Few-Shot Learning Networks: Optimization Techniques and Trends

by

Jeremy Vidaurri

AN HONORS THESIS

Submitted to the
Honors College
at Texas Tech University in
partial fulfillment of the
requirement for
the degree designation of

HIGHEST HONORS

May 2023

Signatures:
Approved by:

[signed] 4/19/23

Dr. Victor Sheng
Edward E Whitacre Jr. College of Engineering, TTU
Thesis Director

Date

[signed] 4/202/23

Dr. Tara Salman
Edward E Whitacre Jr. College of Engineering, TTU
Second Reader

Date

[signed] 4/21/2023

Prof. Kurt Caswell
Instructor of HONS 3300 & 4300, Honors College, TTU

Date

[signed] 5.3.23

Dr. Jill Hernandez
Dean, Honors College, TTU

Date

Contents

***Abstract*—Most modern machine learning systems require to be trained over a large set of data. This is useful when there is readily large amounts of data. In recent years, few-shot learning has been proposed to circumvent this issue. Instead, the machine is given limited amounts of data and can make accurate predictions. Although the initial data may be limited, data can be artificially developed through data augmentation. Unfortunately, few-shot learning systems are not at the state where they can be utilized reliably. There is possibilities for these systems to be optimized through implementing dropout and hyperparameter tuning. This study is designed to analyze any trends and techniques that may allow for higher performance generally.**

***Index Terms*—machine learning, optimization, neural networks, artificial intelligence, learning paradigms, supervised learning, hyperparameter tuning**

I. Introduction

While machine learning systems have seen great performance through training over large amounts of labeled and processed data, there has been sought a solution to reducing the amount of data needed to be collected, labeled, and then processed. Few-shot learning systems have been the solution to tackle this issue. The objective of few-shot learning systems is to learn new concepts and make accurate predictions from minimal labeled examples. In general, few-shot learning can be split into three machine learning systems: N-shot learning, one-shot learning, and zero-shot learning. Each system decreases in the amount of data utilized. To compensate for the lack of data, a method known as data augmentation is utilized. Data augmentation artificially creates additional data by manipulating and transforming existing data. As for zero-shot learning, there are approaches that will either be classifier-oriented or instance-oriented. The goal of this study is to collect and analyze data from varying datasets and networks. Afterwards, the networks are optimized utilizing the bayesian optimization algorithm.

II. Literature Review

A. Network Design

One of the most popular neural networks utilized in machine learning is the residual networks designed by Microsoft researchers, He et al., *Residual Energy Services Network* (ResNet) [1].It was introduced after the *convolutional neural network* (CNN). To overcome problems that occurred in the CNN design, the residual network includes additional layers and Skip Connections. Also, compared to other networks such as VGG, ResNet utilizes exotic architecture which relies on micro-architectures (also known as network-in-network architecture).

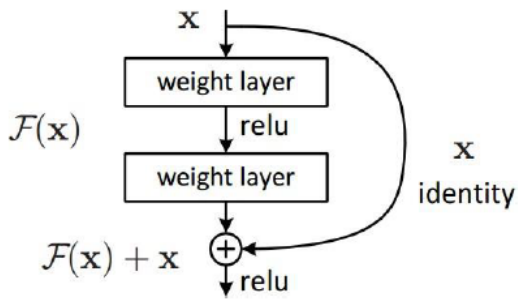

Fig. 1. Residual Block Architecture

ResNet is formed of residual blocks as seen in Fig. 1. These blocks are meant to resolve the issue with degradation. In short, the issue is a side effect of creating deeper networks. With deeper networks, it was expected to yield higher performance but instead it had significantly lower performance. Residual blocks solve this issue through skip connections. Skip connections allow systems to utilize both the previous layer's output and the current layer's output through matrix addition. From here, the researchers also propose an optimized version of residual blocks through the extension of a bottleneck. This reduces the number of parameters and matrix multiplication operations. This allows us to make residual blocks as thin as possible, which in turn provides scalability.

ResNet also has varying layer amounts with different architectures for deeper networks. For reference, the typical naming scheme is "ResNet" followed by the number of layers. In each of the models, the first two layers are a 7x7 kernel convolution and a max pooling layer with stride 2. They then differ in the amount of iterations per layer. For ResNet50, the amount of iterations is [3, 4, 6,3]. In ResNet101, the amount of iterations is [3, 4, 23,3]. For ResNet152, the amount of iterations is [3, 8, 36,3]. Exact details for each layer type can be seen in Fig. 2

Another aspect of network design that may beneficial in a few-shot learning system is implementing dropout as proposed by Srivastava et al. [2] Dropout is the concept of randomly dropping nodes and connections to prevent the nodes from co-adapting too much from backpropagation learning. This prevents networks from overfitting datasets. They experimented with multiple datasets that span multiple domains. In this wide variety of domains, it was found to be efficient for each of them. The issue with implementing dropout is the additional time it takes to train each model.

For further considerations, Sung et al. propose a new type of network known as Relational Network which can be utilized for efficient few-shot learning [3]. They use different models for the zero-shot learning system which utilizes ResNet as a supplement to the main model. They trained these two models on two datasets which resulted in about 1% increase in accuracy to competing networks such as Long Short-Term Memory networks and prototypical networks.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Fig. 2. ResNet Architecture

B. N-Shot Learning

Sun et al. propose a method for deep neural networks such as ResNet to be able to perform well in N-shot learning systems. They discuss previous attempts at utilizing meta-learning and the challenges faced such as tasks needing to be similar, and tasks being modeled by shallow neural networks to avoid overfitting. Instead, they take a combination of meta-learning and transfer learning to reap the benefits of both methods. This allows a system to learn based on previously trained weights. They also utilize methods that enable a system to learn quicker and stronger through hardships. Through experimentation with two datasets, they found that this method of learning was highly effective and could be generalized for other datasets [4].

C. One-Shot Learning

One-shot learning utilizes both data augmentation and a Bayesian approach to classify data from a single image. Fei-Fei et al. prove that a Bayesian approach in which the system utilizes previous learned categories to create future classifications is possible. The framework developed by these researchers was able to produce 70-95% accuracy on the specific datasets tested. This specific implementation may work well for increasing performance on other datasets, but it had not yet been proven at the time [5].

D. Zero-Shot Learning

Wang et al. provide a comprehensive foundation for zero-shot learning and its nuances. They also provide a categorization of semantic spaces in previous works based on whether this information is developed or learned. In terms of methods, they demonstrate how previous approaches either take a classifier-oriented approach or an instance-oriented approach. A classifier-oriented approach focuses on how to directly learn a classifier from unseen classes. Instance-oriented approaches on the other hand, focus on obtaining labeled instances that belong to unseen classes so that they may be utilized in classifier learning. Finally, they provide in-depth advantages and disadvantages for each of the methods. The method utilized depends on the system's requirements [6].

Pourpanah et al. then provide a more generalized approach to zero-shot learning methods known as *Generalized Zero-Shot Learning* (GZSL) [7]. They go over the issues with utilizing GZSL and possible improvements. They also compare the different GZSL methods and provide benchmarks to compare each method.

Liu et al. propose a solution to Indirect Attribute Prediction so that similar techniques can be used for GZSL. This is done through a framework they call label-activating framework. They utilize two mappings for each label activation. It utilizes both visual to label activating and semantics to label activating. These two spaces are required to reconstruct labels to prevent domain shifting. This method outperforms competing frameworks [8].

Researchers from OpenAI devised methods to train zero-shot learning systems on a variety of datasets. They utilize natural language supervision to obtain these results. They also provide a benchmark for training existing networks such as ResNet. The methods are done during preprocessing as to supply the system with information before it encounters the images during predictions. Through their experimentation, they can create a more effective (compared to competing approaches) method for zero-shot learning. These competing approaches are unsupervised, self-supervised, weakly supervised, and supervised learning. There is still plenty of room for improvement especially in the ResNet models [9].

