# Evaluating VGG and ResNet Architectures on Noisy CIFAR-100 Data

Nastasiu Stefan

Mocanu Octavian

***Abstract*—This paper explores the performance of various deep learning model architectures on a noisy version of the CIFAR-100 dataset, a widely used benchmark in computer vision tasks. The study focuses on understanding the impact of noise on model training and evaluating the robustness of different architectures in such scenarios. Specifically, we compare models based on the VGG and ResNet architectures.**

**The research also explores the use of data augmentation strategies and hyperparameter tuning to mitigate the effects of noisy labels, aiming to improve model generalization.**

**The results demonstrate that the ResNet-18 model, when fine-tuned with transfer learning, achieved the best performance, significantly outperforming other architectures in terms of accuracy and robustness to label noise.**

## I. Introduction

The CIFAR-100 dataset is widely used in computer vision tasks for evaluating image classification models. However, real-world data often contains noise due to various factors such as sensor errors, environmental conditions, and human labeling mistakes. Training models on noisy data remains a significant challenge in machine learning.

This report investigates the robustness of deep learning architectures, specifically different versions of VGG and ResNet, on a noisy variant of the CIFAR-100 dataset. By introducing label noise into the dataset, we simulate real-world imperfections and evaluate how these models perform in terms of accuracy, generalization, and resistance to overfitting. The findings aim to provide insights into which architectures are better suited for handling noisy data and to identify strategies for improving robustness in the presence of label noise.

Tested strategies include data augmentation such as blurring an image and adding imperfections, so that the model learns more generalized patterns or the use of CutMix, a powerful tool that replaces regions with a patch from another image in order to further enhance localization ability. Additionally, for a more direct approach at counteracting noisy labels, soft labels were used in order to tackle overconfidence and limit the impact of noise.

## II. Related Work

Ruixuan Xiao et al. [1] proposed a novel framework, ProMix, to address the challenge of learning with noisy labels (LNL) by maximizing the utility of clean samples for improved model performance. The key contribution of ProMix is a high-confidence selection strategy that dynamically expands a base clean sample set by selecting examples with high confidence scores and predictions that match the given labels. To prevent potential biases introduced by excessive clean sample selection, the authors further developed a semi-supervised learning (SSL) framework capable of training balanced and unbiased classifiers by separating clean and noisy samples during the learning process. Their extensive experiments demonstrate that ProMix achieves significant improvements over state-of-the-art methods across multiple benchmarks with varying noise levels. In particular, the framework delivers an average performance increase of 2. 48% in the CIFAR-N data set.

Paul Albert et al. [2] introduced an innovative approach for addressing label noise by improving the detection and correction of incorrect pseudo-labels during training. Their method employs a state-of-the-art noise detection metric to identify noisy samples and estimate their true labels through a consistency regularization technique. To ensure the reliability of these pseudo-labels, the authors propose the use of a pseudo-loss metric, which demonstrates a strong correlation with pseudo-label correctness. Weight updates associated with pseudo-labels that have a low likelihood of being correct are discarded to prevent noisy data from degrading model performance. Additionally, the authors integrate an interpolated contrastive learning objective, where accurate pseudo-labels contribute to learning meaningful interclass relationships, while incorrectly labeled samples are leveraged in an unsupervised objective to minimize their negative impact. Experimental results show that this method achieves state-of-the-art performance on both synthetic and real-world noisy datasets, highlighting its robustness and effectiveness in handling label noise.

Renyu Zhu et al. [3], a novel sample selection approach aimed at improving learning in noisy annotation scenarios by identifying and utilizing the underlying semantics of noise. The method divides the dataset into two subsets: a confident set containing samples with reliable labels, and an unconfident set comprising potentially noisy samples. To distinguish between different types of annotation noise, Proto-semi employs a distance-based technique using prototype vectors to assess the similarity of samples to class representatives. Based on these distances, the confident and unconfident sets are dynamically adjusted throughout the training process. The authors further incorporate a semi-supervised learning strategy to enhance model performance by leveraging both labeled and unlabeled data more effectively.

