Impact of Training Sample Size on the Effects of Regularization in a Convolutional Neural Network-based Dental X-ray Artifact Prediction Model

Adam Adli ¹ and Pascal N. Tyrrell ^{*,1,2,3}

ABSTRACT

Introduction: Advances in computers have allowed for the practical application of increasingly advanced machine learning models to aid healthcare providers with diagnosis and inspection of medical images. Often, a lack of training data and computation time can be a limiting factor in the development of an accurate machine learning model in the domain of medical imaging. As a possible solution, this study investigated whether L2 regularization moderates the overfitting that occurs as a result of small training sample sizes.

Methods: This study employed transfer learning experiments on a dental x-ray binary classification model to explore L2 regularization with respect to training sample size in five common convolutional neural network architectures. Model testing performance was investigated and technical implementation details including computation times and hardware considerations as well as performance factors and practical feasibility were described.

Results: The experimental results showed a trend that smaller training sample sizes benefitted more from regularization than larger training sample sizes. Further, the results showed that applying L2 regularization did not apply significant computational overhead and that the extra rounds of training L2 regularization were feasible when training sample sizes are relatively small.

Conclusion: Overall, this study found that there is a window of opportunity in which the benefits of employing regularization can be most cost-effective relative to training sample size. It is recommended that training sample size should be carefully considered when forming expectations of achievable generalizability improvements that result from investing computational resources into model regularization.

Keywords: Convolutional neural networks, machine learning, medical imaging, dental imaging, L2 Regularization, computing, overfitting, computer vision

Affiliations

¹Department of Medical Imaging, University of Toronto, Canada

²Department of Statistical Sciences, University of Toronto, Canada

³Institute of Medical Sciences, University of Toronto, Canada

*Please direct correspondence to:

Pascal N. Tyrrell, PhD

Associate Professor

Department of Medical Imaging, University of Toronto

Phone: 416-978-7941 Email: pascal.tyrrell@utoronto.ca

Published online on December 31, 2020

INTRODUCTION

Advances in computers have allowed for the practical application of increasingly advanced machine learning models to aid healthcare providers with diagnosis and inspection of medical images. Despite the increasing possible complexity of machine learning models, the quality and size of training data are dominant factors affecting a machine learning model's performance in practice¹. Of particular interest in medical diagnostic imaging, supervised learning methods require labeled data to generate efficient models. Unfortunately, readily available human-annotated data is difficult to come by with its creation being expensive in terms of money and time^{2,3}. Data labeling requires a group of domain-expert reviewers to manually annotate available data, which is often only a fraction of the available un-labeled data. Thus, training data availability can be a bottleneck in the development of a machine learning model in the domain of medical imaging.

Noting that convolutional neural networks (CNNs) analyzing images extract an immense number of features (proportionate to image resolution) from training data, it is intuitive that the pairing of small training sets and a large number of features faced in the domain of medical imaging make overfitting an expected challenge in model development⁴. Previous literature suggests that regularization, a technique that decreases model complexity by imposing model weight penalties on the loss function, may effectively help alleviate some of the overfitting that may occur in a model trained on a small training sample size⁵⁻⁷. Acknowledging the existence of multiple regularization methods, this experiment only focused on L2 regularization which is also known as ridge regression and weight decay⁶. L2 regularization penalizes the squared sum of the model's weights in order to prevent weights from becoming too large by adding onto the loss function⁸.

$$Loss(\hat{y}, y) = LossFunction(\hat{y}, y) + \lambda \sum_{i=1}^n weight_i^2$$

Equation 1: L2 regularization of a loss function

Mathematically, the L2 regularization penalty is represented by Equation-1, where n is the number of weights in the model, $\hat{y}$ is the predicted output of input, and y is the actual label of the input.

