Project Document: Reproducing "Pure Noise to the Rescue of Insufficient Data"
qwerty

Ayan Hussain
Zi Yang Lee
Yusup Orazov
Daniel Roland

Contents

1 Milestone 1: Project Ideas **1**
- 1.1 Introduction . . . 1
- 1.2 Project Idea 1: Reproducing "Pure Noise to the Rescue of Insufficient Data" . . . 1
- 1.3 Project Idea 2: Reducing Memory Usage via Quantized Weights and Other Methods . . . 2
- 1.4 Conclusions . . . 3

2 Milestone 2: Project Selection **4**
- 2.1 Introduction . . . 4
- 2.2 Problem Specification . . . 4
  - 2.2.1 Problem Statement . . . 4
  - 2.2.2 Motivation . . . 5
  - 2.2.3 Resources . . . 5
- 2.3 Related Work . . . 5
- 2.4 Proposed Method 1: OPeN Implementation . . . 5
- 2.5 Proposed Method 2: OPeN and DAR-BN Integration . . . 6
- 2.6 Conclusions . . . 7

3 Milestone 3: Progress Report 1 **8**
- 3.1 Introduction . . . 8
- 3.2 Related Work . . . 8
- 3.3 Experimental Setup . . . 9
- 3.4 Experimental Results . . . 9
- 3.5 Discussion . . . 9
- 3.6 Work Plan . . . 10
- 3.7 Conclusion . . . 10

4 Milestone 4: Progress Report 2 **12**
- 4.1 Introduction . . . 12
- 4.2 Related Work . . . 13
- 4.3 Experimental Setup . . . 13
  - 4.3.1 Data Processing . . . 13
  - 4.3.2 Model Definition . . . 13

4.3.3 Running the Model 14
4.3.4 Imbalanced (Long-Tail) Datasets for CIFAR-10 and CIFAR-100 14
4.3.5 Oversampling with Pure Noise Images (OPeN) 14
4.3.6 Distribution-Aware Routing Batch Norm (DAR-BN) . . . 15
4.4 Experimental Results 15
4.5 Discussion 17
4.6 Work Plan 17
4.7 Conclusion 17

Abstract

For our semester project, we've honed our focus on Project Idea 1, "Reproducing 'Pure Noise to the Rescue of Insufficient Data'," a decision rooted in its practical scope for the semester and alignment with our current expertise. Our ambition is to dive deep into the design, training, and deployment of neural network models for image recognition, thereby enriching our understanding and skills in the field. This endeavor will involve closely following the innovative methods outlined in Zada, Benou, and Irani's paper, specifically the Oversampling with Pure Noise Images (OPeN) technique and the Differentiable Augmentation Regularization in Batch Normalization (DAR-BN), aiming to verify their claims through direct experimentation.

You are required to update your abstract for each Milestone.

Chapter 1

Milestone 1: Project Ideas

1.1 Introduction

In Milestone 1, we will introduce two project ideas in the field of deep-learning and AI. In order to get a solid grasp of our options, we used the *Project Ideas* page on Canvas and the journals linked there to examine prior research. We additionally discussed with former *Deep Learning* students to get an understand of what scope would be appropriate for a semester project.

The two problems we put forward are, in brief:

1. Reproducing the results of a seemingly counter-intuitive approach to enhancing the generalization performance of image classificiation models, while avoiding the overfitting of other methods.

2. Exploring approaches to building memory-efficient deep neural networks that aren't computationally intensive, with a particular focus on quantized weighting.

With the resources used, we believe both of our project ideas are reasonably appropriate for the semester project.

