Residual Network Architecture with Dynamic Data Augmentation, Squeeze-and-Excitation, and Stochastic Weight Averaging for Robust CIFAR-10 Image Classification

Hamza, A., & Zubairi, S.
Department of Electrical and Computer Engineering
New York University Tandon School of Engineering
ah7072@nyu.edu | shz2020@nyu.edu
Github Repository — Kaggle Notebook

Abstract

We propose a compact ResNet for CIFAR-10 that meets a 5M parameter constraint while achieving 95.31% accuracy (Kaggle score 0.85485). Our design incorporates SE blocks, advanced regularization (dropout, label smoothing, stochastic depth), and dynamic data augmentation. A training strategy with warmup, cosine annealing, and SWA with BatchNorm freezing further optimizes performance.

Introduction

CIFAR-10 is a common benchmark with 60,000 32×32 color images in 10 classes. ResNets, known for their skip connections, ease the training of deep models. Here, we modify a ResNet to meet a 5M parameter limit while achieving state-of-the-art performance on CIFAR-10 by combining architectural enhancements, robust regularization, and optimized training techniques.

Methodology

Architecture

Our modified architecture builds upon a ResNet-18 foundation by enhancing its core residual block. In the traditional residual block, using the formulation

$$ReLU(S(x) + F(x))$$

as described in (He et al. 2015) where S(x) is the identity skip connection and F(x) is a sequence of layers comprising a 3×3 convolution, batch normalization, ReLU activation, followed by another 3×3 convolution and batch normalization. In our design, we integrate Squeeze-and-Excitation (SE) blocks (Hu, Shen, and Sun 2018) immediately after the second convolution. These SE blocks perform global average pooling on the feature maps, then pass the resulting channel descriptors through a two-layer fully connected network (with a reduction ratio of 16) to learn channel-wise scaling factors. This adaptive recalibration of feature channels enhances representational power without significant parameter overhead.

Furthermore, we experimented with different residual block configurations, ultimately selecting a (3,3,2,1) block structure. The network starts with an initial convolution that maps the input image to 32 channels and progressively increases the number of channels (to 64, then 128, and finally 256) while reducing spatial dimensions via striding. An adaptive average pooling layer then aggregates spatial features into a fixed-size representation, which is fed into a final fully connected layer for classification into 10 classes.

This architectural design—combining a deeper block configuration, SE enhancements, and a balanced channel progression—enables the model to achieve high accuracy on CIFAR-10 while remaining under the 5M parameter constraint.

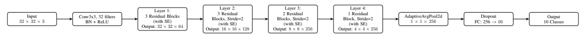

Figure 1: Final Modified ResNet Architecture with SE Blocks: Feature extraction through successive convolutional and residual layers, followed by global pooling and a classification head.

Training Methodology

Our training strategy was designed to maximize generalization while respecting a 5M parameter constraint. In this section, we detail our key design choices, including hyperparameter tuning, optimization, regularization, and data augmentation.

Hyperparameter Tuning

We extensively tuned the architectural and training hyperparameters:

- **Block Configuration and Network Depth:** We experimented with several ResNet variants. Our best configuration was ResNet (3,3,2,1), which increases the number of channels gradually from a base of 32 up to 256. All convolutions used 3×3 kernels.
- **Channel and Kernel Sizes:** The network starts with 32 channels and progresses to 64, 128, and finally 256 channels, balancing expressiveness with parameter efficiency.
- **Batch Size and Total Epochs:** We primarily used a batch size of 128 and varied the total training duration from 35 to 150 epochs. Extended training (up to 150 epochs) allowed for better convergence and fine-tuning.

Optimization

We evaluated different optimizers and learning rate strategies:

- **Optimizer Selection:** We compared SGD and AdamW. AdamW, (Loshchilov and Hutter 2019) with a learning rate of 0.003 and weight decay between 0.0007 and 0.00075, provided more stable convergence and superior generalization compared to SGD, which required a higher learning rate (around 0.01).
- **Learning Rate Scheduling:** Our learning rate schedule combined a warmup phase using LinearLR (starting at a factor of 0.01 for 5 iterations) with cosine annealing (Loshchilov and Hutter 2017) implemented via SequentialLR. The cosine annealing phase used a $T_{\max}$ between 50 and 80 epochs and decayed the learning rate to an $\eta_{\min}$ of approximately 1×10^{-5}.
- **Stochastic Weight Averaging (SWA):** To further enhance generalization, SWA (Izmailov et al. 2018) was incorporated in the latter phase of training. Specifically, SWA was initiated at roughly 62.5% of the total epochs (e.g., epoch 93 for a 150-epoch schedule) using an SWA learning rate between 3×10^{-5} and 5×10^{-5}. This technique averages weights from multiple epochs to obtain a smoother, more robust final model.

