Paleo Oceanography Model Report

Ethan Chung
University of Hawaii
ICS 637: Deep Learning with Neural Networks
[redacted]@hawaii.edu

November 20, 2024

1 Introduction

This project applies image classification to identify single-celled organisms known as foraminifera (forams) from microscope images. These images are sourced from borehole samples taken from the Santa Barbara Basin, spanning an 800-year interval during the Common Era (1249–2008 CE) [1]. The primary goal of this project is to classify the input images into one of 53 foraminifera species labels, with an additional label for "non-forams" classifications.

Foraminifera play a crucial role in climate change as marine calcifiers. When foraminifera die, their shells, which are built up by calcium carbonate, sink to the ocean floor, neutralizing acidity and helping regulate atmospheric carbon dioxide over time. Rising CO2 levels and ocean acidification threaten foraminifera by causing stress, reducing metabolism, and impairing shell repair. This decreases calcification, weakening their ability to neutralize acidity, which results in an increase of ocean acidification, thereby disrupting the carbon cycle [2].

2 Dataset

The dataset for this model is originally sourced from Kahanamoku et al. (2023); however, only a portion of the entire dataset is provided for the Kaggle competition. The dataset provided is pre-split into training and test sets, consisting of 8,653 and 2,174 images, respectively. The training set is further divided into an 80/20 split for training and validation. Each image in the training set is labeled with one of 53 foraminifera species, including an extra label to classify non-forams. The dataset is imbalanced, as shown in Figure 1.

The model is trained on the training set, while the validation set is used for learning rate adjustment and model selection. To evaluate the model, predictions are made on the test set, outputted to a file, and submitted to the Kaggle competition, where they are evaluated on public and private leaderboards to compute the corresponding zero-one loss (accuracy) scores.

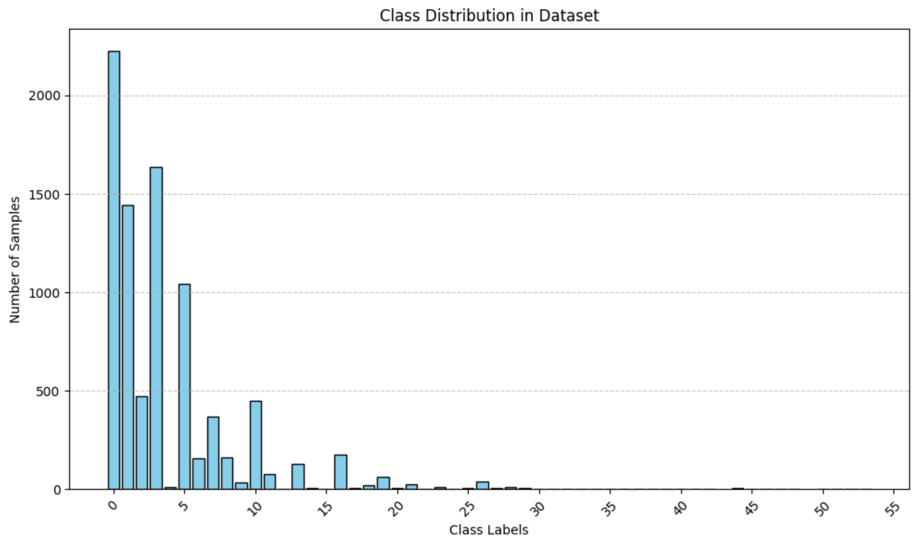

Figure 1: Most samples in the dataset are concentrated in a few classes, while many others have very few samples. All classes had at least one sample.

3 Model Architecture

Convolutional Neural Networks (CNNs) are well-suited for image classification by learning patterns like edges, textures, and shapes directly from image data. This project employs a pre-trained ResNet-50 model for transfer learning. The early layers of the pre-trained ResNet-50 are frozen to retain general features, while the last three layers are unfrozen for fine-tuning to learn specific features for foraminifera classification.

