Effects of different SOTA Data Augmentation Techniques on the Calibration of Deep Neural Networks

Riyanshu Jain
B20AI060
jain.59@iitj.ac.in

Ruthvik K
B20AI037
ruthvik.2@iitj.ac.in

Shubh Soni
B20BB039
soni.11@iitj.ac.in

Abstract

Deep Learning architectures are prone to show undesirable behaviors such as tendency to memorization and are vulnerable to adversarial samples. Although they have the capacity to develop strong representational spaces, which are essential for tackling complex learning tasks, but, these representations are frequently prone to overfitting due to the model capacity needed to capture them. So in order for DNN's to generalise well, proper regularization is needed. In our work, we propose implementing various research papers related to the field of data augmentations such as mixup, cutout and CutMix. We further compare their performances on the basis of accuracy and calibration metrics such as expected calibration error to show their robustness. Finally, we combine all the individual data augmentation techniques and propose a new framework that apply mixup, cutout and CutMix randomly on every image and study the effects of applying them together. We find that applying all the augmentations certainly increases the performance of the model. We also perform an ablation study where we apply all the augmentations on a single image and then a combination of two augmentations to further compare the results. The performance is also tested on OOD datasets such as CIFAR-10C and CIFAR-100C to strengthen our argument about robustness and calibration.
Keywords *- Data Augmentation, Calibration, Deep Neural Networks, Image classification*

1. Introduction and Related Work

One of the major challenges that modern DNN's possess is that a model is needed to be accurately calibrated [3] to capture the uncertainty of the predictions [2]. This is where data augmentation techniques come into picture to improve the generalization capabilities instead of overfitting [6]. **Mixup** was first introduced by Hongyi Zhang [10], in which we train a neural network on convex combinations of pairs of examples and their labels. By doing so, mixup regularizes the neural network to favor simple linear behavior in-between training examples. In the papers [10] [8] the authors claim that mixup improves the generalization and calibration of state-of-the-art neural network architectures. Another such technique introduced is **cutout** [1], where we randomly mask out square regions of input during training to improve robustness and overall performance of DNN's. The authors claim that it can be also be used in conjunction with existing forms of data augmentation and other regularizers to further improve model performance. But this method for regional dropout removes informative pixels on training images by overlaying a patch of black pixels. Such removal is not desirable because it leads to information loss and inefficiency during training. **CutMix** [9] solves this problem by combining both mixup and cutout into one algorithm where patches are cut and pasted among training images along with the ground truth labels are also mixed proportionally to the area of the patches. The authors claim that CutMix improves the model robustness against input corruptions and its out-of-distribution detection performances. The paper [7] talks about how data augmentation improves robustness.

Our work is fundamentally built on top of these data augmentation techniques where we compare the performance in all the three cases and further studies the effects of combining them together. We apply all of the stated augmentation techniques on every image sequentially and compare their performance with the baselines.

2. Methodology

Mixup: In mixup, we take two input images randomly selected from the mini batch having data as x_A and x_B and the targets as y_A and y_B. The classifier is now trained not only on the original training data but also in the vicinity of each training sample as the mixup linearly combines the two images as given in the Eq. (1).

$$x_m = \lambda * x_A + (1 - \lambda) * x_B \tag{1}$$

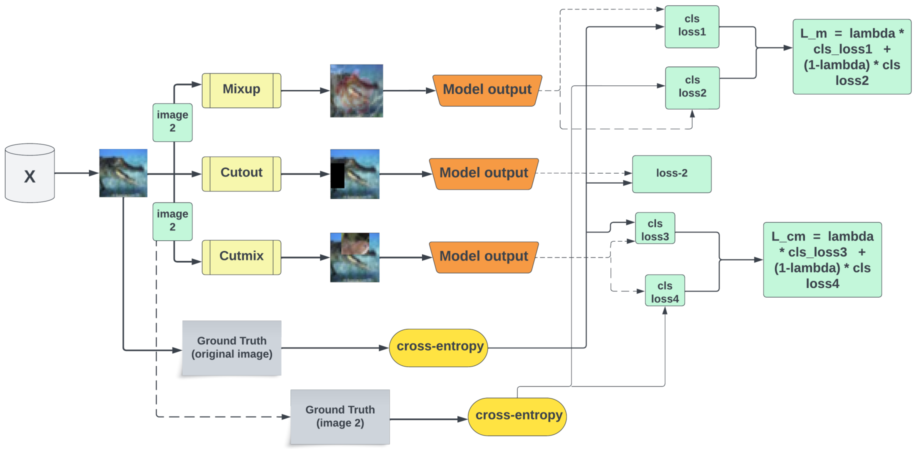

Figure 1. Framework of different augmentations including a sample image of each and with the respective loss function

