Improving image classification on limited data with data augmentations and adversarial training: Final Report

G000 (s1893731)

Abstract

This project aims to improve image classification performance of a baseline ResNet-50 model on a small subset of the Imagenet dataset consisting of 50,000 training images and 50,000 test images by carrying out ablation studies on various data augmentation techniques (such as RandAugment and AutoAugment in addition to CutMix and Mixup), activation functions (such as Mish, Swish, ReLU, GELU), loss functions (such as label smoothing cross entropy and soft target cross entropy) and training techniques (such as adversarial training) in addition to slight improvements in the model architecture with the addition of 3 Spatial Pyramid Pooling layers.

1. Introduction

In the field of deep learning there is a huge dependence on large scale datasets for purposes of model pretraining. Learning with limited data is a challenging task. For example, The ResNet-50 convolutional neural network achieves a Top1-Accuracy of 76.3% on the original ImageNet dataset which contains 1 million training images, 50,000 validation images and 100,000 test images. In contrast, using standard training techniques and cross entropy loss, the best Top-1 Accuracy achieved on the minimized Imagenet dataset is 45% only. In order to reduce dependency on large datasets, there is a need to come up with a combination of training methods that could effectively improve CNN performance on limited data. The ResNet-50 backbone is utilised for carrying out image classification on a subset of the Imagenet dataset. Ablation studies were carried out using varous data augmentation techniques (such as RandAugment and AutoAugment in addition to CutMix and Mixup), loss functions such as label smoothing cross entropy and soft target cross entropy , activation functions like Mish, Swish and GELU in addition to adversarial training techniques along with a slight improvement in the model architecture with the addition of 3 Spatial Pyramid Pooling layers in order to improve **Top-1 Accuracy** on the subset to **40.8%**.

2. Data set and task

A small subset of the Imagenet dataset was utilised, consisting of 50,000 training images and 50,000 test images as opposed to 1 million training images, 50,000 validation images and 100,000 test images of the original Imagenet dataset. The number of samples per class were artificially reduced from 1000 samples per class to 50 samples per class. The task was to use various techniques to improve image classification accuracy from a given baseline of 31%.

3. Methodology

3.1. Activation Functions Utilized -

3.1.1. Swish

Is a scalar activation functon defined by - $\mathbf{f(x) = x.sigmoid(\beta x)}$. Where β is either a constant or a trainable parameter, with $\beta = 1$ Swish acts like a Sigmoid Weighted Linear Unit (SiLU), with $\beta = 0$, Swish acts like the scaled function $f(x) = \frac{x}{2}$ and with $\beta -> \infty$, Swish acts like a ReLU activation function with the sigmoid component approaching a 0-1 function. Swish can therefore be seen as a smooth function which non-linearly interpolates between a linear function and the ReLU function. The smooth, continuous profile of Swish proved essential in better information propagation as compared to ReLU. Since Swish conveyed a very small amount of negative information, it is effective at dealing with the dying ReLU problem. (Ramachandran et al., 2017)

3.1.2. Mish

Mish activation function is a smooth, continuous, self regularized, non-monotonic activation function defined as $\mathbf{f(x) = x.tanh.softplus(x)}$

It uses a self-gating property inspired by Swish where the non-modulated input is multiplied with the output of a non-linear function of the input. Mish by design is built to avoid the dying ReLU problem much like Swish. Additionally, Mish avoids saturation which causes training to slow down due to non-zero gradients. Mish is also continuously differentiable, unlike ReLU making it preferable since it avoids singularities. (Misra, 2019)