Experiments conducted on the CIFAR-N dataset validate the effectiveness of Proto-semi in handling real-world label noise, demonstrating notable improvements over existing methods. The authors suggest that future research could focus on extending Proto-semi to other real-world datasets with noisy annotations to further explore its generalizability and robustness.

Ross Wightman et al. [4] introduced a set of optimized training procedures for the widely used ResNet-50 architecture, focusing on improving performance under varying resource constraints. Their work integrates several enhancements to the standard training pipeline and systematically explores different strategies to establish a new state-of-the-art for training ResNet-50. In addition, the authors propose two alternative procedures aimed at achieving strong model performance with reduced computational requirements. However, they emphasize that these procedures are not universally applicable across all architectures. Instead, they advocate for joint optimization of the model architecture and training strategy. Although their methods yield impressive results on ResNet-50 and certain other models, they note that the same procedures may deliver suboptimal performance on deeper networks, which typically require more extensive regularization. This highlights the importance of tailoring training approaches to the specific characteristics of each model to achieve optimal results.

## III. Methodology

### A. Dataset

The dataset used for this study consists of 100 classes with 600 images each. It represents a slightly modified version of CIFAR100, and it's called CIFAR-100N. This dataset was developed by Jiaheng Wei et al. [5], and it provides CIFAR-100 train images with human annotated noisy labels obtained from Amazon Mechanical Turk.

### B. Data Augmentation

Data augmentation has proven to be a powerful technique for improving the performance of machine learning models, especially when dealing with noisy datasets. By artificially increasing the diversity of training data, augmentation can help models generalize better and mitigate the impact of noise.

For noisy datasets, augmentation methods are particularly valuable as they can introduce variations that simulate real-world conditions, thus making the model more robust to discrepancies and inconsistencies in the input data.

A few data augmentation methods that proved effective in our case study were:

- *random cropping*
- *blurring an image using gaussian blur*
- *random horizontal flip*
- *random augmentation*
- *color jittering*
- *random erasing*
- *changing perspective*

*1) Runtime transforms:* We used runtime transforms, meaning we applied the transformations above to the dataset during training. This resulted in an increase of the execution time for each epoch, but slightly increased the accuracy as well.

*2) CutMix and Mixup:* For all the models trained, we applied CutMix and Mixup to the batch of images during training. One of these 2 is chosen randomly before applying, by using *v2.RandomChoice*. CutMix and Mixup are popular augmentation strategies that can improve classification accuracy. Both CutMix and MixUp are effective data augmentation techniques, but they work in different ways:

- MixUp: This method creates new images by blending two original images, combining their pixel values using a weighted average. The result is a smooth mix of the two images.
- CutMix: Instead of blending, CutMix cuts out a section of one image and pastes it onto another. This approach preserves distinct parts of the images, leading to more varied and diverse training samples.

### C. Model architectures

*1) VGG models:* VGG16 is a widely used fully connected model that consists of 16 weight layers: 13 convolutional layers and 3 fully connected layers. All convolutional layers use small 3×3 filters with a stride of 1 and padding to preserve spatial dimensions followed by the use of max pooling with a 2x2 window and ReLU activation.