The purpose of this study was to explore how training sample size impacts the effects of L2 regularization in a convolutional neural network-based dental x-ray artifact prediction model that employs transfer learning. While the benefits of using regularization in convolutional neural networks have

already been established in literature, this study aims to take an academic approach in providing guidance on the practical benefits and costs of its use. It is hypothesized that L2 regularization could be a significant and feasible method to improve accuracy generalizability of a convolutional neural network model. The objectives of this study were to:

1. Demonstrate existing relationships between training sample size and accuracy generalizability improvements from L2 regularization in a convolutional neural network model based on established architectures.
2. Establish real-world feasibility of employing and optimizing L2 regularization with respect to computational resources and other model testing performance metrics such as sensitivity and specificity.
3. Provide guidance by exploring a window of opportunity where employing L2 regularization can be considered most beneficial and cost-effective with respect to training sample size.
4. Outline a series of good practices for optimizing computational performance and modern hardware-usage in a Python and PyTorch development environment.

METHODS

Experiments in this study were based on a dental x-ray binary classification model that classifies whether a given dental x-ray scan has been taken on a plate with significant damage ("yes") or not ("no"). The dataset used consists of 2928 augmented phosphor-storage plate x-ray images where 1404 have "no" significant artifacts while the remaining 1524 do have ("yes") significant artifacts provided by the Department of Medical Imaging at the University of Toronto. These 2928 x-ray images were originally generated using image augmentation methods such as segmentation and superimposition on 339 phosphor-storage plates of varying degrees of damage that were provided by the Faculty of Dentistry at the University of Toronto.

Experiment Environment

Experiments were run using the standard Python-3.5.6 interpreter on a machine running Ubuntu 16.04 with a stock-clocked Intel i7-8700 processor, 32 GB of 2133 MHz DDR4 RAM, and a stock-clock Nvidia GeForce GTX 1080 Ti with 12 GB GDDR5 VRAM. Relevant hardware specifications have been tabulated as supplementary information in Table-S1.

The development tools used in this study reflect the hardware components used in the experimental environment in order to maximize computational performance where possible. Special consideration is given to CPU vectorization and CUDA acceleration.

Development Tools and Libraries

PyTorch (pip: pytorch)

This experiment based all its machine learning procedures on the procedures contained in the PyTorch library⁹. PyTorch is an open-source deep learning library for Python that provides a high-level application programming interface (API) for developing deep learning models. It provides several modules that handle essential operations such as tensor computations, data loading, neural network design, model training, and evaluation. Under the hood, the PyTorch is developed using C++ with CUDA support in order to maximize the library's performance.

Pre-trained ImageNet Models for PyTorch (pip: torchvision, pretrainedmodels)

ImageNet is a project that aims to provide a large image database containing over 14 million images for research use. The neural network architectures used in this study are implemented in the torchvision-0.2.2 and pretrainedmodels-0.7.4 Python libraries that provide classes that implement the layers of an architecture while downloading pre-trained weights from training on ImageNet.

Pillow-SIMD (pip: pillow-simd)

Pillow is a commonly used API that offers a set of standard image I/O and manipulation procedures, including per-pixel manipulations, enhancing, and filtering. Pillow was used in this experiment in order to decode TIF formatted x-ray images and to apply appropriate resizing and normalization transformations to the x-ray images as they are loaded to be processed by the CNN. The experiment implementations of this study use the Pillow-SIMD fork of the Pillow library which is compiled to include Intel's "single instruction, multiple data" instructions that provide data-level parallelism in a single thread by performing the same CPU instruction on multiple data points simultaneously in a single CPU cycle¹⁰. By using SIMD (specifically, AVX2) instructions in the Pillow implementation instead of the regular Pillow, faster image processing performance and better CPU utilization are achieved on the experiment machine's Intel i7-8700 CPU.

CNN Architectures

The five CNN architectures studied are listed as follows: Inception-V3¹¹, Inception-v4¹², Inception-ResNet-v2¹², ResNet-18 and ResNet-50¹³. These five well-established CNN architectures were studied due to their relatively smaller sizes compared to other architectures (such as the VGG family) and successes in the annual ImageNet Large Scale Visual Recognition Challenge (ILSVRC).

Further, for each of the CNN architectures studied, transfer-learning was employed. Transfer-learning is the usage of pre-trained models; previous image-understanding knowledge from training on possibly

unrelated images benefits learning to differentiate dental x-ray image artifacts. Literature suggests that transfer-learning is useful even when the previously trained model was trained in a different domain of images (i.e. non-medical) and that it allows for increased model accuracy with less available training data¹⁴. When using transfer-learning, training computation time is also significantly reduced in most cases as earlier layers of the CNN are frozen and the weights of the final layers are primarily operated on during training¹⁵. This plays an important role in making model development using lower training sample sizes in practice.