3.1.3. GELU - Gaussian Error Linear Units

The GELU activation is defined by $\mathbf{x\Phi(x)}$ where $\mathbf{\Phi(x)}$ is a Gaussian cumulative distribution function (CDF). GELU weights inputs by their percentile instead of gating inputs by their sign in the case of ReLUs. GELU can be thought of as a smoother version of ReLU. (Hendrycks & Gimpel, 2016)

$$GELU(x) = xP(X \leq x) = x\Phi(x) = x.\frac{1}{2}[1 + erf(x/\sqrt{2})]$$

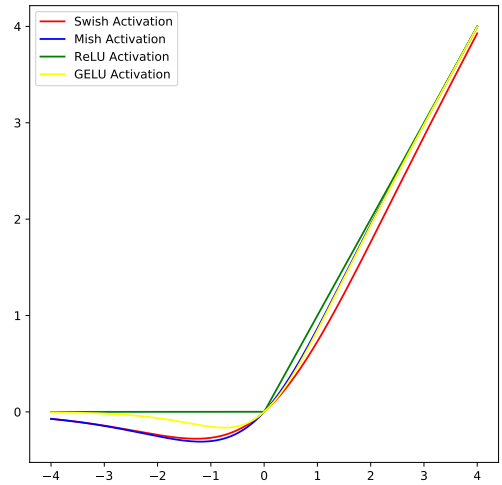

Figure 1. Comparison of Mish, Swish, ReLU and GELU activations

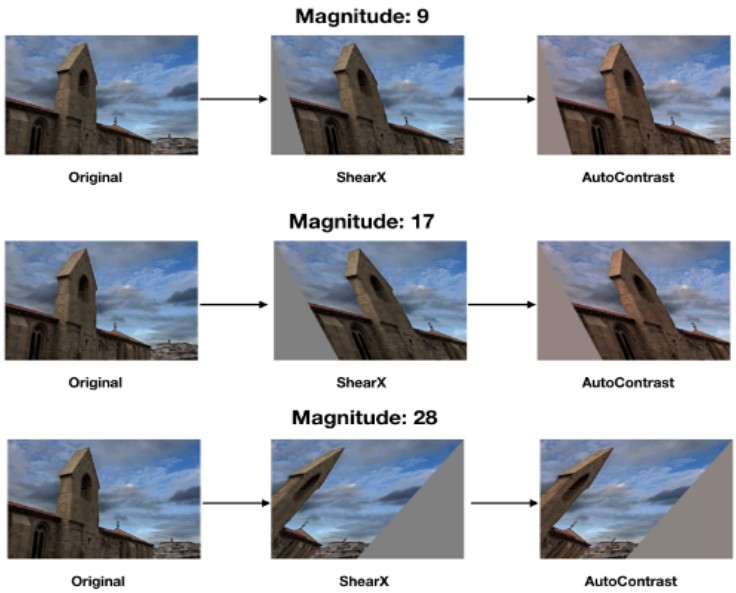

Figure 2. Examples of the RandAugment (Cubuk et al., 2019) Augmentation utilised. The figure shows the change in the image using the Shear and AutoContrast image augmentations with varying magnitude/ϵ of RandAugment

if $X \sim N(0, 1)$

Which can be approximated with

$0.5x(1 + tanh[\sqrt{2}/\pi(x + 0.044715x^3)])$ or $x\sigma(1.702x)$

GELU activations are used in Transformers including Vision Transformers like ViT and Swin Transformers in addition to modern Convnets such as ConvNeXt (Liu et al., 2022) .

3.2. AutoAugment -

AutoAugment (Cubuk et al., 2018) frames the problem of learning best data augmentations (shearing, rotation, inversion, shearing etc.) for image classification as a Reinforcement Learning Problem. AutoAugment looks for the augmentation combinations that lead to the highest accuracy on the evaluation set. It consists of 2 components - search algorithm and search space.

3.3. Rand-Augment -

The purpose of RandAugment (Cubuk et al., 2019) was to remove the need for a separate search space for auto augment by controlling the magnitudes of different transformation operations with a single magnitude parameter.

3.4. Spatial Pyramid Pooling

In order to improve the performance of the ResNet-50 backbone model, 3 Spatial Pyramid Pooling Blocks (He et al., 2014) the first of which is used after the first 3 bottleneck layers, the second is used after the next 4 bottleneck layers and the 3rd and final SPP-Block is used after the next 6 bottleneck layers.

