CMPT 489 Final Report

Derek Kunkel
University of Saskatchewan
105 Administration Place, Saskatoon, Saskatchewan, Canada
ddk960@mail.usask.ca

1. Abstract

This paper presents a study on image classification using the Sorghum-100 dataset, comprising 100 cultivars of sorghum, a drought-resistant grain. The dataset's small inter-class variance and environmental factors affecting image quality make this a challenging multiple-class classification problem. The dataset, derived from the TERRA-REF field in Arizona, includes 22,193 training and 23,639 test 1024x1024 RGB images. A subset of 10 classes with the highest number of samples was derived to facilitate the time and hardware constraints of the experiment.

Two models were employed: a basic LeNet model and the more complex IBN-ResNet-a, used by the third-place team in the 2022 Kaggle competition this paper is based on. Experiments focused on the effects of training epochs, dataset size, and data augmentation on model performance. The baseline LeNet model showed limited success, highlighting the complexity of fine-grained visual classification in this context. An ablation experiment demonstrated that data augmentation might reduce accuracy in specific scenarios.

The study underscores the inadequacy of simpler CNNs like LeNet for complex image classification tasks, as seen in the stark performance contrast with advanced models such as IBN-ResNet. Finally, the paper explores the computational intensity of CLAHE, a tool used for enhancing image contrast, suggesting alternative methods like SWAHE or SUACE for future research. The research contributes to understanding the nuances of image classification in agricultural contexts and offers insights into the potential of deep learning in addressing such complex challenges.

2. Introduction

Sorghum is a species of grass, much the same as corn, and its high nutritional and sugar content allow for it to be used in a diverse array of applications. Approximately half of sorghum grown globally is utilized as food crop [1]. In the United States sorghum is used primarily as livestock feed, though roughly 30% is used to produce ethanol as biofuel. In India and much of Africa, sorghum's ability to resist drought and high temperatures makes it a staple grain in the local diet [2]. A cultivar is a plant of a given species whose phenotype is a result of cultivation to select for a desired set of characteristics [3].

This task is in the domain of images of plants, specifically images of crops. It is a multiple-class classification problem, which is transferrable to several domains where the classification of images is required such as identifying images of animals, vehicles, commercial products, handwritten text, or indeed other types of plants like trees or fungi. The Sorghum-100 dataset consists of 100 different cultivars of sorghum, which translates into 100 different possible classes. This particular classification problem is challenging because the variance between each class (cultivar) is very small, with each phenotype having visual characteristics very similar or near identical to one another [4]. This could be further complicated by environmental factors like weather conditions, which can alter an image's lighting and change or obscure relevant details. If an accurate deep learning solution can be developed to this problem (identifying which phenotype a given sorghum plant is from a provided image), it would allow us to determine if plant seed is incorrectly labeled, if it was non-uniformly seeded on a given plot of land, or if the planter machinery used for seeding developed an undetected jam. These errors are frequent in the crop planting process and, when taken in the context of an experiment utilizing complex, large-scale crop fields involving hundreds of cultivars, can have a significant deleterious effect on the experimental outcome.

The competition for this task is from 2022 and is hosted on Kaggle [5]. 252 teams competed, with the top score being set by DeepBlueAI at 0.965 and the following 3 runners-up in the 0.950 to 0.960 range. While at the time of submitting this paper there is a web page available for FGVC10 in CVPR 2023, there is no competition listed on it for the Sorghum-100 dataset. There is also no 2023 Kaggle competition available. The samples in this dataset are RGB photos of sorghum cultivars taken from a vertical perspective [5].

3. Methods

3.1.1. Dataset

As previously mentioned, the Sorghum-100 dataset is

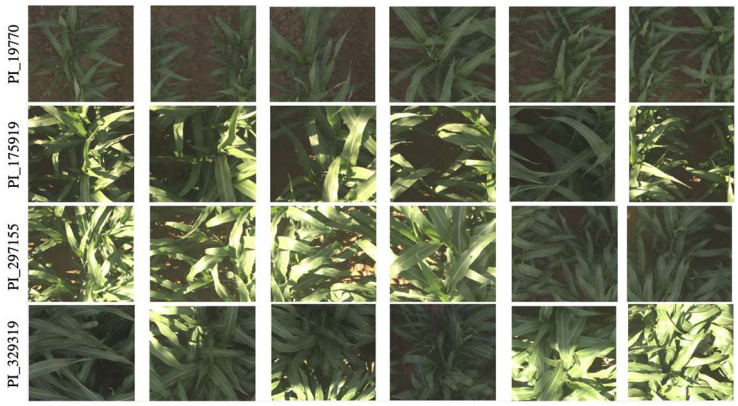

Figure 1: Four rows of example input images. Each row consists of six photos of different sorghum plants of the same cultivar (i.e., the same class), with each column representing photos taken from different days.

