Image Transformations for Domain Generalization on Fruit Datasets

Brett O'Connor
Electrical and Computer Engineering
Cornell Tech
New York, NY 10044
bwo7@cornell.edu

Max Bonzulak
Computer Science
Cornell Tech
New York, NY 10044
mwb233@cornell.edu

Abstract

Domain Generalization is considered to be an "open problem" in the field of deep learning. In this paper, we apply image transformation techniques to training data, aiming to improve generalizability of trained models. Specifically, train a CNN to classify fruit based on cropped images with no background. Our findings suggest that preprocessing training images with transformations such as random translations and noise offer a minimal improvement in context-introduction generalizability. Our findings do not indicate that dataset size has a significant impact on these technique's success.

1 Introduction

Convolutional neural networks (CNN) enable many crucial technologies in today's world [15], [16], [9]. However, applying trained CNNs in practice can result in some problems. Specifically, models trained on datasets from a specific Independent and Identical Distribution (IID) might not perform well when applied to out-of-distribution data [3]. The effort to address this problem is known as Domain Generalization (DG) [2], where an attempt is made to improve model resilience across a domain shift. In this paper, we introduce an instance of domain shift: context-introduction shift. We define this as the shift from out-of-context images to in-context images. Improving generalization across this particular type of domain shift is important because many datasets are created in a controlled environment, documenting idealized versions of their subject matter [10]. Models trained on this out-of-context data are particularly at-risk to perform poorly in the wild, where subject matter is embedded in the context of a complex scene. We have selected two datasets with a substantial context-introduction shift between them: Fruit360 [10], which consists of contextless, cropped, "idealized" images of fruit, and VegFru [11], which contains a wide array of fruit images within rich contexts, such as markets or farms. We selected these datasets because they serve as optimal counterparts in a context-introduction shift. We aim to improve the ability of a model *trained on the out-of-context Fruit360 dataset* to generalize to in-context VegFru examples. To do this, we adopt the technique of augmenting training data with random translations [4]. Additionally, we propose a new technique of augmenting training data with random noise perturbations, inspired by techniques that do the same on latent feature representations [12]. We evaluate the generalizability of models trained with both large and small datasets with these augmentations.

Our main contributions are:

1. We show the efficacy of random image translations for context-introduction generalizability.
2. We show the efficacy of random image perturbation for context-introduction generalizability.
3. For both techniques, we show the degree of improvement for multiple dataset sizes.

2 Background

The need for domain generalization was first introduced as related to medical imaging from patient to patient [1]. Since the problem of Domain Generalization (DG) was formally introduced [2], major strides have been made. Today, there are many techniques that aim to improve generalizability, including domain alignment, ensemble learning, and data augmentation [3]. Within the category of Data Augmentation techniques (DA), there are several approaches. However, many of these approaches require data from the target distribution to be available before model training. This can be impractical in real-world applications such as traffic scene segmentation, where data representative of all possible driving conditions may not be available [3]. Instead, we are focusing on transformation techniques that do not rely on the out-of-distribution target data. We consider two of these approaches: image transformations, and embedded feature augmentation. Image transformations such as flips, rotations, and color augmentation have been shown to improve domain generalizability [4], [5], [6], [7], [8]. Theoretically, transforming images can introduce some of the variability that a model might encounter in the wild. As such, transformations can be hand-picked to introduce variations that correspond with the anticipated deployment environment [3]. Because our training set uses centered, cut-out images, we suspect that a weakness of the model will be adapting to examples that are off-center and in-context, with the background included. Therefore, for our first attempted technique, we augment the training data with random translations. The second transformation technique we consider is embedded feature augmentation, which perturbs latent features with Gaussian noise [12], [13], [14]. For our second attempted technique, we draw inspiration from embedded feature augmentation and image transformations: we apply a Gaussian perturbation directly on the training data, with the expectation that the model might better adapt to complex backgrounds and also to general variations of the subject itself.