1.2 Project Idea 1: Reproducing "Pure Noise to the Rescue of Insufficient Data"

In their article, Zada, Benou, and Irani [6] explore a method to increase the generalization performance of image recognition models by adding pure noise images to the training dataset. In their follow-up article, Lee and Lee [4] aimed to test the following claims made in Zada, Benou, and Irani [6]'s paper:

1. OPeN improves model performance on CIFAR-10/100-LT by improving accuracies on classes with lower frequencies.

2. DAR-BN improves the performance of OPeN on CIFAR-10/100-LT compared to baseline Batch Normalization methods.

3. The performance improvement of OPeN is robust under various data augmentation methods.

4. OPeN improves performance on the full CIFAR-10/100 dataset.

Our goal will also be to reproduce Zada, Benou, and Irani [6]'s results by following a similar methods to Lee and Lee [4]. Therefore, we will focus on a color-image classification problem, based off the CIFAR-10/100 dataset with random pure noise images inserted. Additionally, Zada, Benou, and Irani [6] claims that this method avoids overfitting caused by re-sampling and re-weighting during training, a claim which we are considering testing against directly.

Should time permit, we will extend our analysis to other datasets and possibly other models, in order to test and confirm the generalization ability of this method.

1.3 Project Idea 2: Reducing Memory Usage via Quantized Weights and Other Methods

According to Han, Mao, and Dally [2], neural networks often consume a lot of memory and computational resources, making it difficult to be adapted in portable devices and embedded systems. For this project, we will be exploring the following techniques and solutions to address this issue, brought up by Han, Mao, and Dally [2] and Howard et al. [3] for building small, low-latency neural network models tailored for mobile and embedded vision applications:

1. Pruning: Remove redundant connections in the network, keeping only the most informative ones. This reduces the number of parameters.

2. Trained quantization: Quantize the weights to enforce weight sharing, so multiple connections share the same weight, reducing the number of bits needed to represent each connection.

3. Huffman coding: Take advantage of the biased distribution of effective weights and encode them with Huffman coding to further compress the storage.

4. MobileNets: A model architecture that introduces depthwise separable convolutions and two hyperparameters called width multiplier and resolution multiplier. These features allow mobile devices with harware resources to efficiently adapt neural networks while maintaining precise accuracy.

We will experiment with Han, Mao, and Dally [2] and Howard et al. [3]'s techniques and evaluate their performance and efficiency to decide if they are suitable for mobile devices and embedded systems.

1.4 Conclusions

For our project, we considered the following objectives:

1. Testing the effectiveness of inserting pure noise into a dataset in order to reproduce "Pure Noise to the Rescue of Insufficient Data", and
2. Exploring the memory-reduction technique of using quantized weighting in the training and final model, among other compression and simplification techniques.

We found both of these to be strong candidates for the semester project. Moving forward, our most pressing questions are:

- Would either/both of these projects be sufficient as semester projects?
- Is the scope of either/both of these projects reasonably within the constraints of the course and the semester time period?

Please refer to Table 1.1 for qwerty's team members and their contributions for Milestone 1.

Table 1.1: Contributions by team member for Milestone 1.

Team Member	**Contribution**
Ayan Hussain	Project discovery, research.
Zi Yang Lee	Project discovery, report writing.
Yusup Orazov	Project discovery, research.
Daniel Roland	Report setup, project discovery, report writing.

Chapter 2

Milestone 2: Project Selection

2.1 Introduction

For our project, we have decided to work on Project Idea 1, "Reproducing 'Pure Noise to the Rescue of Insufficient Data"', initially proposed in Milestone 1. We decided on Project Idea 1 for a number of reasons, but primarily:

- We believe it to be sufficiently scoped for the semester length.
- It accurately reflects our current experience and understanding compared to the topics of Project Idea 2.

Additionally, we hope to gain hands-on experience in designing, training, and deploying neural network models for image recognition tasks. This project would contribute to our understanding and interest in image recognition by providing neural network models that have been tested and proven to experiment with.

2.2 Problem Specification

2.2.1 Problem Statement

In "Pure Noise to the Rescue of Insufficient Data", Zada, Benou, and Irani [6] explore a method to address insufficient training data in image classification tasks. The method, Oversampling with Pure Noise Images (OPeN), attempts to improve the model performance by augmenting minority-class images with pure noise and introducing a new batch normalization layer ("Differentiable Augmentation Regularization in Batch Normalization", or DAR-BN) to handle noise images separately. The authors also state that OPeN's performance improvement is robust across different data augmentation techniques and can even enhance classification on balanced datasets.

