The unreasonable effectiveness of the final batch normalization layer

Kocaman, V.; Shir, O.M.; Bäck, T.H.W.; Bebis, G.; Athitsos, V.

The Unreasonable Effectiveness of the Final Batch Normalization Layer

Veysel Kocaman[1] **Ofer M. Shir**[2] **Thomas Bäck**[1]

[1]LIACS, Leiden University, Leiden, The Netherlands
[2]Computer Science Department, Tel-Hai College and Migal Institute, Upper Galilee, Israel

Abstract

Early-stage disease indications are rarely recorded in real-world domains, such as Agriculture and Healthcare, and yet, their accurate identification is critical in that point of time. In this type of highly imbalanced classification problems, which encompass complex features, deep learning (DL) is much needed because of its strong detection capabilities. At the same time, DL is observed in practice to favor majority over minority classes and consequently suffer from inaccurate detection of the targeted early-stage indications. In this work, we extend the study done by [Kocaman et al., 2020], showing that the final BN layer, when placed before the softmax output layer, has a considerable impact in highly imbalanced image classification problems as well as undermines the role of the softmax outputs as an uncertainty measure. This current study addresses additional hypotheses and reports on the following findings: (i) the performance gain after adding the final BN layer in highly imbalanced settings could still be achieved after removing this additional BN layer in inference; (ii) there is a certain threshold for the *imbalance ratio* upon which the progress gained by the final BN layer reaches its peak; (iii) the batch size also plays a role and affects the outcome of the final BN application; (iv) the impact of the BN application is also reproducible on other datasets and when utilizing much simpler neural architectures; (v) the reported BN effect occurs only per a single majority class and multiple minority classes – i.e., no improvements are evident when there are two majority classes; and finally, (vi) utilizing this BN layer with sigmoid activation has almost no impact when dealing with a strongly imbalanced image classification tasks.

1 INTRODUCTION

Detecting anomalies that are hardly distinguishable from the majority of observations is a challenging task that often requires strong learning capabilities since anomalies appear scarcely, and in instances of diverse nature, a labeled dataset representative of all forms is typically unattainable. Despite tremendous advances in computer vision and object recognition algorithms in the past few years, their effectiveness remains strongly dependent upon the datasets' size and distribution, which are usually limited under real-world settings. This work is mostly concerned with hard classification problems at early-stages of abnormalities in certain domains (i.e. crop, human diseases, chip manufacturing), which suffer from lack of data instances, and whose effective treatment would make a dramatic impact in these domains. For instance, fungus's visual cues on crops in agriculture or early-stage malignant tumors in the medical domain are hardly detectable in the relevant time-window, while the highly infectious nature leads rapidly to devastation in a large scale. Other examples include detecting the faults in chip manufacturing industry, automated insulation defect detection with thermography data, assessments of installed solar capacity based on earth observation data, and nature reserve monitoring with remote sensing and deep learning. However, class imbalance poses an obstacle when addressing each of these applications.

In recent years, reliable models capable of learning from small samples have been obtained through various approaches, such as autoencoders [Beggel et al., 2019], class-balanced loss (CBL) to find the effective number of samples required [Cui et al., 2019], fine tuning with transfer learning [Hussain et al., 2018], data augmentation [Shorten and Khoshgoftaar, 2019], cosine loss utilizing (replacing categorical cross entropy) [Barz and Denzler, 2020], or prior knowledge [Lake et al., 2015].

[Kocaman et al., 2020] presented an effective modification in the neural network architecture, with a surprising simplicity and without a computational overhead, which en-

Accepted for the 16th International Symposium on Visual Computing (ISVC 2021).

ables a substantial improvement using a smaller number of anomaly samples in the training set. The authors empirically showed that the final BN layer before the softmax output layer has a considerable impact in highly imbalanced image classification problems. They reported that under artificially-generated skewness of 99% vs. 1% in the PlantVillage (PV) image dataset [Mohanty et al., 2016], the initial F1 test score increased from the 0.29-0.56 range to the 0.95-0.98 range (almost triple) for the minority class when BN modification applied. They also argued that, a model might perform better even if it is not confident enough while making a prediction, hence the softmax output may not serve as a good uncertainty measure for DNNs (see Figure 1).