E. Data Augmentation

Data augmentation is the primary method utilized to develop artificial data. One focus with data augmentation is preventing overfitting. Also, with any form of data augmentation, there needs to be careful attention to preserving the label. In other words, augmentations should not change which category an image belongs. For example, in MNIST, if we rotate the six or nine digit, it effectively swaps the label with one another. In a survey conducted by researchers from Florida Atlantic University, they explain the different data augmentation methods and their effectiveness with not overfitting data [10]. There are a variety of methods shown such as geometric transformations, color space transformations, kernel filters,

mixing images, random erasing, feature space augmentation, adversarial training, GAN-based augmentation, neural style transfer, and meta-learning schemes. The paper utilizes the ResNet network to show the differences between each of the methods alongside a few other novel data augmentations.

There are four geometric transformations provided by Shorten and Khoshgoftaar: flipping, rotating, cropping, and translation. Flipping is most commonly done on the horizontal axis as flipping on the vertical axis may cause loss of the label. It also is known to not preserve labels on the MNIST dataset which includes a collection of handwritten digits.

Rotating is done by rotating the image in either direction at an axis between 1 and 359 degrees. This variance of degree can also directly affect the safety of preserving the label. For example, minor rotations may be helpful for MNIST, but rotations greater than 20 degrees may cause the label to be lost.

Cropping is done by either only preserving a center patch of the image or through random cropping. Either method may lead to label loss. In random cropping, a random subset of the image is created. This is typically done in systems where the object should be detectable with minimal visibility.

Translation is the operation of shifting images left, right, up, or down. This can be utilized to restrict positional bias in the data. The remaining space in the image can either be filled with a constant (typically 0 or 255) or with Gaussian noise. Gaussian noise is created through injecting a matrix of random values drawn from a Gaussian distribution. Translations allows us to preserve the image dimensions post transformation.

Shorten and Khoshgoftaar also compare geometric transformations to color space transformations. Color jittering is where an image has its brightness, contrast, and saturation randomly changed. This method, however, has been shown to reduce performance by roughly 3%. This is more drastic in datasets where the color of the subject matters. Color jittering can also lead to label loss.

In all, Shorten and Khoshgoftaar compared the results of each data augmentation to one another alongside the baseline that is utilized as a benchmark. The highest performing data augmentations were cropping at a $79.1\% \pm 0.80$ top-5 accuracy. Every other augmentation performed with under 70% top-5 accuracy. This is similar results to others who have studied data augmentation such as Tabik et al. [11].

Kernel filters are utilized as an image processing technique to either blur or sharpen an image. In both scenarios, a kernel filter is applied through sliding a square matrix across the image. To blur the image, a Gaussian blur filter is utilized. Blurring has the issue where is may create a bias against motion blur. On the other hand, to sharpen the image, a high contrast vertical or horizontal edge filter is utilized. Sharpening an image can also cause the image to lose necessary details to label the data properly.

Another technique of data augmentation is to take multiple images and averaging their pixel values. The resultant image may be appear to be rather useless to an average human, but there have been studies indicating the effectiveness of this technique by Inoue [12]. In Inoue's work, he combines multiple data augmentations to prepare the images before mixing the two images. Both images are first cropped randomly down to a 224x224 image and then randomly flipped horizontally. An interesting detail from his experimentation is that better images were produced when the two separate images were obtained from two separate classes. This allows one to form a dataset of size $n^2 + n$ from a dataset of size n. The downside to mixing images is that the new images may be subject to overfitting if there are a high number of classes.

Random erasing is a technique developed by Zhong et al. It works by randomly selecting a patch of an image and masking it with either 0 or 255 mean pixel values or random values. Shorten and Khoshgoftaar found that utilizing random values was more effective, reducing the error rate on CIFAR-10 from 5.17 to 4.31%. The method of random erasing is meant to allow the system to focus on minor details rather than similarities of how the image was taken. For example, if a system was trained on a dataset with only center, front-facing portraits, it would not be able to detect faces at other angles. By removing a random patch of the image, the computer is forced to learn other descriptive characteristics. The main disadvantage for random erasing is that it may fail to preserve labels post transformation. This can be resolved through manual intervention.

Instead of augmenting the images within the input space, one can instead augment the feature space. The feature space is the lower-dimensional representations found in high-level layers of a convolutional neural network.

Adversarial training is a solution for searching for possible augmentations. This is done through using two or more networks with contrasting objectives encoded in their loss functions. The utilization of adversarial training can benefit a network by learning and straying away from augmentations that may result in misclassification. It also can strengthen classes that may be more limited in data. Shorten does however mention that this area of machine learning is still fairly unexplored and requires further testing to signify its effectiveness over any general dataset.

F. Hyperparameter Tuning

Hyperparameters are the configurations for a machine learning model that can be used to optimized via functions such as manual search, grid search, Bayesian search, and randomized search [13], [14]. Hyperparameters are divided into three categories: model hyperparameters, optimizer hyperparameters, and data hyperparameters. Model hyperparameters include filter size, pooling, stride, and padding. These are preset in the ResNet models as it is a predefined model. Optimizer hyperparameters describe how the model learn the patterns based on the data. This includes optimizers such as Adam and RMSprop. Data hyperparameters is related to the attributes of the dataset. It is typically utilized for limited datasets through the aforementioned data augmentation methods.

With each ResNet model, the deeper the network, the more hyperparameters there are to tune. On the lower end there

is ResNet50 with over 20 million hyperparameters. On the opposite end of the spectrum, ResNet152 has nearly 60 million hyperparameters [1]. Utilizing the searches allow developers to optimize a network efficiently.

Manual search involves setting hyperparameters to extreme values and then tuning according to the performance of the system. Manual search is useful in shallow networks where the amount of hyperparamters is minimal. Unfortunately, it is not a viable approach for tuning deep neural networks such as ResNet due to the high amount of hyperparameters. Instead, an algorithmic approach can be taken.

Grid search is another naive approach to hyperparameter tuning. In this approach, a grid is defined on n dimensions. Each of these map to a hyperparameter such as learning rate or dropout rate. From there, the range of possible values is defined in an array. Then, all possible combinations of each hyperparameter is tested and the best configuration can be established. Similar to manual search, this approach works well on shallow networks, but is inefficient for deep networks. Grid search becomes impracticable as when dimensions are added, the time complexity grows exponentially. This is due to the fact the grid search does not store the history of computations.

Bayesian search is similar to grid search in that it utilizes a hyperparameter grid to test configurations provided. The main difference is that Bayesian search optimizes its parameter selection in each round according to the previous round's score. By doing so, the system can reach an optimal solution much quicker than grid search. Also, in contrast to grid search, Bayesian search does not require exact values to be provided but rather a range of values. This should be much more efficient on deep networks compared to grid search.

Randomized search is similar to grid search, but does only part of the hyperparameter values are tested. The values are instead pulled randomly from a specified distribution. The amount of values tested is determined by the parameter n_iter. Randomized search allows for extending distributions without increasing the time complexity of the search. The main downside to randomized search is that it requires more trial and error to reach an optimal solution compared to the previous two methods.

G. Review Conclusion

In all, there is probable cause to believe that utilizing Bayesian Search to tune the hyperparameters of ResNet with dropout may provide an increase in accuracy. Randomized search also has the possibility to create an optimized network, but may have longer computational times. It is unclear whether the same tuning can be utilized for all subsets of few-shot learning.

III. Methodology

A. Metrics

To properly assess the efficiency of each configuration, we calculated the accuracy, precision, recall, and F1-Score of each system.

The first metric that was recorded is accuracy. Accuracy is simple, but can be ineffective for determining overall usefulness of a system. Accuracy can be represented in the equation below.

$$Accuracy = \frac{CorrectPredictions}{TotalPredictions} \tag{1}$$

To understand why accuracy is not sufficient for determining a system's effectiveness, think of a machine learning algorithm that predicts hurricanes. If the system simply always predicts that a hurricane will not happen, it will have a high accuracy as majority of the year there are no hurricanes. The issue arises when the system's false negatives are used. It simply does not serve the purpose that is required of the system.

