2021 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD) | 978-1-7281-8592-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICABCD51485.2021.9519341

Investigating Popular CNN Architectures for Plant Disease Detection

Louise Poole[1] and Dane Brown[2]
Computer Science, *Rhodes University*
Grahamstown, South Africa
[1]louisecarmenpoole@gmail.com, [2]d.brown@ru.ac.za

***Abstract*—Food production and food security have become increasingly important due to climate change and rising population numbers. One method to prevent crop loss is to develop a system to allow for early, efficient and accurate identification of plant diseases. CNNs often outperform previously popular machine learning algorithms. There are many existing CNN architectures. We compared and analysed the popular state-of-the-art architectures, namely ResNet, GoogLeNet and VGG, when trained for plant disease classification. We found that ResNet performed the best on the balanced Mendeley Leaves and PlantVillage datasets, obtaining 91.95% and 95.80% accuracy respectively. However, the ResNet architecture was relatively computationally expensive and slow to train. GoogLeNet obtained accuracies very close to those of ResNet with 89.35% and 94.59% achieved on the Mendeley Leaves and PlantVillage datasets respectively and could be considered a less computationally expensive alternative.**

***Index Terms*—neural network architecture, machine vision, image classification, deep learning, plant disease**

I. Introduction

Food production and food security have become increasingly important due to climate change and rising population numbers. One method to prevent crop loss is to develop a system that allows for early, efficient and accurate identification of plant diseases. Plant disease identification, however, is a non-trivial problem made challenging due to the varying nature of the visual appearance of various diseases on different plant species. Many previously proposed methods utilize a combination of image processing with machine learning in order to achieve this.

Previously popular machine learning algorithms such as k-Nearest Neighbours (k-NN), Support Vector Machines (SVM), Naive Bayes and Probabilistic Neural Networks (PNN) perform well on plant disease data [1, 2, 3], but are significantly outperformed by Convolutional Neural Networks (CNNs) [4]. It is also noted that unlike the other machine learning algorithms listed, CNNs perform well even with image processing steps omitted.

Many CNN architectures have been developed to perform well on image datasets. The performance of each architecture depends on the nature of the dataset that is being evaluated. While one architecture may do well on a dataset for facial recognition, that does not necessarily imply it will do well on a plant disease dataset. This paper analyses and compares the popular CNN architectures known as ResNet [5], GoogleNet [6], and VGG-16 [7] and investigates their discriminative abilities on plant disease classification problems of a multi-class nature.

This study was funded by National Research Foundation (120654). This work was undertaken in the Distributed Multimedia CoE at Rhodes University.

II. Machine Learning

Machine learning can be defined as automated learning performed by a program to make predictions. This is done through the use of learning algorithms which use available data to create an estimation of some unknown mapping between system inputs and outputs. This can be used to generate future predictions based on the previously observed samples. One of the most widely researched fields of machine learning is supervised learning. Data samples with both known inputs and known outputs are used during supervised learning [8].

Furthermore, the models developed using supervised learning can generally be divided into three subcategories according to the task they are designed to accomplish: classification, regression and probability models. Classification models generate an estimation of class decision boundaries and are utilized when a limited number of distinct outputs are possible. Regression models generate real-valued outputs, and probability models generate probability density estimations [8].

Convolutional Neural Networks fall under the classification category and will be explored in more detail below.

III. Convolutional Neural Networks

A Convolutional Neutral Network (CNN) is a deep neural network algorithm that utilizes convolutional layers. Convolutional layers are considered to be more specialized and efficient than fully connected layers.

In a fully connected layer, each neuron within the layer is connected to every neuron in the layer before. Each connection also has its own weight. This makes using fully connected layers computationally expensive as well as memory heavy.

In contrast, in a convolutional layer, every neuron within the layer is connected to only a few local neurons in the previous layer. This proves to be very useful for spatial data where features are locally clustered. It also effectively enables the CNN to prioritize salient features and ignore or prune unnecessary input. Therefore, feature extraction during preprocessing is often unnecessary for CNNs compared to traditional classifiers. The same set of weights are used for every neuron's set of connections. This is called parameter sharing and is useful when features are equally likely to be found anywhere in the input. Hence CNNs are a popular choice for learning objects in image datasets.

