Activations And Augmentations: Pushing The Limits Of Isotropic ConvNets

Keifer Lee,[1] **Shubham Gupta,**[1] **Karan Sharma**[1]

[1] New York University
kl3866@nyu.edu, sg7761@nyu.edu, ks6421@nyu.edu

Abstract

Isotropic architectures have recently gained focus for solving computer vision problems for their ability to capture better spatial information. In this work, we experiment with training a ConvMixer model, an isotropic convolutional neural net architecture on the CIFAR-10 dataset. We propose a new architecture: ConvMixer-XL consisting of 66 layers and just under $5M$ parameters. To maximize its performance, various configurations of the architecture, augmentations and activations were tried in our ablation study to further fine-tune the model. Our experiments show applying augmentations and using the Swish (SiLU) activation function for deeper models gives the best results with a top-1 accuracy of 94.52%. Our code can be found at `https://github.com/datacrisis/ConvMixerXL`.

1 Introduction

Image classification is an old problem in computer vision and great success has been achieved by taking advantage of convolutional neural networks (Lecun et al. 1998). While ConvNets have existed for a long time now, it is only in the last five years that we observed improvements in ConvNets, patch embedding for images, and Vision Transformers (Trockman and Kolter 2022; Woo et al. 2023; Dosovitskiy et al. 2020).

With the inception of ResNets (He et al. 2016), researchers have been able to improve performance of networks by building deeper networks and training them for longer epochs. This has been made possible by introducing skip connections into the architecture which circumvents the vanishing gradient problem that has plagued prior deep networks (Hochreiter 1998). A representation of a ResNet block can be seen in Figure 1.

To boost the performance of networks further, image augmentations have proven to aid with improving the generalization capacity of the model on out of distribution samples. Some examples of said augmentations are random resized crops (), random horizontal flips, random augmentation, color jitters, random erasing (Zhong et al. 2020), cutmix (Yun et al. 2019), mixup (Zhang et al. 2017).

Furthermore, it is well-known that by subjecting the inputs to non-linear activation functions, it enables the model to learn the complex relationships between the features and expected outputs. Recently, activations such as leaky ReLU (Agarap 2018), GELU (Hendrycks and Gimpel 2016), SiLU (Elfwing, Uchibe, and Doya 2018) and many more have proven to be a viable alternative to the widely adopted non-negative piecewise linear ReLU (Agarap 2018). By switching out the ReLU activation in existing models with functions that are continuous when differentiated, we expect better performance with deeper networks.

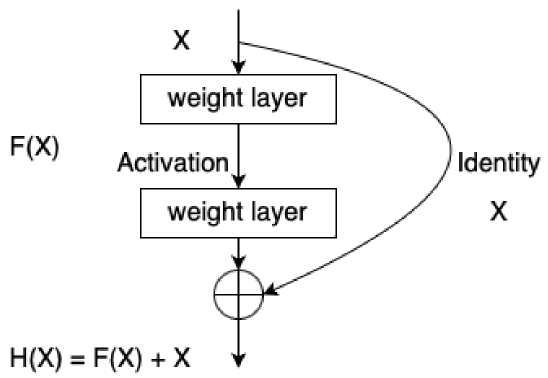

Figure 1: ResNet block (He et al. 2016).

Our contributions in this work can be summarized as:

1. First, we propose a viable deeper ConvMixer architecture, ConvMixer-XL, with added skip connections.
2. Second, we demonstrate the importance of using image augmentation while training deeper networks.
3. Third, we compare the performance of different activation functions on ConvMixer and ConvMixer-XL.

2 Methodology

The base architecture of our model is made from ResNet blocks and can be classified as an isotropic architecture (Dosovitskiy et al. 2020). The network breaks the input into patch embeddings which it passes through blocks of constant width to learn the representations. This is unlike the