This shows that DNNs have the tendency of becoming 'overconfident' in their predictions during training, and this can reduce their ability to generalize and thus perform as well on unseen data. In addition, large datasets can often comprise incorrectly labeled data, meaning inherently the DNN should be a bit skeptical of the 'correct answer' to avoid being overconfident on bad answers. This was the main motivation of Müller et al. [Müller et al., 2019] for proposing the *label smoothing*, a loss function modification that has been shown to be effective for training DNNs. Label smoothing encourages the activations of the penultimate layer to be close to the template of the correct class and equally distant to the templates of the incorrect classes [Müller et al., 2019]. Despite its relevancy, [Kocaman et al., 2020] also reports that label smoothing did not do well in their study as previously mentioned by Kornblith et al. [Chelombiev et al., 2019] who demonstrated that label smoothing impairs the accuracy of transfer learning, which similarly depends on the presence of non-class-relevant information in the final layers of the network.

In this study, we extend previous efforts done by [Kocaman et al., 2020] to devise an effective approach to enable learning of minority classes, given the surprising evidence of applying the final Batch Normalization (BN) layer.

Given these recent findings, we formulate and test additional hypotheses and report our observations in what follows. The concrete contributions of this paper are the following:

- The performance gain after adding the final BN layer in highly imbalanced settings could still be achieved after removing this additional BN layer during inference; in turn enabling us to get a performance boost with no additional cost in production.
- There is a certain threshold for the ratio of the imbalance for this specific PV dataset, upon which the progress is the most obvious after adding the final BN layer.
- The batch size also plays a role and significantly affects the outcome.
- We replicated the similar imbalanced scenarios in MNIST dataset, reproduced the same BN impact, and furthermore demonstrated that the final BN layer has a considerable impact not just in modern CNN architectures but also in simple CNNs and even in one-layered feed-forward fully connected (FC) networks.
- We illustrate that the performance gain occurs only when there is a single majority class and multiple minority classes; and no improvement observed regardless of the final BN layer when there are two majority classes.
- We argue that using the final BN layer with sigmoid activation has almost no impact when dealing with a strongly imbalanced image classification tasks.

The remainder of the paper is organized as follows: Section 2 gives some background concerning the role of the BN layer in neural networks. Section 3 summarizes the previous findings and existing hypotheses in the previous work done by [Kocaman et al., 2020] and then lists the derived hypotheses that will be addressed throughout this study. Section 4 elaborates the implementation details and settings for our new experiments and presents results. Section 5 discusses the findings and proposes possible *mechanistic* explanations. Section 6 concludes this paper by pointing out key points and future directions.

2 BACKGROUND

In order to better understand the novel contributions of this study, in this section, we give some background information about the Batch Normalization (BN) [Ioffe and Szegedy, 2015] concept. Since the various applications of BN in similar studies and related work have already been investigated thoroughly in our previous work [Kocaman et al., 2020], we will only focus on the fundamentals of BN in this chapter.

Training deep neural networks with dozens of layers is challenging as the networks can be sensitive to the initial random weights and configuration of the learning algorithm. One possible reason for this difficulty is that the distribution of the inputs to layers deep in the network may change after each mini-batch when the weights are updated. This slows down the training by requiring lower learning rates and careful parameter initialization, makes it notoriously hard to train models with saturating nonlinearities [Ioffe and Szegedy, 2015], and can cause the learning algorithm to forever chase a moving target. This change in the distribution of inputs to layers in the network is referred to by the technical name *"internal covariate shift"* (ICS).

BN is a widely adopted technique that is designed to combat ICS and to enable faster and more stable training of deep neural networks (DNNs). It is an operation added to the model before activation which normalizes the inputs and then applies learnable scale (γ) and shift (β) parameters