For this project, we aim to follow the methods taken by Zada, Benou, and Irani [6] using OPeN and DAR-BN in order to try and recreate the results found in their paper.

2.2.2 Motivation

Our main goal for this project is to acquire further practical experience in creating, training, and implementing neural network models for tasks involving image recognition. Additionally, we hope to verify the claims made by Zada, Benou, and Irani [6] and the performance results reported through hands-on experimentation.

2.2.3 Resources

For this project, we expect to use the TensorFlow and Scikit-Learn Python libraries and frameworks. We also plan to use the CIFAR-10-LT and CIFAR-100-LT datasets, and, if we choose to explore further datasets, the ImageNet dataset. We don't expect that we will need a computational resource load larger than the current limitations, but this may change as our implementation is realized.

2.3 Related Work

Previously, Lee and Lee [4] have attempted to replicate the experiment conducted by Zada, Benou, and Irani [6]. Overall, Lee and Lee [4] were able to successfully recreate the performance results. Additionally, Lee and Lee [4] extended their investigation to include experiments with the ResNet-32 architecture. They found that while OPeN encourages the model to classify certain images as minority classes, it does not necessarily encode noise or out-of-distribution images to be similar to minority-class images.

We do not plan to experiment with the ResNet-32 architecture unless time permits. However, this paper does provide confirmation that the claims made by Zada, Benou, and Irani [6] are correct.

2.4 Proposed Method 1: OPeN Implementation

In our first proposed method, "OPeN Implementation", we will augment our dataset via the OPeN method and train a model for analyzing performance and accuracy, following the procedure put forward by Zada, Benou, and Irani [6].

Please see Table 2.1 for the individual steps that we expect to perform, as well as the expected time to implement. In total, we expect this method to take approximately 5 weeks.

Table 2.1: OPeN Implementation Procedure

Step	**Expected Time**
Preprocess the CIFAR-10/100-LT dataset via the OPeN method.	1 week
Implement the model architecture.	1 week
Train two instances of the model using the original dataset and the modified dataset.	1 week
Compare accuracy metrics for classes with lower frequencies, comparing model performance with and without the OPeN method.	1 week
Prepare a report of our findings.	1 week

2.5 Proposed Method 2: OPeN and DAR-BN Integration

In our second proposed method, "OPeN and DAR-BN Integration", we will augment our dataset via the OPeN method and train a model for analyzing performance and accuracy. Our model architecture will further analyze the perforfance of the DAR-BN architecture, also put forward by Zada, Benou, and Irani [6].

Please see Table 2.2 for the individual steps that we expect to perform, as well as the expected time to implement. In total, we expect this method to take approximately 5 weeks.

Table 2.2: OPeN and DAR-BN Implementation Procedure

Step	**Expected Time**
Preprocess the CIFAR-10/100-LT dataset via the OPeN method.	1 week
Implement the model architecture, utilizing the DAR-BN architecture method and baseline Batch Normalization.	1 week
Train four instances of the model using the original dataset and the modified dataset, as well as comparing a model with and without DAR-BN.	1 week
Compare accuracy metrics for classes with lower frequencies, comparing model performance with and without the OPeN method and the DAR-BN-enhanced OPeN method.	1 week
Prepare a report of our findings.	1 week

2.6 Conclusions

In this milestone, we decided on reproducing Zada, Benou, and Irani [6]'s paper, "Pure Noise to the Rescue of Insufficient Data". We put forward two methods, with the primary focus of examining the OPeN technique, and optionally, the additional use of DAR-BN to enhance the performance.

While we do not have a strong preference, we are currently leaning towards Method 2, "OPeN and DAR-BN Integration", as this allows us to more fully and accurately recreate the results found by Zada, Benou, and Irani [6].

We have no questions at the time.

