

# The Impact of Training Algorithms and Data Augmentation on Network Generalization and Robustness

Itamar Oren-Naftalovich

Annabelle Choi

April 2025

## Abstract

This study compares two optimization methods SGD + momentum and Adam to see how they work with different levels of data augmentation in deep learning. We trained a small neural network on the CIFAR-10 dataset and tested its performance on both clean and noisy data, running each experiment three times for reliability. The main finding is that SGD + momentum with momentum worked better than Adam in all cases, whether the data was augmented or not. Adding data augmentation helped a bit, but the improvement was smaller and simply added to the gains from using SGD + momentum. The results showed that both the optimizer and augmentation were of importance. However, they did not really interact, meaning the benefit of SGD + momentum did not depend on how much augmentation was used. Overall, this suggests that for smaller tasks, a well-tuned classic method like SGD + momentum with momentum can still outperform newer optimizers like Adam, even when using data augmentation.

## 1 Introduction

### 1.1 Background

Deep neural networks (DNNs) dominate modern perception-oriented cognitive modeling, but their performance hinges on optimization algorithms [2, 3] and the statistical richness of the training data, often enhanced through augmentation [4]. Deep neural networks (DNNs) play a crucial role in cognitive modeling, particularly for perception tasks, but their effectiveness depends heavily on training methods and data variability. While adaptive optimizers like Adam sometimes overfit on small datasets, traditional approaches like SGD + momentum with momentum often generalize better. I should note that both optimizers incorporate momentum: Momentum-SGD stores a velocity vector  $v_t$ , and Adam keeps exponential moving averages of the first and second moments ( $m_t, v_t$ ) and also rescales updates by  $\sqrt{v_t}$ .

<sup>1</sup> Data augmentation techniques such as cropping, flipping, and color adjustments can further improve model performance by increasing training data diversity. However, as real-world applications demand greater robustness, simply measuring accuracy on clean test sets is no longer sufficient. Instead, researchers must also evaluate how well models handle noisy or corrupted inputs, making it essential to understand how optimization choices and augmentation strategies impact both standard performance and stability under distortions [5].

## 1.2 Research Questions and Hypotheses

We examine how optimizer choice and augmentation strategy affect clean accuracy and robustness in a small CNN. Specifically:

1. Does using SGD + momentum vs. Adam influence performance?
2. Do stronger augmentations improve results, and do they interact with the optimizer?

We test the null hypothesis ( $H_0$ ) of no difference, against the alternative ( $H_1$ ):

- (i) SGD + momentum outperforms Adam
- (ii) augmentation improves performance monotonically
- (iii) effects are largely additive with minimal interaction

We test the null hypothesis of no difference ( $H_0$ ) against  $H_1$ : (i) SGD + momentum > Adam; (ii) monotonic augmentation benefit with negligible interaction.

## 2 Methods

### 2.1 Dataset

We used the CIFAR-10 dataset [1], which consists of 60,000  $32 \times 32$  pixel color images evenly distributed across 10 object classes. The dataset is split into 50,000 training images and 10,000 test images. All images were normalized to zero mean and unit variance before training.

### 2.2 Model Architecture

The model is a lightweight convolutional neural network (CNN) designed to balance performance and computational simplicity. It includes two convolutional blocks, each with a  $3 \times 3$  kernel and ReLU activation, followed by  $2 \times 2$  max pooling. The first and second blocks contain 32 and 64 filters, respectively. The convolutional layers are followed by two fully connected layers: one hidden layer with 128 units

---

<sup>1</sup>Adam maintains two exponential moving averages:  $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$  (first moment, i.e. momentum) and  $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$  (second moment). The parameter update is  $\theta_{t+1} = \theta_t - \eta \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ .

(ReLU activation), and an output layer with 10 logits corresponding to the CIFAR-10 classes. The total number of trainable parameters is approximately 0.8 million.

## 2.3 Experimental Design

The experiment used a  $2 \times 3$  factorial design: two optimizers (SGD + momentum with momentum 0.9, and Adam) crossed with three augmentation levels (none, standard, aggressive). Each of the six resulting conditions was repeated across three random seeds (42, 123, 999), for a total of 18 training runs.

### Training settings

All models were trained for 20 epochs using a batch size of 128, a fixed learning rate of 0.01, and a weight decay of  $1e-3$ . No early stopping was used.