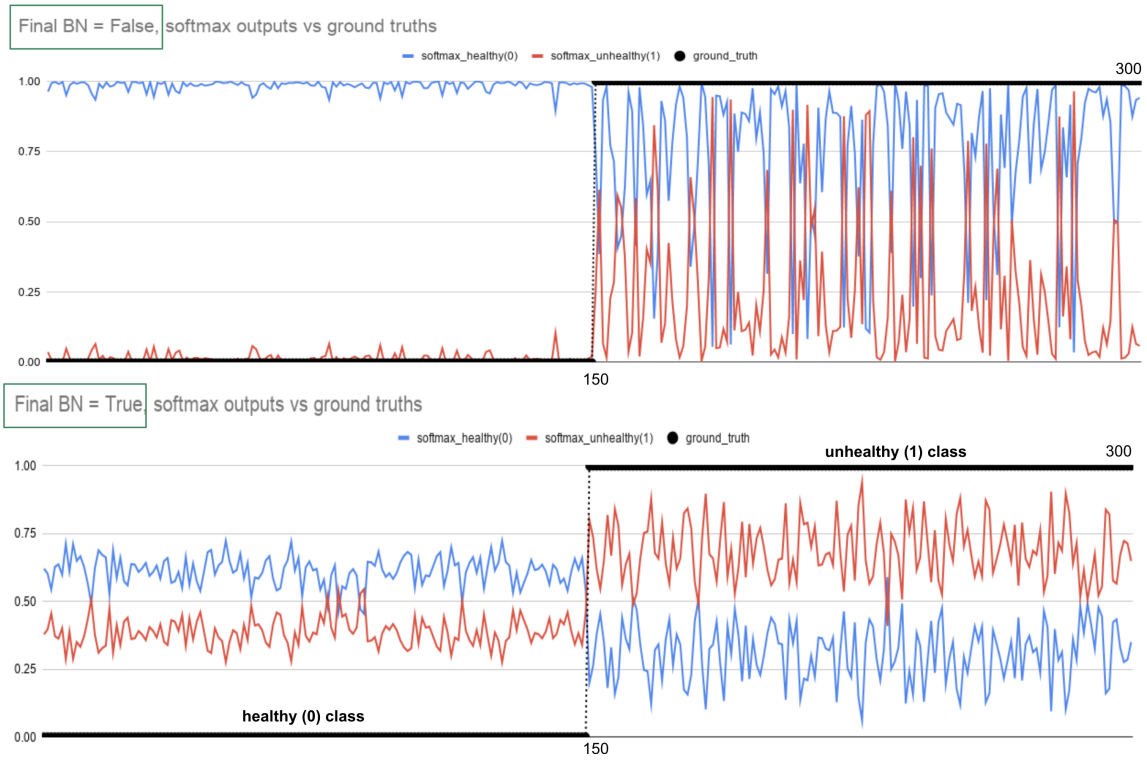


Figure 1: The x-axis represents the ground truth for all 150 healthy (0) and 150 unhealthy (1) images in the test set while red and blue lines represent final softmax output values between 0 and 1 for each image. **Top chart (without final BN):** When ground truth (black) is class = 0 (healthy), the softmax output for class = 0 is around 1.0 (blue, predicting correctly). But when ground truth (black) is class = 1 (unhealthy), the softmax output for class = 1 (red points) changes between 0.0 and 1.0 (mostly below 0.5, NOT predicting correctly). **Bottom chart (with final BN):** When ground truth (black) is class = 0 (healthy), the softmax output is between 0.5 and 0.75 (blue, predicting correctly). When ground truth (black) is class = 1 (unhealthy), the softmax output (red points) changes between 0.5 and 1.0 (mostly above 0.5, predicting correctly).

to preserve model performance. Given m activation values $x_1 \ldots, x_m$ from a mini-batch $\mathscr{B}$ for any particular layer input $x^{(j)}$ and any dimension $j \in \{1, \ldots, d\}$, the transformation uses the mini-batch mean $\mu_{\mathscr{B}} = 1/m \sum_{i=1}^{m} x_i$ and variance $\sigma^2_{\mathscr{B}} = 1/m \sum_{i=1}^{m} (x_i - \mu_{\mathscr{B}})^2$ for normalizing the x_i according to $\hat{x}_i = (x_i - \mu_{\mathscr{B}})/\sqrt{\sigma^2_{\mathscr{B}} + \varepsilon}$ and then applies the scale and shift to obtain the transformed values $y_i = \gamma \hat{x}_i + \beta$. The constant $\varepsilon > 0$ assures numerical stability of the transformation.

BN has the effect of stabilizing the learning process and dramatically reducing the number of training epochs required to train deep networks; and using BN makes the network more stable during training. This may require the use of much larger learning rates, which in turn may further speed up the learning process.

Though BN has been around for a few years and has become common in deep architectures, it remains one of the DL concepts that is not fully understood, having many studies discussing why and how it works. Most notably, Santurkar et al. [Santurkar et al., 2018] recently demonstrated that such distributional stability of layer inputs has little to do with the success of BN and the relationship between ICS and BN is tenuous. Instead, they uncovered a more fundamental impact of BN on the training process: it makes the optimization landscape significantly smoother. This smoothness induces a more predictive and stable behavior of the gradients, allowing for faster training. Bjorck et al. [Bjorck et al., 2018] also makes similar statements that the success of BN can be explained without ICS. They argue that being able to use larger learning rate increases the implicit regularization of the gradient, which improves generalization.

Even though BN adds an overhead to each iteration (estimated as additional 30% computation [Mishkin and Matas, 2015]), the following advantages of BN outweigh the overhead shortcoming:

- It improves gradient flow and allows training deeper models (e.g., ResNet).
- It enables using higher learning rates because it eliminates outliers' activation, hence the learning process

may be accelerated using those high rates.