Experimental Methodology

In order to investigate relationships between training sample size and generalizability improvements from regularization in a CNN model, an experiment was designed to map the following variables for a given CNN architecture conceptually:

- **Independent variable:** Training Sample Size (n).
- **Dependent variables:** differences between model testing performance metrics when $\lambda = 0$ and $\lambda = \lambda_{\text{optimal}}$ (e.g. accuracy, specificity, sensitivity, computation time).

The algorithm designed to run this experiment is simplified to the following steps:

1. First, randomly set aside 10% of the x-ray images as a testing set for hold-out testing.
 - a. This way, holdout validation is done on the same images for all the different experimental trials.
2. Then, randomly create a training set (70%) and validation set (20%) that compose the remaining 90%, which can be subsampled. These base-sets are kept constant and sub-sampled in a predictive, ordered manner for experimental trials with different training sample sizes.
 - a. That is, if T_n and V_n are training and validation sets of size n , for $n \in N$:
 - i. $n \leq 70\% \text{ of } |\text{available data}|$
 - ii. $\forall m \in N, 0 \leq m \leq n \Leftrightarrow T_m \subseteq T_n \wedge V_m \subseteq V_n$
 - iii. $|V_n| \div |T_n| = 20\% \div 70\%$
3. Iterate through the different CNN architectures as $arch$.
4. Nested iterate through the different training sample set sizes as n .
5. Nested iterate through the different lambdas in a grid-search as λ .
6. For each ($arch, n, \lambda$) grouping of independent variables, run an experiment trial:
 - a. initialize a CNN model with the $arch$ architecture,
 - b. train based on a subsample of size n of the training set, its corresponding validation set, and L2 regularization weight decay hyper-parameter equal to λ ,
 - c. test the model using the testing set,

- d. and gather the experiment results (accuracy, sensitivity, specificity, computation time).
7. Once the triple-nested loops are done and all the data-points are gathered, return the data to be outputted into a CSV spreadsheet for analysis.

With the steps of the experimental methodology expressed as a pseudo-code procedure as supplementary data in Figure-S1, it can be seen that the data is iteratively generated for all combinations of the three independent variables (architecture '*arch*', training set size '*n*', and lambda '*λ*'). There are three important design choices to notice: the set splitting schema, validation method, and the lambda search algorithm.

This experiment followed a commonly used 70:20:10 training to validation to testing set split ratio^{16,17} as it would result in large enough available training set sizes to experiment on while providing usable amounts of testing and validation data.

With relation to the set splitting, hold-out validation was used as opposed to more thorough validation methods such as k-fold cross-validation to alleviate computation time in the experiment. Literature suggests that cross-validation may be a stronger validation method, but there is a significant computation time trade-off¹⁸. Although the dataset used in this experiment is not particularly large, the experimental procedure involves a massive amount of repeated training and testing iterations (number of architectures × number of training sets × number of lambda search queries); so, the hold-out validation is employed as its decreased computation time is essential in practice.

To implement regularization, the lambda search algorithm employed in this algorithm is a two-pass grid search. In the initial pass, a set of constant lambda values are queried for an architecture and training set pairing. Afterward, using a very basic interpolation heuristic, the three quartile values between the top two performing lambda values are also trialed. Rather than employing methods such as hyper-parameter learning and random search, a two-pass grid search is used in the procedure for experimental simplicity.

With the gathered experimental results for accuracy and computation times, regression analysis was conducted to establish any relationships between training sample size and the accuracy improvements from L2 regularization (with successful lambda search) and computation time amongst the five different CNN architectures being investigated.

RESULTS

The experiment was run for each of the CNN architectures (Inception and ResNet families), and data was collected for training sample sizes of {16, 32, 64, 128, 256, 512, 1024, 1536, and 2048} into a table. In the table containing all raw data, there are 675 rows—

one for each experiment trial (which is a combination of the three independent variables architecture '*arch*', training set size '*n*', and regularization lambda '*λ*'). Testing accuracy, sensitivity, and specificity were the dependent variables recorded in the rows for each experiment trial.