Regularization and Loss Settings

To reduce overfitting, we applied multiple regularization techniques:

- **Label Smoothing:** (Szegedy et al. 2016) We experimented with label smoothing values of 0.0, 0.05, and 0.1. Label smoothing helps by softening the target labels and reducing the model's overconfidence.
- **Dropout:** Dropout was applied in the fully connected layers with rates between 0.1 and 0.35, which forces the network to learn redundant representations.
- **Stochastic Depth Dropout:** We incorporated stochastic depth dropout (Huang et al. 2016) (with probabilities of 0.0, 0.05, or 0.1) in the residual blocks. This technique randomly skips residual connections during training, encouraging the model to learn more robust features.
- **Squeeze-and-Excitation (SE) Blocks:** SE blocks were integrated within each residual block. They perform global average pooling followed by a two-layer fully connected network (with a reduction ratio of 16) to produce channel-wise scaling factors. This mechanism allows the network to recalibrate channel responses adaptively with minimal parameter overhead.

Data Augmentation and Fine-Tuning

We adopted a dynamic data augmentation strategy to further improve generalization:

- **Early Training (Heavy Augmentation):** In the early phase, we used a heavy augmentation pipeline including:
  - **RandAugment:** (Cubuk et al. 2020) Applied with 3 operations at a magnitude of 9.
  - **ColorJitter:** Adjusting brightness, contrast, and saturation by 0.2 and hue by 0.1.
  - **Random Erasing:** (Zhong et al. 2017) With a probability of 0.1–0.2 to simulate occlusions.

  In addition, we applied MixUp (Zhang et al. 2018) and CutMix (Yun et al. 2019) augmentation:

  - **MixUp:** Blending two images using a mixing coefficient sampled from a Beta distribution with an alpha value of 0.2 or 0.4.
  - **CutMix:** Replacing a region of one image with a patch from another, using an alpha value of 1.0.

  These augmentations were applied with a 35% probability during early epochs to promote robustness.
- **Fine-Tuning Phase:** During the final 25% of epochs, we transitioned to a lighter augmentation pipeline that consisted of basic random cropping, horizontal flipping, and normalization. Concurrently, the probability of applying MixUp/CutMix was reduced to 15% to allow the model to refine its decision boundaries on less perturbed data. Furthermore, BatchNorm layers were frozen in the final epochs (starting around 87.5% of total epochs) to stabilize normalization statistics.

Together, these strategies enabled our modified ResNet to achieve high test accuracy on CIFAR-10 while remaining within the 5M parameter constraint. Each component—from architectural adjustments to the dynamic augmentation and SWA—played a crucial role in enhancing the model's generalization performance.

Iterative Model Improvements

From the original ResNet-18 architecture described in (He et al. 2015), we made several modifications to improve performance while adhering to a 5M parameter constraint. Our evolution can be summarized as follows:

1. **Model 1: Baseline.** Standard ResNet (2,2,2,2) with 2.7M parameters, trained using SGD (LR=0.01, WD=0.0005) and advanced augmentation (random cropping, flipping, ColorJitter). Minimal regularization yields 88.3% accuracy (Kaggle: 0.76536). See Figure 2.

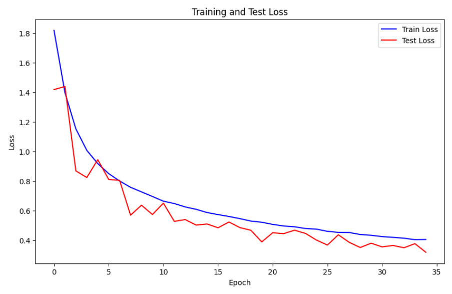

Figure 2: Train/test loss over 35 epochs (Model 1).

2. **Model 2: Regularization Improvements.** Introducing label smoothing (0.05–0.1) and increasing dropout (up to 0.35) improves accuracy to 89.8% (Kaggle: 0.78232).