The next metric that was measured is precision. Precision describes the ratio of true positive predictions to total positive predictions. This metric alongside accuracy allows us to more accurately determine a system's efficiency. Precision is calculated using the following formula.

$$Precision = \frac{TruePositive}{TotalPositivePredicted} \tag{2}$$

In contrast to accuracy, precision allows us to determine when the cost of false positives is high. High precision also means that the system is providing useful information.

The third metric that was observed is recall. Recall calculates how many of the actual positives the model captures as labeling as positive (true positive). Recall can be determined using the following equation.

$$Recall = \frac{TruePositive}{TotalActualPositive} \tag{3}$$

In contrast to precision, recall allows us to determine when there is a high cost associated with false negatives. This allows us to observe issues with overfitting or underfitting classifications.

The final metric that has been recorded is known as F1 score. F1 scores are utilized to determine a balance between precision and recall. In contrast to accuracy, F1 score is affected by both false positives and false positives. These two scores tend to cost businesses and systems much more than true negatives. F1 score is determined through the equation below.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4}$$

By utilizing each of these metrics, we can determine efficient machine learning systems for specific tests.

B. Test Methods

In order to consistently assess each system, the data, as seen in Table I [15]–[22], is pre-processed into a normalized structure. Each tensor of data is split into image and label components. The image is casted into a float32 tensor type and set to a 244, 244, 3 shape. It is then normalized into the ResNet structure through the Keras function `preprocess_input()`. As for the label, it is one hot encoded as to match the shape required by ResNet's output

TABLE I
DATASETS

Dataset	Description	Total Images
MNIST	Black and white images that contain handwritten digits.	60,000 for training. 10 classes.
Fashion MNIST	Black and white images that contain various clothing articles.	60,000 for training. 10 classes.
CIFAR 10	Images containing various objects.	50,000 for training. 10 classes.
CIFAR 100	A more refined classification of CIFAR 10.	50,000 for training. 100 classes.
Chessman	Images containing the 6 types of chess pieces.	556 images total. 6 classes.
Malaria	Images containing blood cells that may or may not have been parasitized.	27,558 images total. 2 classes.
Oxford Pets	Images containing 37 types of pets.	3,669 for training. 37 classes.
SVHN Cropped	Images containing digits from street view house numbers.	73,257 for training. 10 classes.
Oxford Flowers 102	Images containing 102 types of flowers found in the United Kingdom.	6,149 for training. 102 classes.
Plant Village	Images containing both healthy and unhealthy plants categorized by species and disease.	54,303 for training. 38 classes.

layer. After the entire dataset can be properly processed, the original training data is split to form new training, testing, and validation data. 50% of the training dataset is set aside for validation. The other 50% is split with five ratios to analyze how the difference in dataset sizes affects overall performance. The training dataset set to the following ratios for each experiment: 2%, 4%, 6%, 8%, and 10%. The testing dataset is populated with the remaining data.

The ResNet model also needs to be adjusted to fit the specific dataset by adding an additional dense layer with a unit count corresponding to the number of classes. The model is compiled and evaluated on the metrics previously mentioned. To ensure that the results are accurate for the specific system, each experiment was ran ten times. For each set of trials, the average and standard deviation is calculated for each metric. Each method is also tested on three respective networks: ResNet50, ResNet101, and ResNet152. With sufficient testing, trends such as plateauing can be observed.

For the hyperparameter tuning, the library bayesian-optimization was utilized. The networks were trained on F1-Score with varying batch sizes (16-256), learning rates (0.001-0.1), and regularization strengths (0.001-0.1). The optimizer was iterated through a total of 12 times (10 steps of bayesian optimization with 2 random exploration points) for each trial to allow the the optimizer to achieve more precise results. The two random exploration points allow for a more diverse exploration space.

IV. RESULTS

From the performance of the few-shot learning networks, a few things have an effect on the overall performance: the number of images utilized during testing, the number of possible outputs (classes) and whether the images are colored or in black and white. As a note, the Malaria dataset is a binary classification problem which causes the metrics to be equal.

A. Training Image Count

There is not an end-all point where diminishing returns occur. The amount of images utilized during training has different effects depending on the complexity of the dataset. For example, simpler datasets, such as MNIST or Fashion, have an overall steady increase between 1200 and 6000 images utilized. Comparatively, more complex datasets such as the CIFAR 10 dataset, the increase in performance is large at first but begins to plateau around 3000-4000 images. In other cases, where there are few images, such as the Chessman dataset, there is no sign of plateauing. The performance of the network on Chessman was consistently low. As for variance, there is a point roughly between 2000 and 3000 images where the variance is relatively low (less than 1%).

B. Number of Classes

The datasets that contained 10 or less classes performed marginally better than those with up to 50 classes. For example, the Street View House Numbers (SVHN) dataset performed only slightly better than the Oxford Pets dataset. At roughly the 50 classes mark, there appears to be a significant drop in performance. This can be seen by comparing the data from the Oxford Pets with the CIFAR 100 dataset. When training any of the networks on the Oxford Pets dataset, it performs twice as well as the CIFAR 100 dataset.

C. Color

Another factor to consider when training a few-shot learning network is the number of colors in the input images. The system more effectively learns features of an image when there is less information required to classify it. Observing the data gathered from the MNIST and CIFAR 10 datasets, there is a clear performance boost in MNIST which is purely black and white images (in contrast to CIFAR 10 containing images in color).

D. Bayesian Tuning

As previously stated, Bayesian optimization was performed with the following parameters: batch size (16-256), learning rate (0.001-0.1), and regularization strength (0.001-0.1). The optimizer utilized the F1-score as it's target metric. The best settings for each configuration can be seen in tables V,VI, and VII. By separating the focus based on the previously mentioned factors, we can observe trends such as plateauing.

For the first parameter, batch size, we can see that most of the datasets excelled wit batch sizes around the 80-120 mark. Those that utilized smaller batch sizes were accompanied by

large number of outputs and small number of images for training. In particular, we can further observe how the number of classes affects the performance by the results obtained with the Oxford Flowers 102 dataset compared to the Oxford plants dataset. Despite having roughly the same number of images for the trials 10% of Oxford Pets and 6% of Oxford Flowers. Both of these datasets have roughly 367 images (+/- 1). Despite using roughly the same number of images, the optimal batch sizes varied. For those with few classes, the batch size is regular. For larger amounts, the optimal batch size is roughly between 15 and 70.

For the other parameters, learning rate and regularization strength, there are no clear trends with how learning rate affects each dataset. This may be due to a lack of iterations during bayesian optimization. With further testing, there is possibilities for more visible trends.

V. Discussion

The results from this research could provide a pathway to further optimization of neural networks without the need for extensive data collection. If there is a scalable method to optimize these networks, we could test among other datasets that may lack data. The data gathered lines up with other researchers experiments as mentioned in the literature review. Less complex datasets are able to easily reach 60-80% on each performance metric while the more complex datasets can barely get above 30%. Another factor that was discovered to affect performance was batch size which allowed datasets such as CIFAR10 to grow up to 10%.

VI. Conclusion

There are many factors that can affect the overall success of neural networks with limited datasets. Data augmentation can be utilized to create artificial data. Otherwise, we can force the network to focus on fewer inputs or outputs in order to predict with higher performance. Also, by providing 80-120 images per batch, it appears to provide an additional increase by roughly 10% for less complex datasets such as MNIST. On the other hand, complex datasets, such as Oxford Pets, are able to observe increases by nearly 50% There are other factors that may also alter the overall performance such as learning rate and regularization strength, but the results are not providing a statistical trend.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, pp. 770–778.

[2] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[3] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. S. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," 2017.

[4] Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele, "Meta-transfer learning for few-shot learning," 2018.