- It reduces the dependency on initialization and then reduces overfitting due to its minor regularization effect. Similarly to dropout, it adds some noise to each hidden layer's activation.
- Since the scale of input features would not differ significantly, the gradient descent may reduce the oscillations when approaching the optimum and thus converge faster.
- BN reduces the impacts of earlier layers on the following layers in DNNs. Therefore, it takes more time to train the model to converge. However, the use of BN can reduce the impact of earlier layers by keeping the mean and variance fixed, which in some way makes the layers independent from each other. Consequently, the convergence becomes faster.

3 PREVIOUS FINDINGS AND EXISTING HYPOTHESES

In the work done by [Kocaman et al., 2020], the authors focused their efforts on the role of BN layer in DNNs, where in the first part of the experiments ResNet34 [Simonyan and Zisserman, 2014] and VGG19 CNN architectures [He et al., 2016] are utilized. They first addressed the complete PV original dataset and trained a ResNet34 model for 38 classes. Using scheduled learning rates [Smith, 2017], they obtained 99.782% accuracy after 10 epochs – slightly improving the PV project's record of 99.34% when employing GoogleNet [Mohanty et al., 2016]. In what follows, we summarize the previous observations borrowed from [Kocaman et al., 2020] and then formulate the derived hypotheses that became the core of the current study.

3.1 ADDING A FINAL BATCH NORM LAYER BEFORE THE OUTPUT LAYER

By using the imbalanced datasets for certain plant types (1,000/10 in the training set, 150/7 in the validation set and 150/150 in the test set), [Kocaman et al., 2020] performed several experiments with the VGG19 and ResNet34 architectures. The selected plant types were Apple, Pepper and Tomato - being the only datasets of sufficient size to enable the 99%-1% skewness generation. All the tests are run with batch size 64.

In order to fine-tune the network for the PV dataset, the final classification layer of CNN architectures is replaced by Adaptive Average Pooling (AAP), BN, Dropout, Dense, ReLU, BN and Dropout followed by the Dense and BN layer again. The last layer of an image classification network is often a FC layer with a hidden size being equal to the number of labels to output the predicted confidence scores that are normalized by the softmax operator to obtain predicted probabilities. In their implementation, they added another 2-input BN layer after the last dense layer (before softmax output) in addition to existing BN layers in the tail and 4 BN layers in the head of the DL architecture (e.g., ResNet34 possesses a BN layer after each convolutional layer, having altogether 38 BN layers given the additional 4 in the head).

At first they run experiments with VGG19 architectures for selected plant types by adding the final BN layer. When they train this model for 10 epochs and repeat this for 10 times, they observed that the F1 test score is increased from 0.2942 to 0.9562 for unhealthy Apple, from 0.7237 to 0.9575 for unhealthy Pepper and from 0.5688 to 0.9786 for unhealthy Tomato leaves. They also achieved significant improvements in healthy samples (being the majority in the training set). See Table 1 for details.

3.2 EXPERIMENTATION ON PLANTVILLAGE DATASET SUBJECT TO DIFFERENT CONFIGURATIONS

Using the following six configuration variations with two options each, the authors created 64 different configurations which they tested with ResNet34 (training for 10 epochs only): Adding (✓) a final BN layer just before the output layer (BN), using (✓) weighted cross-entropy loss [Goodfellow et al., 2016] according to class imbalance (WL), using (✓) data augmentation (DA), using (✓) mixup (MX) [Zhang et al., 2017], unfreezing (✓) or freezing (learnable vs pretrained weights) the previous BN layers in ResNet34 (UF), and using (✓) weight decay (WD) [Krogh and Hertz, 1992]. Checkmarks (✓) and two-letter abbreviations are used in Table 2 to denote configurations. When an option is disabled across all configurations, its associated column is dropped.

As shown in Table 2, just adding the final BN layer was enough to get the highest F1 test score in both classes. Surprisingly, although there is already a BN layer after each convolutional layer in the backbone CNN architecture, adding one more BN layer just before the output layer boosts the test scores. Notably, the 3rd best score (average score for configuration 31 in Table 2) is achieved just by adding a single BN layer before the output layer, even without unfreezing the previous BN layers.