3. **Model 3: Augmentation & Architecture Change.** Switching to a ResNet (2,2,2,1) and applying MixUp and CutMix (35% probability) with SWA starting after 30 epochs yields approximately 89.4% accuracy (Kaggle: 0.77075).
4. **Model 5: Optimizer & Tuning.** Using AdamW (LR=0.003, WD≈0.0007–0.00075), reducing label smoothing (0.05), increasing dropout (0.35), and starting SWA earlier (after 25 epochs) raises accuracy to 92.70% (Kaggle: 0.81584).
5. **Models 6–8: Minor Tweaks.** Further refinements—such as a deeper ResNet (3,3,2,1) with no label smoothing, moderate dropout (0.2), and fine-tuning with AdamW (LR=0.01)—yield accuracies of 93.18% and 94.06% (Kaggle: 0.82482 and 0.82621), which are combined in our narrative.
6. **Model 9: SE Blocks Integration.** Incorporating SE blocks (reduction ratio=16), reducing dropout (0.1), and maintaining label smoothing (0.1), with SWA starting after 50 epochs, increases accuracy to 94.62% (Kaggle: 0.84497). See Figure 3.

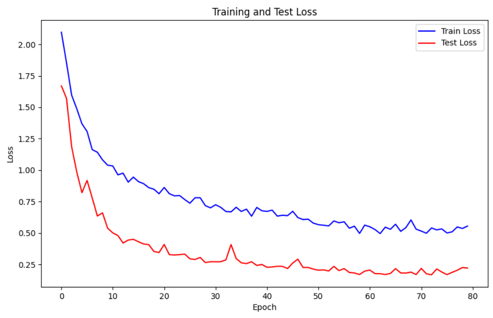

Figure 3: Train/test loss over 80 epochs (Model 9).

7. **Model 10: Final Model.** Extending training to 150 epochs, using lighter augmentation in the final 25% (MixUp/CutMix probability reduced to 15%), freezing BatchNorm layers in the final 10% (from epoch 131), and starting SWA at 62.5% (epoch 93) with SWA LR of 5×10^{-4}, the final model achieves 95.31% accuracy (Kaggle: 0.85485). See Figure 4.

These changes increased the accuracy to 95.31% and the Kaggle score to **0.85485**.

This evolution demonstrates how systematic modifications to architecture, regularization, and training strategies collectively enhanced our model's performance and generalization on CIFAR-10. The iterative model accuracy improvements can be seen in 5

Observations and Analysis

Training Dynamics

Figure 4 shows the evolution of training and validation losses over 150 epochs. Early in training, aggressive data augmentation (RandAugment, ColorJitter, Random Erasing,

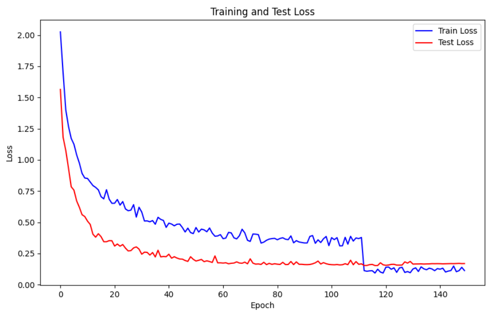

Figure 4: Train/test loss over 150 epochs (Model 10).

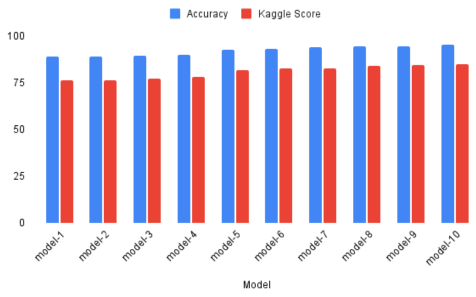

Figure 5: Training performance over successive model iterations showing progressive improvements in accuracy and kaggle score.

MixUp, and CutMix) drives a rapid decrease in loss. In later epochs, a transition to a lighter augmentation scheme stabilizes the gradients and reduces noise, indicating effective fine-tuning and convergence.

Ablation Studies

Our systematic ablation experiments reveal the impact of individual design choices:

- **SWA and BatchNorm Freezing:** Activating Stochastic Weight Averaging (SWA) from 62.5% of the training, together with freezing BatchNorm layers in the final 10%, results in smoother loss curves and improved generalization. Omitting either technique increases variance and degrades performance.
- **Regularization Techniques:** The combined use of dropout, stochastic depth, and label smoothing effectively minimizes overfitting. Removing any one of these methods causes a larger gap between training and validation losses.