| Layer Type | Filter Size / Stride | Output Dimensions |
|---|---|---|
| Input | - | $224 \times 224 \times 3$ |
| Convolution | $3 \times 3$, 64 filters | $224 \times 224 \times 64$ |
| Convolution | $3 \times 3$, 64 filters | $224 \times 224 \times 64$ |
| Max Pooling | $2 \times 2$, stride 2 | $112 \times 112 \times 64$ |
| Convolution | $3 \times 3$, 128 filters | $112 \times 112 \times 128$ |
| Convolution | $3 \times 3$, 128 filters | $112 \times 112 \times 128$ |
| Max Pooling | $2 \times 2$, stride 2 | $56 \times 56 \times 128$ |
| Convolution | $3 \times 3$, 256 filters | $56 \times 56 \times 256$ |
| Convolution | $3 \times 3$, 256 filters | $56 \times 56 \times 256$ |
| Convolution | $3 \times 3$, 256 filters | $56 \times 56 \times 256$ |
| Max Pooling | $2 \times 2$, stride 2 | $28 \times 28 \times 256$ |
| Convolution | $3 \times 3$, 512 filters | $28 \times 28 \times 512$ |
| Convolution | $3 \times 3$, 512 filters | $28 \times 28 \times 512$ |
| Convolution | $3 \times 3$, 512 filters | $28 \times 28 \times 512$ |
| Max Pooling | $2 \times 2$, stride 2 | $14 \times 14 \times 512$ |
| Convolution | $3 \times 3$, 512 filters | $14 \times 14 \times 512$ |
| Convolution | $3 \times 3$, 512 filters | $14 \times 14 \times 512$ |
| Convolution | $3 \times 3$, 512 filters | $14 \times 14 \times 512$ |
| Max Pooling | $2 \times 2$, stride 2 | $7 \times 7 \times 512$ |
| Fully Connected | - | 4096 |
| Fully Connected | - | 4096 |
| Fully Connected | - | 100 (class probabilities) |

TABLE I
SUMMARY OF THE VGG16 MODEL 1 ARCHITECTURE. EACH CONVOLUTION LAYER USES RELU ACTIVATION.

Another VGG16 Model 2 used for training is a slightly modified version of the VGG16 architecture, with only one fully connected layer packing 2048 neurons. This value is a result of using enlarged 64x64 images as input, as opposed to the original 32x32.

*2) Resnet models:* The Resnet model 1 used is based on a ResNet-18 architecture with transfer learning, modified for a custom classification task (CIFAR100 noisy). The initial convolutional layer (conv1) is replaced with a 3x3 kernel with stride 1 and padding 1, instead of the original 7x7 kernel. Additionally, the max-pooling layer that follows conv1 is removed by replacing it with an identity layer. The fully connected (fc) layer is adjusted to output 100 classes (for CIFAR100). These modifications aim to retain more spatial information and adapt the model to a new classification problem.

| Layer | Original ResNet-18 | Modified ResNet-18 |
|---|---|---|
| `conv1` | 7x7 kernel, stride 2 | 3x3 kernel, stride 1 |
| `maxpool` | Max-pooling layer | Removed (replaced with Identity) |
| `fc` | 1,000 output classes | `100` output classes |

TABLE II
SUMMARY OF RESNET-18 MODEL 1 MODIFICATIONS

We also trained an unmodified ResNet-18 Model 2 with transfer learning, which obtained weaker results, as described in the *Results* section.

*3) EncoderDecoderNet:* The EncoderDecoderNet is an encoder-decoder architecture that uses Residual Blocks with Squeeze-and-Excitation (SE) modules to enhance feature extraction. The encoder consists of four stacked residual blocks, progressively increasing the number of channels and reducing spatial dimensions. The bottleneck block maintains a constant number of channels (512) while further refining the feature representation. The decoder consists of three transposed convolutional layers, progressively upsampling the feature maps to restore spatial dimensions. The final feature map is processed through a global average pooling (GAP) layer followed by a fully connected (fc) layer that outputs the class predictions. The SEBlock applies channel-wise attention by recalibrating the feature maps, improving the network's capacity to focus on important features.