[5] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

[6] W. Wang, V. Zheng, H. Yu, and C. Miao, "A survey of zero-shot learning: Settings, methods, and applications," vol. 10, no. 2. ACM, 2019, pp. 1–37.

[7] F. Pourpanah, M. Abdar, Y. Luo, X. Zhou, R. Wang, C. P. Lim, X.-Z. Wang, and Q. M. J. Wu, "A review of generalized zero-shot learning methods," *IEEE transactions on pattern analysis and machine intelligence*, vol. PP, pp. 1–20, 2022.

[8] Y. Liu, X. Gao, Q. Gao, J. Han, and L. Shao, "Label-activating framework for zero-shot learning," *Neural networks*, vol. 121, pp. 1–9, 2020.

[9] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," 2021.

[10] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of big data*, vol. 6, no. 1, pp. 1–48, 2019.

[11] S. Tabik, D. Peralta, A. Herrera-Poyatos, and F. Herrera, "A snapshot of image pre-processing for convolutional neural networks: case study of mnist," vol. 10, no. 1. Springer, 2017, p. 555.

[12] H. Inoue, "Data augmentation by pairing samples for images classification," 2018.

[13] L. Owen, *Hyperparameter Tuning with Python: Boost Your Machine Learning Model's Performance Via Hyperparameter Tuning*. Birmingham: Packt Publishing, Limited, 2022.

[14] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing (Amsterdam)*, vol. 415, pp. 295–316, 2020.

[15] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[16] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," 2017, cite arxiv:1708.07747Comment: Dataset is freely available at https://github.com/zalandoresearch/fashion-mnist Benchmark is available at http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/. [Online]. Available: http://arxiv.org/abs/1708.07747

[17] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (canadian institute for advanced research)." [Online]. Available: http://www.cs.toronto.edu/ kriz/cifar.html

[18] N. Yadav, "Chessman image dataset," Kaggle, Tech. Rep., 2019. [Online]. Available: https://www.kaggle.com/datasets/niteshfre/chessman-image-dataset

[19] S. Rajaraman, S. K. Antani, M. Poostchi, K. Silamut, M. A. Hossain, R. J. Maude, S. Jaeger, and G. R. Thoma, "Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images," *PeerJ*, vol. 6, p. e4568, 2018.

[20] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar, "Cats and dogs," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[21] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.

[22] D. P. Hughes and M. Salathé , "An open access repository of images on plant health to enable the development of mobile disease diagnostics through machine learning and crowdsourcing," *CoRR*, vol. abs/1511.08060, 2015. [Online]. Available: http://arxiv.org/abs/1511.08060

TABLE II
RESNET50 DATA

	Training Size				
Dataset	2%	4%	6%	8%	10%
MNIST (Accuracy)	61.37% (+/- 1.61%)	72.99% (+/- 1.06%)	75.88% (+/- 0.87%)	79.85% (+/- 1.06%)	82.04% (+/- 0.62%)
MNIST (Precision)	67.95% (+/- 1.66%)	77.53% (+/- 1.03%)	79.63% (+/- 0.88%)	83.14% (+/- 1.16%)	85.04% (+/- 0.61%)
MNIST (Recall)	55.51% (+/- 1.79%)	69.13% (+/- 1.11%)	72.64% (+/- 0.89%)	77.00% (+/- 1.00%)	79.58% (+/- 0.57%)
MNIST (F1-Score)	61.09% (+/- 1.62%)	73.09% (+/- 1.05%)	75.97% (+/- 0.86%)	79.96% (+/- 1.07%)	82.22% (+/- 0.58%)
Fashion (Accuracy)	60.78% (+/- 1.43%)	68.22% (+/- 0.64%)	70.85% (+/- 0.96%)	73.36% (+/- 0.53%)	74.43% (+/- 0.51%)
Fashion (Precision)	65.83% (+/- 1.55%)	71.98% (+/- 0.62%)	74.26% (+/- 1.00%)	77.07% (+/- 0.56%)	78.10% (+/- 0.66%)
Fashion (Recall)	57.08% (+/- 1.50%)	65.36% (+/- 0.93%)	68.34% (+/- 1.03%)	70.64% (+/- 0.78%)	71.65% (+/- 0.56%)
Fashion (F1-Score)	61.14% (+/- 1.43%)	68.51% (+/- 0.70%)	71.18% (+/- 0.99%)	73.72% (+/- 0.55%)	74.73% (+/- 0.54%)
CIFAR10 (Accuracy)	32.72% (+/- 1.03%)	40.96% (+/- 0.51%)	45.18% (+/- 0.55%)	47.43% (+/- 0.73%)	49.01% (+/- 0.39%)
CIFAR10 (Precision)	36.74% (+/- 1.19%)	45.79% (+/- 0.55%)	50.54% (+/- 0.66%)	53.14% (+/- 0.90%)	55.18% (+/- 0.62%)
CIFAR10 (Recall)	25.94% (+/- 1.05%)	34.70% (+/- 0.57%)	38.58% (+/- 0.78%)	40.68% (+/- 0.80%)	41.83% (+/- 0.41%)
CIFAR10 (F1-Score)	30.41% (+/- 1.08%)	39.48% (+/- 0.51%)	43.75% (+/- 0.63%)	46.08% (+/- 0.81%)	47.58% (+/- 0.39%)
CIFAR100 (Accuracy)	6.86% (+/- 0.23%)	12.20% (+/- 0.56%)	15.31% (+/- 0.49%)	17.84% (+/- 0.36%)	19.74% (+/- 0.39%)
CIFAR100 (Precision)	11.24% (+/- 0.63%)	19.81% (+/- 1.03%)	24.37% (+/- 0.79%)	28.34% (+/- 0.76%)	30.88% (+/- 0.61%)
CIFAR100 (Recall)	3.66% (+/- 0.18%)	7.27% (+/- 0.48%)	9.68% (+/- 0.44%)	11.59% (+/- 0.44%)	13.12% (+/- 0.35%)
CIFAR100 (F1-Score)	5.52% (+/- 0.27%)	10.63% (+/- 0.65%)	13.85% (+/- 0.57%)	16.45% (+/- 0.54%)	18.41% (+/- 0.43%)
Chessman (Accuracy)	22.51% (+/- 3.87%)	24.52% (+/- 4.45%)	25.87% (+/- 4.74%)	26.20% (+/- 4.50%)	27.08% (+/- 4.98%)
Chessman (Precision)	29.54% (+/- 9.43%)	34.78% (+/- 10.95%)	31.70% (+/- 7.30%)	30.82% (+/- 7.25%)	31.00% (+/- 9.92%)
Chessman (Recall)	7.03% (+/- 3.23%)	5.48% (+/- 1.67%)	6.36% (+/- 2.10%)	5.88% (+/- 1.63%)	5.95% (+/- 1.87%)
Chessman (F1-Score)	11.13% (+/- 4.47%)	9.40% (+/- 2.74%)	10.53% (+/- 3.27%)	9.84% (+/- 2.60%)	9.92% (+/- 2.99%)
Malaria (Accuracy)	71.42% (+/- 4.32%)	80.98% (+/- 2.23%)	84.02% (+/- 1.13%)	85.26% (+/- 1.98%)	87.10% (+/- 0.93%)
Malaria (Precision)	71.42% (+/- 4.32%)	80.98% (+/- 2.23%)	84.02% (+/- 1.13%)	85.26% (+/- 1.98%)	87.10% (+/- 0.93%)
Malaria (Recall)	71.42% (+/- 4.32%)	80.98% (+/- 2.23%)	84.02% (+/- 1.13%)	85.26% (+/- 1.98%)	87.10% (+/- 0.93%)
Malaria (F1-Score)	71.42% (+/- 4.32%)	80.98% (+/- 2.23%)	84.02% (+/- 1.13%)	85.26% (+/- 1.98%)	87.10% (+/- 0.93%)
Oxford Pets (Accuracy)	22.88% (+/- 4.63%)	27.89% (+/- 2.60%)	31.95% (+/- 3.38%)	46.56% (+/- 1.75%)	47.99% (+/- 2.38%)
Oxford Pets (Precision)	39.56% (+/- 8.99%)	41.47% (+/- 8.09%)	57.89% (+/- 6.81%)	74.10% (+/- 4.63%)	74.80% (+/- 2.60%)
Oxford Pets (Recall)	5.75% (+/- 1.60%)	8.37% (+/- 2.34%)	9.55% (+/- 1.52%)	16.87% (+/- 1.57%)	18.40% (+/- 1.90%)
Oxford Pets (F1-Score)	9.97% (+/- 2.60%)	13.90% (+/- 3.67%)	16.37% (+/- 2.48%)	27.47% (+/- 2.34%)	29.48% (+/- 2.51%)
Houses (Accuracy)	35.77% (+/- 1.06%)	43.75% (+/- 0.74%)	46.66% (+/- 0.35%)	49.51% (+/- 0.34%)	50.89% (+/- 0.46%)
Houses (Precision)	53.81% (+/- 2.10%)	64.77% (+/- 0.95%)	68.80% (+/- 0.98%)	71.68% (+/- 0.71%)	72.71% (+/- 0.55%)
Houses (Recall)	17.37% (+/- 0.71%)	23.60% (+/- 0.44%)	25.99% (+/- 0.56%)	28.68% (+/- 0.39%)	30.60% (+/- 0.23%)
Houses (F1-Score)	26.25% (+/- 0.91%)	34.59% (+/- 0.52%)	37.73% (+/- 0.72%)	40.97% (+/- 0.48%)	43.07% (+/- 0.28%)
Oxford Flowers (Accuracy)	13.50% (+/- 7.09%)	14.00% (+/- 3.91%)	10.98% (+/- 2.75%)	9.38% (+/- 2.15%)	10.88% (+/- 3.60%)
Oxford Flowers (Precision)	30.00% (+/- 45.83%)	28.33% (+/- 39.48%)	25.00% (+/- 33.54%)	33.33% (+/- 32.49%)	26.67% (+/- 31.80%)
Oxford Flowers (Recall)	2.00% (+/- 3.32%)	1.00% (+/- 1.22%)	0.82% (+/- 1.10%)	0.86% (+/- 0.79%)	0.59% (+/- 0.65%)
Oxford Flowers (F1-Score)	3.72% (+/- 6.11%)	1.92% (+/- 2.35%)	1.59% (+/- 2.13%)	1.68% (+/- 1.53%)	1.14% (+/- 1.26%)
Plant Village (Accuracy)	72.61% (+/- 0.89%)	84.38% (+/- 0.64%)	88.13% (+/- 0.38%)	90.54% (+/- 0.37%)	92.08% (+/- 0.20%)
Plant Village (Precision)	91.53% (+/- 1.12%)	94.23% (+/- 0.67%)	94.95% (+/- 0.27%)	95.61% (+/- 0.35%)	96.08% (+/- 0.17%)
Plant Village (Recall)	51.35% (+/- 1.15%)	71.53% (+/- 0.70%)	79.02% (+/- 0.54%)	83.52% (+/- 0.64%)	86.45% (+/- 0.13%)
Plant Village (F1-Score)	65.78% (+/- 1.01%)	81.32% (+/- 0.60%)	86.26% (+/- 0.37%)	89.16% (+/- 0.45%)	91.01% (+/- 0.11%)