These analyses confirm that each component—when applied individually or in combination—contributes significantly to robust convergence and improved generalization.

Confusion Matrix Analysis

To further assess class-level performance, we visualize the confusion matrix in Figure 6. The strong diagonal dominance indicates the model accurately classifies most examples within each class. However, certain classes (e.g., *cat* and *dog*) exhibit minor confusion, reflecting their visual similarity at low resolution. Understanding these misclassifications can guide targeted augmentations or specialized fine-tuning strategies for even better results.

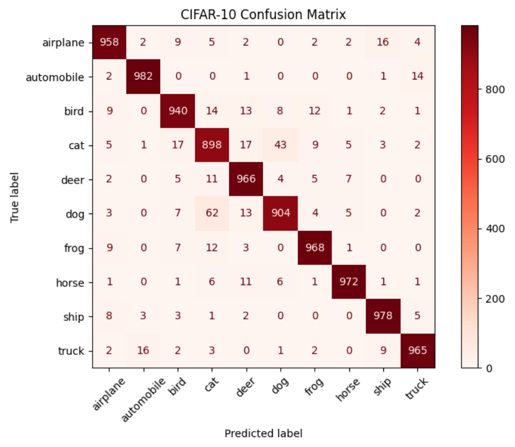

Figure 6: Confusion matrix for the final model on the CIFAR-10 test set. Diagonal dominance shows robust class performance, with minor off-diagonal entries indicating occasional misclassifications.

Conclusion

We present a compact Residual Network that achieves state-of-the-art CIFAR-10 performance with only 4.4 million parameters. Our final model reaches a Kaggle score of **0.85485** (95.31% accuracy), proving that strategic architectural and training innovations can yield robust performance within a tight parameter budget. Key drivers include:

- **Modified Architecture:** A compact ResNet with Squeeze-and-Excitation blocks for dynamic channel recalibration.
- **Advanced Regularization:** The combined use of FC dropout, stochastic depth, and label smoothing to mitigate overfitting.
- **Dynamic Data Augmentation:** An aggressive early-stage augmentation strategy that transitions to a lighter scheme during fine-tuning.
- **Optimized Training Strategy:** AdamW with a warmup phase, cosine annealing, and SWA to enhance convergence and stability.

Table 1 summarizes the performance evolution, highlighting the cumulative benefits of these design choices. Our results demonstrate that even under a 5M parameter constraint, systematic tuning and strategic model design can achieve competitive, real-world performance, setting a new benchmark for efficient deep learning on CIFAR-10.

Variant	Accuracy (%)	Kaggle Score
Baseline	88.3	0.76536
+ Regularization	89.8	0.78232
+ Data Augmentation	89.4	0.77075
+ AdamW + SWA	92.7	0.81584
+ Architecture Change	94.06	0.8261
+ SE Blocks	94.6	0.84497
Final Model	**95.31**	**0.85485**

Table 1: Performance evolution across model iterations.

References

[Cubuk et al. 2020] 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.

[He et al. 2015] He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*. Submitted on 10 Dec 2015.

[Hu, Shen, and Sun 2018] Hu, J.; Shen, L.; and Sun, G. 2018. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7132–7141.

[Huang et al. 2016] Huang, G.; Sun, Y.; Liu, Z.; Sedra, D.; and Weinberger, K. Q. 2016. Deep networks with stochastic depth. In *European Conference on Computer Vision*, 646–661.

[Izmailov et al. 2018] Izmailov, P.; Podoprikhin, D.; Garipov, T.; Vetrov, D.; and Wilson, A. G. 2018. Averaging weights leads to wider optima and better generalization. In *Proceedings of the International Conference on Machine Learning*.

[Loshchilov and Hutter 2017] Loshchilov, I., and Hutter, F. 2017. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*.

[Loshchilov and Hutter 2019] Loshchilov, I., and Hutter, F. 2019. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

[Szegedy et al. 2016] Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2818–2826.

[Yun et al. 2019] 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 International Conference on Computer Vision*, 6023–6032.

[Zhang et al. 2018] Zhang, H.; Cisse, M.; Dauphin, Y. N.; and Lopez-Paz, D. 2018. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*.

[Zhong et al. 2017] Zhong, Z.; Zheng, L.; Li, G.; Zhou, S.; Li, S.; and Yang, Y. 2017. Random erasing data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 13001–13008.