings of the European conference on computer vision (ECCV)*, pp. 3–19.

Yun, Sangdoo et al. (2019). "Cutmix: Regularization strategy to train strong classifiers with localizable features". In: *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6023–6032.