Please refer to Table 2.3 for qwerty's team members and their contributions for Milestone 2.

Table 2.3: Contributions by team member for Milestone 2.

Team Member	**Contribution**
Ayan Hussain	Proposed Methods.
Zi Yang Lee	Introduction, Problem Specification, and Related Work.
Yusup Orazov	Proposed Methods.
Daniel Roland	Conclusion, abstract, editing.

Chapter 3

Milestone 3: Progress Report 1

3.1 Introduction

For this milestone, we proceeded with Method 2 outlined in Milestone 2. We focused on the implementation provided by Lee and Lee [4] in their paper. Alongside the paper, Lee and Lee [4] provided a repository with their codebase. In the interest of time, we have decided to adapt this to help in reproducing the results found by Zada, Benou, and Irani [6]. As this was a foreign code base, we focused our efforts leading up to this Progress Report on:

- Understanding Lee and Lee [4]'s implementation.
- Simplifying the codebase by removing features and tests we are not presently evaluating.
- Gaining familiarity with PyTorch, the machine learning library used in Lee and Lee [4]'s implementation.

While we were able to gain some insight into the implementation, due to general availability contraints, we were not able to finish the code evaluation or perform training prior to the Progress Report.

3.2 Related Work

Alongside their paper, Lee and Lee [4] provide a complete codebase to help with reproducing their results, which has been of great help to us. Furthermore, Zada, Benou, and Irani [6] provide implementation snippets, which have helped to clarify understanding and ensure the accuracy of the implementation.

3.3 Experimental Setup

During the initial setup, we had to implement the code from Lee and Lee [4]. According to Lee and Lee [4], the model is the base WideResNet model from Matsubara [5] modified with the DAR-BN layer with the long-tailed dataset from Cao et al. [1]. We are still in the process of getting the code set up and have encountered difficulties getting one of the pip packages `wandb` to function as intended.

Below is the current directory structure of the code, however, we will remove some unwanted files based on our needs and requirements in the upcoming milestones.

Table 3.1: Current Directory Structure

Directory	Purpose
`/`	The main files that run the main functionality of the code such as training the model, plotting confusion matrices, histograms, analyzing the accuracies, saving and loading checkpoints.
`/bash_scripts`	Shell scripts that can be used to run and executed to get the results and accuracies of the experiments.
`/datasets`	Contains datasets such as celeba5, cifar10lt and files to compute the dataset's statistics and imbalancing the cifar10 dataset.
`/initializers`	Contains files that prepares the images before training such as input normalization and transformation
`/models`	Defines the main structure of the model used in the experiments such as the DAR-BN layer and the WideResNet model.

3.4 Experimental Results

As this was predominantly preliminary setup and code analysis, we do not have experimental results prepared for this Progress Report.

3.5 Discussion

So far, we have identified several critical libraries essential for the implementation of our proposed methods. Our project heavily relies on deep learning frameworks, notably PyTorch, for model development and training. Additionally, we utilize libraries such as NumPy for data manipulation andMatplotlib for data visualization.

One significant challenge we encountered was the installation and utilization of specific packages within the SWAN environment. Given the restricted access and limited installation privileges, deploying certain external libraries that are crucial for our project's success became a considerable obstacle. This limitation has prompted us to explore a way to create our own environment with help of Holland Computing Center's staff by utilizing their office hours on Tuesdays and Thursdays.

We will also be analyzing the rest of the files from the source code of the project we are referencing to, to determine if there are more files we can use for our project.

3.6 Work Plan

As we advance through the project, our immediate focus is on the comprehensive analysis, revision, and testing of the existing code base. This phase is critical for ensuring that the foundation of our project is robust and reliable before proceeding with further developments.

Subsequently, our work plan includes the following key activities:

1. **Data Augmentation and Pre-processing:** Utilizing the OPeN method for augmenting the CIFAR-10/100-LT dataset, aiming to achieve a more balanced data distribution that facilitates improved model training.