The SPP-Block carries out 3 MaxPooling operations, for each a kernel size of 5x5 was chosen. The input featuremap of shape HxWxC (F_1) undergoes a pointwise convolution to generate a featuremap (F_2) of shape HxWxC/2, on this featuremap, 3 MaxPooling operations are applied one after the other after which the featuremaps F_2, F_3, F_4, F_5 are concatenated where F_3 is the featuremap generated by the first MaxPooling operation, F_4 is the featuremap generated by the second MaxPooling operation and F_5 by the third respectively. After which the concatenated featuremap of shape HxWx4C is subjected to another pointwise convolution mapping it to a featuremap of size HxWxC as can be seen in Figure 4.

3.5. Label Smoothing-

Label smoothing is a regularization strategy that introduces noise for the labels accounting for the fact that datasets might have mistakes in them, due to this maximising $logp(y|x)$ might be harmful. Label smoothing assumes a small constant ϵ for which y is correct with probability $1-\epsilon$, it then regularizes the model with the help of a softmax with k output values by replacing the hard 0 and 1 classification targets with targets of $\frac{\epsilon}{k-1}$ and $1 - \epsilon$ respectively. (Goodfellow et al., 2016)

3.6. Mixup

Mixup (Zhang et al., 2017) is a data augmentation strategy that helps regularize the convolutional neural network to favour simple linear behaviour in between training examples by training the neural network on a convex combination of pairs of examples and their labels. It improves the gen-

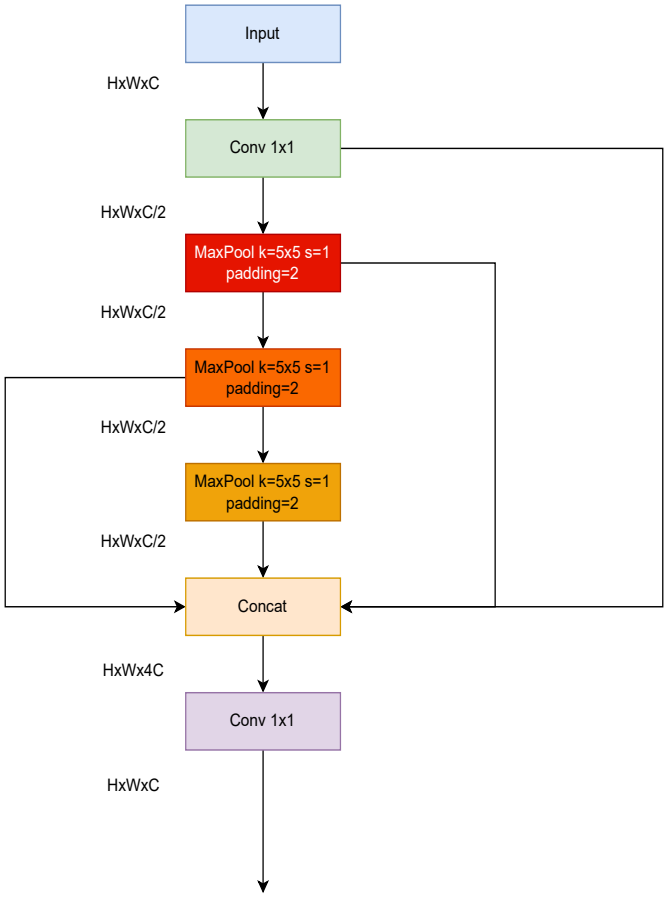

Figure 3. Spatial Pyramid Pooling Block (He et al., 2014)

eralization ability of architectures like the ResNet-50 and improves sensitivity to adversarial examples.

