Project - Noisy CIFAR-100

Brassat Alexandru

I. Introduction

Deep Learning necessitates a huge quantity of data in order to achieve satisfactory results. However, as datasets expand, the presence of noise becomes more likely. This issue presents a significant challenge to the training of machine learning models, as noise can introduce biases and degrade overall performance.

Research suggests that the quality of data might be more important than the architecture of models which train on it. Thus, the collection, filtering and usage of the data is an open field for study which is

In this project, the aim is to counteract the noise in the dataset and produce satisfying results. Below are some experiments which explore the impact of different techniques, model architectures and augmentations on the final performance.

II. Dataset

The project uses the CIFAR-100N-fine [1] dataset, which is based on CIFAR-100. This dataset consists of 40.2% real human annotation errors, as opposed to synthetic label noise. The errors follow a different distribution and are closer to real-world use cases.

III. Augmentation

Several augmentation techniques were used to improve the training process, and their performance results are detailed in table I.

TABLE I
Baseline Augmentation study

	Basic	**Flipped inputs**		**Gaussian Noise (σ)**			**Random Rotation**		
		Random Flip	**Flipped Dataset**	**0.005**	**0.01**	**0.1**	**$\pm$2°**	**$\pm$5°**	**$\pm$10°**
resnet18 (baseline)	28.09(14s)	27.24(14s)	31.19(41s)	27.55(14s)	28.09(14s)	27.09(14s)	28.00(14s)	27.99(14s)	27.25(14s)
resnext101_32x8d	44.98(93s)	44.5(93s)	48.74(186s)	45.78(93s)	45.78(93s)	38.69(93s)	45.67(93s)	42.86(93s)	41.5(93s)

Results of each augmentation method by model. Table cells denote validation accuracy, with time per epoch (in seconds) in parenthesis.

A. Image Flipping

Horizontal Image Flipping is a simple augmentation technique which has been shown to aid in training. I used two methods of performing this transformation: using pytorch's Random Flip transformation, and including the flipped images in the training dataset, effectively doubling its size.

The first method obtained slightly lower accuracy, while the second method showed a 3% improvement, at the cost of doubling the training time.

B. Gaussian Noise

Gaussian Noise is used in the cases when slight variations in the input should not entail changes in the output. This is the case in the task at hand, as the predicted object class of an image should remain the same, even if the input is noisy, as long as the object is still discernible. For this reason, only small values of σ were considered, namely 0.005, 0.01 and 0.1.

Using pytorch's image transformations, small values for σ showed a little improvement for resnext101 over no added noise. For resnet18, only $\sigma = 0.01$ had the best score, with no decrease from the non-augmented dataset.

C. Random Rotations

Random Rotations are another way to augment the input images without modifying their class. In my experiments, larger rotations only diminish the accuracy of the model. The only case when random rotations perform slightly better (+0.69% validation accuracy) than the basic model is for 2 degree rotations for the resnext model.

IV. Network Architecture

There is evidence [2] regularization techniques play a major part in the robustness of models, as sufficiently large models have an easy time memorizing the training data, regardless of the actual content (even when the data consists of purely random noise).

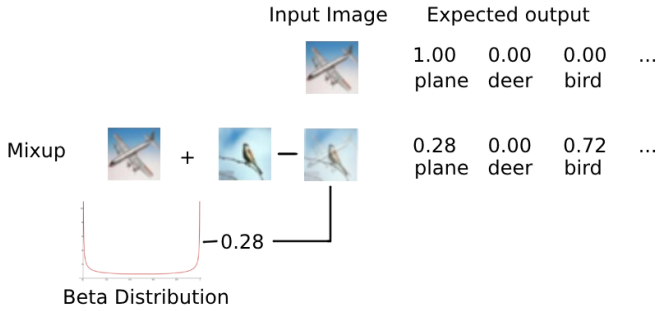
An interesting result that arose from the experiments below was that smaller networks stopped improving after reaching 50-60% accuracy on the training data, which is close to the ratio of clean data in the training set. While the same is true for larger networks, the size increase meant a faster convergence, with a steeper increase in training accuracies, indicating a predisposition to overfitting.

A. Loss Function

The best results were achieved using Cross Entropy Loss. MSE Loss was also considered, but proved to be significantly slower to train.

The Cross Entropy Loss in pytorch had to be customized in order to support probability vectors as input, so it could work with a one-hot representation of the output classes.

B. Mixup



Mixup [3] is a data-agnostic augmentation technique which leverages one-hot encodings of the output data. *Mixup* augments the training process by altering each input image to be a linear interpolation between two training data points.

Mixup uses a Beta Distribution with $\alpha = \beta$ in order to pick the mix ratio between two examples, introduing a new hyperparameter in the process: α. Values between 0 and 1 generate images which closely resemble one of the two inputs, while values over 1 generate a more even mix of the two. A value of $\alpha = 1$ transforms the distribution into a uniform one.