In general, a CNN architecture comprises a combination of one or more convolutional, pooling and fully connected layers. A few of the most popular architectures for image datasets include ResNet [5], GoogleNet [6] and VGG-16 [7]. While one architecture may outperform another on average, in special cases, this may prove untrue [9]. Thus, it is important to find and optimize an architecture that performs best on the specific problem or dataset in question.

An important practice when using CNNs is to obtain a training set that is as large as possible [9]. This allows the algorithm to learn slight variations in the data and thus generalize the problem better. Smaller datasets can be expanded by augmenting the existing data through the use of transformations. Simple ones include translations, rotations and skewing [10]. By introducing these transformations, the learning algorithm can infer the transformation invariance and generalize it effectively.

IV. Architectures

The design of CNN architectures plays essential roles in discriminative performance. In particular, network depth has proven to be of critical importance when designing a CNN architecture [7]. However, simply stacking more layers leads to problems such as vanishing or exploding gradients [11], which hinder convergence, increasing training error. Interestingly this increase in training error is not due to overfitting. The addition of more layers causes the system's accuracy to plateau, followed by a rapid drop. Hence increasing the number of layers has the potential to be beneficial. However, it must be done in such a way as to avoid the aforementioned problems.

The following architectures make use of deep networks:

A. ResNet

He et al. [5] makes use of a deep residual learning framework in their architecture in order to combat the degradation issue that arises with high network depth. This is done by using 'shortcut connections' [12] which allows connections to skip one or more layers. In this case, these shortcut connections are simple identity mappings where their outputs are added to the outputs of the stacked layers. The system learns residual functions without adding computational complexity and thus making the deep network easier to optimize.

He et al. [5] compare their proposed architecture to popular existing state-of-the-art architectures such as VGG-16, GoogLeNet and PReLU-net. They performed the comparison using the ImageNet 2012 classification dataset [13]. This image dataset consists of 1000 classes and over 1.4 million images. They used just over 1.2 million images for training, 50k for validation and 100k for testing. Comparing the top-5 error rates of the models generated for each architecture, ResNet obtained consistently lower error rates than the other architectures considered. The top performing existing architecture models obtained an error equal to the poorest performing ResNet model. The ResNet models showed a trend of decreasing error rate with increasing layer depth. The 152 layer ResNet model obtained the lowest error rate of 4.49%.

In this paper, a 30 layer ResNet model based on the work presented by He et al. [5] was implemented and utilized to obtain the results in section VI.

B. VGG-16

Simonyan and Zisserman [7] counters the typical issues that arise with very deep networks by utilizing many convolutional layers with small 3x3 filter sizes. The chosen 3x3 filter size is the smallest filter size to capture the notion of left and right, up and down, and centre.

Maximum Pooling (maxpool) layers are used in between various 3x3 convolutional layers to reduce volume size. The maxpool layer achieves this by using a 2x2 filter to calculate the maximum value of a patch of the current feature map. The values in that patch are replaced with the maximum value only, reducing the data's size. This results in a feature map that highlights the most prevalent feature in each patch.

VGG-16 is comprised of 16 such layers, making a large network with approximatly 138 million parameters.

C. GoogleNet

Szegedy et al. [6] focus on computational efficiency and practicality with their architecture design. They counter the vanishing gradient problem by adding auxiliary classifiers connected to the network's middle layers, providing regularization.

Similar to VGG, they make use of maxpool to reduce volume size in the middle layers. However, they use average pooling, followed by fully connected layers directly before each activation layer.

Szegedy et al. [6] utilized the ImageNet 2012 classification dataset [13] for testing. They achieved a top-5 error rate of 6.67% while VGG, MSRA and other top architectures achieved top-5 error rates of 7.32%, 7.35% and above 11.2%, respectively.

V. Experimental Setup

The following section describes the structure and setup of the experiments conducted.