3.7. Cutmix

CutMix (Yun et al., 2019) is a data augmentation strategy similar to Mixup in that it combines two image samples where the ground truth of the label is provided by the linear interpolation of one-hot labels. In CutMix, patches are cut and pasted among training images and the ground truth labels are also mixed proportionally to the size of the area of the patches. The regularization effect of regional dropout helps improve CNN performance on vision tasks.

3.8. Adversarial Training -

Gradient based adversarial attacks were utilised for training the ResNet-50 model, this helps regularize the ResNet-50 model and makes it resistant to adversarial examples and serves as the ultimate data augmentation.

The Projected Gradient Descent Attack (PGD Attack) was utilised in a white box setting (i.e. the attacker knows the details of the model architecture). (Madry et al., 2017)

Figure 4. Example of MixUp (Zhang et al., 2017), CutOut and CutMix (Yun et al., 2019) data augmentation strategies

Setup	Mish	GELU	Swish
GPU	Tesla P-100	Tesla P-100	Tesla P-100
Batch Size	70	70	70
Image Size	256x256	256x256	256x256
Number of epochs	250	250	250
Optimizer	Adam	Adam	Adam
Initial LR	$1e-5$	$1e-5$	$1e-5$
RandAugment (ϵ)	15	15	15
BatchNorm	Yes	Yes	Yes
SPP-Layer	No	No	No
Top-1 Accuracy	**33.8%**	**33.3%**	**33.1%**

Table 1. Mish vs Swish vs GELU

4. Experiments

4.1. Experimental Setup

All experiments were carried out on Kaggle on 1 NVIDIA-Tesla-P100 GPU with 16GB RAM and 2 vCPUs.

4.2. Mish vs Swish vs GELU

For the first experiment, the ResNet-50 model was trained with the 3 best activation functions - Mish, Swish and GELU all of which are smoother than ReLU, Leaky-ReLU and ELU. The task was to determine which activation function would give the best performance. The ResNet-50 model was trained for 250 epochs, with a batch size of 70, with FP-16 floating point precision, using image size of 256x256, the Adam optimizer was utilised with a learning rate of $1e-5$ with BatchNorm and RandAugment augmentation with $\epsilon = 15$ for each activation function as can be seen in Table 1.

The Mish activation managed to outperform the Swish and GELU on the test set achieving a **Top-1 Accuracy of 33.8%**. Therefore, the Mish activation was selected to be used with BatchNorm for further experiments.

4.3. Mish + BatchNorm vs GELU + LayerNorm

It was important to carry out a comparison of a combination of Mish and BatchNorm with GELU and LayerNorm which was inspired by (Liu et al., 2022). This was done to analyse the performance of GELU + LayerNorm on the ResNet-50 model in order to verify the observations of (Liu et al., 2022)

Setup	Mish + BatchNorm	GELU + LayerNorm
GPU	Tesla P-100	Tesla P-100
Batch Size	70	70
Image Size	256x256	256x256
Number of epochs	250	250
Optimizer	Adam	Adam
Initial LR	$1e-5$	$1e-5$
RandAugment (ϵ)	15	15
SPP-Layer	No	No
Top-1 Accuracy	**33.8%**	**32.5%**

Table 2. Mish + Batchnorm vs GELU + LayerNorm

who had concluded that using LayerNorm worsened performance of the ResNet-50 on the Imagenet dataset, however using GELU + Layer Normalization helped improve the performance of the ConvNext model architecture.

The ResNet-50 model was trained for 250 epochs, using a batch size of 70, with FP-16 floating point precision, using image-size of 256x256, utilising the Adam optimizer with an initial learning rate of $1e-5$ and a minimum learning rate of $1e-6$ with RandAugment augmentation for both Mish + BatchNorm and GELU + LayerNorm and the performance of both was compared as can be observed in Table 2.

The Mish + BatchNorm training strategy outperformed the GELU + LayerNorm training strategy achieving a **Top-1 Accuracy of 33.8%** on the provided Test set, one of the possible reasons for this could be due to the better performance of BatchNorm over LayerNorm due to FP-16 floating point precision training which allows for larger batch sizes in training. Therefore, Mish + BatchNorm was the combination used for the final experiments.