The raw data was then partitioned by architecture and training sample size pairings, and aggregate data grouped by architecture and training sample size comparing the performance of the model trained without regularization and with regularization (using the optimal lambda amongst those found) was tabulated in Table-1. In particular, a proportional difference was calculated between the accuracies, sensitivities, and specificities of the model trained with an optimal lambda and a model trained without regularization, using the following formula in Equation-2 where a performance measure $P[\lambda]$ represents accuracy, sensitivity, or specificity for a particular lambda value given a fixed architecture training sample size:

$$\left(\frac{P[\lambda = \lambda_{optimal}] - P[\lambda = 0]}{P[\lambda = 0]} \right) \times 100\%$$

Equation 2: Formula used to calculate proportional differences

In addition to proportional differences being tabulated, the lambda values were highlighted in orange in the cases where a beneficial lambda was not found for an architecture and training sample size in Table-1. Finally, there is a column that represents a Boolean value whether or not the second-pass heuristic in the lambda search algorithm resulted in the optimal lambda's discovery in the experiment denoted as "Heuristic Find?".

To better visualize the data and any trends that may exist in the data particularly for the accuracy, the proportional accuracy improvements for the different architectures were plotted using scatter plots for each architecture in Figures 1-5. In each of the plots, power regression was used to fit a power formula to capture the trend between training set size and proportional testing accuracy improvements. Cases where a beneficial lambda value was not found (and thus there was no accuracy improvement) were considered as outliers and thus not considered for any trends. Through experimentation, power regression was deemed to be the best fit in comparison to linear and logarithm regression; polynomial regression was not considered in order to fit a monotonic trend function.

The power regression trend line information from the plots for each architecture was tabulated in Table-2. While the power regression trends between proportional accuracy improvements and training set size fitted very nicely for the experiments run using three CNN architectures. Inception-V3, Inception-