One of the important observations is that the model without the final BN layer is pretty confident even if it predicts falsely. But the proposed model with the final BN layer predicts correctly even though it is less confident. They basically ended up with less confident but more accurate models in less than 10 epochs. The classification probabilities for five sample images from the unhealthy class (class = 1) with final BN layer (right column) and without final BN layer (left column) are shown in Table 3. As explained above, without the final BN layer, these anomalies

Table 1: Averaged **F1 test set** performance values over 10 runs, alongside BN's total improvement, using 10 epochs with VGG19, with/without BN and with Weighted Loss (WL) without BN.

plant	class	without final BN	with WL (no BN)	with final BN (no WL)	BN total improvement
Apple	Unhealthy	**0.2942**	**0.7947**	**0.9562**	**0.1615**
	Healthy	0.7075	0.8596	0.9577	0.0981
Pepper	Unhealthy	**0.7237**	**0.8939**	**0.9575**	**0.0636**
	Healthy	0.8229	0.9121	0.9558	0.0437
Tomato	Unhealthy	**0.5688**	**0.8671**	**0.9786**	**0.1115**
	Healthy	0.7708	0.9121	0.9780	0.0659

Table 2: Best performance metrics over the Apple dataset under various configurations using ResNet34.

Class	Config Id	Test set precision	Test set recall	Test set F1-score	Epoch	BN	DA	UF	WD
Unhealthy	**31**	0.9856	0.9133	**0.9481**	6	✓			
(class = 1)	23	0.9718	0.9200	0.9452	6	✓	✓		
	20	0.9926	0.8933	0.9404	7	✓	✓	✓	✓

are all falsely classified (recall that $\mathscr{P}_{\text{softmax}}(\text{class} = 0) = 1 - \mathscr{P}_{\text{softmax}}(\text{class} = 1)$).

Table 3: Softmax output values (representing class probabilities) for five sample images of unhealthy plants. Left column: Without final BN layer, softmax output values for unhealthy, resulting in a wrong classification in each case. Right column: With final BN layer, softmax output value for unhealthy, resulting in correct but less "confident" classifications.

Without final BN layer	With final BN layer
0.1082	0.5108
0.1464	0.6369
0.1999	0.6082
0.2725	0.6866
0.3338	0.7032

3.3 DERIVED HYPOTHESES

Under all these observations and findings mentioned above, we derived the following hypotheses for our new study:

- The added complexity to the network by adding the final BN layer could be eliminated by removing the final BN layer in inference without compromising the performance gain achieved.
- There might be a certain level of skewness upon which the progress reaches its peak without further sizing the minority class.
- Since the trainable parameters in a BN layer also depend on the batch size (i.e., number of samples) in each iteration (mini-batch), its sizing could also play a role on the level of progress with the final BN layer.
- The observations and performance gain with respect to the PV dataset, upon utilizing ResNet and VGG architectures, may not be reproduced with any other dataset or with much simpler neural architectures.
- Since the number of units in the output layer depends on the number of classes in the dataset, the performance gain may not be achieved in multi-classification settings and the number of majority and minority classes can affect the role of the final BN layer.
- Since sigmoid activation is also one of the most widely used activation functions in the output layer for the binary classification problems, the performance gain could be achieved with sigmoid outputs as well.

Next, we report on addressing these hypotheses, one by one, and describe our empirical findings in detail.

4 IMPLEMENTATION DETAILS AND EXPERIMENTAL RESULTS

4.1 REMOVING THE ADDITIONAL BN LAYER DURING INFERENCE (H-1)

Since adding the final BN layer adds a small overhead (four new parameters) to the network at each iteration, we experimented if the final BN layer could be dropped once the training is finished so that we can avoid the cost. Dropping this final BN layer means that training the network from end to end, and then chopping off the final BN layer from the network before saving the weights. We tested this hypothesis for Apple, Pepper and Tomato images from PV dataset

under three conditions with 1% imbalance ratio: *Without final BN, with final BN and then removing the final BN during testing*. We observed that removing the final BN layer in inference would still give us a considerable boost on minority class without losing any performance gain on the majority class. The results in Table 4 show that the performance gain is very close to the configuration in which we used the final BN layer both in training and inference time. As a consequence, we confirm hypothesis (H-1) by showing that the final BN layer can indeed be removed in inference without compromising the performance gain.