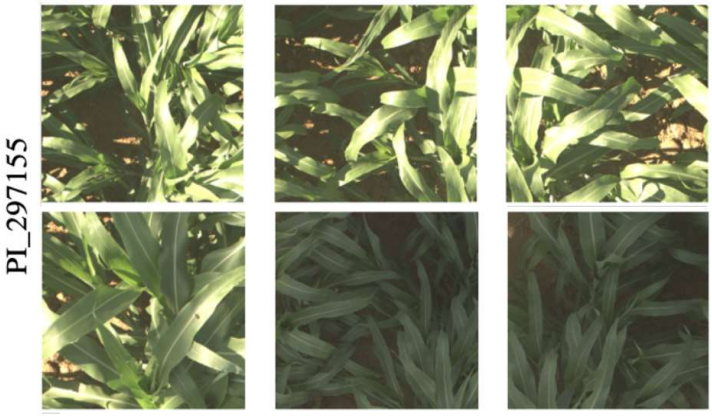

Figure 2: A subset of figure 1, showing sample input images of just one cultivar with a range of lighting conditions.

comprised of 100 different classes, where each class label is the ID of 1 of the 100 genetically distinct cultivars. The images were taken from the TERRA-REF (Terraphenotyping Reference Platform) field in Arizona in June of 2017, and used a sensor package mounted on an automated, mobile gantry to take the photos. TERRA-REF planted each sorghum cultivar in two separate plots to account for any localized extreme conditions that could affect crop growth for a given plot and to prevent models from overtraining on irrelevant details like soil patterns. They then utilized these two plots as a respective train dataset and blind test dataset. Both datasets are comprised of 1024x1024 24-bit RGB images in PNG format. The training dataset has 22,193 images, while the test dataset has 23,639 images. Intriguingly, the third-place solution utilized the same dataset converted into JPEG format at the same resolution and bit depth for their solution to reduce space requirements, stating in their provided solution that this would have negligible effects on accuracy. This reduced-size dataset has also been made publicly available. Unfortunately, only the labels for the training dataset have been made available, and not those for the test dataset.

Figure 1, taken from the Kaggle competition website, shows sample cultivars from 4 different species [5]. Each row consists of 6 images of different plants of the same cultivar (i.e., the same class), taken throughout the day. This figure demonstrates the extreme inter-class similarity between each of the cultivars, which is what make this classification problem so difficult. It also demonstrates the diverse range of lighting conditions due to differences in weather and time of day for each image, which further increases classification difficulty.

In figure 2 we see a clearer depiction of the spectrum of lighting conditions encountered within a single class from the dataset, as each photo was taken on a different day. The top row's lighter conditions allow for more distinct leaf shapes and features, while the bottom row's darker lighting conditions make the cultivar's features less distinct. This may significantly increase classification difficulty.

In figure 3, also taken from the Kaggle competition website, we see the example output required by the Kaggle competition, which is a CSV file with the image's name in a column on the left, and which one of the 100 classes the algorithm predicted for that image in a column on the right [5].

23639 unique values	1 unique value
1000005362.png	PI_152923
1000099707.png	PI_152923
1000135300.png	PI_152923
1000136796.png	PI_152923
1000292439.png	PI_152923
1000350798.png	PI_152923
100043618.png	PI_152923

Figure 3: An example of the algorithm's output, with the image name on the left and the cultivar's identity (label) predicted by the algorithm on the right.

3.1.2. Visualizing the Labels

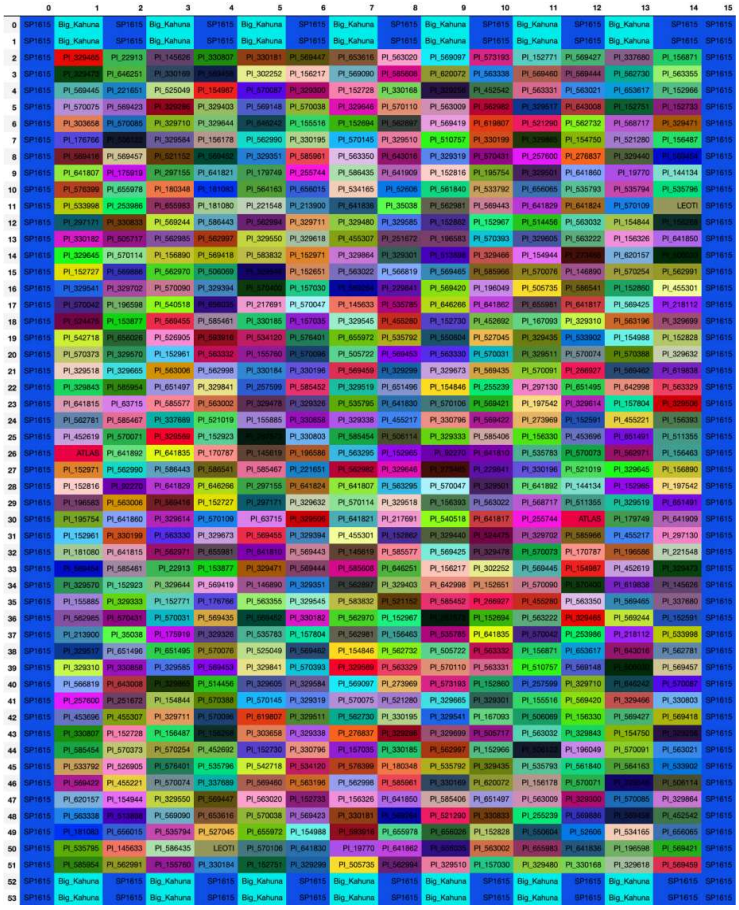

Figure 4: The geographical layout of sorghum cultivar plots, with cultivars of the same species being color coded together.