This model is trained with mini-batch gradient descent with batch sizes of `100` for `2` iterations per epoch. The loss function being optimized is categorical cross-entropy loss. The optimization is performed using the AdamW optimizer, with a learning rate of `0.001` and weight decay of `1e-5`. The learning rate is dynamically adjusted using the ReduceLROnPlateau scheduler, which monitors the validation loss and reduces the learning rate by a factor of `0.5` if the validation loss does not improve for `2` consecutive epochs. Hyper-parameter values were chosen manually as sensible defaults, and no further hyper-parameter tuning was performed.

4 Features & Pre-processing

4.1 Image Pre-processing

All images are rescaled to a resolution of `224x224` pixels, which aligns with the original input size of the images used to train the ResNet-50 model. Additionally, the image channels are normalized to a mean of `[0.5, 0.5, 0.5]` and a standard deviation of `[0.225, 0.225, 0.225]`.

4.2 Data Augmentation

As the dataset is rather small, data augmentation is used to increase diversity by generating modified variants transformed by translation, rotation, contrast, inversion, etc. The following auto-augmentation methods are tested:

- AutoAugment: Applies augmentations based on learned policies, which specify the probability of applying a particular augmentation and the magnitude of applying that augmentation.

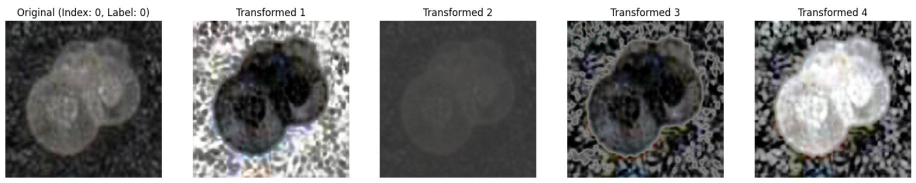

Figure 2: AutoAugment sample augmentations

- RandAugment: Randomly selects augmentations from a predefined set of transformations, applying them with uniform probability.

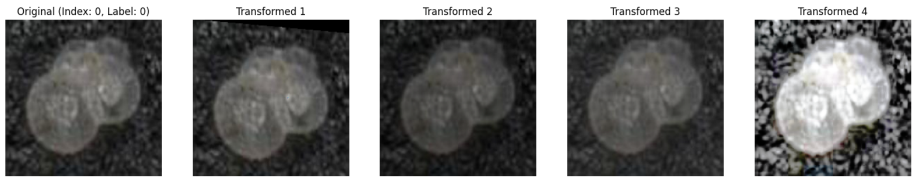

Figure 3: RandAugment sample augmentations

- TrivialAugmentWide: Applies a single augmentation per image, with the transformation randomly selected for each image.

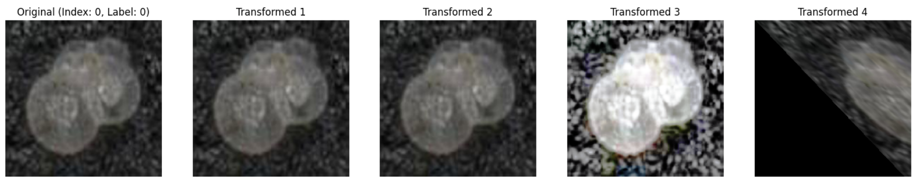

Figure 4: TrivialAugmentWide sample augmentations

- AugMix: Combines multiple augmentations to maintain semantic similarity while introducing variability.

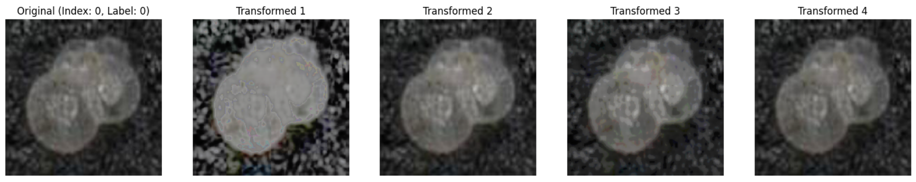

Figure 5: AugMix sample augmentations

5 Results

The classification model's performance was assessed using **zero-one loss (accuracy)** on the validation set during training and on a held-out test set. The predictions obtained from evaluating the model on the test set are then submitted to the Kaggle competition, where two held-out public and private leaderboard sets are evaluated using the predictions from the test set to obtain an accuracy score. Table 1 summarizes the results for the various auto augmentation strategies utilized.