3 Experiment

3.1 Structure

Our experiment consists of one control group and two treatment groups. The datasets we used for our experiment consist of different types of fruit, where one dataset contains images of single fruits with white backgrounds and the other dataset contains images of fruits with different backgrounds. For our control group, we trained a model on the dataset that contains only white backgrounds and we evaluated this model on both a test dataset with white backgrounds and the dataset with different backgrounds. Additionally, we trained a model on a smaller training split of white background data and evaluated our model on both the test split of white background data and the dataset of different background data. Figure 1 depicts a flow diagram of our control group:

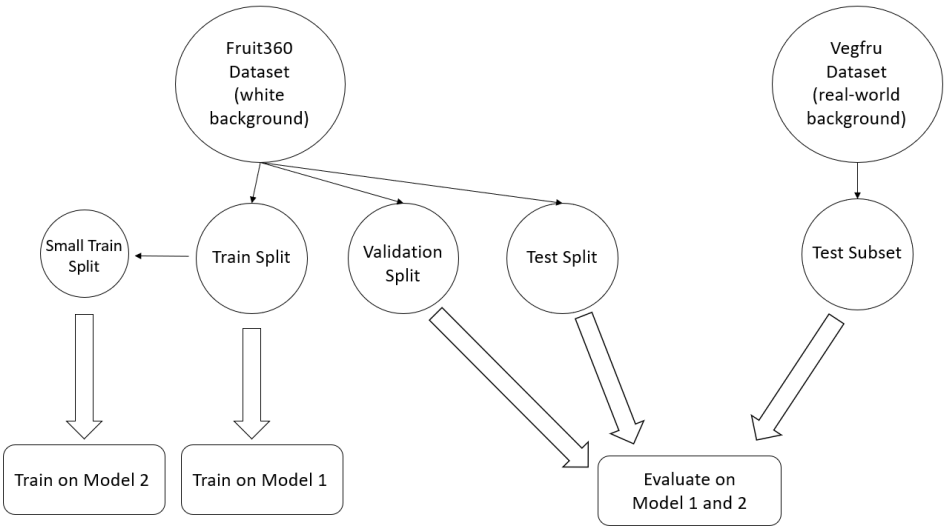

Figure 1: Control Group Experiment

For our first treatment group, we took the white background dataset and augmented the training split with randomized noise. We then trained a model on this data and tested it on a test split of white background data and the different background data. As we did in our control group, we also trained a model on a smaller subset of the training split of the augmented data and tested it on the same test datasets previously mentioned. Figure 2 shows the method of our first treatment group:

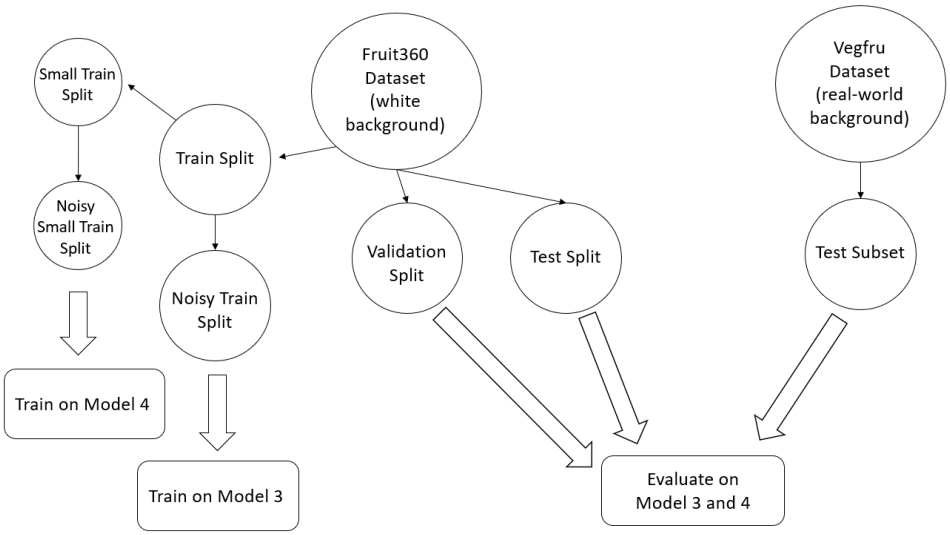

Figure 2: Treatment Group 1 Experiment

For our second treatment group, we again used the white background dataset and augmented the training split using image transformations. The specific image transformations that we used were random transposing, random cropping, reflections, and padding. With this augmented training split, we trained a model and tested it on the same test data, specified previously. Additionally, like our other groups, we trained a model on a smaller subset of this augmented training data and tested it on the same test data we have been using for each of our groups. Figure 3 shows the method of our second treatment group:

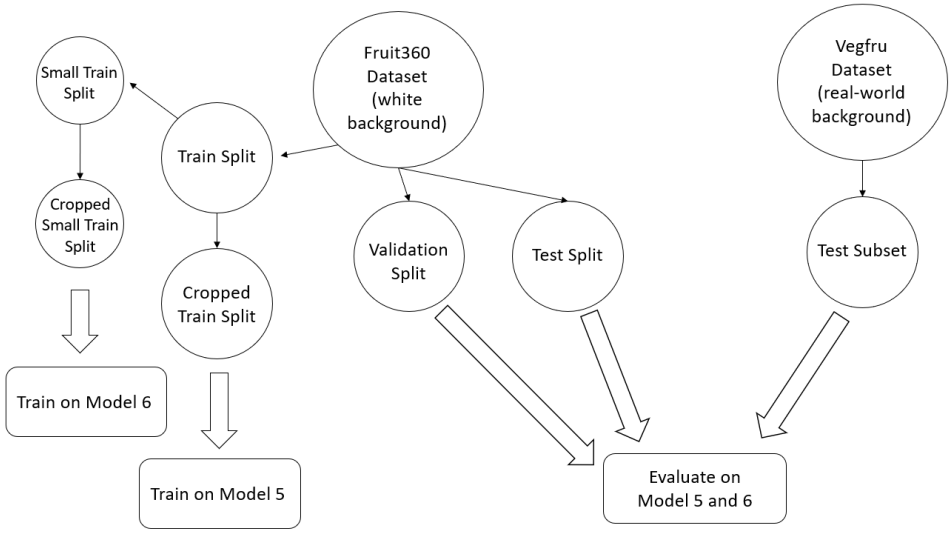

Figure 3: Treatment Group 2 Experiment

3.2 Datasets

As mentioned before, we are using two primary datasets: one dataset contains single fruits with white backgrounds, which was obtained from the Fruit360 [10] dataset, and the other dataset contains different backgrounds with an arbitrary number of fruits in each image, which was obtained from the Vegfru dataset [11]. Figure 4 is a single image from the Fruit360 dataset:

Figure 4: Fruit360 Example Image

Figure 5 is a single image from the Vegfru dataset:

Figure 5: Vegfru Example Image

In total the Fruit360 data we used contained 27,136 images and the Vegfru data we used contained 31,305 images. These amounts were obtained using preprocessing, as we made sure to use the data from both datasets that had mutual classes. Figures 6 and 7 show the class distributions of the Fruit360 dataset and the Vegfru dataset respectively:

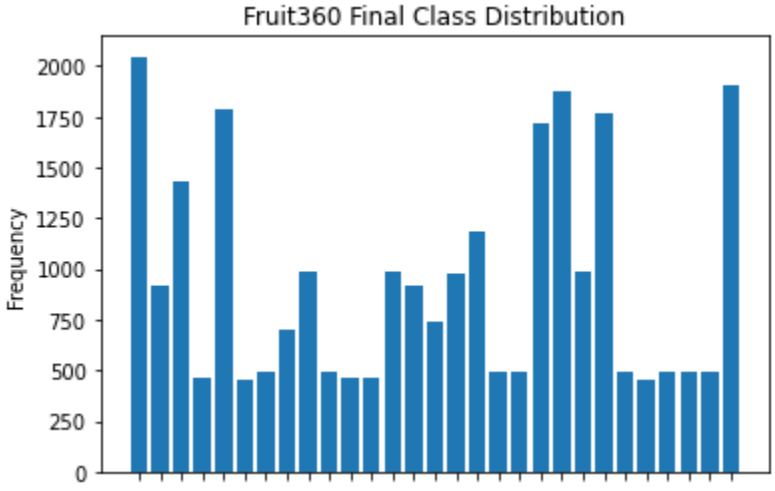

Figure 6: Fruit360 Class Distribution

X-Axis Labels from left to right: 'apple', 'avocado', 'banana', 'pear', 'blueberry', 'nut', 'cherry', 'tomato', 'chestnut', 'fig', 'peach', 'melon', 'grapefruit', 'dates', 'guava', 'hazelnut', 'kiwi', 'lemon', 'mango', 'nectarine', 'papaya', 'passion', 'pineapple', 'plum', 'pomegranate', 'pomelo', 'rambutan', 'raspberry', 'salak'

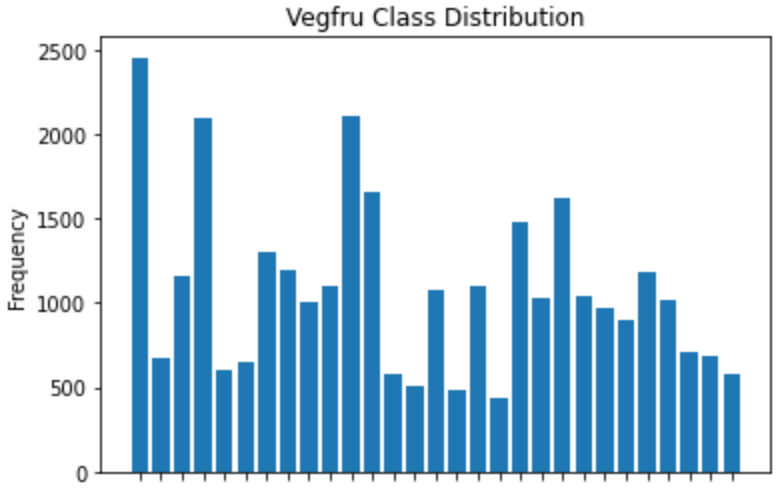

Figure 7: Vegfru Class Distribution

X-Axis Labels from left to right: 'apple', 'avocado', 'banana', 'pear', 'blueberry', 'nut', 'cherry', 'tomato', 'chestnut', 'fig', 'peach', 'melon', 'grapefruit', 'dates', 'guava', 'hazelnut', 'kiwi', 'lemon', 'mango', 'nectarine', 'papaya', 'passion', 'pineapple', 'plum', 'pomegranate', 'pomelo', 'rambutan', 'raspberry', 'salak'

For the test data from our Vegfru dataset, we decided to take 20% of our Vegfru dataset to have 6,251 Vegfru images for testing.

With our Fruit360 dataset, we created splits for training, validation, and test data. The following percentages were used to create these splits: 60% training data, 20% validation data, 20% test data. This gave us 16,281 images for training, 5,427 images for validation, and 5,427 images for testing. Additionally, we took 25% of our training data to create a small training dataset that we would use for another training session. This gives us 4,070 images for our small training dataset.

For both augmentation sets, we used the same proportions previously stated for our training dataset and our small training dataset. The augmentations were specifically applied to the Fruit360 dataset images.