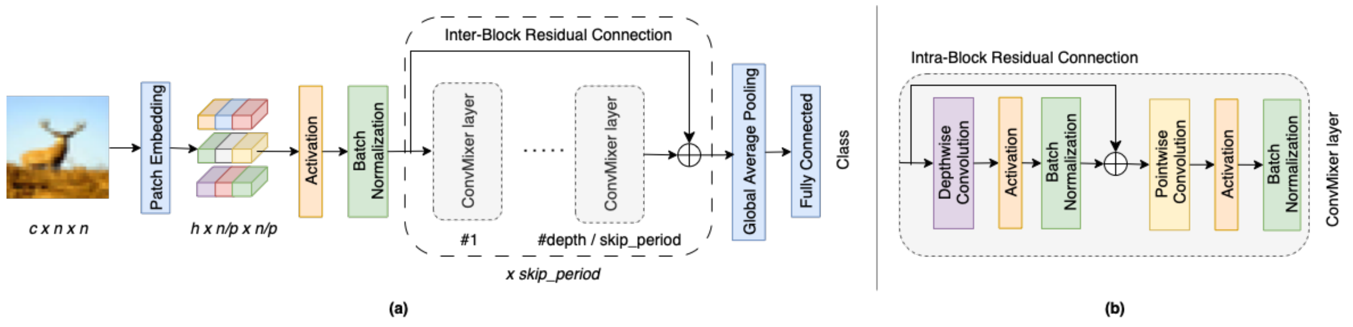

Figure 2: Overview of proposed architecture, ConvMixer-XL. Sub-figure **(a)** shows the high-level structure with inter-block residual connection that spans $depth/skip\_period$ number of *ConvMixer* layers, while **(b)** shows the structure of an individual *ConvMixer* layer.

typical ConvNet designs where we incrementally increase the channel dimension like a pyramid structure (Trockman and Kolter 2022). We chose this architecture for the following reasons: first, it is easily extendable by increasing the number of blocks, second, generating patch embedding is still an area of active research and we can further modify how the embedding layer generates its patches, third, the authors of ConvMixer (Trockman and Kolter 2022) suggested their model may not yet be fully optimized for performance which we wanted to explore.

In order to improve the representational capability of the model, we've also increased the number of parameters in the model to the limit by vastly upping network depth from 8-blocks to 66-blocks deep with just under $5M$ parameters. We dub this architecture as 'ConvMixer-XL'. However, just blindly increasing the depth and parameter count as such introduces other drawbacks such as risk of overfitting and convergence failures due to vanishing gradients (Hochreiter 1998). To combat the aforementioned issues, a new set of long range inter-block skip connections were introduced alongside a new hyperparameter called $skip\_period$ that controls the number of ConvMixer blocks that the inter-block skip connection will span, as seen in figure 2. With this, we've found that ConvMixer-XL is able to learn and converge almost like the much shallower vanilla ConvMixer model, thereby alleviating the vanishing gradient problem. On the risks of overfitting, this particular issue was dealt with by subjecting the training dataset to a battery of augmentations. Results of experimentation with the model architecture will be discussed in section 4.

Apart from building a deeper model, we wanted to push the performance of ConvMixer-XL further; following the original paper again, we also tried to tune the augmentations and regularizers used to train the model including Random Erasing (Zhong et al. 2020) and Random Augmentation (Cubuk et al. 2020). The idea behind random erasing is to occlude random regions of the image with noise which forces the model to learn features less specific to the training set. Random Augment as the name suggests applies a range of random augmentations on images on multiple rounds resulting in a very diverse dataset. This reduces the chances of the model overfitting to features that only exist the training set. Further discussion on the augmentations used is in section 3. We keep most augmentations implemented by the author except for CutMix (Yun et al. 2019) and MixUp (Zhang et al. 2017) due to implementation issues. The idea of using augmentations is to better generalize the model weights for out of distribution samples which should yield better performance on the test and validation sets.

Activations in a network play an important role of adding non-linearity in order to disentangle complex relationships between the input and output features. The ReLU activation function used in the original ResNet paper (He et al. 2016) has proven its effectiveness. The same way, ConvMixer (Trockman and Kolter 2022) shows both ReLU and GELU have almost identical performance, GELU being slightly better for isotropic architecture. In this work we compare ReLU, GELU and SiLU activations for the ConvMixer and ConvMixerXL architectures. We experiment with SiLU as it has proven to perform better on deeper networks (Ramachandran, Zoph, and Le 2017). The biggest differentiating factor between ReLU and the rest would be the graph shape, having a kink. GELU is a smooth approximation of ReLU derived from a gaussian distribution while SilU is a smooth approximation of the ReLU function derived from the sigmoid function. We compare the performance of our model with different activation, augmentation and depth on the CIFAR-10 (Krizhevsky 2009) dataset in section 4.

3 Experiment setup

We ran our experiments using a NYU Greene HPC node with 16 CPU cores, an NVIDIA RTX8000 GPU and 64GB of DRAM. We set a seed to 1024 to make our experiments reproducible. We use a batch size of 128 and the following hyperparameters, if enabled: scale$= 0.75$, random erasing$=$ 0.2, 2 random augmentations of magnitude 12, color jitter$=$ 0.2, patch size$= 2$, kernel size$= 5$, horizontal dimension$=$ 256, maximum learning rate$= 0.005$, weight decay$= 0.01$ and 8 workers.