A. Datasets

Two popular publicly available plant disease image datasets were used in the testing and comparison of the aforementioned architectures:

1) Mendeley Leaves: a dataset consisting of 4500 images of diseased and healthy plant leaves. It contains 10 plant species with only two target classes per plant species: diseased and healthy. The dataset does not specify what the diseases are. The entire dataset was comprised of images of a single leaf with a dark grey background. This dataset was useful to train a model to identify a leaf's plant species as well as if the leaf is diseased or not. However, it cannot determine what the visible disease is [14].

2) PlantVillage: a dataset consisting of over 54 000 images of diseased and healthy plant leaves. It contains 20 plant diseases across 14 plant species. The majority of the dataset comprised images of a single leaf with a monotone background created by holding a purple/light piece of paper behind the leaf. However, several also contained other leaves of the same species in the background. These images were not removed as they should not negatively affect the final classification.

This dataset was comprised of images of both diseased and healthy leaves for multiple plant species, all of which were used in training. This allowed the models created to determine the species of leaf, if the leaf is diseased or not and if so, what disease was visible.

The acquired datasets contain varied sample sizes and so required balancing to achieve realistic results. The datasets were balanced by shuffling and splitting using a random number generator with a seed. This ensured that all test data across all experiments was unseen and balanced. After balancing, the Mendeley Leaves dataset consisted of 1540 images, and the PlantVillage dataset consisted of 5624 images.

A 75:25 split was used to allocate the data for training and testing. The (previously imbalanced) class instances determined the number of images used per model based on the smallest number of instances. Data augmentation was applied to the training dataset only.

B. Measuring Model Performance

Multiple metrics are available for gauging the performance of a given machine learning model. The most commonly used are accuracy, precision, recall, and f1 score – their harmonic mean. We use accuracy as it is the most popular metric and vital for comparing results to other research.

1) Accuracy: Classification accuracy is the favoured metric for comparing the performance of machine learning models in the majority of research papers available.

Accuracy is a measurement for the system's overall correctness and is found by calculating the ratio between the number of correctly classified samples and the total number of input samples.

$$Accuracy = \frac{\text{number of correct classifications}}{\text{total number of classifications attempted}}$$

This gives a true reflection of a given model's performance if the test data used is balanced.

C. Parameter Tuning

Keras Tuner[1], a hyperparameter optimizer developed by the Google team, was used to determine the best parameter values for the tested architectures. The optimizer used was Hyperband [15]. Hyperband is based on the popular Random Search optimizer, however, uses early stopping and adaptive resource allocation to decrease the system's computational expense and increase the tuning speed of the system. The chosen parameters for tuning were *learning rate* and *epochs*.

[1]https://github.com/keras-team/keras-tuner

D. Test Models

For each test model produced, the system first underwent parameter tuning, followed by the retraining of the model with the chosen parameters and finally testing on the test data.

Each dataset was split into three sections for this process: training, validation, and test data. The training data was used for parameter tuning with the validation data used to evaluate the comparative performance of each of the tuner's models. Once the best combination of parameters was determined on the validation data, a new model was trained on the combined training and validation data. This final model was tested on the unseen test data to reflect the model's unbiased performance accurately. Furthermore, data contamination was avoided during data augmentation as it was applied only to the training data, not to the validation data during tuning or the test data. A visual overview of the process described above is shown in Figure 1.

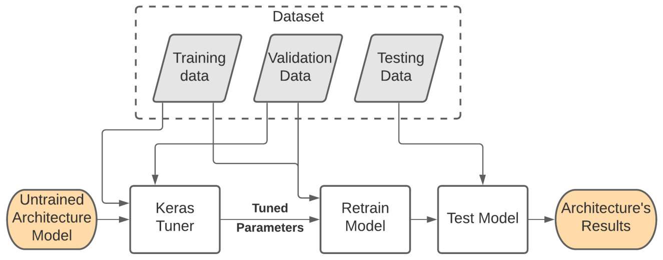

Fig. 1. Overview of system used to test the architecture models.