The loss term for the classifier is also linearly combined as shown in Eq. (2). Here, $f(x)$ is the output from model.

$$.\mathcal{L}_m = \lambda * \mathcal{L}_{CE}(f(x_m), y_A) + (1-\lambda) * \mathcal{L}_{CE}(f(x_m), y_B) \tag{2}$$

Cutout: In cutout, the input image x_i is zeroed out in some parts as shown in Eq. (3). Basically a binary mask M is created with black pixels (0 denotes a zero matrix of appropriate H*W) indicating where to drop out and a certain section of image is replaced with this binary mask. The loss term for the classifier is cross entropy loss L_{CE}. The motivation behind cutout comes from dropout.

$$x_{co} = M \odot x_i + (1-M) \odot 0 \tag{3}$$

CutMix: CutMix uses the motivation of both the mixup and cutout, and the input images x_A and x_B are mixed using a binary mask M indicating a certain section of image is replaced with this binary mask. The difference here is that instead of using zero pixels, the section of x_A gets filled by the same section of x_B as shown in Eq. (4). The term λ is sampled from beta distribution Beta(α, α). The main advantages of using CutMix over mixup and cutout is that it generates more locally natural image than mixup does and cutout suffers from loss of information due to black pixels.

$$x_{cm} = M \odot x_A + (1-M) \odot x_B \tag{4}$$

The coordinates of the box (cropping region) is $B = (r_x, r_y, r_w, r_h)$ and the sampling of the coordinates is done using Eq. (5).

$$\begin{aligned} r_x &\sim U(0, W), \quad r_w = W\sqrt{1-\lambda} \\ r_y &\sim U(0, H), \quad r_h = H\sqrt{1-\lambda} \end{aligned} \tag{5}$$

Here, one thing to note is that in cutout, the length of the box is pre-defined but in cutmix it is not pre-defined and is changed as per the sampling.

The loss term for the classifier defined in Eq. (6) is same as miuxp loss. Here, $f(x)$ is the output from model.

$$\mathcal{L}_{cm} = \lambda * \mathcal{L}_{CE}(f(x_{cm}), y_A) + (1-\lambda) * \mathcal{L}_{CE}(f(x_{cm}), y_B) \tag{6}$$

Algorithm 1 Our approach

Require: A model to be trained f
Require: The original training Dataset $(X, Y) \in D$, factor α, β_{cm}, n_{holes}, $length$ of the box in cutout
Initialization: Model training $f(x)$
for `i = 1, ... , Max_epoch` **do**
  Sample a batch (x, y) from D, $r = U(0, 1)$
  if $0 <= r < 0.3$ **then**
    Apply mixup on each image x_A in x and another randomly selected image x_B from x
    Calculate mixup loss using $\lambda = \beta(\alpha, \alpha)$ and backpropagate
  else if $0.3 <= r < 0.6$ **then**
    Apply cutout on images in x using n_{holes} and $length$
    Calculate L_{CE} loss and backpropagate
  else if $0.6 <= r < 0.9$ **then**
    Apply CutMix on each image x_A in x and another randomly selected image x_B from x
    Calculate CutMix loss using $\lambda = \beta(\beta_{cm}, \beta_{cm})$ and backpropagate
  else
    No augmentations on images in x
    Calculate L_{CE} loss and backpropagate
  end if
end for

Our contribution: We define a random variable r, sampled from a Uniform distribution $U(0, 1)$. Then we assign

Model	RN32x4		WRN40-2	
Augmentation	Acc↑	ECE↓	Acc↑	ECE↓
none	94.26	0.0342	**95.18**	0.0306
mixup	95.23	0.0328	94.73	0.0429
cutout	95.54	**0.0223**	94.56	0.0208
CutMix	95.82	0.0248	94.8	0.0291
Ours(1)	92.5	0.3097	91.28	0.2687
Ours(2)	**95.9**	0.1149	94.86	**0.0187**