TABLE III
RESNET101 DATA

	Training Size				
Dataset	2%	4%	6%	8%	10%
MNIST (Accuracy)	62.10% (+/- 2.71%)	73.64% (+/- 0.97%)	77.27% (+/- 0.92%)	80.75% (+/- 0.51%)	82.14% (+/- 0.82%)
MNIST (Precision)	69.79% (+/- 3.05%)	79.02% (+/- 0.96%)	81.75% (+/- 0.87%)	84.49% (+/- 0.58%)	85.71% (+/- 0.80%)
MNIST (Recall)	53.68% (+/- 2.96%)	69.01% (+/- 0.96%)	73.56% (+/- 1.05%)	77.58% (+/- 0.51%)	79.08% (+/- 0.81%)
MNIST (F1-Score)	60.68% (+/- 3.00%)	73.68% (+/- 0.90%)	77.44% (+/- 0.94%)	80.88% (+/- 0.53%)	82.27% (+/- 0.80%)
Fashion (Accuracy)	65.30% (+/- 1.18%)	71.50% (+/- 0.61%)	73.24% (+/- 0.43%)	75.52% (+/- 0.60%)	76.48% (+/- 0.25%)
Fashion (Precision)	70.80% (+/- 1.48%)	75.65% (+/- 0.88%)	76.84% (+/- 0.54%)	79.23% (+/- 0.83%)	80.26% (+/- 0.29%)
Fashion (Recall)	60.43% (+/- 1.35%)	68.10% (+/- 0.58%)	70.33% (+/- 0.47%)	72.38% (+/- 0.45%)	73.36% (+/- 0.38%)
Fashion (F1-Score)	65.20% (+/- 1.28%)	71.67% (+/- 0.60%)	73.44% (+/- 0.40%)	75.65% (+/- 0.57%)	76.65% (+/- 0.25%)
CIFAR10 (Accuracy)	38.10% (+/- 1.04%)	46.35% (+/- 0.83%)	50.85% (+/- 0.44%)	53.05% (+/- 0.47%)	54.77% (+/- 0.48%)
CIFAR10 (Precision)	45.03% (+/- 1.34%)	53.10% (+/- 0.72%)	57.64% (+/- 0.52%)	60.30% (+/- 0.63%)	62.18% (+/- 0.54%)
CIFAR10 (Recall)	28.76% (+/- 0.99%)	38.18% (+/- 1.23%)	43.12% (+/- 0.72%)	44.90% (+/- 0.44%)	46.61% (+/- 0.66%)
CIFAR10 (F1-Score)	35.10% (+/- 1.07%)	44.42% (+/- 1.07%)	49.33% (+/- 0.63%)	51.47% (+/- 0.42%)	53.28% (+/- 0.57%)
CIFAR100 (Accuracy)	6.91% (+/- 0.29%)	12.17% (+/- 0.50%)	15.10% (+/- 0.46%)	17.35% (+/- 0.21%)	19.09% (+/- 0.23%)
CIFAR100 (Precision)	10.39% (+/- 0.57%)	17.80% (+/- 0.87%)	21.91% (+/- 0.86%)	25.15% (+/- 0.59%)	27.51% (+/- 0.70%)
CIFAR100 (Recall)	4.31% (+/- 0.17%)	8.28% (+/- 0.55%)	10.59% (+/- 0.43%)	12.47% (+/- 0.28%)	13.84% (+/- 0.23%)
CIFAR100 (F1-Score)	6.10% (+/- 0.26%)	11.30% (+/- 0.68%)	14.28% (+/- 0.57%)	16.67% (+/- 0.35%)	18.41% (+/- 0.35%)
Chessman (Accuracy)	22.30% (+/- 3.82%)	27.10% (+/- 3.72%)	28.83% (+/- 4.28%)	29.40% (+/- 4.21%)	30.63% (+/- 3.49%)
Chessman (Precision)	28.11% (+/- 11.67%)	33.59% (+/- 9.37%)	37.33% (+/- 10.93%)	41.60% (+/- 9.13%)	39.98% (+/- 6.72%)
Chessman (Recall)	7.63% (+/- 2.20%)	6.36% (+/- 1.87%)	7.95% (+/- 2.91%)	7.71% (+/- 2.34%)	7.96% (+/- 2.59%)
Chessman (F1-Score)	11.83% (+/- 3.36%)	10.60% (+/- 2.94%)	12.97% (+/- 4.41%)	12.91% (+/- 3.62%)	13.15% (+/- 3.77%)
Malaria (Accuracy)	75.15% (+/- 4.35%)	84.33% (+/- 1.79%)	86.76% (+/- 0.86%)	87.89% (+/- 0.67%)	88.93% (+/- 1.03%)
Malaria (Precision)	75.15% (+/- 4.35%)	84.33% (+/- 1.79%)	86.76% (+/- 0.86%)	87.89% (+/- 0.67%)	88.93% (+/- 1.03%)
Malaria (Recall)	75.15% (+/- 4.35%)	84.33% (+/- 1.79%)	86.76% (+/- 0.86%)	87.89% (+/- 0.67%)	88.93% (+/- 1.03%)
Malaria (F1-Score)	75.15% (+/- 4.35%)	84.33% (+/- 1.79%)	86.76% (+/- 0.86%)	87.89% (+/- 0.67%)	88.93% (+/- 1.03%)
Oxford Pets (Accuracy)	19.59% (+/- 3.83%)	27.82% (+/- 3.49%)	38.14% (+/- 4.43%)	48.30% (+/- 3.67%)	51.20% (+/- 2.39%)
Oxford Pets (Precision)	40.24% (+/- 12.60%)	48.76% (+/- 12.44%)	61.47% (+/- 6.95%)	75.85% (+/- 3.22%)	77.41% (+/- 3.36%)
Oxford Pets (Recall)	6.30% (+/- 3.53%)	9.86% (+/- 3.35%)	10.86% (+/- 1.94%)	19.73% (+/- 3.15%)	20.08% (+/- 1.82%)
Oxford Pets (F1-Score)	10.78% (+/- 5.63%)	16.33% (+/- 5.25%)	18.45% (+/- 3.11%)	31.21% (+/- 4.10%)	31.84% (+/- 2.34%)
Houses (Accuracy)	35.04% (+/- 0.78%)	42.47% (+/- 0.63%)	46.25% (+/- 0.41%)	48.55% (+/- 0.41%)	49.93% (+/- 0.48%)
Houses (Precision)	51.71% (+/- 1.82%)	61.94% (+/- 1.25%)	67.54% (+/- 0.68%)	70.06% (+/- 0.90%)	71.14% (+/- 0.51%)
Houses (Recall)	17.92% (+/- 1.07%)	23.12% (+/- 0.54%)	25.60% (+/- 0.59%)	27.87% (+/- 0.57%)	29.58% (+/- 0.42%)
Houses (F1-Score)	26.61% (+/- 1.37%)	33.67% (+/- 0.68%)	37.12% (+/- 0.72%)	39.87% (+/- 0.70%)	41.78% (+/- 0.48%)
Oxford Flowers (Accuracy)	20.50% (+/- 6.87%)	12.00% (+/- 5.57%)	9.84% (+/- 3.67%)	9.26% (+/- 2.42%)	7.35% (+/- 1.60%)
Oxford Flowers (Precision)	53.00% (+/- 36.28%)	41.67% (+/- 38.19%)	31.67% (+/- 31.14%)	20.94% (+/- 16.80%)	9.70% (+/- 10.67%)
Oxford Flowers (Recall)	5.50% (+/- 3.50%)	2.00% (+/- 1.87%)	1.48% (+/- 1.36%)	1.11% (+/- 0.86%)	0.69% (+/- 0.77%)
Oxford Flowers (F1-Score)	9.57% (+/- 5.73%)	3.77% (+/- 3.47%)	2.79% (+/- 2.56%)	2.09% (+/- 1.61%)	1.28% (+/- 1.43%)
Plant Village (Accuracy)	71.95% (+/- 0.89%)	84.50% (+/- 0.37%)	88.42% (+/- 0.26%)	90.48% (+/- 0.50%)	91.66% (+/- 0.27%)
Plant Village (Precision)	91.42% (+/- 0.84%)	93.89% (+/- 0.22%)	94.79% (+/- 0.23%)	95.48% (+/- 0.37%)	95.73% (+/- 0.21%)
Plant Village (Recall)	51.65% (+/- 0.82%)	72.13% (+/- 0.82%)	79.65% (+/- 0.40%)	83.87% (+/- 0.37%)	86.39% (+/- 0.29%)
Plant Village (F1-Score)	66.00% (+/- 0.64%)	81.58% (+/- 0.56%)	86.56% (+/- 0.25%)	89.30% (+/- 0.33%)	90.82% (+/- 0.24%)