| Stage | Layer | Output Shape |
|---|---|---|
| Input | RGB Image | $3 \times 224 \times 224$ |
| Encoder Stage 1 | ResidualBlock(3, 64) | $64 \times 224 \times 224$ |
| Encoder Stage 2 | ResidualBlock(64, 128, stride=2) | $128 \times 112 \times 112$ |
| Encoder Stage 3 | ResidualBlock(128, 256, stride=2) | $256 \times 56 \times 56$ |
| Encoder Stage 4 | ResidualBlock(256, 512, stride=2) | $512 \times 28 \times 28$ |
| Bottleneck | ResidualBlock(512, 512) | $512 \times 28 \times 28$ |
| Decoder Stage 1 | ConvTranspose2d(512, 256) | $256 \times 56 \times 56$ |
| Decoder Stage 2 | ConvTranspose2d(256, 128) | $128 \times 112 \times 112$ |
| Decoder Stage 3 | ConvTranspose2d(128, 64) | $64 \times 224 \times 224$ |
| Classifier | GAP + Linear(64, num_classes) | $num\_classes$ |

TABLE III
ENCODERDECODERNET ARCHITECTURE WITH RESIDUAL AND SE BLOCKS

### D. Experimental Setup

For training the deep learning models, we used Kaggle notebooks, with Nvidia P100 as GPU. All code was written in Python, and the model architectures were implemented with PyTorch framework.
Below are the hyperparameters used for every model.

| Hyperparameter | Value |
|---|---|
| Batch Size | 128 |
| Learning Rate | 0.005 |
| Optimizer | SGD |
| Momentum | 0.9 |
| Weight Decay | 0.0005 |
| Nesterov | True |
| Scheduler | CosineAnnealingLR |
| Criterion | CrossEntropyLoss (label smoothing = 0.2) |
| Best Result | 168 epochs, 3 hours 40 minutes |

TABLE IV
VGG16 MODEL 1 HYPERPARAMETERS

| Hyperparameter | Value |
|---|---|
| Batch Size | 128 |
| Learning Rate | 0.001 |
| Optimizer | AdamW |
| Weight Decay | 1e-4 |
| Scheduler | CosineAnnealingLR |
| Criterion | CrossEntropyLoss (label smoothing = 0.2) |
| Best Result | 15 epochs, 36 minutes |

TABLE V
VGG16 MODEL 2 HYPERPARAMETERS

| Hyperparameter | Value |
|---|---|
| Batch Size | 128 |
| Learning Rate | 0.01 |
| Optimizer | SGD |
| Momentum | 0.9 |
| Weight Decay | 5e-4 |
| Nesterov | True |
| Fused Optimizer | True |
| Scheduler | CosineAnnealingLR |
| Criterion | CrossEntropyLoss (label smoothing = 0.1) |
| Best Result | 42 epochs, 56 minutes |

TABLE VI
RESNET18 MODEL 1 HYPERPARAMETERS

| Hyperparameter | Value |
|---|---|
| Batch Size | 128 |
| Learning Rate | 0.01 |
| Optimizer | SGD |
| Momentum | 0.9 |
| Weight Decay | 5e-4 |
| Nesterov | True |
| Fused Optimizer | True |
| Scheduler | CosineAnnealingLR |
| Criterion | CrossEntropyLoss (label smoothing = 0.1) |
| Best Result | 3 epochs, 2 minutes |

TABLE VII
RESNET18 MODEL 2 HYPERPARAMETERS

| Hyperparameter | Value |
|---|---|
| Batch Size | 128 |
| Learning Rate | 0.002 |
| Optimizer | Adam |
| Weight Decay | 1e-4 |
| Scheduler | ReduceLROnPlateau |
| Criterion | CrossEntropyLoss (label smoothing = 0.1) |
| Best Result | 20 epochs, 1 hour 12 minutes |

TABLE VIII
ENCODER-DECODER MODEL HYPERPARAMETERS

## IV. Theoretical Analysis

In this section, we discuss the theoretical concepts that aim to explain how deep learning architectures and data augmentation techniques perform when trained on noisy datasets. Our focus is on understanding how label noise impacts learning, how different models respond to this noise, and how augmentations contribute to better generalization.

### *A. Learning with Noisy Labels*

The presence of noisy labels is a common issue in many real-world datasets and poses a significant challenge for machine learning models. Labels may be incorrect due to human errors, ambiguities in data annotation, or biases in the labeling process. Such inaccuracies can degrade model performance, especially when noise is not accounted for during training.