Table 1. Results on the CIFAR-10 dataset. Acc↑ denotes that higher is better and ECE↓ denotes that lower is better. In Ours(1) algorithm, we apply all the above augmentations on each image which suffers from a lot of information loss as shown in Appendix A, but Ours(2) algorihtm is the one described in Algorithm 1

Model	RN32x4		WRN40-2	
Aug	Avg.Acc↑	Avg.ECE↓	Avg.Acc↑	Avg.ECE↓
none	78.35	0.1505	76.44	0.1559
mixup	**79.88**	**0.0832**	**78.02**	**0.0760**
cutout	76.34	0.1454	75.05	0.1341
CutMix	77.80	0.0887	75.15	0.0921
Ours(1)	72.98	0.1949	72.97	0.1676
Ours(2)	76.75	0.1027	76.01	0.1208

Table 2. Results on the CIFAR10-C dataset. Avg.Acc and Avg. ECE are calculated over the 19 corruptions.

Model	RN32x4		CIFAR100-C (RN32x4)	
Augmentation	Acc↑	ECE↓	Avg. Acc↑	Avg. ECE↓
none	76.88	0.0703	52.52	0.1995
mixup	78.07	**0.0353**	**55.16**	**0.0811**
cutout	78.02	0.0773	51.78	0.2254
CutMix	79.27	0.0417	52.39	0.1778
mixup+cutout	77.84	0.1862	54.05	0.1082
mixup+cutmix	77.34	0.2076	52.68	0.1382
cutmix+cutout	78.86	0.0467	50.97	0.2048
Ours(1)	74.94	0.2500	50.68	0.1530
Ours(2)	**79.53**	0.0412	52.56	0.1759

Table 3. Results on the CIFAR-100 dataset. Avg. Acc denotes the average accuracy calculated over 19 corruptions in the CIFAR-100 dataset, ↑ means higher is better and Avg. ECE denotes the average ECE over the 19 corruptions, ↓ means lower is better.

randomly a data augmentation approach to each batch of images based on this variable r. This means the classifier is now trained on various different kinds of images and not only with one given augmentation based images. As per the intuition this approach must outperforms all of the above techniques becuase the dataset representation is now more diverse and the model cannot show unidentified biases. Our algorithm is thus summarised in Algorithm Algorithm 1.

3. Experiments and Results

We have done an extensive set of experiments to compare the results between mixup, cutout and CutMix in terms of test accuracy and expected calibration error [5]. The datasets used are CIFAR-10 and CIFAR-100 containing 10 and 100 classes respectively. Each dataset consists of 32x32 colored images in 3 channels with 50000 training and 10000 test images. Framework used is PyTorch and Python.

The experiment settings are kept the same throughout all experiments by keeping a batch size of 128 and using SGD optimizer with initial learning rate of 0.05, momentum of 0.9 and weight decay of 5×10^{-4}. Trained for a total of 120 epochs, the learning rate is multiplied by 0.1 at $50, 75, 100$ epoch. Models considered for CIFAR-10 dataset are WideResNet-40-2 and ResNet-32x4 and for CIFAR-100 dataset, ResNet-32x4 is used. In case of cutout, only 1 cut ($n_holes = 1$) is made of size 16x16, in case of mixup $\alpha = 0.3$ is used and for CutMix $\beta_{cm} = 1$ is used.

After applying a standard data augmentation of CIFAR datasets like normalisation using mean and standard deviation, random crop of 32, random horizontal flip and random rotation of 15 degrees, the training is done on the configuration stated above. The results are shown in Tabs. 1 to 3. The framework of different augmentations is shown in Fig. 1

As evident from the results, in case of CIFAR-10, Ours(2) approach is best performing among all the augmentations with an accuracy score beating the rest of the values but in case of RN32x4, the ECE value is on a higher side which implies the calibration of the model suffers after applying the technique. In case of CIFAR-100 we did a more thourough study of each pair of augmentation technqiues applied in a combination and found that CutMix and cutout applied together outperforms the other pairs involving mixup. In this case also, accuracy of Ours(2) approach is best. But the ECE is also comparable to the best ECE obtained. Among mixup, cutout and CutMix we can see that using CutMix creates a good accuracy and ECE result indicating that it has best generalisation performance out of the three.