Aggregate Optimal Lambda Testing Performance Comparisons for Architecture-Training Sample Size Experiment Trials								
Arch	N	λ_{optimal}	Accuracy		Proportion al Δ Accuracy	Proportion al Δ Sensitivity	Proportion al Δ Specificity	Heuristic Find?
			$\lambda = 0$	$\lambda = \lambda_{\text{optimal}}$				
inception_v3	2048	0.01	0.842465753	0.839041096	-0.41%	-2.38%	1.67%	FALSE
	1536	0.5	0.832191781	0.839041096	0.82%	0.00%	1.68%	FALSE
	1024	0.001	0.835616438	0.842465753	0.82%	-1.60%	3.36%	FALSE
	512	0.0001	0.808219178	0.828767123	2.54%	-2.34%	8.33%	FALSE
	256	0.0001	0.726027397	0.746575342	2.83%	-6.48%	12.50%	FALSE
	128	0.5	0.602739726	0.650684932	7.95%	-16.03%	77.78%	FALSE
	64	0.75	0.455479452	0.523972603	15.04%	-18.18%	350.00%	FALSE
	32	0.03125	0.506849315	0.630136986	24.32%	95.92%	-11.11%	TRUE
inception_v4	16	1.00E-05	0.565068493	0.54109589	-4.24%	-66.29%	68.42%	FALSE
	2048	0.05	0.815068493	0.839041096	2.94%	5.04%	0.84%	FALSE
	1536	0.025	0.815068493	0.842465753	3.36%	2.40%	4.42%	FALSE
	1024	0.5	0.811643836	0.845890411	4.22%	-0.81%	4.42%	FALSE
	512	0.75	0.797945205	0.815068493	2.15%	-0.82%	5.41%	FALSE
	256	1.00E-05	0.760273973	0.77739726	2.25%	8.82%	-3.33%	FALSE
	128	0.000775	0.688356164	0.712328767	3.48%	-2.94%	10.10%	TRUE
	64	0.25075	0.530821918	0.647260274	21.94%	5.68%	43.28%	TRUE
inception_resnet_v2	32	1.00E-05	0.517123288	0.619863014	19.87%	-13.04%	47.56%	FALSE
	16	0.0001	0.51369863	0.609589041	18.67%	217.24%	-28.93%	TRUE
	2048	0.075	0.815068493	0.839041096	2.94%	5.08%	0.83%	TRUE
	1536	0.0775	0.821917808	0.845890411	2.92%	4.10%	1.69%	TRUE
	1024	0.001	0.825342466	0.849315068	2.90%	2.48%	3.33%	FALSE
	512	1.00E-05	0.825342466	0.825342466	0.00%	2.42%	-2.56%	FALSE
	256	0.5	0.746575342	0.818493151	9.63%	0.00%	20.19%	FALSE
	128	0.38	0.72260274	0.787671233	9.00%	18.18%	-0.99%	TRUE
resnet18	64	0.05	0.534246575	0.729452055	36.54%	-7.75%	485.71%	FALSE
	32	0.05	0.561643836	0.647260274	15.24%	327.78%	-23.29%	FALSE
	16	0.025	0.48630137	0.654109589	34.51%	-21.58%	2633.33%	FALSE
	2048	0.5	0.811643836	0.845890411	4.22%	0.00%	8.70%	FALSE
	1536	0.75	0.825342466	0.845890411	2.49%	4.07%	0.85%	FALSE
	1024	0.75	0.832191781	0.842465753	1.23%	1.65%	0.82%	FALSE
	512	1	0.79109589	0.842465753	6.49%	2.78%	9.76%	FALSE
	256	0.625	0.732876712	0.743150685	1.40%	25.00%	-11.59%	TRUE
resnet50	128	0.187575	0.643835616	0.746575342	15.96%	20.21%	11.70%	TRUE
	64	0.75	0.571917808	0.613013699	7.19%	-59.12%	310.00%	FALSE
	32	1	0.534246575	0.684931507	28.21%	-40.85%	728.57%	FALSE
	16	0.0001	0.565068493	0.619863014	9.70%	322.22%	-51.45%	FALSE
	2048	0.575	0.821917808	0.835616438	1.67%	-3.20%	6.96%	TRUE
	1536	0.1	0.832191781	0.839041096	0.82%	1.63%	0.00%	FALSE
	1024	0.75	0.832191781	0.849315068	2.06%	1.64%	2.48%	FALSE
	512	0.1	0.845890411	0.859589041	1.62%	6.84%	-3.08%	FALSE
resnet101	256	1.00E-05	0.698630137	0.760273973	8.82%	41.67%	-4.86%	FALSE
	128	0.38125	0.729452055	0.773972603	6.10%	17.20%	-2.50%	TRUE
	64	1	0.739726027	0.691780822	-6.48%	-7.50%	-5.21%	FALSE
	32	0.1	0.547945205	0.667808219	21.88%	-38.84%	210.26%	FALSE
	16	0.05	0.510273973	0.712328767	39.60%	∞	-18.79%	FALSE

Table 1 Aggregate Optimal Lambda Testing Performance Comparisons for Architecture-Training Sample Size Experiment Trials

ResNet-V2, and ResNet-50 all fit the power regression trend with R^2 values beyond 0.87; in fact, an almost perfect fit with an R^2 value of 0.97 is observed for the Inception-V3 experiments. It is also observed that a poorer power regression fit is found in the Inception-V4 and ResNet-18 experiments. Despite the varying line fits, all the architectures show lower general accuracy improvement from regularization and they all exhibit sporadic behavior with too small a training sample size. Beyond machine learning model performance, training runtime was also recorded during experimentation and tabulated in the raw data table. Similarly, the computation times have been aggregated by architecture and training sample size as supplementary information in

Table-S2.

Overall, the experiment consisting of 675 trials (5 architectures $\times$ 9 training sample sizes $\times$ 15 searched lambda values) took 24 hours, 39 minutes, and 55 seconds (for a total of 88795.28121 seconds) to compute.

DISCUSSION

Overall, data gathered in the experiment showed that in the cases where an optimal lambda could be found, regularization did benefit models trained on smaller training sets more than larger ones generally. Additionally, there was an observed

Figures 1-5 Proportional Testing Accuracy Improvement from Optimal Regularization for Training Set Sizes by Architecture

distinction between the behavior of regularization in the different architectures with the observed regression lines having varying R^2 fits. Finally, computation times for any given experimental training sample size and neural network architecture did not vary significantly with the employment of regularization and choice of lambda.