2. **Model Implementation and Training:** Implementing the DAR-BN architecture within our model and training multiple instances of the model. This includes variations of the dataset (original and modified) and configurations (with and without DAR-BN) to comprehensively evaluate the model's performance.

3. **Evaluation and Comparison:** Assessing the accuracy metrics, particularly for classes with lower frequencies, to determine the impact of our proposed methods on model performance. This analysis will focus on comparing the effectiveness of the OPeN method and the enhancement introduced by integrating DAR-BN.

4. **Documentation and Reporting:** Preparing a detailed report of our findings, which will encapsulate the insights gained, the challenges encountered, and the overall performance of the proposed methods.

3.7 Conclusion

We've navigated through the initial challenges of adapting the codebase by Lee and Lee [4]. This foundation allows us to explore the Oversampling with Pure Noise Images (OPeN) technique and Differentiable Augmentation Regularization in Batch Normalization (DAR-BN), as proposed by Zada, Benou, and Irani [6]. Our groundwork, inspired by the practical implementations and architectural

insights provided by Lee and Lee [4] and further guided by Matsubara [5], sets the stage for our experimental work ahead.

Our future efforts will focus on implementing and evaluating the DAR-BN within either the WideResNet model or our own model, using both augmented and original datasets. This phase aims to rigorously assess the impact of these methods on enhancing model performance, particularly for imbalanced datasets. Through detailed evaluation and documentation, we aspire to contribute valuable insights to the field, substantiating the effectiveness of leveraging noise for data augmentation.

In essence, our project is poised to transition from a preliminary setup to an experimental and evaluative phase. We are committed to advancing our understanding of neural network training for image recognition tasks, especially in the context of imbalanced datasets, with the anticipation of providing meaningful contributions to the domain.

Please refer to Table 3.2 for qwerty's team members and their contributions for Milestone 3.

Table 3.2: Contributions by team member for Milestone 3.

Team Member	**Contribution**
Ayan Hussain	Code Review, Conclusion
Zi Yang Lee	Code Review, Project Setup, Experimental Setup
Yusup Orazov	Code Review, Discussion, Work Plan
Daniel Roland	Code Review, Introduction, Related Work

Chapter 4

Milestone 4: Progress Report 2

4.1 Introduction

For our semester project, we are attempting to reproduce the findings in Zada, Benou, and Irani [6]'s paper. In our previous milestone, we had expressed interest in adapting the code used by Lee and Lee [4]. After considering the code, and with Dr. Stephen Scott's advice, we have decided to instead reimplement Zada, Benou, and Irani [6]'s original specification detailed in their report, as Lee and Lee [4]'s implementation goes beyond the scope of the project and uses unfamiliar libraries.

For this Milestone, we have implemented the basic structure of Zada, Benou, and Irani [6]'s specification. Currently, we have implemented methods for inserting noise data, as well as the basic structure of the Distribution-Aware Routing Batch Norm (DAR-BN) layer. Notable shortcomings currently include:

- We do not currently utilize the Long Tail variants of the CIFAR-10 and CIFAR-100 datasets. From our knowledge, these are not datasets supplied by Tensorflow, and so we need to develop our own filtering/selection process.
- While the intended noise-addition implementation uses a normal distribution based on the dataset (i.e., with the same mean and variance in each color channel), ours does not yet.

Despite these shortcomings, we have provided in Section 4.4 some preliminary tests of the implementation.

4.2 Related Work

In our work, we have mostly referenced Zada, Benou, and Irani [6] for implementing OPeN and DAR-BN into code instead of Lee and Lee [4]. We wanted to understand the reason for implementing OPeN and DAR-BN and to reproduce OPeN and DAR-BN from their code into Tensorflow to replicate their experimental results.

4.3 Experimental Setup

In this section, we outline the steps and methodologies used in preparing, defining, and running our experimental models, focusing specifically on the CIFAR-10 dataset. Do note we will expand it to the CIFAR-100 dataset as well as their long-tailed versions.