4.2 IMPACT LEVEL REGARDING THE IMBALANCE RATIO (H-2) AND THE BATCH SIZE (H-3)

[Kocaman et al., 2020] empirically shows that the final BN layer, when placed be-fore the softmax output layer, has a considerable impact in highly imbalanced image classification problems but they fail to explain the impact of using the final BN layer as a function of level of imbalance in the training set. In order to find if there is a certain ratio in which the impact is maximized, we tested this hypothesis (H-1) over various levels of imbalance ratios and conditions explained below. In sum, we ended up with 430 model runs, each with 10 epochs (basically tested with 5, 10, 15, .. 100 unhealthy vs 1000 healthy samples). During the experiments, we observed that the impact of final BN on highly imbalanced settings is the most obvious when the ratio of minority class to the majority is less than 10%; above that almost no impact. As expected, the impact of the final BN layer is more obvious on minority class than it is on majority class, albeit the level of impact with respect to the imbalance ratio is almost same, and levels off around 10%. It is mainly because of the fact that the backbone architecture (ResNet34) is already good enough to converge faster on such data set (Plant Village) and the model does well on both classes after 10% imbalance (having more than 100 unhealthy with respect to 1000 healthy samples can already be handled regardless of final BN trick). During these experiments, we also tested if unfreezing the previous layers in the backbone CNN architecture (ResNet34) would also matter. We observed that unfreezing the pretrained layers helps without even final BN layer, but unfreezing adds more computation as the gradient loss will be calculated for each one of them. After adding the final BN layer and freezing the previous pretrained layers, we observed similar metrics and learning pattern as we did with unfreezing but not with final BN layer. This is another advantage of using the final BN that allow us to freeze the previous pretrained layers. The results are displayed as charts in Figure 2a.

We also experimented if the batch size would also be an important parameter for the minority class test accuracy when the final BN is added and found out that the highest score is gained when the batch size is around 64, whereas the accuracy drops afterwards with larger batches. The results are exhibited in Figure 2b. Consequently, given the reported observations, we confirm hypotheses (H-2) and (H-3).

4.3 EXPERIMENTATION ON THE MNIST DATASET: THE IMPACT OF FINAL BN LAYER IN BASIC CNNS AND FC NETWORKS [(H-4),(H-5),(H-6)]

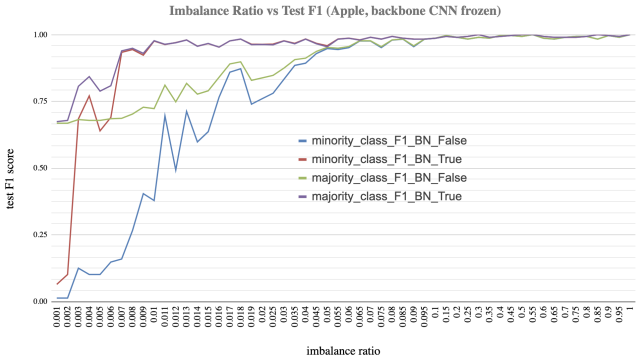
In order to reproduce equivalent results on a well known benchmark dataset when utilizing different DL architectures, we set up two simple DL architectures: a CNN network with five Conv2D layers and a one-layer (128-node) FC feed forward NN. Then we sampled several pairs of digits (2 vs 8, 3 vs 8, 3 vs 5 and 5 vs 8) from MNIST dataset that are mostly confused in a digit recognition task due to similar patterns in the pixels. During the experiments, the ratio of minority class to majority is kept as 0.1 and experiments with the simple CNN architecture indicates that, after adding the final BN layer, we gain $\sim 20\%$ boost in minority class and $\sim 10\%$ in majority class (see Table 5). Lower standard deviations across the runs with final BN layer also indicates the regularization effect of using the final BN layer. As a result, we rejected the fourth hypothesis (H-4) that we derived in section 3.3 by showing that the performance gain through the final BN layer can be reproduced with another dataset or with much simpler neural architectures. Another important finding is that the final BN layer boosts the minority class' F1 scores only when there is a single majority class. When two majority classes are set, no improvements are evident regardless of the usage of the final BN layer (see 4th and 5th settings in Table 6). Therefore, we partially confirm hypothesis (H-5) by showing that the performance gain through the final BN layer can also be achieved in multi-classification settings but the number of majority and minority classes may affect the role of the final BN layer. In the experiments with a one-layer FC network, we enriched the scope and also tested whether using different loss functions and output activation functions would have an impact on model performance using final BN layer with or without another BN layer after the hidden layer. We observed that adding the final BN layer, softmax output layer and categorical crossentropy (CCE) as a loss function have the highest test F1 scores for both classes. It is also important to note that we observe no improvement even after adding the final BN layer when we use sigmoid activation in the output layer (Table 6). Accordingly, we reject hypothesis (H-6).

5 DISCUSSION

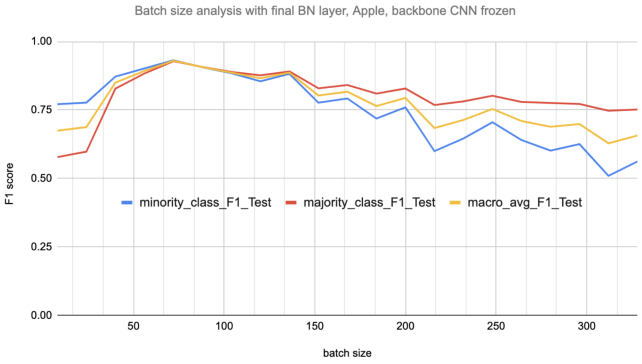
In a previous work done in [Kocaman et al., 2020], the authors suggested that by applying BN to dense layers, the gap between activations is reduced (normalized) and then

Table 4: Training with the final BN layer, and then dropping this layer while evaluating on the test set proved to be still useful in terms of improving the classification score on minority classes, albeit not as much as with the final BN layer kept (imbalance ratio 0.01, epoch 10, batch size 64).