Generalizability Improvements with Respect to Training Sample Size

Concerning the trends observed in the experimental results, two proposed patterns describe the relationship between accuracy improvement from regularization and training sample size in the data for

all the architectures studied:

- 1. Minimum Information Threshold:** Having too small a training sample size below a threshold results in erratic regularization behavior, and
- 2. Diminishing Returns:** Generally, larger training sample sizes exhibit less proportionate accuracy improvements from regularization if an optimal lambda value is found.

The first pattern was expected when designing the experiment in terms of training sample sizes. Regularization prevents the weights in a CNN model from growing too large; this, in theory, prevents a model from overfitting to the training data⁵⁻⁷. Although

regularization may have prevented the model from overfitting to the training data, the extremely small amount of training data resulted in a lack of information. Since transfer learning was employed in these cases, the models were being tested on their previous knowledge from ImageNet with very little learning done on the domain of dental x-ray artifacts resulting in sporadic testing performance. The information threshold pattern can be seen in all the architectures' experimental trials, but this behavior is most pronounced in the ResNet-18 experiment trials as suggested by the poor regression fit between proportional testing accuracy improvement and training sample size and the greatest number of local minima in the relation between those two variables.

Beyond the sporadic behavior of small training sample sizes, the second pattern of larger training sample sizes exhibiting smaller proportionate improvements in accuracy was also an expected result of the experiment; every architecture studied in this experiment exhibited decreasing accuracy improvements relative to training sample size. This relationship between the effectiveness of regularization in terms of generalization and training sample can be rationalized through foundational theory in machine learning. Disregarding other factors that affect a machine learning model's testing performance, it is theoretically accepted that training on a small data sample is expected to result in more overfitting (and poorer generalizability) in comparison to training on a larger data sample¹⁹. Noting that regularization helps to prevent overfitting (and improve generalizability), then it can be concluded that a machine learning model that's more heavily over-fitted as a result of a small training sample size can be expected to exhibit a more pronounced improvement in its generalizability (which is measured as testing accuracy in this study) through the employment of regularization.

With these two observed patterns in mind, the once form a theoretical window of opportunity with respect to training sample size in which the real-world benefit from regularization can be maximized. When observing the aggregate performance comparison with respect to training sample size, the different architectures exhibited different regularization behaviors. Regardless, all the architectures show lower general accuracy improvement from regularization with increasing training sample sizes. Thus, a recommendation is given that special consideration should be given in the usage of regularization on a model-by-model basis, with realistic expectations (derived from the training sample size) regarding potential generalizability improvements that can be achieved from invested computation time.

Sensitivity and Specificity

Despite the models in the experiment achieving a better overall testing accuracy from regularization

when a beneficial lambda value was found, accuracy by itself may not be a sufficient performance measure in certain real-world machine learning model applications. Sensitivity (the true positive rate) and specificity (the true negative rate) are also significant performance measures, especially in medicine. No particular pattern can be uncovered for testing sensitivity and specificity when applying L2 regularization on different training sample sizes. The lack of an underlying pattern for sensitivity and specificity can be justified, as no measures were taken to treat these metrics in this study (although sensitivity-driven regularization methods exist in literature)²⁰.

Computational Performance and Efficiency of Regularization

While a machine learning model's testing performance is an important consideration, the available computational resources and the expected computational cost of model development are also important considerations in practice. The experimental computation times gathered in this study showed very little variation with regularization as seen in the standard deviations. As expected, the results show that training a model on less data requires a less computation time.

Overall, each architecture had differing average computation times with respect to training sample size, but they had one important statistic in common: a negligible variability of training time with respect to lambda choice and regularization. From this, it can be empirically and practically concluded that the addition of the L2 regularization to the loss function in the PyTorch implementation can be considered a computationally cheap operation. The most significant expense of employing L2 regularization is in tuning and searching for an optimal lambda hyperparameter value as it requires repeated model training.