4.3.1 Data Processing

The CIFAR-10 dataset, consisting of 60,000 images split into training and test sets, was used. Each image was first normalized to have pixel values between 0 and 1 to facilitate faster convergence during training. To enhance the dataset and test the model's robustness, noise images generated with a uniform distribution were integrated into the training dataset. The noise images accounted for approximately 50% of the training data, which were then mixed with the original images to create a new augmented training set.

4.3.2 Model Definition

The model's architecture was defined using a convolutional neural network (CNN) framework tailored for image classification tasks. Key components of the model included:

- Convolutional layers to extract spatial hierarchies of features.
- Custom Distribution-Aware Routing Batch Normalization (DAR-BN) layers designed to normalize activation maps differently during training and inference, based on the batch statistics and the entire dataset statistics, respectively.
- Activation functions, specifically ReLU, to introduce non-linearity into the model, allowing it to learn more complex patterns.
- A final dense layer with softmax activation to classify the images into one of ten categories.

4.3.3 Running the Model

The model was compiled with the Adam optimizer, chosen for its efficiency in handling sparse gradients and adaptive learning rate capabilities. Training was conducted over 25 epochs with a batch size of 64, where both training and validation losses and accuracies were monitored to gauge the model's performance and generalization capabilities. This setup was particularly focused on observing the impact of the introduced noise on the model's ability to generalize from training to unseen data.

4.3.4 Imbalanced (Long-Tail) Datasets for CIFAR-10 and CIFAR-100

We will be using CIFAR-10-LT and CIFAR-100-LT for the experiment, which are the long-tailed imbalanced datasets of CIFAR-10 and CIFAR-100. For each of the two datasets, each will contain C classes $\{1, 2, ..., C\}$ where each class c_i will have n_i training images. Based on Zada, Benou, and Irani [6]'s implementation, the number of training images will decrease as it moves from the first class to the last class such that $n_1 \geq n_2 \geq ... \geq n_C$ to simulate a scenario of imbalanced training data with majority and minority classes.

CIFAR-10-LT

CIFAR-10 has 6000 images per class, 5000 training images per class, and 1000 testing images per class. In CIFAR-10-LT, n_1 will have 5000 training images and n_{10} will have 100 training images. The number of training images will exponentially decrease from each class to simulate a scenario of having an imbalanced dataset. According to Zada, Benou, and Irani [6], we will have a total of 13,996 training images for the CIFAR-10-LT dataset. The test images will remain the same as CIFAR-10 during testing.

CIFAR-100-LT

CIFAR-100 has 600 images per class, 500 training images per class, and 100 testing images per class. In CIFAR-100-LT, n_1 will have 500 training images and n_{10} will have 10 training images. The number of training images will exponentially decrease from each class to simulate a scenario of having an imbalanced dataset. According to Zada, Benou, and Irani [6], we will have a total of 12,608 training images for the CIFAR-100-LT dataset. The test images will remain the same as CIFAR-100 during testing.

4.3.5 Oversampling with Pure Noise Images (OPeN)

1. Oversample the minority classes to have the same number of training images as the largest majority class itself. So $n_1 = n_2 = ... = n_C$.

2. Replace part of the oversampled images with pure random noise images, with the following probability:

   $P(\text{replace } x \text{ with } x_{noise} \text{ for each class}) = (1 - n_i/n_{max}) \times \delta$

   x = an oversampled image
   x_{noise} = pure random noise image
   n_i = number of training images in class i
   n_{max} = max number of training images in a class
   $\delta \in [0, 1]$ = a hyperparameter defining the ratio between pure noise images and natural images

3. Add in the pure random noise by first finding the mean, μ, and standard deviation, σ, for each color channel. Then, it is sampled from the normal distribution within the domain $[0, 1]$: $x_{noise} \sim N(\mu, \sigma)$

This can be seen in Figure 4.1.