Figure 4 is taken from the original paper that introduces the Sorghum-100 dataset and demonstrates the geographical layout of cultivars in the TERRA-REF sorghum plots [4]. Since the Sorghum-100 dataset was photographed in the TERRA-REF fields using an automated gantry and sensor setup in this highly organized geographical grid, and because each cultivar was planted in its respective grid space as figure 4 shows, the likelihood of label noise is very low. However, the extreme inter-class similarity makes finding label noise from visual inspection of samples essentially impossible, so existing label noise due to human error during the planting process cannot be ruled out.

In figure 5 we see the distribution of labels (and thus, image samples) across all cultivar classes in the Sorghum-100 dataset. They range from 134 labels in the PI_257600 class to 298 labels in the PI_156393 class. While all classes have at least 130 sample labels/images, and the median number of labels is 235, the standard deviation amongst the number of labels per class is 37.8, indicating a clear discrepancy in available samples per class.

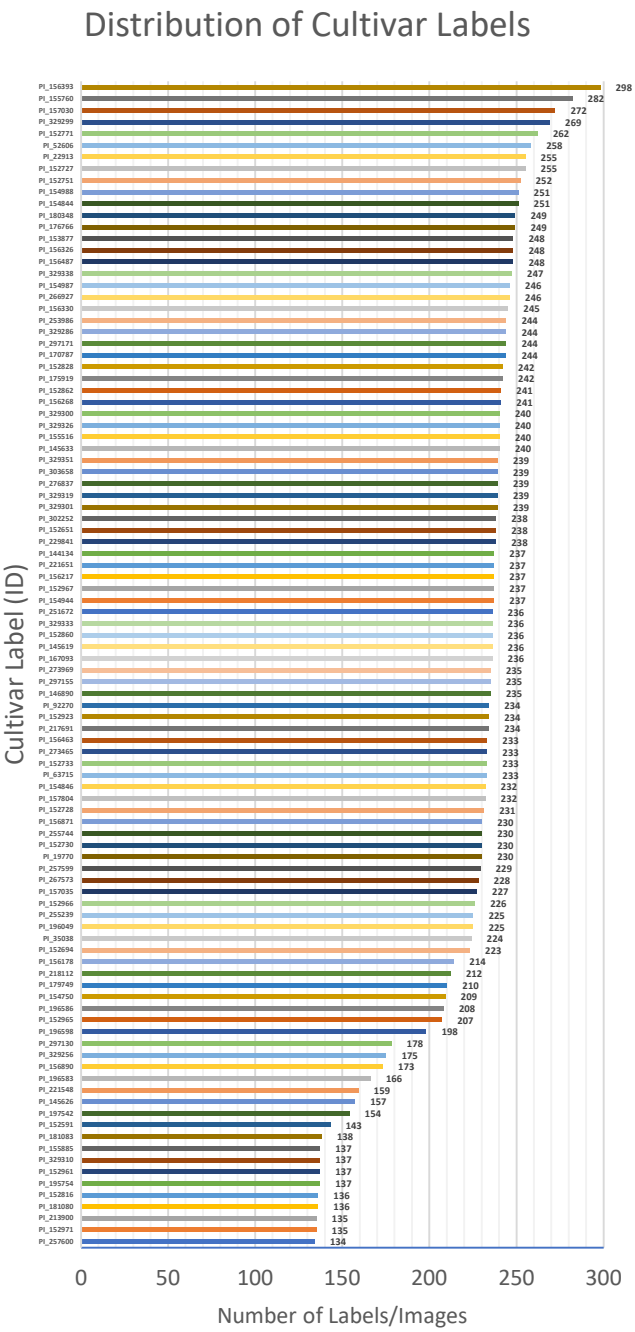

Figure 5: A bar graph showing the distribution of cultivar labels in the Sorghum-100 dataset.

Figure 6 shows a PCA plot performed on a sub-dataset of 1000 images from the training data, as I did not have the available computational power to perform it on the entire dataset. This required batching the images in sets of 100 and took roughly an hour to complete. The number of components for the PCA was also set to 100, as there are

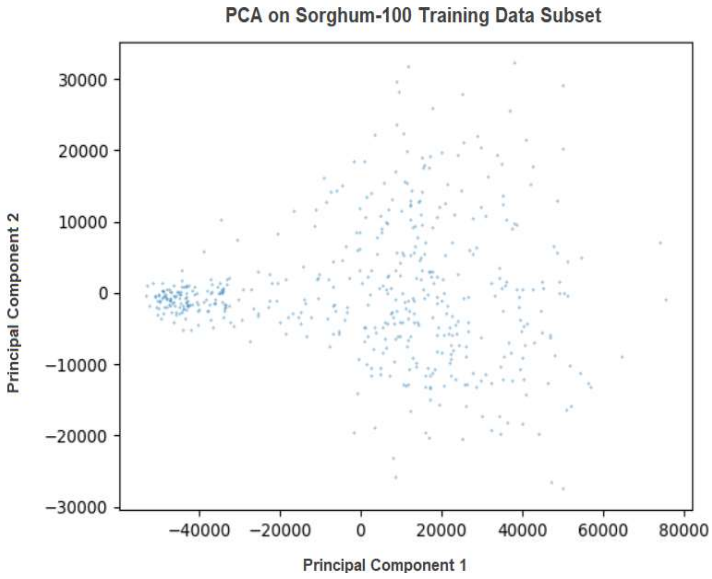

Figure 6: A plot of a principal component analysis conducted on a subset of 1000 images from the training set of the Sorghum-100 dataset, with the number of components set to 100.