	Apple		Pepper		Tomoto	
	healthy	unhealthy	healthy	unhealthy	healthy	unhealthy
with no final BN	0.71	0.22	0.74	0.45	0.74	0.46
with final BN	0.92	0.91	0.94	0.94	0.98	0.98
train with final BN remove while testing	0.74	**0.83**	0.75	**0.82**	0.78	**0.85**



(a) The impact of final BN layer on the F1 test score of each class for Apple plant. The impact is the most obvious when the ratio of minority class to the majority is less than 0.1.



(b) The impact of batch size on the F1 test score when the final BN layer added. The highest score is gained when the batch size is 64 and then the accuracy starts declining.

Figure 2: Imbalance ratio and batch size analysis with respect to the final BN layer added

Table 5: Average test metrics with a simple CNN network (5xConv2D) to classify 3 (minority) and 8 (majority) images from MNIST dataset with 0.01 imbalance ratio, 10 runs, and 20 epoch per run.

	with BN minority	without BN minority	with BN majority	without BN majority
Test F1	0.9299	0.7378	0.9447	0.8354
Std. Dev.	0.0791	0.1126	0.0487	0.0506

softmax is applied on normalized outputs, which are centered around the mean. Therefore, ending up with centered probabilities (around 0.5), but favoring the minority class by a small margin. They also argued that DNNs have the tendency of becoming 'over-confident' in their predictions during training, and this can reduce their ability to generalize further and thus perform as well on unseen data. Then they also concluded that a DNN with the final BN layer is more calibrated [Guo et al., 2017]. We assert that 'being less confident' in terms of softmax outputs might be fundamentally wrong and a network shouldn't be discarded or embraced due to its capacity of producing confident or less confident results in the softmax layer as it may not even be interpreted as a 'confidence'.

In this study, we empirically demonstrated that the final BN layer could still be eliminated in inference without compromising the attained performance gain. This finding supports the assertion that adding the BN layer makes the optimization landscape significantly smoother, which in turn renders the gradients' behavior more predictive and stable – as suggested by [Santurkar et al., 2018]. We argue that the learned parameters, which were affected by the addition of the final BN layer under imbalanced settings, are likely sufficiently robust to further generalization on the unseen samples, even without normalization prior to the softmax layer.

The observation of locating a sweet spot (10%) for the imbalance ratio, at which we can utilize the final BN layer, might be explained by the fact that the backbone ResNet architecture is already strong enough to easily generalize on the PV dataset and the model does not need any other regularization once the number of samples from the minority class exceeds a certain threshold. As the threshold found in our experiments is highly related to the DL architecture and the utilized dataset, it is clear that it may not apply to other datasets, but can be found in a similar way.

As discussed before, the BN layer calculates mean and variance to normalize the previous outputs across the batch, whereas the accuracy of this statistical estimation increases

Table 6: Using the same ResNet-34 architecture and skewness (1% vs 99%) and setting up five different configurations across various confusing classes from MNIST dataset, it is clear that adding the final BN layer boosts the minority class F1 scores by 10% to 30% only when there is a single majority class. When we have two majority classes, there is no improvement observed regardless of the final BN layer is used or not. (✓) indicates minority classes.

	Setting-1		Setting-2		Setting-3		Setting-4			Setting-5		
	3 (✓)	8	2 (✓)	8	3 (✓)	5	3 (✓)	5	8	3 (✓)	5 (✓)	8
without final BN	0.19	0.69	0.25	0.70	0.24	0.70	0.06	0.77	0.78	0.28	0.32	0.56
with final BN	0.55	0.76	0.50	0.74	0.54	0.76	0.00	0.77	0.78	0.58	0.59	0.69

Table 7: Using one-layer (128 node) NN and the 0.01 skewness ratio, with nine different settings for 3 (minority) and 8 (majority) classes from MNIST dataset (100-epoch). Adding the final BN layer, softmax output layer and CCE as a loss function has the highest test F1 scores for both classes. (CCE - categorical cross entropy, BCE - binary cross entropy, first BN - a BN layer after the hidden layer).