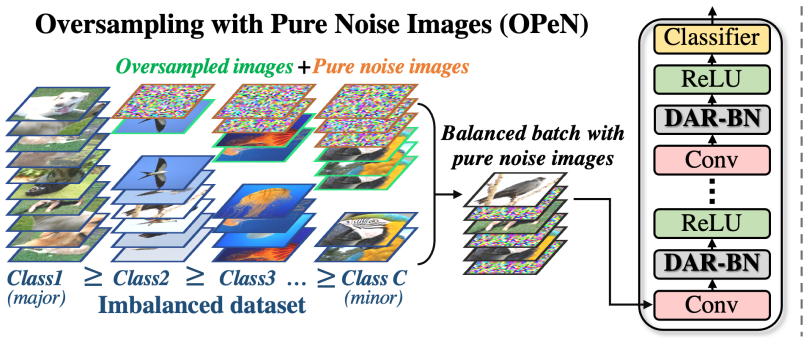

Figure 4.1: A depiction of the OPeN augmentation method.

4.3.6 Distribution-Aware Routing Batch Norm (DAR-BN)

Instead of a standard batch normalization layer, a modified version of the batch normalization layer is used alongside OPeN to compensate for the shift in different distributions of natural and noise images. DAR-BN distributes the training images in the batch into two categories, natural images and noise images. It will normalize the inputs of natural images and noise images separately and obtain their respective outputs, out_{nat} and out_{noise}. This can be seen in Figure 4.2.

4.4 Experimental Results

We tested the algorithm on a small model for baseline performance analysis. The model consists of:

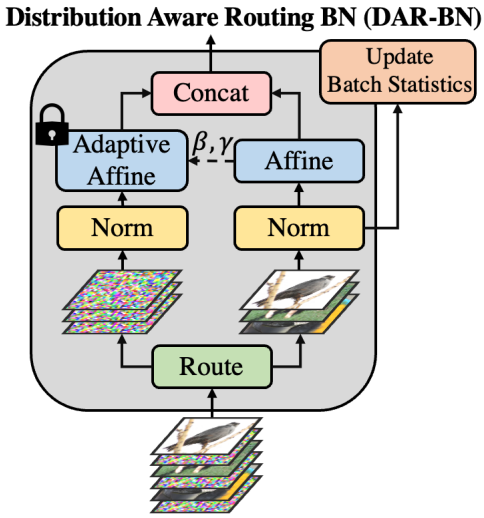

Figure 4.2: A depiction of the DAR-BN modified batch normalization layer.

1. A Convolutional 2D layer with a ReLU activation function.
2. The Batch Normalization layer (DAR-BN or standard.)
3. The output layer.

The DAR-BN model uses an augmented dataset, while the standard model uses the original CIFAR-10 dataset.

In Figure 4.3, we report the testing and validation accuracy during training. We did not compute accuracy against the testing set during this testing.

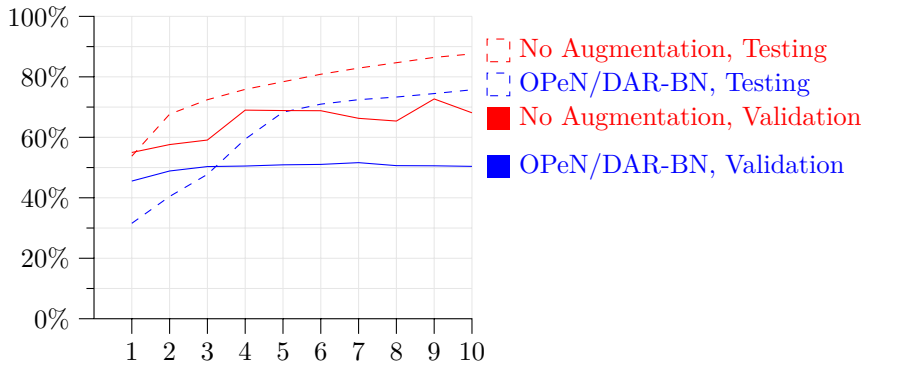

Figure 4.3: CIFAR-10 Performance

4.5 Discussion

Our results show that the model using the OPeN/DAR-BN technique underperformed the unaugmented model. We suspect that there are three primary causes:

- As we are not using the LT datasets, we applied a fixed number of noisy images to each class, which almost certainly degraded performance.
- As mentioned in Section 4.1, our noise generation implementation does not respect the input dataset, so the noise may be 'too noisy' compared to the other images in each class, serving to confuse the model.
- The models are also quite small and trained for a very short time, so there was likely an upper bound on performance.