3.3 Preprocessing

In order to introduce our data to the models we will be training and evaluating with, we spent a large amount of effort and time preprocessing. This process began with us trimming down both of the initial Fruit360 and Vegfru datasets that we downloaded from Kaggle. The reason why we trimmed the data was because the raw data classes were not mutual between each dataset. Therefore, to have a mutual list of classes, we strategically analyzed both datasets' respective class names and constructed a "master class list" that included classes that were in common with both datasets. This mutual class list contained 28 different classes.

After trimming this data, we decided to take a look at the distribution of data between each class in each dataset. We found that the Vegfru data was nicely distributed, with Figure 7 depicting the class distribution. However, the Fruit360 data had abnormally large amounts of data within the "apple", "cherry", "pear", and "tomato" classes, as shown in Figure 8 below:

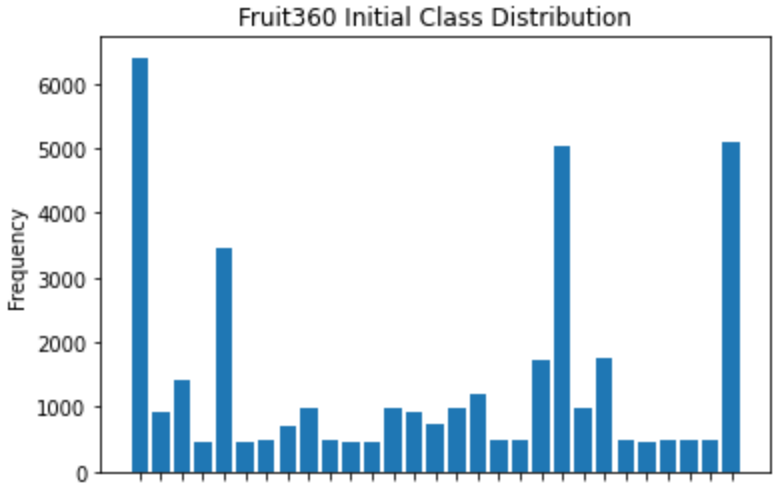

Figure 8: Fruit360 Disproportionate Class Distribution

X-Axis Labels from left to right: 'apple', 'avocado', 'banana', 'pear', 'blueberry', 'nut', 'cherry', 'tomato', 'chestnut', 'fig', 'peach', 'melon', 'grapefruit', 'dates', 'guava', 'hazelnut', 'kiwi', 'lemon', 'mango', 'nectarine', 'papaya', 'passion', 'pineapple', 'plum', 'pomegranate', 'pomelo', 'rambutan', 'raspberry', 'salak'

To handle this issue, we randomly removed samples from each of these "problem" classes to achieve a better class distribution. Figure 6, displayed above, shows the final class distribution of the Fruits360 dataset that we used.

Following improving our class distributions, we wanted all of our images in both datasets to be the same size. Because the Fruit360 dataset came downloaded as 100 pixel by 100 pixel images, we felt that this was extremely nice to work with, and therefore, we resized each image in our Vegfru dataset to 100 pixels by 100 pixels.

From here, we split our data into training, small training, validation, and test splits for the Fruit360 dataset and we obtained a test subset from the Vegfru dataset. The exact sizes of these data splits and subsets are specified above in the "Datasets" section of this paper. We also ensured that we shuffled our data before doing these splits.

After performing these data splits, we then generated two different augmentations of our training data and our small training data. The two different augmentations were 1) adding noise and 2) doing random translations, random cropping, padding, and reflections. For adding noise, for each image in our training and small training datasets, we created a 100 pixel by 100 pixel random Gaussian noise image and merged this image with the image in our dataset. For this merger, we used an alpha parameter of 0.3, which signified how heavily the noise was applied to the original image. This gave us a training dataset with added noise and a small training dataset with added noise (example shown below in Figure 9). For our augmentation involving random translations, random cropping, padding, and reflections, we created a custom Dataset class with Pytorch where we used the Random Cropping method within Pytorch's Transforms to apply these image transformations on our original training and small training datasets. Specifically, we randomly cropped, which also translated our image, to an image size of 224 pixels by 224 pixels (as wanted by our model), we padded with a weight of 30, and we set the padding mode to "reflect" to implement a reflection based on where the image was translated (example shown below in Figure 10).