4. Analysis and Conclusion

Upon testing the Out-of-distribution performance on the CIFAR10-C and CIFAR100-C datasets [4], the mixup technique consistently outperforms rest of the other, followed by mixup+cutout. This proves that mixup is best in terms of robustness to noise and attacks. At the end, through this project we analyse the calibration and robustness of various data augmentation techniques and compared them to find an effective strategy and trade-off between accuracy and ECE. Our findings suggest using a random augmentation (as described in Algorithm 1) in every batch gives the best result and followed by CutMix. Cutout suffers from a loss of information due to its regional dropout nature which is not effective in OOD experiments as observed by the ECE value (it is getting under confident in its prediction). Finally mixup, as suggested in the paper [10], offers vicinal risk minimization and helps to avoid memorization of corropt labels. It can be further used in adversarial training as its OOD experiments suggest that it should be less sensitive to adversarial examples.

References

[1] Terrance Devries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout. *ArXiv*, abs/1708.04552, 2017. 1

[2] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseo Lee, Matthias Humt, Jianxiang Feng, Anna M. Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, M. Shahzad, Wen Yang, Richard Bamler, and Xiaoxiang Zhu. A survey of uncertainty in deep neural networks. *ArXiv*, abs/2107.03342, 2021. 1

[3] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1321–1330, 2017. 1

[4] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019. 3

[5] Jeremy Nixon, Michael W. Dusenberry, Linchuan Zhang, Ghassen Jerfel, and Dustin Tran. Measuring calibration in deep learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019. 3

[6] Sylvestre-Alvise Rebuffi, Sven Gowal, Dan Andrei Calian, Florian Stimberg, Olivia Wiles, and Timothy A Mann. Data augmentation can improve robustness. In *Advances in Neural Information Processing Systems*, volume 34, pages 29935–29948, 2021. 1

[7] Sylvestre-Alvise Rebuffi, Sven Gowal, Dan Andrei Calian, Florian Stimberg, Olivia Wiles, and Timothy A Mann. Data augmentation can improve robustness. In *Advances in Neural Information Processing Systems*, volume 34, pages 29935–29948, 2021. 1

[8] Sunil Thulasidasan, Gopinath Chennupati, Jeff A Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019. 1

[9] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 1

[10] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018. 1, 4

A. Appendix

Ours1 approach: We have also tried another approach where we applied all the augmentations on a single image, i.e., first applied mixup using $\alpha = 0.3$, then on the output images applied cutout with 1 hole of length 8x8 and finally on the output images applied CutMix with $\beta = 1$ (Fig. 4). The loss term used in this approach is the sum of all individual losses defined in Eq. (7).

$$\begin{aligned} \mathcal{L}_{total} = 1/3 \times [&L_{CE}(f(x_{out}), y_A) \\ &+\mathcal{L}_m(x_{out}, y_A, y_B) \\ &+L_{cm}(x_{out}, y_A, y_C)] \end{aligned} \tag{7}$$

where, x_{out} is the final image generated by combining all the augmentations, $f(x_{out})$ is the model's classification on the image x_{out}, y_A is the actual target, y_B is the label corresponding to the image used to create mixup augmentation and y_C is the label corresponding to the image used to create CutMix image.

The results of the above approach shows that the combination suffers from higher loss of information in images because of which the model is not able to generalise well and using more than one augmentation may not be as effective.

Figure 2. Original image

Figure 3. All augmentation

Figure 4. Comparison of information loss between the images

ECE: Expected Calibration Error is calculated as:

$$\text{ECE} = \sum_{i=1}^{n} \frac{|B_i|}{N} \big|\text{acc}(B_i) - \text{conf}(B_i)\big| \tag{8}$$

where, $|B_i|$ denotes the number of samples in bin B_i, $acc(B_i)$ is the average accuracy of samples in bin B_i and $conf(B_i)$ is the average confidence of samples in bin B_i calculated as average probability of prediction of samples in the corresponding bin.