100 classes in the dataset. However, given the small size of the subset I was reasonably capable of sampling, not all classes may have been represented in the sample. The plot demonstrates an astounding range along its axes and has observable perpendicularity.

Inspection of 50 randomly chosen images from the training dataset does reveal some image noise in images with very high sunlight conditions, which makes features of the sorghum leaves struck by the sun indiscernible due to "whiteout". This happens to varying degrees throughout the inspected subsample, with one third of the images being exposed to a problematic quantity of sunlight. This is, I suspect, an artifact of the gantry and mounted sensor package, which obscures the sun and casts a shadow on the other two thirds of many of the images and makes adjusting the sensor's contrast difficult. There is a similar problem with some of the images having extremely dark lighting conditions, which also makes leaf features difficult to discern. Furthermore, random inspection appears to show that lighting conditions are more similar in photos taken on the same day, which the dataset is organized by. Inspecting these same 50 images shows that the dataset has consistent and exceptional image quality, with no observable weather effects, lens artifacts, or other easily identifiable issues aside from the adverse lighting conditions. This is to be suspected given the automated nature of the gantry camera setup.

3.1.3. Selecting Sub-Dataset

As previously mentioned, only the labels for the training dataset have been made available, and not those for the test dataset. Posts in the competition's discussion forum have requested that the testing labels be released but have gone unanswered by the competition organizers. As such, a smaller subset of images can be extracted from the training subset, but not from the testing subset. Evaluation of any algorithm on the testing subset is currently impossible, as the competition is archived and has not been repeated for this year's FGVC 10, making submission and evaluation of the results of a testing run unavailable. Emails reaching out to the competition organizers have not received a response. As a result, I will need to use only the 22,193 images in the training set to create both a training and test dataset so that may verify my algorithm's results. This has two significant detriments, in that training and test datasets were grown on different plots of land to ensure that extreme localized conditions would not have an outsized adverse effect on the image data, and that geographically distinct plots of land were thought to prevent against the algorithm training on soil details. Thus, the advantage inherent to having these distinct subsets is lost.

As was also previously discussed, the very small genetic differences between all cultivars in the dataset makes the inter-class visual variability similarly small, to the point of individual cultivars being indistinguishable to expert observers. Observations of the PCA from figure 6 did not meaningfully inform as to what samples would best demonstrate the most prominent features. As previously discussed, figure 2 demonstrates the vast range of intra-class lighting conditions within the dataset. Thus, the sub-dataset should be representative of these lighting conditions, to ensure the algorithm performs robustly on images taken at different times of day and under different weather conditions. Using cultivars with a larger number of available labels/samples allows for a greater variability in the lighting conditions on which to train, allowing for development of a more robust model. In addition, it also increases the likelihood of images being taken from different days which, as has also been previously mentioned, also increases the likelihood of them having dissimilar lighting conditions.

The third-place team mentions in the discussion boards of their solution that they used a NVidia GeForce RTX 3090 as their computational unit, and that the full experimental setup used 45 epochs for training and took approximately 340 hours to converge. They also mention that, because of the cosine restart learning rate used in their algorithm, the training time can be shortened to 20 epochs and should obtain similar results [6]. Although I am fortunate to have a similar computational unit (an NVidia 3080) it is not feasible, given the constraints of the project timeline, to train and validate all iterations of my algorithm for this length of time. Assuming, very naively, that the relationships between number of epochs, the number of training samples, and the time-to-train hold in a linear fashion, a sub-sample of using only the training dataset subdivided into a train and validate set will take

approximately 86 hours to train. As such, I needed to select a smaller sub-dataset with which to do my project.

Taking into consideration the aforementioned factors, I created a smaller sub-dataset of 10 classes (cultivars) that have the median or higher number of samples available in the training dataset, ranging from 251 to 298 images each. Originally, I had decided to choose 100 images from each for a total of 2000 images, using 1000 for training and 1000 for validation, which I hoped would provide enough data to meaningfully train and validate my model while still reducing the number of samples to a more manageable portion. Discussions on the viability of my experimental setup with Professor Stavness elucidated that this 100/100 train/validation set had both an inadequate number of training samples and too many validation samples. This led to altering my dataset to use the entirety of samples available for each of the classes with the largest sample sizes and change the ratio of training/validation sample split to 0.8/0.2, respectively. This brought the total sub-dataset size to 2,654 images, and the minimum number of images in the training set to 200, and in the validation set to 50, which is what I used in my experiments. If the naïve calculation of train/validate time holds linearly, a full experimental setup would take approximately 15 hours to run using my GPU, given that my model has a similar number of layers and types of preprocessing as the 3rd place winner of the Kaggle contest. While initially I had made my sub-dataset by hand via separating the desired classes into folders, review of the Kaggle competition's 3rd place solution informed me of the sci-kit learn library, which automatically creates train/test splits of a provided dataset in whichever ratio is desired and is easily changeable as a function parameter. I implemented this library in my baseline model, which makes my solution more generalizable (in the coding sense, not the deep learning sense) to datasets that aren't sub-divided into folders by class.