Figure 9: Noisy Fruit360 Example

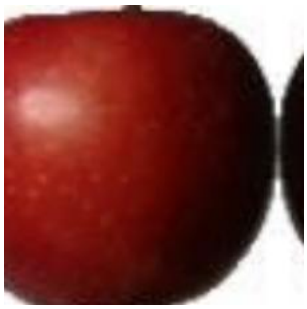

Figure 10: Cropped Fruit360 Example

At this point our data was fully processed, so we created a custom Dataset class with Pytorch so that we could use Pytorch's Data Loader structure with our model. For creating our Pytorch Datasets, we applied the necessary transformations desired by our models. We then loaded these Datasets as Data Loaders, with a batch size of 64 and ensured that our data was shuffled.

3.4 Model

For the models we used, we decided to use an untrained Resnet18 neural network for each training session that we did. We contemplated on creating our own neural network, however, as proposed in the feedback of our project proposal, we found it would be best to use a well-established convolutional neural network, and therefore, we chose Resnet. We did all of our training first, which included checking our progress on our validation datasets, and then we did our evaluations. The procedures with our models underwent the following structure: initialize the model, enter the model into training mode, train the model, save these parameters, enter the model into evaluation mode, evaluate the model on our validation data, tweak hyperparameters and re-train until we achieved high accuracies for our validation data, evaluate the model on our test data.

3.5 Hyperparameters

Through numerous training sessions for each training dataset size, we were able to find that our models' loss values all converged after around the 50 epoch region. Therefore, we decided to run our final training sessions to 75 epochs and saved the model parameters that yielded the best lost. We used stochastic gradient descent with momentum to optimize our models. In terms of our hyperparameters, we primarily tweaked our learning rate for each model, but we found that the best learning rate for each model was 0.001. We initially attempted to make changes to momentum with our small training datasets, however, after numerous changes we found that the value of 0.9 was working well for these datasets. We selected cross-entropy loss as our objective function.

4 Results

Throughout each training session, our models experienced a convergence to a low loss value after the 75 epochs we trained them for. The following are our training loss plots, where the y-axis consists of the training loss and the x-axis consists of the epoch during training. Each graph is properly labeled via its title.