TABLE IV
RESNET152 DATA

	Training Size				
Dataset	2%	4%	6%	8%	10%
MNIST (Accuracy)	59.49% (+/- 3.30%)	72.07% (+/- 0.73%)	75.43% (+/- 0.83%)	79.23% (+/- 0.52%)	81.34% (+/- 0.63%)
MNIST (Precision)	64.69% (+/- 3.70%)	75.79% (+/- 0.63%)	78.65% (+/- 0.82%)	82.10% (+/- 0.51%)	84.03% (+/- 0.68%)
MNIST (Recall)	54.71% (+/- 3.28%)	69.06% (+/- 0.85%)	72.89% (+/- 0.87%)	76.99% (+/- 0.55%)	79.26% (+/- 0.47%)
MNIST (F1-Score)	59.28% (+/- 3.45%)	72.27% (+/- 0.70%)	75.66% (+/- 0.82%)	79.46% (+/- 0.52%)	81.58% (+/- 0.56%)
Fashion (Accuracy)	62.23% (+/- 1.23%)	69.81% (+/- 0.68%)	72.26% (+/- 0.34%)	74.42% (+/- 0.53%)	75.36% (+/- 0.57%)
Fashion (Precision)	65.44% (+/- 1.32%)	72.53% (+/- 0.64%)	74.59% (+/- 0.42%)	76.63% (+/- 0.58%)	77.92% (+/- 0.69%)
Fashion (Recall)	59.65% (+/- 1.16%)	67.71% (+/- 0.80%)	70.49% (+/- 0.43%)	72.75% (+/- 0.55%)	73.38% (+/- 0.47%)
Fashion (F1-Score)	62.41% (+/- 1.20%)	70.04% (+/- 0.69%)	72.48% (+/- 0.36%)	74.64% (+/- 0.53%)	75.59% (+/- 0.55%)
CIFAR10 (Accuracy)	37.60% (+/- 1.06%)	47.29% (+/- 0.97%)	51.15% (+/- 0.53%)	53.62% (+/- 0.39%)	53.69% (+/- 0.58%)
CIFAR10 (Precision)	43.45% (+/- 1.30%)	53.48% (+/- 1.12%)	57.58% (+/- 0.68%)	60.14% (+/- 0.53%)	60.12% (+/- 0.60%)
CIFAR10 (Recall)	29.57% (+/- 1.28%)	40.48% (+/- 1.07%)	44.16% (+/- 0.63%)	46.65% (+/- 0.63%)	46.81% (+/- 0.49%)
CIFAR10 (F1-Score)	35.19% (+/- 1.30%)	46.08% (+/- 1.04%)	49.98% (+/- 0.61%)	52.54% (+/- 0.53%)	52.64% (+/- 0.51%)
CIFAR100 (Accuracy)	7.33% (+/- 0.38%)	12.33% (+/- 0.51%)	15.35% (+/- 0.42%)	18.09% (+/- 0.17%)	19.36% (+/- 0.36%)
CIFAR100 (Precision)	10.66% (+/- 0.71%)	17.95% (+/- 0.83%)	22.18% (+/- 0.61%)	25.94% (+/- 0.53%)	27.87% (+/- 0.60%)
CIFAR100 (Recall)	4.63% (+/- 0.33%)	8.62% (+/- 0.42%)	11.14% (+/- 0.33%)	13.38% (+/- 0.35%)	14.43% (+/- 0.34%)
CIFAR100 (F1-Score)	6.46% (+/- 0.44%)	11.65% (+/- 0.54%)	14.83% (+/- 0.41%)	17.65% (+/- 0.40%)	19.02% (+/- 0.41%)
Chessman (Accuracy)	23.92% (+/- 4.45%)	27.88% (+/- 3.47%)	28.90% (+/- 3.77%)	29.58% (+/- 3.79%)	30.70% (+/- 2.92%)
Chessman (Precision)	32.09% (+/- 5.51%)	38.93% (+/- 10.86%)	42.54% (+/- 8.35%)	40.68% (+/- 10.71%)	42.21% (+/- 9.42%)
Chessman (Recall)	8.13% (+/- 2.49%)	8.41% (+/- 4.22%)	10.07% (+/- 3.60%)	7.96% (+/- 2.84%)	9.05% (+/- 2.93%)
Chessman (F1-Score)	12.86% (+/- 3.56%)	13.66% (+/- 6.23%)	16.06% (+/- 4.96%)	13.26% (+/- 4.53%)	14.76% (+/- 4.42%)
Malaria (Accuracy)	78.82% (+/- 2.77%)	85.36% (+/- 1.64%)	87.17% (+/- 1.15%)	88.83% (+/- 0.66%)	90.19% (+/- 0.61%)
Malaria (Precision)	78.82% (+/- 2.77%)	85.36% (+/- 1.64%)	87.17% (+/- 1.15%)	88.83% (+/- 0.66%)	90.19% (+/- 0.61%)
Malaria (Recall)	78.82% (+/- 2.77%)	85.36% (+/- 1.64%)	87.17% (+/- 1.15%)	88.83% (+/- 0.66%)	90.19% (+/- 0.61%)
Malaria (F1-Score)	78.82% (+/- 2.77%)	85.36% (+/- 1.64%)	87.17% (+/- 1.15%)	88.83% (+/- 0.66%)	90.19% (+/- 0.61%)
Oxford Pets (Accuracy)	24.66% (+/- 3.73%)	30.48% (+/- 1.72%)	38.05% (+/- 4.01%)	49.66% (+/- 3.10%)	51.25% (+/- 1.77%)
Oxford Pets (Precision)	46.50% (+/- 13.53%)	55.14% (+/- 7.64%)	65.46% (+/- 6.13%)	77.57% (+/- 4.98%)	76.77% (+/- 3.50%)
Oxford Pets (Recall)	6.99% (+/- 2.48%)	10.07% (+/- 2.17%)	12.00% (+/- 2.45%)	19.80% (+/- 2.08%)	20.92% (+/- 1.47%)
Oxford Pets (F1-Score)	12.06% (+/- 4.15%)	16.94% (+/- 3.20%)	20.17% (+/- 3.62%)	31.50% (+/- 2.87%)	32.87% (+/- 2.05%)
Houses (Accuracy)	34.34% (+/- 1.46%)	41.92% (+/- 0.77%)	45.10% (+/- 0.52%)	47.25% (+/- 0.32%)	48.24% (+/- 0.71%)