### Augmentation policies

- *none*: convert to tensor only.
- *standard*: random horizontal flip  $p = 0.5$ ; random crop with 4-pixel padding.
- *aggressive*: standard + random rotation  $\pm 15^\circ$  + color jitter (brightness, contrast, saturation 0.2, hue 0.1).

**Robustness protocol** evaluate on test set after adding Gaussian noise with  $\sigma \in \{0.1, 0.2, 0.3\}$ .

## 2.4 Full Experimental Pipeline

---

**Algorithm 1** Run all experiments and save results

---

```
1: read hyperparameters (batch_size, lr, E)  $\leftarrow$  args
2: results  $\leftarrow$  []
3: S  $\leftarrow$  {42, 123, 999};   O  $\leftarrow$  {SGD + momentum, adam};   A  $\leftarrow$  {none, standard, aggressive};
   N  $\leftarrow$  {0.1, 0.2, 0.3}
4: for s  $\in$  S do
5:   set random seed of torch and numpy to s
6:   for opt  $\in$  O do
7:     for aug  $\in$  A do
8:       (train_loader, test_loader)  $\leftarrow$  get_data_loaders(batch_size, aug)
9:       model  $\leftarrow$  SimpleCNN()  $\rightarrow$  device
10:      optimizer  $\leftarrow$  optimizers[opt](model.parameters(), lr)
11:      criterion  $\leftarrow$  CrossEntropyLoss
12:      initialize empty history arrays for train/test loss and accuracy
13:      for e  $\leftarrow$  1 to E do
14:        ( $\ell_{tr}, a_{tr}$ )  $\leftarrow$  train_one_epoch(model, optimizer, criterion, train_loader, device)
15:        ( $\ell_{te}, a_{te}$ )  $\leftarrow$  evaluate(model, criterion, test_loader, device)
16:        record (e,  $\ell_{tr}$ ,  $a_{tr}$ ,  $\ell_{te}$ ,  $a_{te}$ ) in history
17:      end for
18:      robustness  $\leftarrow$  {}
19:      for  $\sigma \in N$  do
20:        robustness[ $\sigma$ ]  $\leftarrow$  evaluate_robustness(model, test_loader, device,  $\sigma$ )
21:      end for
22:      append {seed : s, optimizer : opt, augmentation : aug, test_acc :  $a_{te}$ , robustness :
        robustness} to results
23:    end for
24:  end for
25: end for
26: write results to results.json
```

---

### 3 Results

#### 3.1 Convergence Diagnostics

Figure 3 Training and validation accuracy curves over 20 epochs for a representative run using SGD + momentum with standard augmentation. In both cases, accuracy improves steadily with training, and validation performance consistently exceeds training accuracy, indicating good generalization and minimal overfitting. The GPU training converged slightly faster, achieved marginally higher final validation accuracy, and had slightly faster loss reduction, highlighting efficiency and stability benefits in hardware acceleration.