Taking these factors together, we can only determine that the OPeN/DAR-BN technique does not cause the model to degenerate into guessing.

4.6 Work Plan

As noted in Section 4.1, we are currently missing two features: Long-Tail dataset generation and noise based on the input dataset. Thus, we plan to implement these features.

We also plan to expand our tests to include:

- A comparison of performance with the addition of Pure Noise and DAR-BN, to confirm the findings of Zada, Benou, and Irani [6].
- A comparison of performance with the use of DAR-BN versus standard Batch Normalization, to confirm a remark that standard Batch Normalization will reduce performance when Pure Noise is present.
- An analysis on the performance on more datasets, including CIFAR-10, CIFAR-10-LT, CIFAR-100, and CIFAR-100-LT.

We also note that, as a result of technical complications, there are a shortage of visualization aids for the tests in this Milestone. These complications will be corrected or circumvented for the Final Report.

4.7 Conclusion

In our project, we endeavored to reproduce the findings from Zada, Benou, and Irani's study on using Pure Noise to enhance model training in data-scarce environments. Through our efforts, we implemented and tested OPeN and DAR-BN, discovering challenges primarily around noise generation and dataset limitations that impacted model performance. Moving forward, we plan to refine our noise addition protocols, better adapt our models to handle imbalanced

datasets, and extend our testing to more robust datasets to verify and enhance the reproducibility and effectiveness of the methods studied.

We had the following questions regarding the Final Report and presentation:

- Do we need to include our own progress, such as development progress, hurdles, and challenges encountered, in our presentation?

Please refer to Table 4.1 for qwerty's team members and their contributions for Milestone 4.

Table 4.1: Contributions by team member for Milestone 4.

Team Member	**Contribution**
Ayan Hussain	Abstract, conclusion, proofreading.
Zi Yang Lee	Preliminary tests; related work, experimental setup.
Yusup Orazov	Code implementation, preliminary tests.
Daniel Roland	Introduction, discussion, work plan, figures.

Bibliography

[1] Kaidi Cao et al. "Learning Imbalanced Datasets with Label-Distribution-Aware Margin Loss". In: (2019). arXiv: 1906.07413 [cs.LG].

[2] Song Han, Huizi Mao, and William J. Dally. "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding". In: (2016). arXiv: 1510.00149 [cs.CV].

[3] Andrew G. Howard et al. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications". In: *CoRR* abs/1704.04861 (2017). arXiv: 1704.04861. URL: http://arxiv.org/abs/1704.04861.

[4] Seungjay Ryan Lee and Seungmin Brian Lee. "[Re] Pure Noise to the Rescue of Insufficient Data". In: *ReScience* (July 2023). Ed. by Koustuv Sinha, Maurits Bleeker, and Samarth Bhargav. DOI: 10.5281/zenodo.8173763.

[5] Yoshitomo Matsubara. "torchdistill: A Modular, Configuration-Driven Framework for Knowledge Distillation". In: (2021), pp. 24–44. ISSN: 1611-3349. DOI: 10.1007/978-3-030-76423-4_3. URL: http://dx.doi.org/10.1007/978-3-030-76423-4_3.

[6] Shiran Zada, Itay Benou, and Michal Irani. "Pure Noise to the Rescue of Insufficient Data: Improving Imbalanced Classification by Training on Random Noise Images". In: *CoRR* abs/2112.08810 (2021). arXiv: 2112.08810. URL: https://arxiv.org/abs/2112.08810.