4 Results & Discussions

Building off a vanilla ConvMixer (Trockman and Kolter 2022) classifier, various modifications were tested in order

Name	***Activation***	***Depth***	***Inter-Block Skip***	***Augmentations***	***#Params (M)***	***Top 1 %Acc***
CM-Vanilla-NoAug	GELU	8	No	No	0.59	0.8854
CM-Vanilla	GELU	8	No	Yes	0.59	0.9378
CM-Vanilla-ReLU	ReLU	8	No	Yes	0.59	0.9384
CM-Vanilla-SiLU	SiLU	8	No	Yes	0.59	0.9372
CM-XL-NoSkip	GELU	66	No	Yes	4.9	0.4868
CM-XL-Skip	GELU	66	Yes	Yes	4.9	0.9422
CM-XL	**SiLU**	**66**	**Yes**	**Yes**	**4.9**	**0.9452**

Table 1: Details and top-1 test accuracy of each configuration. All instances are trained on CIFAR-10 only for 100 epochs. **Bold** indicates the best result.

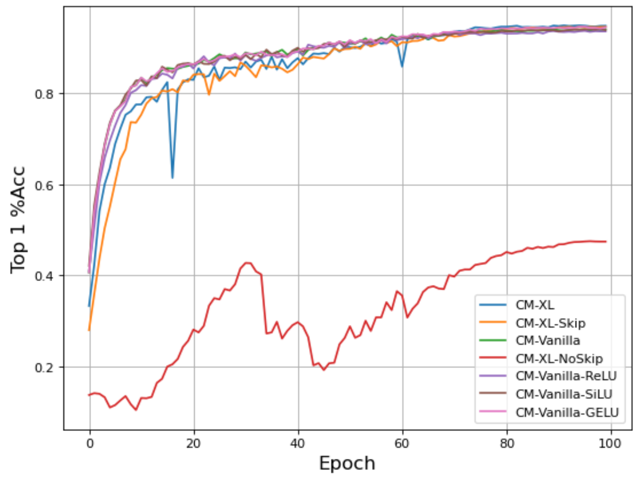

Figure 3: Validation accuracy by epochs for each configurations tested.

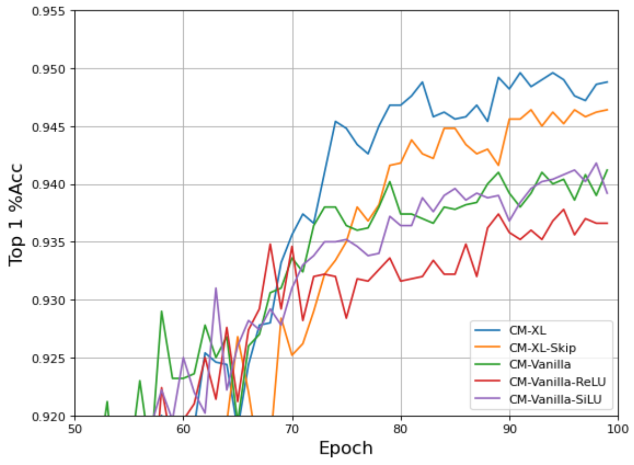

Figure 4: Validation accuracy by epochs filtered (relevant subset only) and zoomed-in for finer distinction.

Figure 5: Example of correctly classified test set samples by ConvMixer-XL. From top-left to bottom-right, the predicted classes are - *horse, plane, bird, truck, ship, bird, deer, car, dog, car, horse, truck, car, deer, dog* and *plane* respectively.

to find new configurations with better performance. Our proposed ConvMixer-XL is like its namesake, a much deeper version of its vanilla counterpart with extra tweaks to aid convergence. The result of each configurations explored can be found in Table 1; this can be viewed as an ablation study as well.