When pairing diminishing performance improvements from applying regularization on models trained on larger amounts of data and the increased computation time of training on larger datasets, it is realized that regularization may not be an efficient means of improving model performance when lack of training data is not a significantly limited resource. More importantly, this highlights the trade-off between two important resources: computation and data. When training data is limited, computational resources could be used in an attempt to accommodate a model's shortcomings through regularization and hyperparameter tuning. Contrarily, if training data is in excess, it may be inefficient to employ regularization and hyperparameter tuning to the same extent.

Limitations

Although this study managed to show a trend between training sample sizes and the effects of L2 regularization on different CNN architectures, some limitations introduce uncertainty in the data. Primarily,

Power Regression Fits		
Architecture	Formula	R ²
inception_v3	$6.4073x^{-0.926}$	0.9773
inception_v4	$0.6403x^{-0.447}$	0.6338
inception_resnet_v2	$1.5926x^{-0.539}$	0.8714
resnet18	$0.509x^{-0.414}$	0.4601
resnet50	$2.815x^{-0.734}$	0.9074

Table 2 Power Regression Fits

they involve the (lack of an) optimal lambda search, dataset sampling, and the granularity of gathered data points. Despite the discovery of a few good performing lambdas, there are a few trials in the experiment where a beneficial, let alone optimal, lambda value was not found. More thorough lambda search algorithms (such as hyperparameter training and randomized search) were not explored in the scope of this study. In addition to the innumerable possible lambda values, there is also an innumerable number of dataset-split combinations; if the dataset has a high variance there could be inconsistent and non-robust experimental results. Beyond lambda and dataset sampling options, the number of actual data points gathered in an experiment are also limited due to practical constraints on computation time (it would rarely be feasible to compute data for ~2000 training sample sizes from {16...2048}). All described limitations exist mainly due to much larger running time costs in their solutions was greater than the available computational time and resources for this experiment.

Future Work

While this study managed to achieve its objectives, with the primary being that it showed an association between generalizability of improvements from L2 regularization and training sample size for different convolutional neural network architectures, future work should consider different experimental conditions (such as using more standardized datasets, cross-validation, hyper-parameter tuning, repeated trials, etc.) to validate the observed trend between testing accuracy improvement and training sample size. Additionally, it should consider exploring other regularization methods (such as L1 lasso regularization) to show if they follow the same trend that is theorized in this study. Finally, a distinction between sensitivity and specificity should be made when exploring regularization methods like sensitivity-based regularization is suggested.

CONCLUSION

When limited training data is available, our study demonstrated that regularization was a feasible measure to employ to enhance a convolution neural

network's generalizability through testing accuracy. Regularization was not the most efficient use of computational resources when training data was not a significant limiting factor for a model's testing performance. Experiments revealed a minimum information threshold paired with the existence of diminishing returns forming a window of opportunity in which employing regularization can be beneficial with respect to training sample size. Thus, it is recommended that training sample size should be carefully considered when forming expectations of achievable generalizability improvements that result from investing computational resources into model regularization.

ACKNOWLEDGEMENTS

This research was supported by the MiDATA lab, the University of Toronto Medical Imaging Department's Data Science Unit. I would like to express my appreciation to my colleagues at the MiDATA lab Ahmed Sigiuk, Ariana Bermúdez, and Mauro Mendez for their consistent support and advice in the experimental development.

COMPETING INTERESTS

No competing interests declared.

Received: March 29, 2020

Accepted: December 28, 2020

Published online: December 31, 2020

No publicly available datasets were used in this study.

Supplementary information available at
<https://doi.org/10.33137/juls.v14i1.35883>

REFERENCES

1. Kohli, M. D., Summers, R. M. & Geis, J. R. Medical Image Data and Datasets in the Era of Machine Learning—Whitepaper from the 2016 C-MIMI Meeting Dataset Session. *J. Digit. Imaging* 30, 392–399 (2017) doi:10.1007/s10278-017-9976-3.
2. Yamashita, R., Nishio, M., Do, R. K. G. & Togashi, K. Convolutional