Six models were trained for the respective experiments listed below:

- Experiment 1 - Compare the performance of the ResNet, GoogleNet and VGG-16 architectures when trained and tuned on the balanced Mendeley Leaves dataset.
- Experiment 2 - Compare the performance of the ResNet, GoogleNet and VGG-16 architectures when trained and tuned on the balanced PlantVillage dataset.

VI. Results and Discussion

A. Parameter Tuning

For parameter tuning, 25 trials were used to find the best learning rate. Learning rates of $1e^{-}2$ to $1e^{-}6$ were explored. A maximum of 1000 epochs were used to find the best epoch. The parameters that produced the best models per dataset-architecture pairing are given in Table I.

TABLE I
BEST PARAMETERS DETERMINED WITH KERAS TUNER

Dataset	**Architecture**	**Best Parameters**	
		Learning Rate	***Epoch***
Mendeley Leaves	ResNet	$1e^{-}3$	799
	GoogLeNet	$1e^{-}4$	480
	VGG-16	$1e^{-}4$	804
PlantVillage	ResNet	$1e^{-}4$	644
	GoogLeNet	$1e^{-}4$	647
	VGG-16	$1e^{-}5$	984

The training and validation accuracies for the ResNet and GoogLeNet models produced on the Mendeley Leaves dataset with the above parameters are graphed in Figure 2 and 3, respectively. Figure 2 shows that the higher learning rate of the ResNet model created a more erratic learning curve than that of GoogLeNet. However, it achieved higher accuracies much sooner, in terms of epochs, than GoogLeNet and obtained a higher maximum validation accuracy overall.

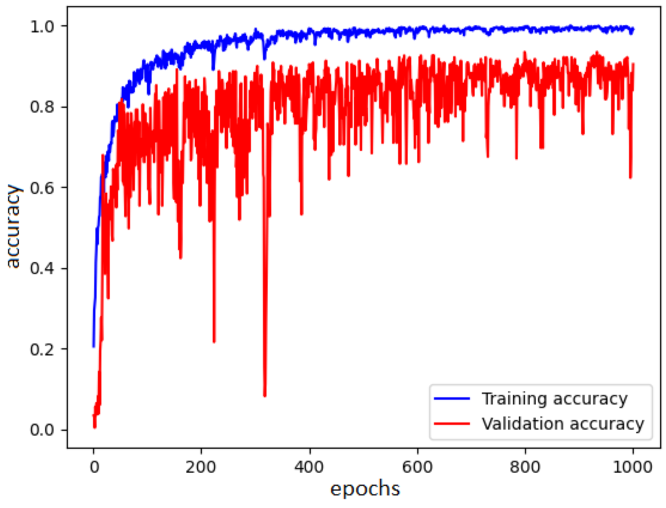

Fig. 2. Training and validation accuracy of the ResNet model trained on Mendeley Leaves dataset.

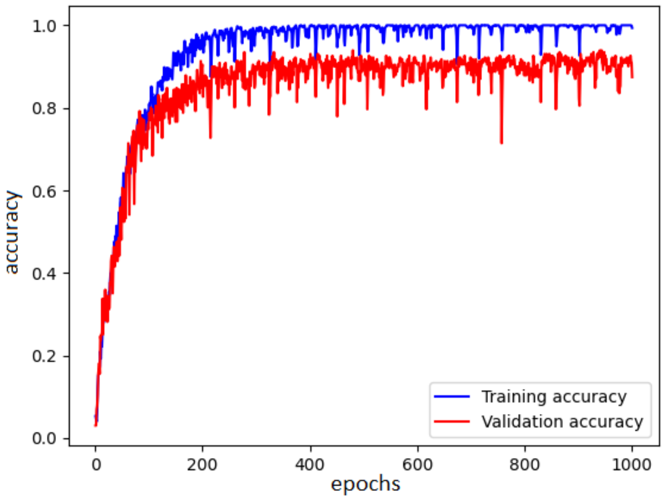

Fig. 3. Training and validation accuracy of the GoogLeNet model trained on Mendeley Leaves dataset.