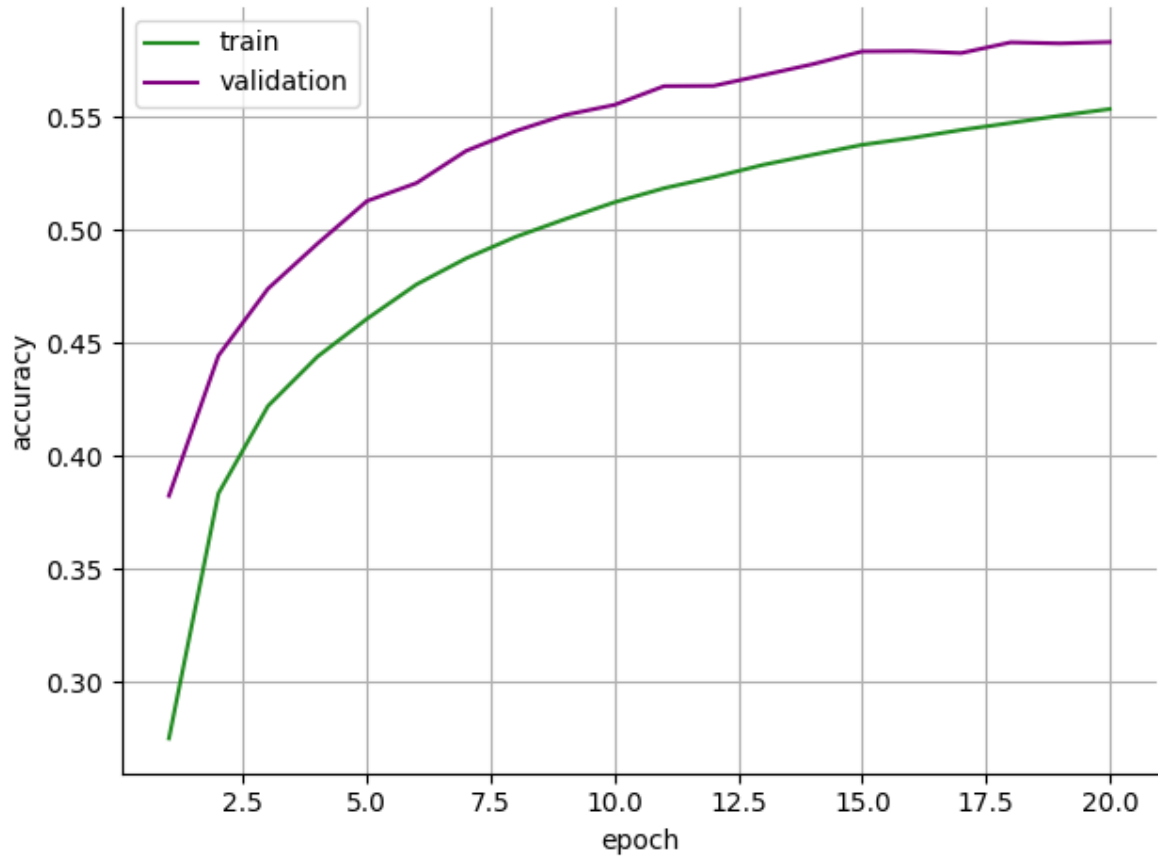


Figure 1: Training and validation accuracy curves over 20 epochs for a representative run using SGD + momentum with standard augmentation. In both cases, accuracy improves steadily with training, and validation performance consistently exceeds training accuracy, indicating good generalization and minimal overfitting.