Houses (Precision)	50.32% (+/- 2.63%)	60.88% (+/- 0.91%)	66.64% (+/- 0.89%)	68.87% (+/- 0.79%)	69.35% (+/- 0.63%)
Houses (Recall)	15.82% (+/- 0.96%)	21.55% (+/- 0.54%)	23.12% (+/- 0.36%)	25.13% (+/- 0.43%)	26.08% (+/- 0.42%)
Houses (F1-Score)	24.06% (+/- 1.33%)	31.83% (+/- 0.65%)	34.33% (+/- 0.47%)	36.82% (+/- 0.53%)	37.91% (+/- 0.52%)
Oxford Flowers (Accuracy)	16.00% (+/- 9.17%)	12.50% (+/- 5.48%)	9.84% (+/- 4.34%)	7.90% (+/- 1.76%)	8.92% (+/- 2.34%)
Oxford Flowers (Precision)	26.67% (+/- 31.80%)	27.50% (+/- 32.50%)	22.86% (+/- 24.79%)	26.43% (+/- 33.58%)	17.61% (+/- 13.73%)
Oxford Flowers (Recall)	3.50% (+/- 3.91%)	1.50% (+/- 1.66%)	1.15% (+/- 1.28%)	0.86% (+/- 0.96%)	0.88% (+/- 0.69%)
Oxford Flowers (F1-Score)	6.13% (+/- 6.83%)	2.80% (+/- 3.06%)	2.16% (+/- 2.39%)	1.65% (+/- 1.83%)	1.67% (+/- 1.30%)
Plant Village (Accuracy)	69.92% (+/- 0.90%)	83.26% (+/- 0.79%)	87.41% (+/- 0.51%)	90.09% (+/- 0.38%)	91.38% (+/- 0.36%)
Plant Village (Precision)	89.78% (+/- 1.59%)	93.39% (+/- 0.64%)	94.43% (+/- 0.41%)	95.24% (+/- 0.32%)	95.63% (+/- 0.35%)
Plant Village (Recall)	50.19% (+/- 1.35%)	70.62% (+/- 0.75%)	78.63% (+/- 0.56%)	83.31% (+/- 0.41%)	85.81% (+/- 0.43%)
Plant Village (F1-Score)	64.38% (+/- 1.37%)	80.42% (+/- 0.62%)	85.81% (+/- 0.41%)	88.88% (+/- 0.35%)	90.45% (+/- 0.38%)

TABLE V
RESNET50 BAYESIAN OPTIMIZED RESULTS

Dataset (Image Count)	Percentage	Images Used	F1-Score	Batch Size	Learning Rate	Regularization Strength
MNIST (60000)	2%	1200	88.13	16.00	0.0010	0.0010
	4%	2400	88.13	16.00	0.0010	0.0010
	6%	3600	88.13	16.00	0.0010	0.0010
	8%	4800	89.19	88.56	0.0155	0.0101
	10%	6000	90.75	116.10	0.0723	0.0010
Fashion MNIST (60000)	2%	1200	84.76	22.55	0.0010	0.0010
	4%	2400	85.68	26.80	0.0011	0.0011
	6%	3600	85.68	26.80	0.0011	0.0011
	8%	4800	85.68	26.80	0.0011	0.0011
	10%	6000	85.68	26.80	0.0011	0.0011
CIFAR10 (50000)	2%	1000	85.85	16.01	0.0273	0.0977
	4%	2000	85.85	16.01	0.0273	0.0977
	6%	3000	85.85	16.01	0.0273	0.0977
	8%	4000	87.23	46.46	0.0010	0.0010
	10%	5000	87.23	46.46	0.0010	0.0010
CIFAR100 (50000)	2%	1000	48.61	88.12	0.0625	0.0095
	4%	2000	57.96	88.56	0.0155	0.0101
	6%	3000	61.29	116.10	0.0723	0.0010
	8%	4000	64.73	86.58	0.0402	0.0050
	10%	5000	66.67	116.10	0.0723	0.0010
Chessman (556)	2%	11	28.57	116.10	0.0723	0.0010
	4%	22	48.65	116.10	0.0723	0.0010
	6%	33	48.65	116.10	0.0723	0.0010
	8%	44	48.65	116.10	0.0723	0.0010
	10%	55	48.65	116.10	0.0723	0.0010
Malaria (27558)	2%	551	79.31	70.60	0.0164	0.0948
	4%	1102	85.12	94.79	0.0101	0.0340
	6%	1653	88.93	16.00	0.0010	0.0010
	8%	2204	88.93	16.00	0.0010	0.0010
	10%	2755	88.93	16.00	0.0010	0.0010
Oxford Pets (3669)	2%	73	15.91	143.80	0.0790	0.0183
	4%	146	29.95	67.84	0.0729	0.0089
	6%	220	77.47	16.00	0.0010	0.1000
	8%	293	87.18	16.00	0.0010	0.0010
	10%	366	87.18	16.00	0.0010	0.0010
SVHN Cropped (73257)	2%	1465	30.50	88.56	0.0155	0.0101
	4%	2930	35.59	88.56	0.0155	0.0101
	6%	4395	39.20	82.21	0.0094	0.0081
	8%	5860	42.64	87.31	0.1000	0.0010
	10%	7325	44.18	116.10	0.0723	0.0010
Oxford Flowers (6149)	2%	122	17.39	16.00	0.0667	0.0071
	4%	245	17.39	16.00	0.0667	0.0071
	6%	368	17.65	16.86	0.0746	0.0769
	8%	491	17.65	16.86	0.0746	0.0769
	10%	614	17.65	16.86	0.0746	0.0769
Plant Village (54303)	2%	1086	71.10	88.57	0.0017	0.0145
	4%	2172	82.09	88.56	0.0155	0.0101
	6%	3258	87.26	115.50	0.0010	0.0010
	8%	4344	90.95	86.05	0.0595	0.0015
	10%	5430	90.95	86.05	0.0595	0.0015