- neural networks: an overview and application in radiology. *Insights Imaging* 9, 611–629 (2018) doi:10.1007/s13244-018-0639-9.
3. Figueroa, R. L., Zeng-Treitler, Q., Kandula, S. & Ngo, L. H. Predicting sample size required for classification performance. *BMC Med. Inform. Decis. Mak.* 12, 8 (2012) doi:10.1186/1472-6947-12-8.
 4. Zheng, Q., Yang, M., Yang, J., Zhang, Q. & Zhang, X. Improvement of Generalization Ability of Deep CNN via Implicit Regularization in Two-Stage Training Process. *IEEE Access* 6, 15844–15869 (2018) doi:10.1109/ACCESS.2018.2810849.
 5. Demir-Kavuk, O., Kamada, M., Akutsu, T. & Knapp, E.-W. Prediction using step-wise L1, L2 regularization and feature selection for small data sets with large number of features. *BMC Bioinformatics* 12, 412 (2011) doi:10.1186/1471-2105-12-412.
 6. Connect, a. K., Krogh, A. & Hertz, J. A. A Simple Weight Decay Can Improve Generalization. *Adv. Neural Inf. Process. Syst.* 4, 950–957 (1992).
 7. Wu Yan & Zhang Liming. The effect of initial weight, learning rate and regularization on generalization performance and efficiency. in 6th International Conference on Signal Processing, 2002. vol. 2 1191–1194 (IEEE, 2002). doi:10.1109/ICOSP.2002.1180003.
 8. Tuia, D., Flamary, R. & Barlaud, M. To be or not to be convex? A study on regularization in hyperspectral image classification. in 2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS) vols 2015-Novem 4947–4950 (IEEE, 2015). doi:10.1109/IGARSS.2015.7326942.
 9. Paszke, A. et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. in *Advances in Neural Information Processing Systems* (eds. Wallach, H. et al.) vol. 32 8026–8037 (Curran Associates, Inc., 2019).
 10. Kristof, P., Yu, H., Li, Z. & Tian, X. Performance Study of SIMD Programming Models on Intel Multicore Processors. in 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum 2423–2432 (IEEE, 2012). doi:10.1109/IPDPSW.2012.299.
 11. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. & Wojna, Z. Rethinking the Inception Architecture for Computer Vision. in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) vols 2016-Decem 2818–2826 (IEEE, 2016). doi:10.1109/CVPR.2016.308.
 12. Szegedy, C., Ioffe, S., Vanhoucke, V. & Alemi, A. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. 31st AAAI Conf. Artif. Intell. AAAI 2017 4278–4284 (2016).
 13. He, K., Zhang, X., Ren, S. & Sun, J. Deep Residual Learning for Image Recognition. in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) vols 2016-Decem 770–778 (IEEE, 2016). doi:10.1109/CVPR.2016.90.
 14. Pan, S. J. & Yang, Q. A Survey on Transfer Learning. *IEEE Trans. Knowl. Data Eng.* 22, 1345–1359 (2010) doi:10.1109/TKDE.2009.191.
 15. Kunze, J. et al. Transfer Learning for Speech Recognition on a Budget. *Proc. 2nd ACL Work. Represent. Learn. NLP* 168–177 (2017).
 16. Vang, Y. S., Chen, Z. & Xie, X. Deep Learning Framework for Multi-class Breast Cancer Histology Image Classification. in *Lecture Notes in Computer Science* (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) vol. 10882 LNCS 914–922 (Springer Verlag, 2018). doi:10.1007/978-3-319-93000-8_104.
 17. Pang, B. C. et al. Hybrid Outcome Prediction Model for Severe Traumatic Brain Injury. *J. Neurotrauma* 24, 136–146 (2007) doi:10.1089/neu.2006.0113.
 18. Yadav, S. & Shukla, S. Analysis of k-Fold Cross-Validation over Hold-Out Validation on Colossal Datasets for Quality Classification. in 2016 IEEE 6th International Conference on Advanced Computing (IACC) 78–83 (IEEE, 2016). doi:10.1109/IACC.2016.25.
 19. Riley, R. D. et al. Minimum sample size for developing a multivariable prediction model: PART II - binary and time-to-event outcomes. *Stat. Med.* 38, 1276–1296 (2019) doi:10.1002/sim.7992.
 20. Tartaglione, E., Lepsøy, S., Fiandrotti, A. & Francini, G. Learning Sparse Neural Networks via Sensitivity-Driven Regularization. *Adv. Neural Inf. Process. Syst.* 2018-December, 3878–3888 (2018).