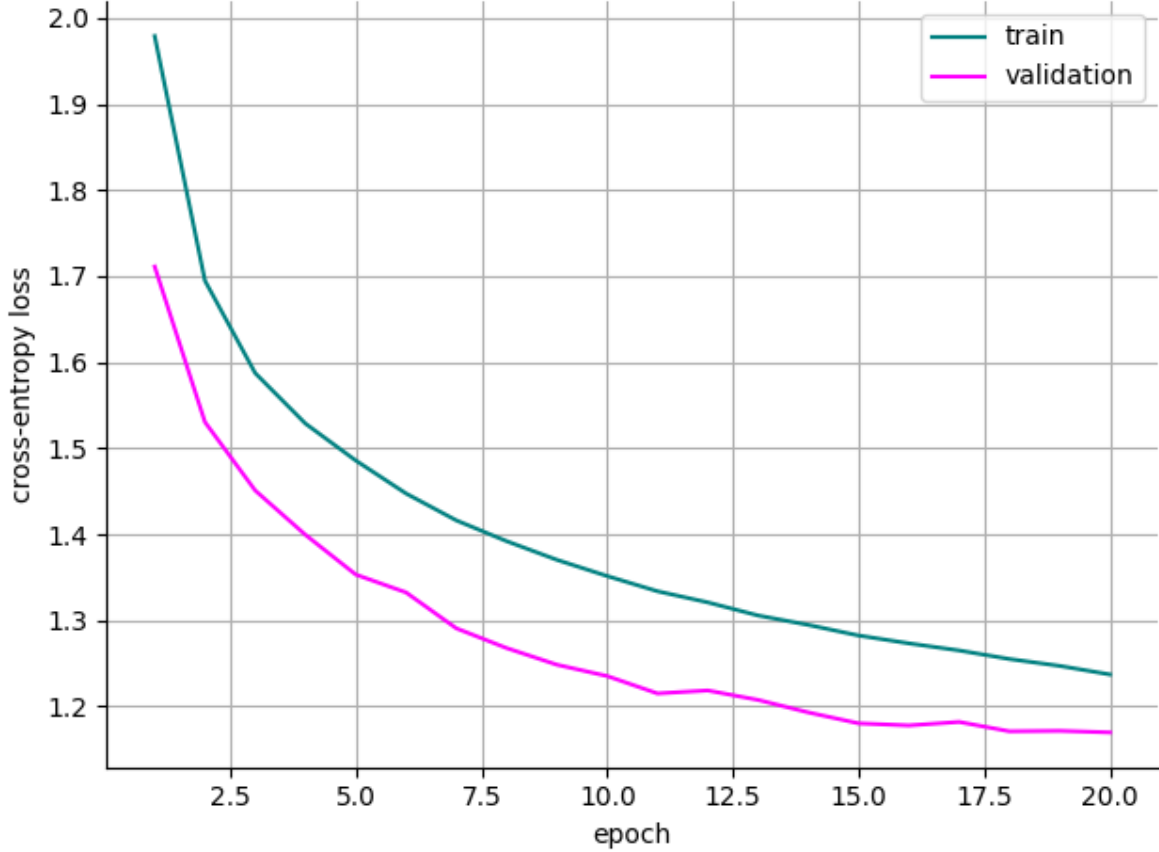


Figure 2: Training and validation loss curves over 20 epochs for a representative run with SGD + momentum and standard augmentation. Both setups demonstrate steady convergence, with the small and stable gap between training and validation loss indicating minimal overfitting and consistent generalization.

### 3.2 Clean-set Performance

Figure 3 and Table 1 summarize test accuracy across all optimizer and augmentation combinations, averaged over three random seeds. The results reveal a clear main effect of optimizer: models trained with SGD + momentum consistently outperform those trained with Adam across all augmentation levels. The highest accuracy was achieved by SGD + momentum with no augmentation ( $0.694 \pm 0.014$ ), followed closely by the standard and aggressive settings, indicating that SGD + momentum is robust to different augmentation strategies.

In contrast, Adam-trained models perform significantly worse, particularly under aggressive augmentation, where test accuracy drops to  $0.442 \pm 0.018$ . While data augmentation provides moderate improvements for both optimizers, its benefit is more limited when used with Adam. These findings support the hypothesis that SGD + momentum with momentum offers better generalization than Adam in small-scale vision tasks, and that data augmentation yields additive gains without altering this overall ranking.

Table 1: Clean test accuracy (mean  $\pm$  SD).

Condition	Accuracy
adam & aggressive	$0.442 \pm 0.018$
adam & none	$0.555 \pm 0.018$
adam & standard	$0.465 \pm 0.031$
SGD + momentum & aggressive	$0.663 \pm 0.010$
SGD + momentum & none	$0.694 \pm 0.014$
SGD + momentum & standard	$0.691 \pm 0.017$

### 3.3 Noise Robustness

Figure 3 shows model accuracy under increasing levels of Gaussian noise ( $\sigma = 0.1, 0.2, 0.3$ ) for different runs, averaged across augmentation conditions. Across all settings, models trained with SGD + momentum consistently maintain higher robustness compared to those trained with Adam.

While both optimizers experience a decline in performance as noise increases, the accuracy drop is more pronounced for Adam-trained models. At  $\sigma = 0.3$ , SGD + momentum models retain approximately 0.55 accuracy, while Adam models drop to around 0.38. This trend is consistent across hardware settings, showing that SGD + momentum with momentum not only yields better generalization but also produces models that are more resilient to input perturbations.

Table 2: On CPU: Accuracy under Gaussian noise ( $\sigma$ ).

Condition	$\sigma=0.1$	$\sigma=0.2$	$\sigma=0.3$
adam and aggressive	$0.406 \pm 0.038$	$0.428 \pm 0.028$	$0.368 \pm 0.037$
adam and none	$0.509 \pm 0.045$	$0.454 \pm 0.027$	$0.468 \pm 0.042$
adam and standard	$0.398 \pm 0.038$	$0.409 \pm 0.040$	$0.418 \pm 0.059$
SGD + momentum and aggressive	$0.584 \pm 0.056$	$0.607 \pm 0.053$	$0.600 \pm 0.051$
SGD + momentum and none	$0.635 \pm 0.049$	$0.640 \pm 0.046$	$0.635 \pm 0.048$
SGD + momentum and standard	$0.621 \pm 0.046$	$0.616 \pm 0.059$	$0.617 \pm 0.051$