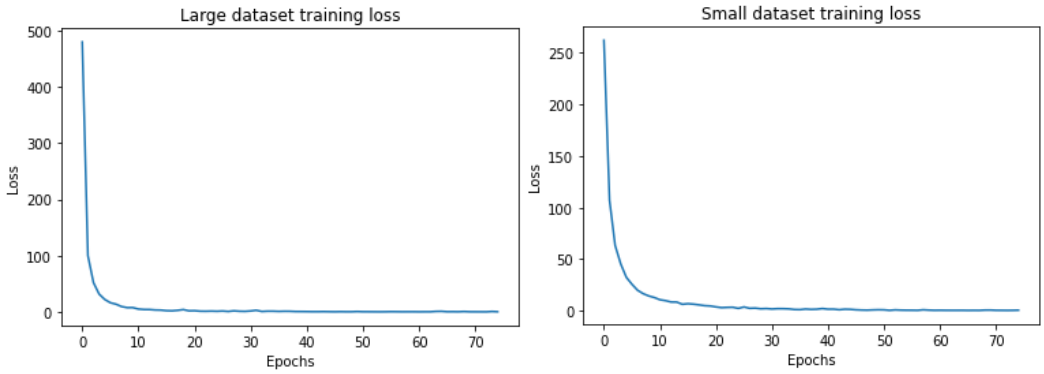

Figure 11: Models Trained with Normal Fruit360 Data

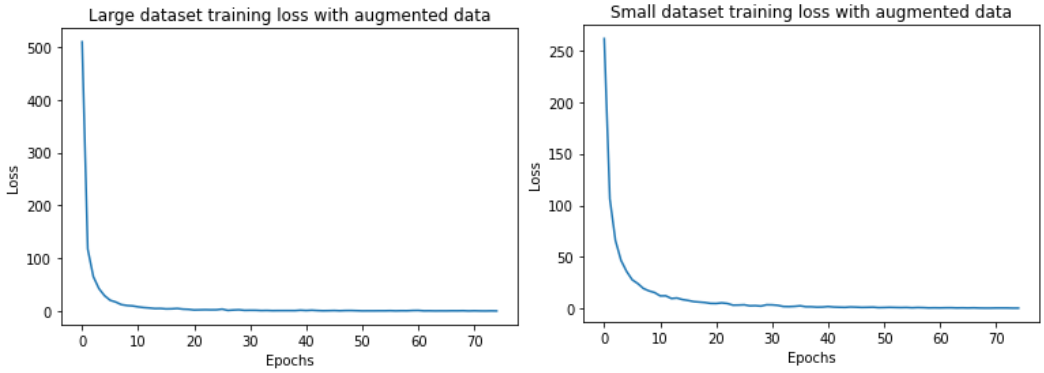

Figure 12: Models Trained with Noisy Fruit360 Data

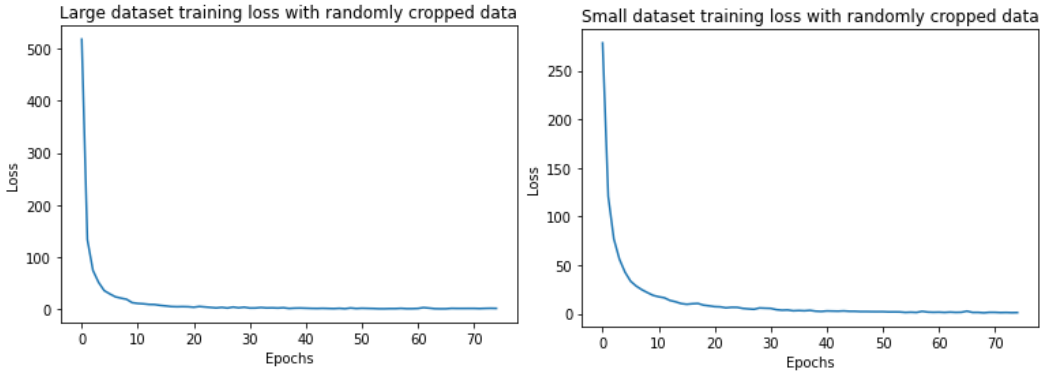

Figure 13: Models Trained with Cropped Fruit360 Data

Once we tuned our models to achieve desirable validation outcomes, we evaluated our models on both our Fruit360 test split and our Vegfru test subset. Figure 14 below demonstrates our results from evaluation:

Training Data	Fruit360 Test Split	Vegfru Test Subset
Large Unaugmented Data	99.982%	8.4626%
Small Unaugmented Data	99.908%	10.222%
Large Cropped Data	100.00%	9.5665%
Small Cropped Data	99.945%	8.1267%
Large Noisy Data	43.615%	7.8547%
Small Noisy Data	40.170%	7.7108%

Figure 14: Results from Evaluation on Models

5 Conclusion

Based on our results from the section above, we are able to conclude that our first augmentation of adding a noise layer to our training images was not helpful. In fact, it was even difficult to achieve high accuracy on our validation set with training using this method. The accuracy we achieved on our Vegfru test subset was substantially lower with this augmentation technique than the model trained with the original Fruit360 images. Contrary to this result, our findings from the second augmentation we performed were interesting. Here, we see that the accuracy of this augmented data model, for the larger training dataset, was larger than the accuracy of our non-augmented data model, for the larger training dataset, on the Vegfru test subset. However, for the smaller training datasets, the augmented data model accuracy was lower than the non-augmented data model. This implies that for large training datasets, better results are yielded when a model is trained with random cropped augmented data, as opposed to non-augmented data.