output activation	loss function	first BN layer	final BN layer	class-3 (minority)	class-8 (majority)
sigmoid	BCE			0.17	0.67
softmax	BCE			0.00	0.67
softmax	BCE	✓		0.60	0.78
softmax	CCE			0.67	0.80
sigmoid	BCE		✓	0.05	0.67
softmax	BCE		✓	0.85	0.88
softmax	BCE	✓	✓	0.83	0.87
softmax	CCE		✓	**0.88**	**0.90**
softmax	CCE	✓	✓	0.78	0.85

as the batch size grows. However, its role seems to change under the imbalanced settings – we found out that a batch size of 64 reaches the highest score, whereas by utilizing larger batches the score consistently drops. As a possible explanation for this observation, we think that the larger the batches, the higher the number of majority samples in a batch and the lesser the chances that the minority samples are fairly represented, resulting in a deteriorated performance.

During our experiments, we expected to see similar behavior with sigmoid activation replacing softmax in the output layer, but, evidently, the final BN layer works best with softmax activations. Although softmax output may not serve as a good uncertainty measure for DNNs compared to sigmoid layer, it can still do well on detecting the under-represented samples when used with the final BN layer.

6 CONCLUSIONS

In this study, we extended the previous efforts done by [Kocaman et al., 2020] to devise an effective approach to enable learning of minority classes, given the surprising evidence of applying the final BN layer. Given these recent findings, we formulated and tested additional hypotheses.

We at first noticed that the performance gain after adding the final BN layer in highly imbalanced settings could still be achieved after removing this additional BN layer during inference; in turn enabling us to get a performance boost with no additional cost in production. Then we explored the dynamics of using the final BN layer as a function of the imbalance ratio within the training set, and found out that the impact of final BN on highly imbalanced settings is the most apparent when the ratio of minority class to the majority is less than 10%; there is hardly any impact above that threshold.

We also ran similar experiments with simpler architectures, namely a basic CNN and a single-layered FC network, when applied to the MNIST dataset under various imbalance settings. The simple CNN experiments exhibited a gain of $\sim$ 20% boost per the minority class and $\sim$ 10% per the majority class after adding the final BN layer. In the FC network experiments, we observed improvements by 10% to 30% only when a single majority class was defined; no improvements were evident for two majority classes, regardless of the usage of the final BN layer. While experimenting with different activation and cost functions, we found out that using the final BN layer with sigmoid activation had almost no impact on the task at hand.

We found the impact of final BN layer in simpler neural networks quite surprising. It is an important finding, which we plan to further investigate in the future, as it requires thorough analysis. We also plan to formulate our findings in a generalized way for any neural model, preferably with a combination of softmax activation or a proper loss function that could be used in imbalanced image classification problems.

REFERENCES

Bjorn Barz and Joachim Denzler. Deep learning on small datasets without pre-training using cosine loss. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 1371–1380, 2020.

Laura Beggel, Michael Pfeiffer, and Bernd Bischl. Robust anomaly detection in images using adversarial autoencoders. *arXiv preprint arXiv:1901.06355*, 2019.

Nils Bjorck, Carla P Gomes, Bart Selman, and Kilian Q Weinberger. Understanding batch normalization. In *Advances in Neural Information Processing Systems*, pages 7694–7705, 2018.

Ivan Chelombiev, Conor Houghton, and Cian O'Donnell. Adaptive estimators show information compression in deep neural networks. *arXiv preprint arXiv:1902.09037*, 2019.

Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9268–9277, 2019.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1321–1330. JMLR. org, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Mahbub Hussain, Jordan J Bird, and Diego R Faria. A study on CNN transfer learning for image classification. In *UK Workshop on Computational Intelligence*, pages 191–202. Springer, 2018.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Veysel Kocaman, Ofer M Shir, and Thomas Bäck. Improving model accuracy for imbalanced image classification tasks by adding a final batch normalization layer: An empirical study. *arXiv preprint arXiv:2011.06319, Accepted to International Conference on Pattern Recognition, ICPR 2020.*, 2020.

Anders Krogh and John A Hertz. A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pages 950–957, 1992.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

Dmytro Mishkin and Jiri Matas. All you need is a good init. *arXiv preprint arXiv:1511.06422*, 2015.

Sharada P Mohanty, David P Hughes, and Marcel Salathé. Using deep learning for image-based plant disease detection. *Frontiers in plant science*, 7:1419, 2016.

Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? In *Advances in Neural Information Processing Systems*, pages 4696–4705, 2019.

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In *Advances in Neural Information Processing Systems*, pages 2483–2493, 2018.

Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472. IEEE, 2017.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.