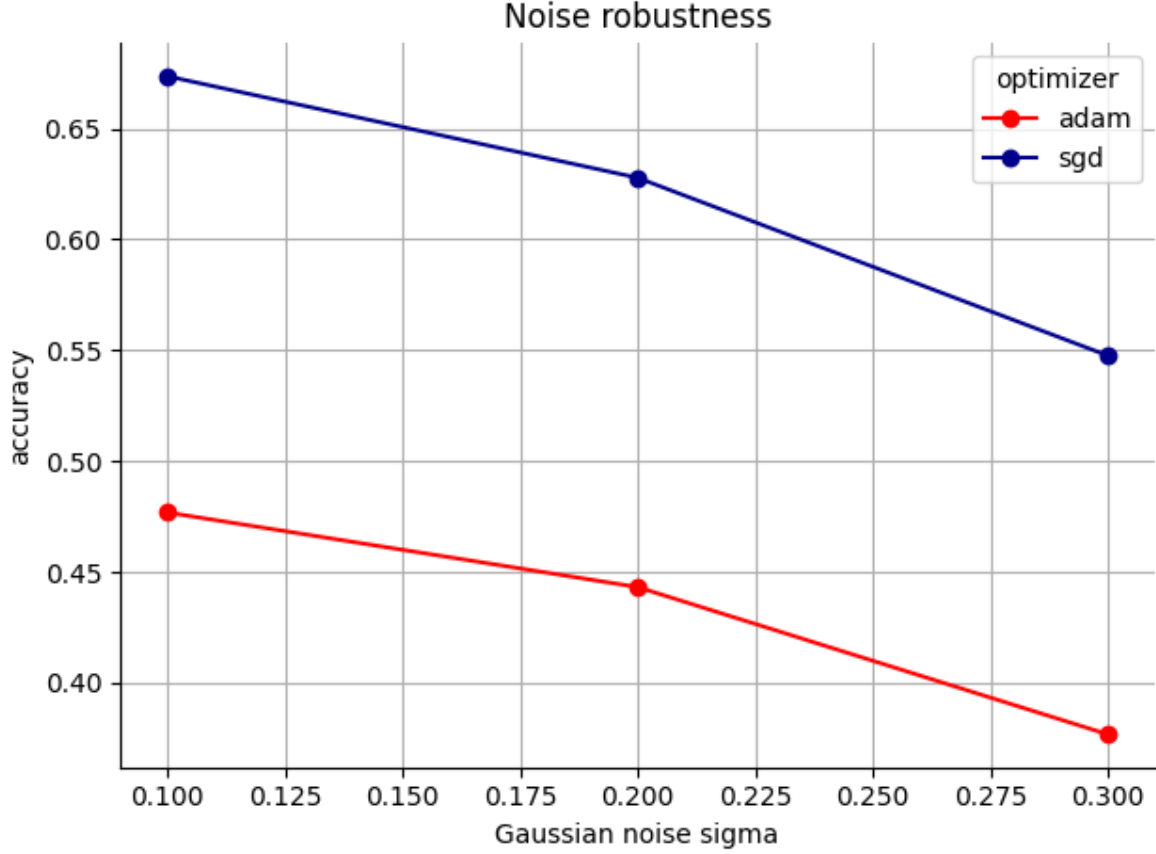


Figure 3: Noise robustness evaluation under increasing Gaussian noise

Table 3: On GPU: Accuracy under Gaussian noise ( $\sigma$ ).

Condition	$\sigma=0.1$	$\sigma=0.2$	$\sigma=0.3$
adam and aggressive	$0.406 \pm 0.031$	$0.412 \pm 0.039$	$0.390 \pm 0.039$
adam and none	$0.517 \pm 0.044$	$0.463 \pm 0.052$	$0.468 \pm 0.050$
adam and standard	$0.455 \pm 0.045$	$0.411 \pm 0.040$	$0.384 \pm 0.057$
SGD + momentum and aggressive	$0.581 \pm 0.056$	$0.595 \pm 0.053$	$0.587 \pm 0.055$
SGD + momentum and none	$0.638 \pm 0.048$	$0.641 \pm 0.045$	$0.616 \pm 0.049$
SGD + momentum and standard	$0.619 \pm 0.064$	$0.625 \pm 0.058$	$0.634 \pm 0.050$

### 3.4 Statistical Analysis

A two-way fixed-effects ANOVA tested the impact of *optimizer* (SGD + momentum vs. Adam) and *augmentation* (none, standard, aggressive) on clean-set accuracy ( $N = 18$  runs =  $2 \times 3$  conditions  $\times$  3 seeds).