Another finding from our results is that the model trained on data manipulated from our second augmentation obtained a better test accuracy on the Fruits360 test data split than the model trained on the original Fruits360 training data. This shows that these image transformations that we conducted for our augmented data aid in preventing overfitting for models and generate better results than a model trained on non-manipulated data.

6 Work Contribution

Overall, we both believe that we contributed to this project equally, even though our tasks did contrast with each other at certain times. To summarize, Brett handled planning and strategizing for the experiment, preprocessing the data, training and evaluating the models (because he owns an NVIDIA GeForce RTX 2070 Super GPU), and writing parts of the report. Max worked on planning and strategizing for the experiment, initially organizing the data, preprocessing the data, doing extra literature research, and writing parts of the report.

References

[1] G. Blanchard, G. Lee, and C. Scott, "Generalizing from several related classification tasks to a new unlabeled sample," in NeurIPS, 2011.

[2] K. Muandet, D. Balduzzi, and B. Scholkopf, "Domain generalization via invariant feature representation," in ICML, 2013.

[3] Zhou, K., Liu, Z., Qiao, Y., Xiang, T. and Loy, C.C., 2021. Domain generalization in vision: A survey. arXiv preprint arXiv:2103.02503, 2021.

[4] R. Volpi and V. Murino, "Addressing model vulnerability to distributional shifts over image transformation sets," in ICCV, 2019.

[5] Y. Shi, X. Yu, K. Sohn, M. Chandraker, and A. K. Jain, "Towards universal representation learning for deep face recognition," in CVPR, 2020.

[6] S. Otalora, M. Atzori, V. Andrearczyk, A. Khan, and H. Muller, ¨ "Staining invariant features for improving generalization of deep convolutional neural networks in computational pathology," Frontiers in bioengineering and biotechnology, 2019.

[7] C. Chen, W. Bai, R. H. Davies, A. N. Bhuva, C. H. Manisty, J. B. Augusto, J. C. Moon, N. Aung, A. M. Lee, M. M. Sanghvi et al., “Improving the generalizability of convolutional neural network based segmentation on cmr images,” Frontiers in cardiovascular medicine, 2020.

[8] L. Zhang, X. Wang, D. Yang, T. Sanford, S. Harmon, B. Turkbey, B. J. Wood, H. Roth, A. Myronenko, D. Xu et al., “Generalizing deep learning for medical image segmentation to unseen domains via deep stacked transformation,” TMI, 2020.

[9] Z. Li, F. Liu, W. Yang, S. Peng and J. Zhou, "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects," in TNNLS, 2021.

[10] Fruits360 Dataset - https://www.kaggle.com/datasets/moltean/fruits?resource=download

[11] Vegfru Dataset - https://www.kaggle.com/datasets/zhaoyj688/vegfru

[12] P. Li, D. Li, W. Li, S. Gong, Y. Fu and T. M. Hospedales, "A Simple Feature Augmentation for Domain Generalization," 2021 in ICCV, 2021.

[13] K. Zhou, Y. Yang, Y. Qiao, and T. Xiang, “Domain generalization with mixstyle,” in ICLR, 2021.

[14] M. Mancini, Z. Akata, E. Ricci, and B. Caputo, “Towards recognizing unseen categories in unseen domains,” in ECCV, 2020.

[15] W. Zhang, “Shift-invariant pattern recognition neural network and its optical architecture,” in Proc. Annu. Conf. Jpn. Soc. Appl. Phys., 1988

[16] Y. LeCun et al., “Backpropagation applied to handwritten zip code recognition,” Neural Comput., vol. 1, no. 4, pp. 541–551, Dec. 1989.