TABLE VI
RESNET101 BAYESIAN OPTIMIZED RESULTS

Dataset (Image Count)	Percentage	Images Used	F1-Score	Batch Size	Learning Rate	Regularization Strength
MNIST (60000)	2%	1200	80.26	33.65	0.0659	0.0103
	4%	2400	81.44	88.57	0.0017	0.0145
	6%	3600	84.77	116.10	0.0723	0.0010
	8%	4800	87.72	88.56	0.0155	0.0101
	10%	6000	88.63	116.10	0.0723	0.0010
Fashion MNIST (60000)	2%	1200	81.71	49.79	0.0594	0.0337
	4%	2400	83.56	95.77	0.1000	0.0514
	6%	3600	84.69	88.56	0.0155	0.0101
	8%	4800	85.25	88.56	0.0155	0.0101
	10%	6000	86.25	90.60	0.0010	0.0010
CIFAR10 (50000)	2%	1000	84.23	44.31	0.1000	0.0010
	4%	2000	87.83	39.98	0.0077	0.0943
	6%	3000	88.16	63.61	0.0017	0.0626
	8%	4000	88.51	96.33	0.1000	0.0010
	10%	5000	89.39	88.56	0.0155	0.0101
CIFAR100 (50000)	2%	1000	53.06	88.56	0.0155	0.0101
	4%	2000	65.20	90.87	0.0029	0.0010
	6%	3000	66.98	116.10	0.0723	0.0010
	8%	4000	69.31	86.58	0.0402	0.0050
	10%	5000	70.64	116.10	0.0723	0.0010
Chessman (556)	2%	11	26.67	177.90	0.0184	0.0555
	4%	22	26.67	177.90	0.0184	0.0555
	6%	33	26.67	177.90	0.0184	0.0555
	8%	44	27.40	221.20	0.0722	0.0913
	10%	55	27.40	221.20	0.0722	0.0913
Malaria (27558)	2%	551	87.84	16.00	0.0010	0.0010
	4%	1102	87.84	16.00	0.0010	0.0010
	6%	1653	90.68	31.41	0.0143	0.0464
	8%	2204	90.68	31.41	0.0143	0.0464
	10%	2755	90.68	31.41	0.0143	0.0464
Oxford Pets (3669)	2%	73	18.37	70.60	0.0164	0.0948
	4%	146	25.84	88.57	0.0017	0.0145
	6%	220	84.56	19.39	0.0088	0.0062
	8%	293	84.56	19.39	0.0088	0.0062
	10%	366	84.56	19.39	0.0088	0.0062
SVHN Cropped (73257)	2%	1465	29.87	84.27	0.0260	0.0076
	4%	2930	35.30	86.82	0.0124	0.0010
	6%	4395	37.30	116.10	0.0723	0.0010
	8%	5860	40.10	88.56	0.0155	0.0101
	10%	7325	41.86	116.10	0.0723	0.0010
Oxford Flowers (6149)	2%	122	18.18	221.20	0.0722	0.0913
	4%	245	18.18	221.20	0.0722	0.0913
	6%	368	24.32	16.00	0.1000	0.0010
	8%	491	24.32	16.00	0.1000	0.0010
	10%	614	24.32	16.00	0.1000	0.0010
Plant Village (54303)	2%	1086	89.41	16.00	0.0010	0.0010
	4%	2172	89.41	16.00	0.0010	0.0010
	6%	3258	89.41	16.00	0.0010	0.0010
	8%	4344	91.23	85.96	0.0013	0.0011
	10%	5430	91.88	85.32	0.0010	0.0014

TABLE VII
RESNET152 BAYESIAN OPTIMIZED RESULTS

Dataset (Image Count)	Percentage	Images Used	F1-Score	Batch Size	Learning Rate	Regularization Strength
MNIST (60000)	2%	1200	70.29	77.18	0.0010	0.0010
	4%	2400	79.37	88.56	0.0155	0.0101
	6%	3600	83.05	88.56	0.0155	0.0101
	8%	4800	85.95	87.16	0.0038	0.0024
	10%	6000	87.80	116.10	0.0723	0.0010
Fashion MNIST (60000)	2%	1200	84.69	16.00	0.0539	0.0446
	4%	2400	85.58	54.22	0.0156	0.0234
	6%	3600	85.92	68.67	0.0208	0.0368
	8%	4800	85.96	88.99	0.0281	0.0068
	10%	6000	87.00	88.56	0.0155	0.0101
CIFAR10 (50000)	2%	1000	90.20	16.00	0.0010	0.0010
	4%	2000	91.16	16.00	0.0010	0.0010
	6%	3000	91.16	16.00	0.0010	0.0010
	8%	4000	91.16	16.00	0.0010	0.0010
	10%	5000	91.16	16.00	0.0010	0.0010
CIFAR100 (50000)	2%	1000	55.01	88.57	0.0017	0.0145
	4%	2000	66.00	116.10	0.0723	0.0010
	6%	3000	69.13	116.10	0.0723	0.0010
	8%	4000	70.80	116.10	0.0723	0.0010
	10%	5000	72.01	86.02	0.0010	0.0033
Chessman (556)	2%	11	35.29	190.30	0.0731	0.0203
	4%	22	35.29	190.30	0.0731	0.0203
	6%	33	35.29	190.30	0.0731	0.0203
	8%	44	35.29	190.30	0.0731	0.0203
	10%	55	35.29	190.30	0.0731	0.0203
Malaria (27558)	2%	551	81.85	116.10	0.0723	0.0010
	4%	1102	88.02	70.60	0.0164	0.0948
	6%	1653	90.32	88.56	0.0155	0.0101
	8%	2204	91.65	38.87	0.0010	0.0010
	10%	2755	91.65	38.87	0.0010	0.0010
Oxford Pets (3669)	2%	73	27.27	190.30	0.0731	0.0203
	4%	146	31.91	89.87	0.0215	0.0274
	6%	220	84.26	20.86	0.0010	0.0010
	8%	293	87.18	17.50	0.1000	0.0010
	10%	366	87.18	17.50	0.1000	0.0010
SVHN Cropped (73257)	2%	1465	27.96	88.56	0.0155	0.0101
	4%	2930	33.36	116.10	0.0723	0.0010
	6%	4395	36.07	92.49	0.0060	0.0010
	8%	5860	38.25	87.49	0.0707	0.0019
	10%	7325	38.79	116.10	0.0723	0.0010
Oxford Flowers (6149)	2%	122	23.08	116.10	0.0723	0.0010
	4%	245	23.08	116.10	0.0723	0.0010
	6%	368	28.17	16.00	0.1000	0.1000
	8%	491	28.17	16.00	0.1000	0.1000
	10%	614	28.17	16.00	0.1000	0.1000
Plant Village (54303)	2%	1086	72.99	88.56	0.0155	0.0101
	4%	2172	82.22	88.56	0.0155	0.0101
	6%	3258	86.19	116.10	0.0723	0.0010
	8%	4344	89.18	116.10	0.0723	0.0010
	10%	5430	90.76	116.10	0.0723	0.0010