Amongst the various instances, the best results are achieved with **ConvMixer-XL** (CM-XL) with a Top 1 accuracy of 94.52% on CIFAR-10. However, the vanilla ConvMixer (CM-Vanilla) is not far behind with a corresponding accuracy of 93.78%. We hypothesize that the CIFAR-10 dataset is probably too simplistic and therefore the vanilla version with a depth of 8 is already sufficient to represent a relatively well performing classifier. Additionally, all the model instances were trained with a 100 epochs only, which may not be optimal in the case of a deep model such as ConvMixer-XL; subjecting ConvMixer-XL to a more complex dataset (e.g. CIFAR-100) and better hyperparameter tuning (e.g. epochs) is an avenue for future studies. We also found that if we simply added more blocks to the vanilla model, the model convergence deteriorates significantly as seen in Figure 3 (CM-XL-NoSkip), which can be attributed to the issue of vanishing gradient. To combat this issue, additional long range skip connections were added into the architecture (e.g. inter-block residual connections; spans 22 layers in ConvMixer-XL-Skip) and convergence performance recovered significantly.

In terms of activation functions, they do not seem to make much difference, at least amongst the selection of functions that were tested. However, we observe that Swish (CM-Vanilla-SiLU) seems to show faster convergence speed in general compared to ReLU (CM-Vanilla-ReLU) and GELU (CM-Vanilla). We hypothesize that the difference observed is likely due to the specific model initialization and dataset at hand since Swish (SiLU) and GELU are very similar functions mathematically. Augmentations however seems to play a significant role in model generalization and performance as expected - with all else equal, CM-Vanilla-NoAug achieved only 88.54% in test time top-1 accuracy, a sizable deterioration compared to CM-Vanilla.

All the model instances explored so far meet the requirement of not exceeding 5 million parameters as shown in Table 1, with the ConvMixer-XL coming in at 4.9M parameters in total. Finally, in terms of generalization on the test set, it can be observed that none of the models are over-fitting by cross-checking the resulting accuracy and ordering between the validation and test sets result in both Figure 4 and Table 1. From Figure 5 and 6 where some example predictions of ConvMixer-XL on the CIFAR-10 test set are shown, we can see that the model does very well on the correct predictions despite the sample variety. For the incorrect classifications, it does appears that the samples are very hard in those particular instances - e.g. the *bird*'s body does somewhat look like a dog's from that angle, and the *deer* is incredibly hard to discern due to its low resolution and distance from the camera, from which it does look like a possible cat.

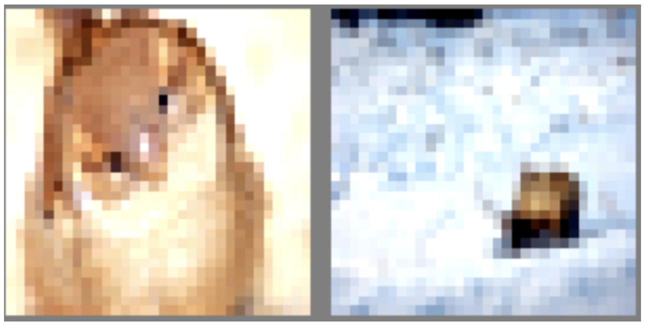

Figure 6: Example of wrongly classified test set samples by ConvMixer-XL. From left to right, the predicted classes are - *dog* and *cat*, and the labels are *bird* and *deer* respectively.

References

Agarap, A. F. 2018. Deep Learning using Rectified Linear Units (ReLU). Cite arxiv:1803.08375Comment: 7 pages, 11 figures, 9 tables.

Cubuk, E. D.; Zoph, B.; Shlens, J.; and Le, Q. V. 2020. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 702–703.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Elfwing, S.; Uchibe, E.; and Doya, K. 2018. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107: 3–11.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hendrycks, D.; and Gimpel, K. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Hochreiter, S. 1998. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 06(02): 107–116.

Krizhevsky, A. 2009. Learning multiple layers of features from tiny images. Technical report.

Lecun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

Ramachandran, P.; Zoph, B.; and Le, Q. V. 2017. Searching for activation functions. *arXiv preprint arXiv:1710.05941*.

Trockman, A.; and Kolter, J. Z. 2022. Patches Are All You Need?

Woo, S.; Debnath, S.; Hu, R.; Chen, X.; Liu, Z.; Kweon, I. S.; and Xie, S. 2023. ConvNeXt V2: Co-designing and Scaling ConvNets with Masked Autoencoders. *arXiv preprint arXiv:2301.00808*.

Yun, S.; Han, D.; Oh, S. J.; Chun, S.; Choe, J.; and Yoo, Y. 2019. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, 6023–6032.

Zhang, H.; Cisse, M.; Dauphin, Y. N.; and Lopez-Paz, D. 2017. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*.

Zhong, Z.; Zheng, L.; Kang, G.; Li, S.; and Yang, Y. 2020. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 13001–13008.