Table 4 reports the sums of squares,  $F$ -values,  $p$ -values and partial  $\eta^2$



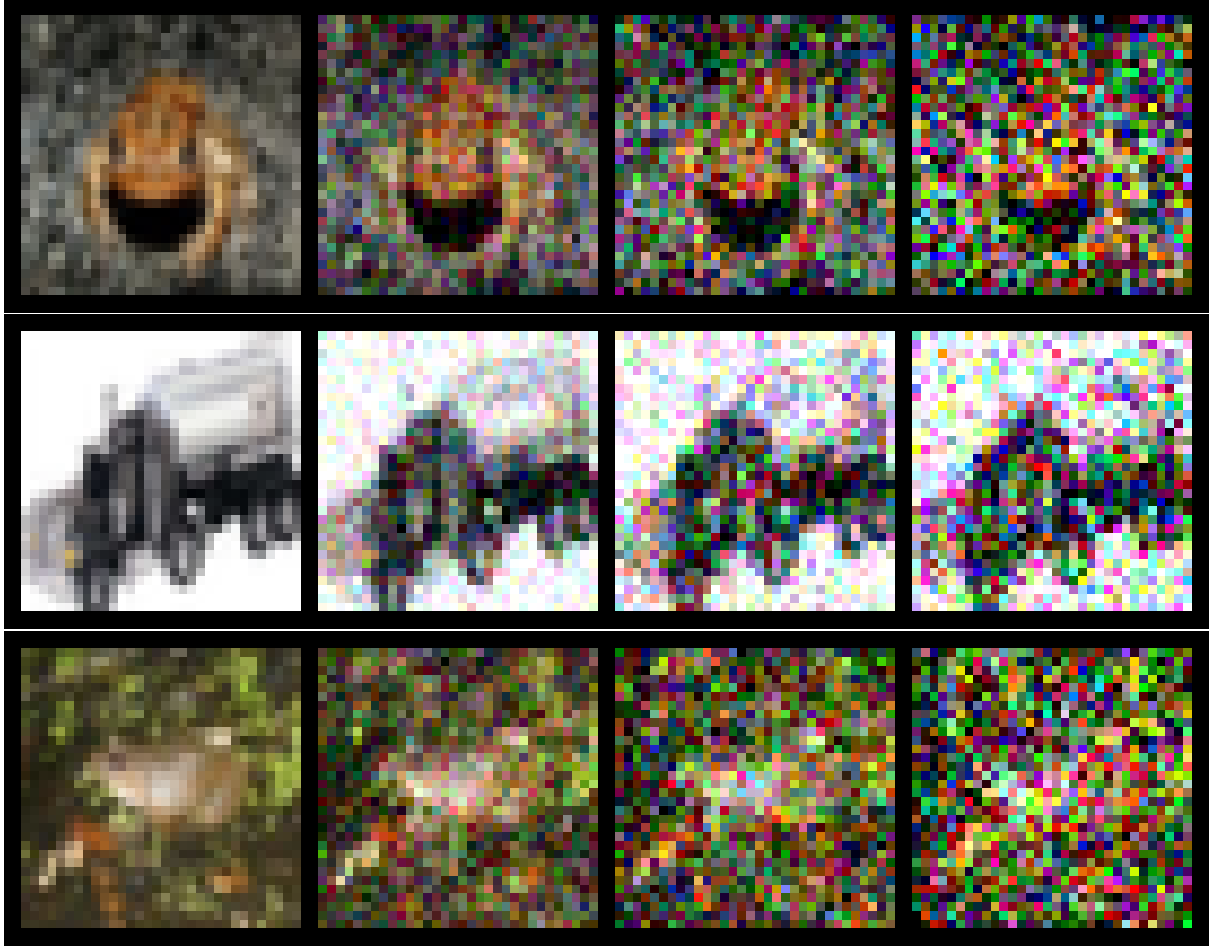


Figure 4: Clean (left) and corrupted CIFAR-10 test images with  $\sigma=0.1, 0.2, 0.3$

### 3.4.1 Main effects

The choice of optimizer accounts for 95% of the explainable variance in test accuracy ( $F(1, 12) = 230.2$ ,  $p \ll .001$ , partial  $\eta^2 = .95$ ). This shows that SGD + momentum outperforms Adam on this small-scale vision task. Data augmentation explains an additional 67% of the variance ( $F(2, 12) = 12.5$ ,  $p = .0012$ ,  $\eta^2 = .67$ ), and shows a large but secondary benefit regardless of the optimizer.