It should be noted that ResNet took significantly longer to train than GoogLeNet. Seeing this, as well as the similar accuracies they obtained during tuning, one may need to consider execution time and computational expense over the insignificant improvement in accuracy when choosing between these architectures for leaf disease detection. While this paper focuses on the recognition accuracy that the models produce, the role that execution time and computational expense may play in some cases are acknowledged.

B. *Experiment 1*

Experiment 1 trains and tests the models produced using the architectures ResNet, GoogleNet and VGG-16 with the relevant parameters determined in section VI-A using the balanced Mendeley Leaves dataset. The results of each are given in Table II.

TABLE II
PERFORMANCE OF ARCHITECTURES ON BALANCED MENDELEY LEAVES DATASET.

Architecture	Training		Testing	
	Loss	***Accuracy***	***Loss***	***Accuracy***
ResNet	0.0123	99.57%	0.3620	91.95%
GoogLeNet	0.0041	100.00%	1.799	89.35%
VGG-16	0.0002	100.00%	0.5885	88.83%

It can be seen that ResNet obtained the best test accuracy of 91.95% while GoogLeNet obtained only 2.60% lower and VGG-16 only 0.52% lower than GoogLeNet. By inspection of the confusion matrix produced by the ResNet model and given in Figure 4, the two significant confusions that can be noted are those between *Lemon Diseased* and *Lemon Healthy*, and *Jamun Diseased* and *Jamun Healthy*. However, these minor confusions inevitably highlight the architecture's strong ability to classify plant species, irrelevant to it being diseased or not, since the correct plant species were still identified in both cases.

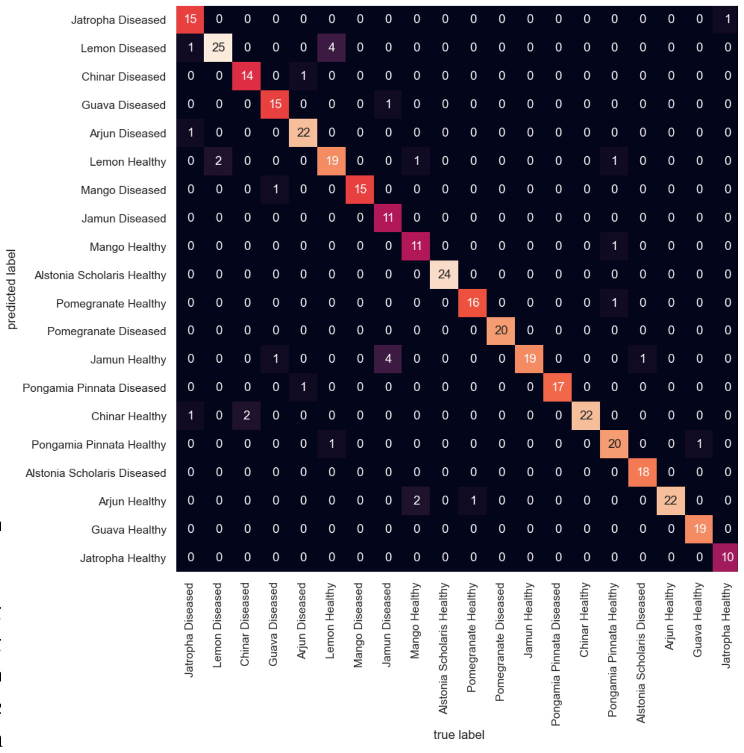

Fig. 4. Confusion Matrix of ResNet model trained on Mendeley Leaves dataset

C. Experiment 2

Experiment 2 trains and tests the models produced using the architectures ResNet, GoogleNet and VGG-16 with the relevant parameters determined in section VI-A on the balanced PlantVillage dataset. The results of each are given in Table III.

TABLE III
PERFORMANCE OF ARCHITECTURES ON BALANCED PLANTVILLAGE DATASET

Architecture	**Training**		**Testing**	
	Loss	***Accuracy***	***Loss***	***Accuracy***
ResNet	0.0091	99.74%	0.2661	95.80%
GoogLeNet	0.0001	100.00%	0.9716	94.59%
VGG-16	0.0350	99.00%	0.8479	87.34%

As with Experiment 1, ResNet obtained the top performance with an accuracy of 95.80%. GoogLeNet's accuracy was only 1.21% lower than that of ResNet's. VGG-16's performance was much lower, with an 8.46% drop from that of ResNet's.