The Kaggle contest's third place solution implements CLAHE (contrast-limited adaptive histogram equalization) on each image as part of preprocessing. Contrast in an image can be improved using AHE (adaptive histogram equalization), which reconstructs an image using histograms computed for each region of that image, and then redistributes the brightness in the image using those histograms [7]. This can alleviate the obstruction of plant features by images with either very high or very low brightness from local lighting effects. CLAHE is a variant of AHE, which reduces the amount of noise amplified by the AHE process by limiting contrast amplification by performing histograms on each pixel in an image relative to its neighboring pixels, rather than each section of an image. While CLAHE is more effective than AHE at delivering a better visualization of an image, it is also much more computationally expensive and thus takes much longer to perform [7]. This is apparent in my own running of the 3rd place solution, as well as in the forums of the Kaggle competition. The issue was so pronounced that one participant in the challenge re-made the dataset pre-processed with CLAHE, to reduce the time required to train and test models for the competition [8]. This CLAHE-processed dataset was made available to the community, and I re-made my 10-class dataset using these pre-processed images to also reduce the training time of my models. One final detail of note is that the CLAHE dataset used PNG images downscaled to 512x512 resolution, as opposed to the 1024x1024 PNG images of the original dataset.

3.2. Models

3.2.1 Baseline Model

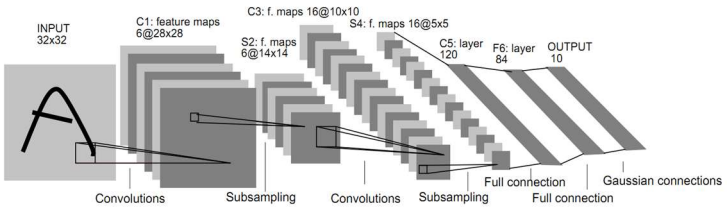

Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Figure 7: A picture of the LeNet CNN architecture. Taken from: https://www.kaggle.com/code/blurredmachine/lenet-architecture-a-complete-guide

The model I chose as my baseline model is LeNet, a CNN we covered in class and one that is classically used and referred to in literature since its creation in 1998 [9]. My implementation of LeNet uses 2 consecutive convolutional layers. The first layer has 6 filters, while the second has 16 filters. Both have a kernel size of 5x5, have ReLu activation functions, and feed into subsequent average pooling layers. Average pooling is used to decrease the spatial dimensions of the incoming feature maps, which is helpful in reducing overfit while also decreasing the number of calculations required, thus decreasing the time required per epoch.

These convolutional layers are followed by 3 fully-connected layers. The final fully connected layer has a variable number of outputs, as I have my code configured to easily switch between the 10 and 100 class datasets for easy experimentation. Classification from this layer is done with softmax activation.

3.2.2 Main Model

The model I chose as my main model is the model created by the 3rd place Kaggle competition winners, which uses IBN-ResNet as its base. IBN-ResNet is a variant of the popular ResNet architecture that integrates instance normalization (IN) and batch normalization (BN) to achieve higher performance [10]. There are several variants of IBN-ResNet, but the model they used for their competition submission is IBN-ResNet50-a. Inspection of their solution's code showed that the team had built implementations for various types of ResNet (18, 34, 50, 101, and 152) in both 'a' and 'b' configurations of IN/BN that could be readily switched for testing purposes.

IBN-Resnet-a's architecture can be observed in figure 8. The original IBN-ResNet paper states that the instance normalization in the model allows the learning of features in a way that's unaffected by changes such as whether the image is taken from real or virtual images, and changes to the feature's color. Batch normalization is used preserver content-related information by operating at the mini-batch level.

The code for my main model is taken directly from the 3rd place winners' solution to the Kaggle contest, which can be accessed at the following github repo: https://github.com/xinchenzju/CV-competition-arsenal/tree/main/CVPR2022-FGVC9-top3

3.3. Metrics

The Kaggle competition asked for accuracy as its metric, and thus both my baseline and main models used accuracy as their primary metric.

Accuracy in the case of image classification is an exceedingly simple calculation. It is simply the number of total correct predictions made by the model divided by the total number of predictions made by the model [11].

An alternate metric to accuracy for image classification is precision, which is how many of the predictions made by the model were correct [12].

3.4. Experiments

Experiments were performed using a NVidia GeForce RTX 3080 GPU, which has 10GB of GDDR6X memory. The machine housing the GPU has an AMD Ryzen 5 5600X CPU with a base clock speed of 3.7 GHz, and 16 GB of RAM.

3.4.1. Main Experiment

The main experiment I chose to conduct was a classic "two-by-two" design, training and validating my baseline and main models (LeNet and IBN-ResNet-a, respectively) on two datasets. These were the previously described full dataset of Sorghum-100 training data processed with CLAHE (22,193 images), and the smaller-subset of that dataset (2,654 images). The train/validation split in both datasets for both models will be set to 80% training to and 20% validation. All models were run for 25 epochs, which

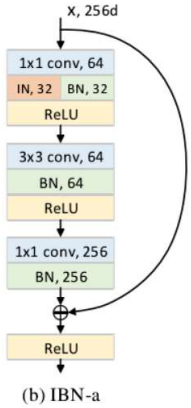

Figure 8: A picture of the IBN-Resnet-a CNN architecture.
Taken from the 3rd place team's Kaggle solution website:
https://www.kaggle.com/competitions/sorghum-id-fgvc-9/discussion/328593

was chosen for three reasons. First, because the 3rd place winners state that the cosine restart learning rate used in their model allows the training time to be shortened from 45 epochs to 20 epochs while obtaining similar results. Second, because the time required to perform the experiment was estimated to be a far-more-manageable 15 hours compared to time required to complete the original experimental setup, as discussed in the "Selecting Sub-Dataset" section. Finally, initial tests on my LeNet model led to results that were difficult to interpret at the 10-epoch range, and I thought that 25 epochs would both align with the previous two reasons while also allowing for more training to elucidate trends in the results.