This method has proved useful in reducing overconfidence in classification models [4], by virtue of the aforementioned fuzzy labels used in training, which better model tasks where uncertainty is a big factor.

C. MentorNet

MentorNet [5] is a technique which uses a neural network in order to train a "Student Network", by dynamically determining a curriculum for the Student to learn from. It has been used for working with noisy data, as the mentor learns to ignore high loss training examples, which are likely wrongly labeled.

The mentor and the student are trained alternatively, one at a time, such that the student can learn from the generated curriculum, and the mentor can adjust according to the performance of the student.

D. MentorMix

MentorMix [6] is a method which combines the previous two techniques in order to create models which are more resilient to noisy labels. It uses a teacher network in order to weight the samples in a mini-batch based on a confidence score from 0 to 1, depending on how likely a sample is to be labeled correctly. This results in filtering out the mislabeled training data, allowing for a more robust training process.

The aforementioned confidence score is used as a probability score for *Mixup*, such that only samples which are likely to be correctly labeled will be mixed. An optional second weighting can be done on the mixed-up mini-batch, in order to entirely remove high loss examples. One variant of *MentorMix* presented in the original paper, which was used in the experiments below, replaces the mentor network with a simpler weighting process: for each mini-batch, assign 0 to any training examples above the γ_pth percentile, while allowing the rest to be part of the Mixup process. An exponential moving average is employed to smooth over sudden changes in mini-batch losses.

V. Results

Multiple network architectures were tested: resnet18 (11.227.812 parameters), efficientnet b5 (28.545.684 parameters) and resnext101 (86.947.236 parameters). Each were trained starting from pretrained weights on ImageNet and finetuned for the task at hand. The models were tested under different configurations regarding augmentation and regularization methods (*MentorMix* and *Mixup*). The models were judged based on accuracy, number of parameters and training speed.

Training was done with mini-batch size of 64 samples and an early stopping patience of 10 epochs (following the best accuracy on the validation set). I used the SGD optimizer with a learning rate of 0.001, and a custom implementation of the Cross Entropy Loss Function, which works with the fuzzy labels of *Mixup*. When running *MentorMix* I also added a weight decay of 0.0001, in order to more closely follow the original paper's formulation.

The best performance, 53.92 accuracy, was obtained (and not shown in the table) with *resnext101_32x8d* mentormix with percentile 70, $\alpha = 0.4$ and dataset flipping.

TABLE II
MentorMix & Mixup Comparison

	Basic	**Mixup (α)**				**MentorMix (percentile)**		
		0.2	**0.4**	**1**	**2**	**30**	**50**	**70**
resnet18 (baseline)	28.09(14s)	29.15(14s)	30.61(14s)	31.30(14s)	32.54(14s)	33.87(20s)	33.19(20s)	33.14(20s)
efficientnet_b5	39.31(110s)	-	-	-	45.67(111s)	-	-	-
resnext101_32x8d	44.98(93s)	48.12(93s)	49.17(93s)	52.17(94s)	51.07(93s)	52.34(120s)	52.63(120s)	52.30(120s)

Results of each augmentation method by model. Table cells denote validation accuracy, with time per epoch (in seconds) in parenthesis. All *MentorMix* tests were run with $\alpha = 1$ for *Mixup*.

VI. Conclusion and Further Work

Out of all augmentation methods, dataset flipping yielded the best accuracy increase. It doubles the size of the training data, but it also doubles the training time. On the other hand, Gaussian Noise and Random Flipping did not show any significant improvements over the basic models.

Mixup had a positive impact on accuracy, with the greatest accuracy increase being achieved when $\alpha = 1$ (uniform distribution). It is a relatively inexpensive method which does not increase the running time of training, but managed to obtain a 7% accuracy increase.

The presented solution could be improved by experimenting with more hyperparameter configurations, and by implementing more complex MentorNets and Mixup methods.

Further experimentation with regularization techniques could improve the accuracy and robustness of the model. Ensambling can also be employed in order to boost the confidence in the end result.

References

[1] Jiaheng Wei, Zhaowei Zhu, Hao Cheng, Tongliang Liu, Gang Niu, and Yang Liu. Learning with noisy labels revisited: A study using real-world human annotations. In *International Conference on Learning Representations*, 2022.

[2] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization, 2017.

[3] Hongyi Zhang, Moustapha Cissé, Yann Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *ArXiv*, abs/1710.09412, 2017.

[4] Sunil Thulasidasan, Gopinath Chennupati, Jeff Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. On mixup training: Improved calibration and predictive uncertainty for deep neural networks, 2020.

[5] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels, 2018.

[6] Lu Jiang, Di Huang, Mason Liu, and Weilong Yang. Beyond synthetic noise: Deep learning on controlled noisy labels, 2020.