4.4. Experiments with CutMix, Mixup and addition of SPP-Blocks

Having chosen the ideal activation function and Normalization technique, further experiments were carried out, testing out operations like CutMix, Mixup in addition to utilising 3 additional Spatial Pyramid Pooling layers for improving the ResNet-50 model architecture.

As can be seen from Table 3, three training approaches are compared, the first of which is trained without FP-16 floating point precision while A2 and A3 are trained with FP-16. Approach A1 has been trained with AutoAugment data augmentation, while A2 and A3 were trained with RandAugment data augmentation and A1 and A2 were trained without CutMix and Mixup image augmentations while A3 was trained with both CutMix and Mixup. Additionally, A3 utilises an improved version of the ResNet-50 model architecture with 3 SPP-Blocks the first of which was inserted after the first three bottlenecks, the next after the next 4 bottlenecks, and the last SPP-Block after the next 6 bottlenecks.

It can be observed that training the model with FP-16 allowed for more than twice the batch size when looking at the batch size of 30 for A1 and batch size of 70 for A2, while the addition of the 3 SPP-Layers slightly increased model complexity reducing the possible batch size to 50 from 70 during training. Although with the application of CutMix and Mixup, A3 took longer to converge - 300 epochs to 250, however, it managed to outperform A2 achieving a Top-1 Accuracy of 35% compared to that of A2 of 33.8% since the A3 model had to be trained from scratch without the use of pretrained weights.

Setup	A1	A2	A3
GPU	Tesla P-100	Tesla P-100	Tesla P-100
Batch Size	30	70	50
FP-16	No	Yes	Yes
Image-size	256x256	256x256	256x256
Number of epochs	250	250	300
Optimizer	Adam	Adam	Adam
Initial LR	$1e-5$	$1e-5$	$1e-5$
H-flip	Yes	Yes	Yes
RandAugment	No	Yes	Yes
AutoAugment	Yes	No	No
Cutmix	No	No	Yes
Mixup	No	No	Yes
SPP-Layer	No	No	Yes
Label Smoothing CE	No	Yes	No
Smooth Target CE	No	No	Yes
Top-1 Accuracy	**31%**	**33.8%**	**35%**

Table 3. Comparison of 3 Approaches for training ResNet-50, the first approach uses the AutoAugment data augmentation without Cutmix and Mixup, the second aproach utilises RandAugment image augmentation with a label smoothing cross entropy loss, whereas the 3rd and best approach utilises RandAugment data augmentation strategy with Cutmix + Mixup, a smooth target cross entropy loss, in addition to 3 Spatial Pyramid Pooling Layers

We can conclude from these experiments that CutMix and Mixup augmentation strategies in addition to utilising SPP-Layers on the ResNet-50 backbone, in combination with RandAugment data augmentation and FP-16 floating point precision helped improve the Top-1 Accuracy of the ResNet-50 model on the Test set.

4.5. Experiments with Adversarial Training

Since it was concluded from the previous experiment that CutMix, MixUp and the utilization of SPP-Layers improved model performance, therefore for the final experiments, A4 and A5 built upon the A3 approach and were trained with Adversarial Training using a PGD Attacker to analyse the benefits of Adversarial Training.

In Table 4, the two new training approaches utilizing Adversarial Training are compared with A3 with the difference being that A4 just like A1, A2, A3 was trained with a 90:10 split while A5 was trained with an 80:20 split with a learning rate of $1e-3$ allowing it to converge much faster.

The A4 and A5 approaches which used adversarial training managed to outperform the A3 approach achieving a Top-1 Accuracy of **37.4%** and **40.8%** respectively.