Various strategies have been proposed to address the issue of noisy labels, with one widely studied approach involving the use of robust loss functions. These loss functions aim to reduce the model's sensitivity to incorrect labels by modifying the learning process. In our study, we use CrossEntropyLoss combined with label smoothing. Label smoothing helps by softening the target distribution, which reduces the model's reliance on exact labels and mitigates the impact of incorrect ones. The theoretical foundation of this technique is linked to overfitting prevention, where smoothing the loss landscape makes the model less prone to memorizing noise.

### *B. Robustness of Deep Learning Architectures*

Deep learning models exhibit varying degrees of robustness to noisy labels, depending on their architecture. Model capacity and complexity play a key role in determining whether a network will memorize noisy labels or learn generalizable features. High-capacity models can overfit noise if not properly managed, but they also have the potential to extract meaningful patterns when regularized appropriately.

For instance, the VGG architecture is known for its deep structure and uniform convolutional layers. Its robustness comes from its ability to capture fine details while maintaining spatial consistency. Techniques such as max-pooling and ReLU activations help VGG models maintain robustness despite encountering noisy inputs. These design choices align with theoretical principles in deep learning that emphasize feature extraction and spatial invariance.

In contrast, ResNet architectures introduce skip connections that create identity mappings between layers. This innovation addresses the vanishing gradient problem, allowing for much deeper networks to be trained efficiently. Theoretically, these skip connections enable residual learning, which simplifies optimization and makes the model less susceptible to overfitting noise. By flattening the loss landscape, residual connections allow gradient-based optimization methods to converge more effectively.

### *C. Impact of Data Augmentation on Generalization*

Data augmentation is a well-known method to improve the generalization capabilities of deep learning models, especially when dealing with noisy data. The core idea is to expand the hypothesis space by introducing diverse variations in the training data, which forces the model to learn more generalized patterns.

In our experiments, we used augmentations such as random cropping, Gaussian blur, horizontal flips, and color jittering. These augmentations are not arbitrary—they are rooted in theoretical concepts. For example, random cropping promotes spatial invariance by ensuring that the model does not rely on specific positions of features in an image. Similarly, Gaussian blur simulates real-world noise, helping the model become more resilient to imperfect inputs.

Horizontal flips and color jittering further diversify the dataset by introducing changes that the model must learn to handle. These transformations push the model to develop more robust representations that are less influenced by noise. The theoretical concept of invariance and equivariance in neural networks supports this practice. Invariance refers to the ability of a model to produce the same output despite input changes, while equivariance ensures that predictable input changes lead to corresponding output changes.

### *D. Trade-offs in Model Complexity and Regularization*

A key factor in determining a model's robustness to noise is the balance between its complexity and regularization. Complex models, such as deep convolutional networks, have a high capacity to fit training data—including noise. This can result in low bias but high variance, which is undesirable when dealing with noisy labels.

Regularization techniques help manage this trade-off by imposing constraints that limit the model's capacity to overfit noise. In our study, we explore methods like weight decay and label smoothing. Weight decay penalizes large weights in the network, encouraging simpler models that are less likely to memorize noise. Label smoothing, as mentioned earlier, adjusts the target distribution to reduce reliance on exact labels.

We observed notable differences between VGG and ResNet architectures in terms of how they handle noise. VGG, with its fully connected layers, tends to have a higher capacity for memorization, which can be detrimental in noisy scenarios. However, incorporating runtime data augmentation helps reduce overfitting in VGG models by continually providing varied inputs during training. ResNet, on the other hand, benefits from its inherent regularization through skip connections. These residual connections act as a built-in form of regularization, making ResNet models naturally more robust to label noise without the need for extensive external regularization techniques.

### *E. Theoretical Foundations of Transfer Learning*

Transfer learning leverages pretrained models to improve performance on new tasks, and its effectiveness is well supported by theoretical insights into feature reuse and domain adaptation. Pretrained models learn general features from large

datasets that can be fine-tuned for specific tasks with smaller labeled datasets.