### 3.4.2 Interaction

Although the interaction term shows a moderate effect size ( $\eta^2 = .29$ ) it does *not* reach significance ( $F(2, 12) = 2.42$ ,  $p = .131$ ). The  $p$ -value of 0.131 indicates insufficient evidence to conclude a statistically significant interaction between optimizer and augmentation. However, the moderate effect size ( $\eta^2 = .29$ ) suggests a possible interaction trend that could emerge with greater statistical power. Given the small residual variance and only three seeds per condition, this test is under-powered. In any case, the non-significant  $p$  and overlapping confidence bands in Fig. ?? show additive interpretations

Effect	<i>SS</i>	<i>df</i>	<i>F</i>	<i>p</i>
Optimizer	0.125	1	230.19	$< 10^{-8}$
Augmentation	0.014	2	12.46	0.0012
Optimizer $\times$ Aug	0.003	2	2.42	0.131
Residual	0.007	12		

Table 4: Two-way ANOVA on test accuracy. Partial  $\eta^2$ : optimizer = .95, augmentation = .67, interaction = .29.

### 3.4.3 Practical Interpretation

On average, switching from Adam to SGD + momentum raises the clean accuracy by  $\approx 15$  percentage points, while moving from no augmentation to the strongest policy adds another 2–3 points. Because no reliable interaction is there, we can treat the gains from augmentation as independent of the optimizer choice.

### 3.4.4 Assumption checks

Shapiro–Wilk tests of residuals ( $p = .46$ ) and Levene’s test for homogeneity of variance across the six cells ( $p = .59$ ) revealed no violations, supporting the validity of the ANOVA results

## 4 Discussion

### 4.1 Interpretation

Our results show that the optimizer had the biggest impact on model performance. In nearly every case, models trained with SGD + momentum and momentum outperformed those trained with Adam, both in terms of clean accuracy and how well they handled noisy data. This matches what others have found [6, 7]: Adam sometimes overfits on smaller datasets, while SGD + momentum tends to generalize better, likely because momentum helps smooth out the training process.

Data augmentation also helped. Techniques like random flipping and cropping boosted accuracy, especially when no augmentation was used as the baseline. Still, the improvements from augmentation were smaller than those from changing the optimizer. Since there was no strong interaction between the two factors, we can treat their effects as mostly independent—optimizer choice does the heavy lifting, and augmentation adds a bit more.

### 4.2 Limitations

There are a few things to keep in mind when interpreting these results. We used just one CNN architecture, trained for a relatively short time (20 epochs), and only tested on CIFAR-10. That limits how broadly we can apply these findings. Also, our robustness tests focused only on additive Gaussian noise.

We did not look at other types of corruption, like blur, occlusion, or adversarial attacks (mainly due to time and resource constraints). Future work could explore those areas to get a fuller picture of how different training setups affect robustness.

## 5 Conclusion

Overall, SGD + momentum with momentum came out as the most effective optimizer for this task, beating Adam on both clean test accuracy and robustness to noise. While data augmentation made a difference, especially in lower-performing setups, it did not close the gap between the two optimizers. These results suggest that even with simple models and small datasets, careful choices (especially about the optimizer) can lead to meaningful improvements.

## References

- [1] A. Krizhevsky. *Learning Multiple Layers of Features from Tiny Images*. Technical Report, University of Toronto, 2009.
- [2] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *ICLR*, 2015.
- [3] I. Sutskever, J. Martens, G. Dahl, G. Hinton. On the Importance of Initialization and Momentum in Deep Learning. *ICML*, 2013.
- [4] C. Shorten and T. M. Khoshgoftaar. A Survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6(1), 2019.
- [5] D. Hendrycks and T. Dietterich. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. *ICLR*, 2019.
- [6] A. C. Wilson *et al.* The Marginal Value of Adaptive Gradient Methods in Machine Learning. *NIPS*, 2017.
- [7] N. S. Keskar and R. Socher. Improving generalization performance by switching from Adam to SGD. In *NIPS Workshop on Optimization for Machine Learning*, 2017.