Setup	A3	A4	A5
GPU	P-100	P-100	P-100
Batch Size	50	50	50
Train:Val split	90:10	90:10	80:20
FP-16	Yes	Yes	Yes
Image-size	256x256	256x256	256x256
Number of epochs	300	325	325
Optimizer	AdamW	AdamW	AdamW
Initial LR	$1e-5$	$1e-5$	$1e-3$
LR-Decay	Cosine	Cosine	Cosine
H-flip	Yes	Yes	Yes
RandAugment (ϵ/std)	7/0.5	7/0.5	7/0.5
Cutmix	Yes	Yes	Yes
Mixup	Yes	Yes	Yes
SPP-Layer	Yes	Yes	Yes
Adversarial-Training	No	Yes	Yes
Smooth Target CE	Yes	Yes	Yes
Top-1 Accuracy	**35%**	**37.4%**	**40.8%**

Table 4. Comparison of A3, A4 and A5, ResNet-50 was trained using **Adversarial Training** in A4 and A5 with a train:val split of 90:10 and 80:20 respectively for 325 epochs.

5. Conclusions

It can be concluded from the above experiments that utilizing data augmentation techniques like RandAugment, AutoAugment in addition to augmentation and regularization strategies like Cutmix and Mixup used alongside adversarial training sufficiently regularizes the model and helps it train with low data. Additionally, the addition of 3 SPP layers to the ResNet-50 model was helpful in improving model performance.The ResNet-50 model architecture trained with a combination of these techniques manages to achieve a Top-1 Accuracy of **40.8%** on the provided test set.

References

Cubuk, Ekin D., Zoph, Barret, Shlens, Jonathon, and Le, Quoc V. Randaugment: Practical data augmentation with no separate search. *CoRR*, abs/1909.13719, 2019. URL http://arxiv.org/abs/1909.13719.

Cubuk, Ekin Dogus, Zoph, Barret, Mané, Dandelion, Vasudevan, Vijay, and Le, Quoc V. Autoaugment: Learning augmentation policies from data. *CoRR*, abs/1805.09501, 2018. URL http://arxiv.org/abs/1805.09501.

Goodfellow, Ian J., Bengio, Yoshua, and Courville, Aaron. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. http://www.deeplearningbook.org.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Spatial pyramid pooling in deep convolutional networks for visual recognition. *CoRR*, abs/1406.4729, 2014. URL http://arxiv.org/abs/1406.4729.

Hendrycks, Dan and Gimpel, Kevin. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415, 2016. URL http://arxiv.org/abs/1606.08415.

Liu, Zhuang, Mao, Hanzi, Wu, Chao-Yuan, Feichtenhofer, Christoph, Darrell, Trevor, and Xie, Saining. A convnet for the 2020s. *CoRR*, abs/2201.03545, 2022. URL https://arxiv.org/abs/2201.03545.

Madry, Aleksander, Makelov, Aleksandar, Schmidt, Ludwig, Tsipras, Dimitris, and Vladu, Adrian. Towards deep learning models resistant to adversarial attacks, 2017. URL https://arxiv.org/abs/1706.06083.

Misra, Diganta. Mish: A self regularized non-monotonic neural activation function. *CoRR*, abs/1908.08681, 2019. URL http://arxiv.org/abs/1908.08681.

Ramachandran, Prajit, Zoph, Barret, and Le, Quoc V. Searching for activation functions. *CoRR*, abs/1710.05941, 2017. URL http://arxiv.org/abs/1710.05941.

Yun, Sangdoo, Han, Dongyoon, Oh, Seong Joon, Chun, Sanghyuk, Choe, Junsuk, and Yoo, Youngjoon. Cutmix: Regularization strategy to train strong classifiers with localizable features. *CoRR*, abs/1905.04899, 2019. URL http://arxiv.org/abs/1905.04899.

Zhang, Hongyi, Cissé, Moustapha, Dauphin, Yann N., and Lopez-Paz, David. mixup: Beyond empirical risk minimization. *CoRR*, abs/1710.09412, 2017. URL http://arxiv.org/abs/1710.09412.