The lower layers of a neural network typically learn basic features such as edges and textures, which are applicable across various domains. Higher layers, in contrast, become more specialized for the target task. This hierarchical feature reuse forms the theoretical basis of transfer learning.

In the context of noisy labels, transfer learning can enhance robustness by relying on pretrained features that are less influenced by noise in the target dataset. This aligns with the concept of inductive bias, where a pretrained model guides the learning process toward more generalized solutions. In our study, we apply this principle by fine-tuning a pretrained ResNet-18 model, which helps mitigate the effects of noise in the CIFAR-100N dataset.

### *F. Conclusion*

The theoretical concepts discussed in this section form the foundation of our approach to handling noisy labels in deep learning. By understanding how robust architectures, data augmentation, regularization techniques, and transfer learning contribute to noise resilience, we can develop models that perform well even when faced with imperfect data. Our findings highlight the importance of balancing model complexity with appropriate regularization strategies and demonstrate the value of leveraging theoretical insights to guide practical implementations.

## V. Results

Below is a detailed comparison between the performance of the baseline and the best custom models we trained for this study.

| Metric | Baseline ResNet-18 |
|---|---|
| Number of Parameters | 11.2 million |
| Epochs | 40 |
| Time per Epoch | 11s |
| Accuracy | 26.11% |

TABLE IX
BASELINE RESNET-18 MODEL RESULTS

| Metric | VGG-16 Model 1 |
|---|---|
| Number of Parameters | 34 million |
| Epochs | 168 |
| Time per Epoch | 78.40s |
| Accuracy | 60.26% |

TABLE X
VGG-16 MODEL 1 RESULTS

| Metric | VGG-16 Model 2 |
|---|---|
| Number of Parameters | 15 million |
| Epochs | 15 |
| Time per Epoch | 144.61s |
| Accuracy | 57.63% |

TABLE XI
VGG-16 MODEL 2 RESULTS

| Metric | Custom ResNet-18 Model 1 |
|---|---|
| Number of Parameters | 11.22 million |
| Epochs | 42 |
| Time per Epoch | 81.24s |
| Accuracy | 64.38% |

TABLE XII
CUSTOM RESNET-18 MODEL 1 RESULTS

| Metric | Custom ResNet-18 Model 2 |
|---|---|
| Number of Parameters | 11.23 million |
| Epochs | 3 |
| Time per Epoch | 35.24s |
| Accuracy | 57.88% |

TABLE XIII
CUSTOM RESNET-18 MODEL 2 RESULTS

| Metric | Encoder-Decoder Model |
|---|---|
| Number of Parameters | 12.4 million |
| Epochs | 101 |
| Time per Epoch | 80.04s |
| Accuracy | 59.23% |

TABLE XIV
ENCODER-DECODER MODEL RESULTS

## VI. Results Analysis

We compared the baseline ResNet-18 model with several custom models to analyze differences in accuracy, training time, and parameter count. These findings help us understand the trade-offs when choosing models for better performance.

### *A. Baseline ResNet-18 Model*

The baseline ResNet-18 model (Table IX) achieved an accuracy of 26.11% with 11.2 million parameters after 40 epochs. Although its accuracy is relatively low, it serves as a reliable starting point for comparisons. With a moderate parameter count and short training time (11 seconds per epoch), this model balances simplicity and efficiency, making it a good baseline.

### *B. VGG-16 Model 1*

VGG-16 Model 1 (Table X) achieved a significantly higher accuracy of 60.26%, thanks to its deeper architecture, which includes more convolutional layers. However, this comes at a cost: the model has over 34 million parameters and takes 78.40 seconds per epoch to train. While it delivers better accuracy, the longer training time and higher computational demands are notable drawbacks.