The expectations of running the IBN-ResNet-a model on the 100-class full training dataset were that the validation accuracy will be the highest of the 4 experiments. This is because the cosine restart learning rate should allow similar results to what the model achieved in the competition with a smaller number of epochs, and because having 100 classes to train on will make the model more generalizable over a larger number of classes. Also, the IBN-ResNet model has a far larger number of layers than the relatively basic LeNet model, allowing the IBN-ResNet model to find more fine-grained and features.

The expectations of running the IBN-ResNet-a model on the 10-class smaller sub-dataset were that it will have the second-highest validation accuracy of the 4 experiments, since the larger number of layers will make it more accurate than the LeNet model, but less accurate than being able to train on a larger number of classes.

The expectations of running the LeNet model on the 100-class full training dataset were that the training accuracy will be higher than the LeNet model trained on the 10-class dataset accuracy, due to a having a larger

quantity of training data to extract features from, and lower than either of the IBN-ResNet model/dataset combinations due to LeNet having fewer layers to find fine-grained features.

The expectations of running the LeNet model on the 10-class smaller sub-dataset were that that it will the lowest training and validation accuracy of any of the models.

3.4.2. Hyper-parameter Experiment

The hyper-parameter I chose to vary during this experiment was the number of epochs used for training. This was for two reasons. First, because the statement by the 3rd place Kaggle competition winners claim that their model could achieve similar accuracy when trained for only 20 epochs as when trained for 45 made me curious as to the effect of the number of training epochs on models used for fine-grain visual categorization. Second, much the same as with the main experiment, because initial testing of my LeNet model showed extremely erratic and difficult to interpret results when training for a smaller number of epochs, and I hoped that training for a wider range would produce more easily discerned and perhaps more performant results.

This experiment was conducted on my LeNet model and not using the IBN-ResNet-a model, because the time to complete this experiment at larger epoch ranges (e.g., 100, or really anything more than 25) for the IBN-ResNet-a model was estimated to be unreasonably high, and unachievable in the time allotted before this report was due. Timeliness was also the reason I chose to run this experiment on the smaller 10 class sub-dataset. Availability of the machine used for training was also an issue, as it needed to be utilized for other projects.

I chose a range of epochs to try: 10, 25, 50, and 100. The highest value of 100 epochs was chosen to see if over-fit would be more clearly visible observable at a larger "timescale", and to see if the fit between training and validation would increase with the number of epochs.

The expectations of running the LeNet model while varying the number of epochs as the hyper-parameter is that the training accuracy will be higher as the number of epochs increases, at first due to having more training data to determine features from, and then continue increasing at higher numbers of epochs due to overfitting. The validation accuracy will eventually peak and be much lower than the training accuracy as the training is no longer generalizable.

3.4.3. Ablation Experiment

The ablation experiment I chose to perform was the removal of data augmentation from the baseline LeNet model. Thus, the ablation model used only the original images, without flips, rotations, or zooms, which were implemented to increase the algorithm's robustness and generalizability during training. This ablation experiment was performed using the smaller sub-dataset of 10 classes.

The reason I chose this was that, during my time reading through the Kaggle competition forums, I came across a post wherein a participant in the competition using a ResNet-34 model was achieving a lower competition score (model accuracy) when using data augmentation than when they were not [13]. This made me curious about the effects of data augmentation on a model trained on a dataset with very difficult to differentiate features between classes.

The expectations of running the LeNet model on the 10-class dataset for 25 epochs, but ablating the image augmentation, were that the ablation model will have lower validation accuracy than the non-ablation model, since the training will be less robust and thus less generalizable to other samples.

4. Results

4.1. Main Experiment Results

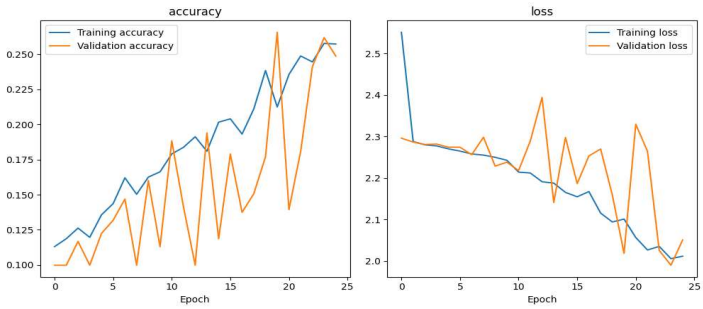

Figure 9: Two graphs showing my normal LeNet model's accuracy and loss for the training and validation datasets over 25 epochs.

Unfortunately, I was unable to complete most of the main experiment, only being able to collect data for running the baseline LeNet model on the smaller 10-class sub-dataset. I was unable to run the IBN-ResNet-a model created by the 3rd place Kaggle competition winners due to issues encountered getting both PyCharm and and PyLance (VSCode) to correctly identify the libraries for Pytorch. This problem is due to how Pytorch organizes its file structure and referencing for its API and is well documented. I also encountered the same problem with the Keras library from Tensorflow [14][15]. By the time a workaround had been found, there was not sufficient time to complete the experimental setup, due to the long run time from the model's depth/complexity. However, if the creators of the IBN-ResNet-a model are correct, it can be assumed that the validation accuracy using their model would achieve similar results at 25 epochs as it would at 50 epochs. Thus, the accuracy for both datasets can be inferred to be approximately 95.7%, as they have demonstrated in the Kaggle competition.