The baseline model (without augmentation) achieved the highest validation accuracy of 84.70%, matched only by the AugMix method. However, AugMix underperformed on the public and private leaderboards, scoring 81.36% and 81.01%, respectively. RandAugment demonstrated strong performance across all metrics, with validation accuracy reaching 84.20% and leaderboard scores of 82.73% (public) and 83.61% (private), making it the most balanced method.

AutoAugment and TrivialAugmentWide displayed slightly lower validation accuracies of 81.90% and 83.20%, respectively. AutoAugment showed consistent performance across metrics, with a marginal -0.08% decrease in the public leaderboard and a +0.78% improvement in the private leaderboard compared to its validation accuracy. TrivialAugmentWide achieved moderate results on the validation set, but experienced significant drops in accuracy on both leaderboards, with -3.20% (public) and -1.17% (private).

While AutoAugment demonstrated consistent performance across metrics, suggesting minimal overfitting on the training dataset, RandAugment ultimately yielded the best overall results on both the public and private leaderboards.

Table 1: Model Accuracies Using Different Auto-Augmentation Techniques

Augmentation Method	**Best Validation Accuracy (%)**	**Public Accuracy (%)**	**Private Accuracy (%)**
None (Baseline)	**84.70**	81.36	82.87
AutoAugment	81.90	81.82	82.68
RandAugment	84.20	**82.73**	**83.61**
TrivialAugmentWide	83.20	80.00	82.03
AugMix	**84.70**	81.36	81.01

6 Conclusion

Among the five models trained, the model using RandAugment achieved the best zero-one loss (accuracy) on both the public and private leaderboards, with 82.73% and 83.61% accuracy, respectively. However, a likely discrepancy between the training and test datasets is that the training set may be more skewed toward certain labels compared to the test set. This imbalance can limit the model's ability to generalize to unseen data. More training data alongside a more balanced dataset could improve both the model's performance and generalization.

Although hyper-parameter tuning was manually performed by testing various auto-augmentation techniques, hyper-parameter tuning was not conducted on parameters like the learning rate or weight decay for the AdamW optimizer. Using a method like GridSearchCV to explore the hyper-parameter search space could yield better performance on the public and private leaderboards. In addition, incorporating evaluation metrics other than accuracy may better represent the model's effectiveness. Yuan et al. (2015) suggests that the Average Precision (AP) metric, derived from the area under the Precision-Recall (PR) curve, provides a more representative measure of performance in cases of highly imbalanced datasets [3].

7 Data Availability

The original dataset can be sourced from Kahanamoku et al. (2023) [1], while the adapted dataset used for this project can be found from the Kaggle competition at `https://www.kaggle.com/competitions/Paleo-Oceanography`.

8 Code Availability

All code for this report is available at `https://github.com/echung32/ics637-paleo-oceanography`.

References

[1] Kahanamoku, S., Samuels-Fair, M., Kamel, S. M., Stewart, D., Kahn, L., Titcomb, M., Mei, Y. A., Bridge, R. C., Li, Y. S., Sinco, C., Epino, J. T., Gonzalez-Marin, G., Latt, C., Fergus, H., & Finnegan, S. (2023). Twenty-two thousand Common Era benthic foraminifera from the Santa Barbara Basin [Data set]. Zenodo. `https://doi.org/10.5281/zenodo.10067274`

[2] Kerlin, K. E. (2017). Tiny Shells Indicate Big Changes to Global Carbon Cycle. UC Davis. `https://www.ucdavis.edu/news/tiny-shells-indicate-big-changes-global-carbon-cycle`

[3] Yuan, Y., Su, W., & Zhu, M. (2015). Threshold-free measures for assessing the performance of medical screening tests. Frontiers in public health, 3, 57. `https://doi.org/10.3389/fpubh.2015.00057`