### *C. VGG-16 Model 2*

The second VGG-16 model (Table XI) reached an accuracy of 57.% with fewer parameters (15 million) and faster training per epoch (144.61 seconds). Despite this, the accuracy is slightly lower than VGG-16 Model 1, possibly because it was trained for fewer epochs (15). This highlights how training duration affects performance, suggesting the model may not have fully optimized in the limited time.

### D. Custom ResNet-18 Models

Custom ResNet-18 Model 1 (Table XII) achieved the best accuracy of 64.38%. It has 11.22 million parameters, close to the baseline ResNet-18, but requires more training time per epoch (81.24 seconds) and was trained for 42 epochs. This model stands out for its strong performance with a reasonable number of parameters.

Custom ResNet-18 Model 2 (Table XIII) had a lower accuracy of 57.88% but required just 3 epochs to train, with each epoch taking 35.24 seconds. This demonstrates the impact of shorter training on accuracy, showing that faster models may sacrifice precision for speed.

### E. Encoder-Decoder Model

The Encoder-Decoder model (Table XIV) achieved 59.23% accuracy with 12.4 million parameters and took 80.04 seconds per epoch to train. It uses advanced techniques like Squeeze-and-Excitation (SE) blocks and residual connections. While it improves over the baseline ResNet-18, it doesn't match the performance of Custom ResNet-18 Model 1, balancing complexity and accuracy.

### *Conclusion*

Custom ResNet-18 Model 1 was the most effective in this comparison, achieving the highest accuracy (64.38%) with a manageable parameter count and training time. The VGG-16 models also performed well but required much more computational power, which could limit their practical use in larger tasks. Custom ResNet-18 Model 2, with its rapid training, highlighted the trade-off between speed and accuracy. Overall, these results show that optimizing for both efficiency and performance requires careful consideration of model design and training strategies.

## VII. Future work

Based on our findings, we recommend the following strategies for training models on noisy datasets:

- Data Augmentation: Introduce synthetic noise during training to improve robustness.
- Early Stopping: Prevents overfitting on noisy data.

Future improvements on these models could include:

- Fine tuning hyper parameters to better deal with noisy labels
- Adding more data augmentation specific to noisy datasets
- Experimenting with different loss functions that are more robust to noise like Mean Absolute Error (MAE), which is less influenced by extreme errors than Cross Entropy Loss or Focal Loss, which downweights the contribution of hard (potentially noisy) examples.
- Adding methods for dealing with noisy labels such as using soft labels or adding a noise layer

## VIII. Conclusion

This study examined the performance and robustness of deep learning architectures, specifically VGG and ResNet models, on a noisy version of the CIFAR-100 dataset. By introducing label noise to simulate real-world imperfections, the study highlighted the challenges of training models on noisy data and explored strategies to enhance their generalization capabilities.

Key techniques such as data augmentation and the use of soft labels proved powerful tools in mitigating the impact of noise. Augmentation methods helped the models learn more generalized patterns and improved their ability to localize meaningful features despite noise. Meanwhile, the adoption of soft labels reduced overconfidence, limiting the detrimental effects of mislabeled data.

The findings demonstrate that robust training techniques combined with carefully chosen architectures, such as ResNet-18, can significantly improve performance on noisy datasets.

## References


[1] XIAO, Ruixuan, et al. Promix: Combating label noise via maximizing clean sample utility. arXiv preprint arXiv:2207.10276, 2022.

[2] ALBERT, Paul, et al. Is your noise correction noisy? PLS: Robustness to label noise with two stage detection. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision. 2023. p. 118-127.

[3] ZHU, Renyu, et al. Rethinking Noisy Label Learning in Real-world Annotation Scenarios from the Noise-type Perspective. arXiv preprint arXiv:2307.16889, 2023.

[4] WIGHTMAN, Ross; TOUVRON, Hugo; JÉGOU, Hervé. Resnet strikes back: An improved training procedure in timm. arXiv preprint arXiv:2110.00476, 2021.

[5] WEI, Jiaheng, et al. Learning with noisy labels revisited: A study using real-world human annotations. arXiv preprint arXiv:2110.12088, 2021.