I was also unable to complete the run of my LeNet model on the larger Sorghum-100 training dataset. While early experiments on the smaller 10-class dataset were

performed without issue, later experimentation on the larger dataset demonstrated that my implementation encountered memory issues with larger datasets. I attempted to implement batching to alleviate the issue but was unable to create a working solution in time to perform the experiment on the larger dataset.

I was, however, able to complete the experimental setup for my LeNet model on the smaller 10-class dataset. Figure 10 shows the performance of my model over 25 epochs. Training accuracy peaked at 25.77% at epoch 24, while validation accuracy peaked at 26.55% at epoch 20. The fit of training to validation accuracy is erratic and at times very poor. Loss drops off sharply to 2.3 after 2 epochs, then slowly decreases to roughly 2.0 after 25 epochs.

4.2. Hyper-parameter Experiment Results

Number of Epochs	10	25	50	100
Training Accuracy	0.154	0.2572	0.1983	0.3363
Training Precision	0.75	0.6847	0.5714	0.6764
Validation Accuracy	0.1337	0.2486	0.194	0.2976
Validation Precision	0.6154	0.8125	0.56	0.6301

Figure 10: A table showing the accuracy and precision achieved by my LeNet model on the 10-class dataset for both the training and validation datasets over varying numbers of epochs.

Since I was unable to successfully run the main experiment using the IBN-ResNet-a model, or on the full training dataset of 100 classes, I chose to add the alternative metric of precision as an additional metric to my LeNet model to see how varying epoch number would affect both accuracy and precision. Figure 10 shows the results of my hyper-parameter experiment, where the number of epochs used to train my LeNet model on the 10-class dataset was varied between 10 epochs and 100 epochs. Training and validation accuracy peaked at 33.63% and 29.76%, respectively, after 100 epochs. Intriguingly, training and validation precision both peaked at 25 epochs, with 68.47% and 81.25%, respectively.

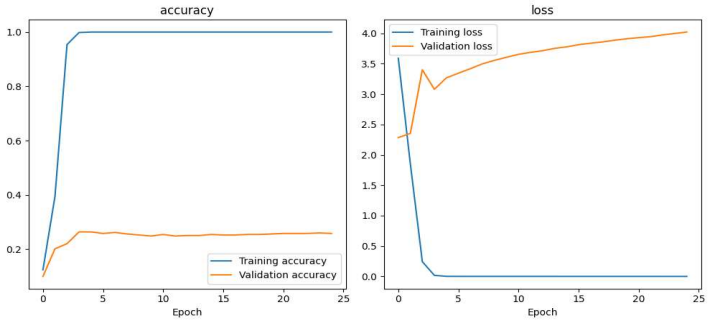

Figure 11: Two graphs showing my Ablation LeNet model's accuracy and loss for the training and validation datasets over 25 epochs. The model has ablation of its data augmentation.

4.3. Ablation Experiment Results

Figure 9 shows the result of training my normal LeNet model, while figure 11 shows the result of training my LeNet model with data augmentation ablation, for comparison. The ablation model shows a peak of 100% training accuracy after 5 epochs, and a peak at 26.37% validation accuracy after 4 epochs.

5. Discussion

5.1. Main Experiment Discussion

LeNet's terrible performance relative to the performance of the Kaggle competition's third-place winner is likely attributable to the fact that the LeNet model is far too simple, lacking sufficient layers to extract features in FGVC. Another possible explanation is that training only on the smaller 10-class dataset did not offer the model enough samples to find features, and that a later implementation of my model using batching to overcome the memory issue could see increased performance with training on the larger dataset. The erratic and poor fit of the training accuracy with the validation accuracy is interesting, and I am uncertain as to why it occurred. One guess would be that the image augmentation made the training accuracy steadily climb, while it also made the features learned less generalizable, which in turn caused (mostly) underperformance in the validation set. The spike in validation accuracy at epoch 18 I have no explanation for.

5.2. Hyper-parameter Experiment Discussion

The considerable increase in accuracy from 25 training epochs observed in the ablation experiment is likely due to LeNet's sensitivity to data augmentation.

5.3. Ablation Experiment Discussion

One possible reason for the ablation model having a slightly higher accuracy than the original model is that LeNet does not implement any features besides average pooling to decrease overfitting, such as dropout or other similar methods, and is thus prone to overfitting. Therefore, it is logical that a less challenging dataset (e.g., one that is not altered by augmentation each epoch) would allow the ablation model to achieve a higher accuracy via overfitting.

A second possible (though unlikely) reason is that the decrease in accuracy is not related to the architecture of the model including data augmentation (neither LeNet nor IBN-ResNet), but to the actual Sorghum-100 dataset itself. While overfitting of LeNet seems far more likely, two different models with very different architectures both seeing decreased performance when using data augmentation does offer an interesting question about whether certain tasks (e.g., fine-grained image classification) or datasets (e.g., pictures of plants with very

small differences between classes) inherently perform worse with data augmentation.

6. Conclusion

Clearly, CNNs such as LeNet with shallow depth or sufficient measures to reduce overfit are not suitable for application in fine-grained visual classification of the Sorghum-100 dataset. LeNet's performance is abysmal compared to that of more complex models such as IBN-ResNet, as is easily observable from the results of my experiment compared to the result of the top-performing models in the Kaggle competition like IBN-ResNet-a.