When compared to Experiment 1, both ResNet and GoogLeNet's accuracy increased while VGG-16's accuracy decreased. The Mendeley Leaves balanced dataset had 77 image samples per target class, while the PlantVillage dataset had around double that amount with 152 images per target class. This additional data could aid in model training and account for the 4% increase by ResNet and GoogLeNet between the two experiments.

However, the Mendeley Leaves dataset had 20 target classes compared to the 37 target classes of the PlantVillage dataset. The increase to almost double the number of target classes could possibly account as to why the VGG-16's performance decreased between the two experiments. This suggests that VGG-16 is indirectly influenced by the number of target classes required by a classification system. Therefore, VGG-16 is empirically shown as a non-ideal architecture for plant disease identification as it consistently obtained lower accuracy than ResNet and GoogLeNet and is susceptible to significant performance loss based on changes in target classes samples.

ResNet and GoogLeNet both obtain the highest accuracies with insignificant differences between the two. With accuracy as the sole metric for comparison, ResNet is considered the top performer. However, if the computational expense is considered, it may not be feasible when one has limited resources.

VII. CONCLUSION

Overall, all three state-of-the-art architectures reviewed performed well on the plant disease classification datasets used. ResNet and GoogLeNet were constant top performers, with VGG-16's accuracy being slightly lower and significantly worse when the number of target classes increased.

Both ResNet and GoogLeNet's performances increased when the dataset's size increased, suggesting they could obtain even higher accuracy with an even larger dataset.

When looking at accuracy as the sold metric for comparison, ResNet obtained the best results of 91.95% and 95.80% on the two images datasets. However, it was noted that ResNet was very computationally expensive. GoogLeNet performed almost as well with 89.35% and 94.59% accuracies on the two image datasets and appeared to be much less computationally expensive. Both architectures are good choices for plant diseases classification systems.

REFERENCES

[1] C. A. Priya, T. Balasaravanan, and A. S. Thanamani, "An efficient leaf recognition algorithm for plant classification using support vector machine," in *International conference on pattern recognition, informatics and medical engineering (PRIME-2012)*. IEEE, 2012, pp. 428–432.

[2] B. Harish, A. Hedge, O. Venkatesh, D. Spoorthy, and D. Sushma, "Classification of plant leaves using morphological features and zernike moments," in *2013 international conference on advances in computing, communications and informatics (ICACCI)*. IEEE, 2013, pp. 1827–1831.

[3] K. Singh, I. Gupta, and S. Gupta, "Svm-bdt pnn and fourier moment technique for classification of leaf shape," *International Journal of Signal Processing, Image Processing and Pattern Recognition*, vol. 3, no. 4, pp. 67–78, 2010.

[4] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," *Frontiers in plant science*, vol. 7, p. 1419, 2016.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[6] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[8] V. Cherkassky and F. M. Mulier, *Learning from data: concepts, theory, and methods*. John Wiley & Sons, 2007.

[9] P. Domingos, "A few useful things to know about machine learning," *Communications of the ACM*, vol. 55, no. 10, pp. 78–87, 2012.

[10] P. Y. Simard, D. Steinkraus, J. C. Platt *et al.*, "Best practices for convolutional neural networks applied to visual document analysis." in *Icdar*, vol. 3, no. 2003, 2003.

[11] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[12] B. D. Ripley, *Pattern recognition and neural networks*. Cambridge university press, 2007.

[13] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[14] S. S. Chouhan, U. P. Singh, A. Kaul, and S. Jain, "A data repository of leaf images: Practice towards plant conservation with plant pathology," in *2019 4th International Conference on Information Systems and Computer Networks (ISCON)*. IEEE, 2019, pp. 700–707.

[15] A. Rostamizadeh, A. Talwalkar, G. DeSalvo, K. Jamieson, and L. Li, "Efficient hyperparameter optimization and infinitely many armed bandits," 2017.