CLAHE, the tool used to make details in dark or overly bright images more visible, is also used in imaging in dental, blood vessel, and other medical imaging. Papers in these fields acknowledge the high computational cost of CLAHE and have suggested (and sometimes implemented) faster versions of adaptive histogram equalization to correct contrast in images in the form of SWAHE (Sliding Window Adaptive Histogram Equalization) and SUACE (Speeded-Up Adaptive Contrast Enhancement) [16], [17]. Future directions of exploration could include implementing SWAHE or SUACE to see if similar accuracy can be achieved with a smaller computational footprint.

7. References

[1] Sorghum | Agricultural Marketing Resource Center. https://www.agmrc.org/commodities-products/grains-oilseeds/sorghum (accessed Jan. 19, 2023).

[2] Sorghum – The Drought-Resistant Super Grain. https://www.globalgap.org/uk_en/media-events/news/articles/Sorghum-The-Drought-Resistant-Super-Grain/ (accessed Jan. 19, 2023).

[3] Allen v. Barker. Cultivated vegetables of the world: A multilingual onamasticon, *HortScience*, vol. 49, no. 9. 2014. doi: 10.3920/978-90-8686-720-2.

[4] Chao Ren *et al.* Multi-resolution outlier pooling for sorghum classification, 2021. doi: 10.1109/CVPRW53098.2021.00328.

[5] Sorghum-100 Cultivar Identification - FGVC 9 | Kaggle. https://www.kaggle.com/c/sorghum-id-fgvc-9/ (accessed Jan. 19, 2023).

[6] Sorghum-100 Cultivar Identification 3rd Place Solution - FGVC 9 | Kaggle. https://www.kaggle.com/competitions/sorghum-id-fgvc-9/discussion/328593 (accessed Feb. 04, 2023).

[7] C. Rubini* and N. Pavithra. Contrast Enhancementof MRI Images using AHE and CLAHE Techniques, *International Journal of Innovative Technology and Exploring Engineering*, vol. 9, no. 2, pages 2442–2445, Dec. 2019, doi: 10.35940/ijitee.B7017.129219.

[8] Sorghum FGVC 9 (equalized image 512x512) | Kaggle. https://www.kaggle.com/datasets/joonasyoon/sorghum-fgvc9-clahe-512 (accessed Apr. 20, 2023).

[9] Yann LeCun *et al.* Gradient-based learning applied to document recognition, *Proceedings of the IEEE*, vol. 86, no. 11, 1998, doi: 10.1109/5.726791.

[10] Xingang Pan *et al.* Two at Once: Enhancing Learning and Generalization Capacities via IBN-Net, in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018, vol. 11208 LNCS. doi: 10.1007/978-3-030-01225-0_29.

[11] Classification: Accuracy | Machine Learning | Google Developers. https://developers.google.com/machine-learning/crash-course/classification/accuracy (accessed Apr. 20, 2023).

[12] Classification: Precision and Recall | Machine Learning | Google Developers. https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall (accessed Apr. 20, 2023).

[13] Sorghum -100 Cultivar Identification - FGVC 9 | Kaggle. https://www.kaggle.com/competitions/sorghum-id-fgvc-9/discussion/323838 (accessed Apr. 20, 2023).

[14] adding additional support for completions related to lazy-loaded modules by bschnurr · Pull Request #54104 · tensorflow/tensorflow. https://github.com/tensorflow/tensorflow/pull/54104 (accessed Apr. 20, 2023).

[15] Unresolved reference: tensorflow.keras : PY-34174. https://youtrack.jetbrains.com/issue/PY-34174 (accessed Apr. 20, 2023).

[16] T Sund and A Møystad. Sliding window adaptive histogram equalization of intraoral radiographs: effect on image quality, *Dentomaxillofacial Radiology*, vol. 35, no. 3, pages 133–138, May 2006, doi: 10.1259/dmfr/21936923.

[17] A. M.R.R. Bandara *et al.* Super-efficient spatially adaptive contrast enhancement algorithm for superficial vein imaging, in *2017 IEEE International Conference on Industrial and Information Systems, ICIIS 2017 - Proceedings*, 2018, vol. 2018-January. doi: 10.1109/ICIINFS.2017.8300427.

8. Code Reference Disclosures

There were three guides/walkthroughs and existing git repos I heavily referenced and either adopted or used code directly from when developing my LeNet baseline model, the first of which being my primary reference. The links to these are listed below:

- https://towardsdatascience.com/convolutional-neural-network-champions-part-1-lenet-5-7a8d6eb98df6
- https://www.analyticsvidhya.com/blog/2021/03/the-architecture-of-lenet-5
- https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/

Though results from the main IBN-ResNet model from the 3rd place Kaggle competition winners was not used, their code for the model can be accessed at the following github link:

- https://github.com/xinchenzju/CV-competition-arsenal/tree/main/CVPR2022-FGVC9-top3

I also heavily referenced the Numpy, Pandas, Matplotlib, and TensorFlow/Keras documentation, links to which are listed below:

- Numpy:
    - https://numpy.org/doc/1.24/
- Pandas:
    - https://pandas.pydata.org/docs/
- Matplotlib:
    - https://matplotlib.org/stable/index.html
- Keras:
    - https://www.tensorflow.org/api_docs/python/tf/keras/