

Batch Normalization and the impact of batch structure on the behavior of deep convolution networks

Mohamed Hajaj Duncan Gillies
Department of Computing, Imperial College London

Abstract

Batch normalization was introduced in 2015 to speed up training of deep convolution networks by normalizing the activations across the current batch to have zero mean and unity variance. The results presented here show an interesting aspect of batch normalization, where controlling the shape of the training batches can influence what the network will learn. If training batches are structured as balanced batches (one image per class), and inference is also carried out on balanced test batches, using the batch's own means and variances, then the conditional results will improve considerably. The network uses the strong information about easy images in a balanced batch, and propagates it through the shared means and variances to help decide the identity of harder images on the same batch. Balancing the test batches requires the labels of the test images, which are not available in practice, however further investigation can be done using batch structures that are less strict and might not require the test image labels. The conditional results show the error rate almost reduced to zero for nontrivial datasets with small number of classes such as the CIFAR10.

1 Introduction

When training deep convolution networks, batch normalization BN [Ioffe and Szegedy, 2015] reduces the dependency of the gradients on the scale of the parameters and their initial values, which allows for much higher learning rates, much faster convergence, and better final results. The experiments presented here show that by the nature of its implementation, batch normalization has more to it than just speeding up training. Because activations are normalized using means and variances that are shared across all images in the current batch, the output activations of one image are influenced by all the other images in the batch. This means that the behavior of the network is not only receptive to the individual images but also to the structure of the training batches. Therefore, with BN there is an extra level of control that can be used to guide the training of these networks, which is based on how batches are constructed. The results presented here show that it is possible to make the network learn something based on how batches are constructed.

Balanced batches are batches that have a single instance from each class with size equal to the number of classes. If the network is trained only on balanced batches, then in addition to learning how to classify single images, BN will allow the network to learn an extra logic based on the structure of balanced batches. Because it was only exposed to balanced batches in the training phase, the network will learn an association mechanism between the images in the batch through the shared means and variances of BN to always expect balanced batches. If the performance of the network is measured in the standard way on single test images, then this association mechanism based on batch structure cannot be noticed. In order to measure it, the network needs to be tested on balanced test batches using each batch's own means and variances. The practical difficulty here is balancing the test batches, which requires the labels of the test images. Therefore, the purpose of these experiments is to investigate this extra dimension of control based on *batch structure* that is made possible through BN, and it is not claiming state of the art results. However, because of the scale of improvement (conditional) achieved on the CIFAR10 dataset, the subject is worth further investigation maybe using other batch structures that do not require the test image labels.

2 Implementation

2.1 Batch Normalization

For a deep neural network, the input distribution of layer l depends on the parameters of all previous layers, and as these parameters change, the input distribution of layer l will also change. Layer l will try to adapt to an input distribution that keeps changing throughout training, and that slows up training. This problem is called internal covariance shifts [Shimodaira, 2000], and BN tries to reduce this problem by performing a simplified version of complete whitening to the inputs of each layer. First, BN assumes that input features are independent, and therefore can be normalized independently to have zero mean and unity variance. Second, the means and variances are calculated across the current batch, and not over the entire training data (hence the name batch normalization).

Because the networks used in this study [He et al., 2016a] are fully convolutional, BN will only be applied to convolution layers. For each output channel a mean and variance are calculated across all images in the current batch. If the channel size is $h \times w$, and the batch size is m images, and $m' = m \times h \times w$, then the mean and variance are calculated using equations (1) and (2), and each location x_i in that output channel across all the images in the batch is normalized using equation (3). A convolution layer with N output channels will need N pairs of means and variances. The algorithm also adds a linear transformation $y = \alpha x + \beta$ (α, β are trainable parameters) after the normalization step to restore the expressive capabilities of the network. For the backward error propagation equations refer to [Ioffe and Szegedy, 2015].

$$\mu = \frac{1}{m'} \sum_{i=1}^{m'} x_i \quad (1)$$

$$\sigma^2 = -\mu^2 + \frac{1}{m'} \sum_{i=1}^{m'} x_i^2 \quad (2)$$

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \varepsilon}} \quad (3)$$

2.2 Balanced Batches

A balanced batch contains a single instance from each class, and therefore the batch size of a balanced batch will always be equal to the number of classes. With balanced batches, the standard BN algorithm will be used with the following considerations at the training and inference stages.

- **Training:** - instead of being constructed randomly, training batches are created to be balanced, and to contain a single instance from each class. If training images were shuffled to prevent an image from always appearing in the same batch (to improve performance), then the shuffling subroutine needs to be changed to always generate balanced batches.
- **Inference:** - Standard BN calculates a fixed set of means and variances over the entire training data to be used at the inference stage instead of using the means and variances of the current test batch. This is done to make the prediction of the network deterministic and depends only on the image itself, and not on the other images in the batch. In order to measure the effect of training the network on balanced batches, test images are arranged as balanced batches and the current means and variances of the test batch itself are used in the inference process.

2.3 Network Structure

Deep residual convolution networks [He et al., 2016a] were used in this study, where a standard deep residual network starts with a single convolution layer, followed by multiple residual building blocks, followed by one fully connected layer, which is the output layer. A residual block figure(1) is made of 2 or 3 stacked convolution layers warped by identity mapping so that the output of the block is the combination of the input signal to that block and the output signal of the stacked convolution layers. Residual learning makes it possible to train very deep networks with up to hundreds of layers by eliminating the degradation problem [He and Sun, 2015, Srivastava et al., 2015] exhibited by standard very deep stacked networks.

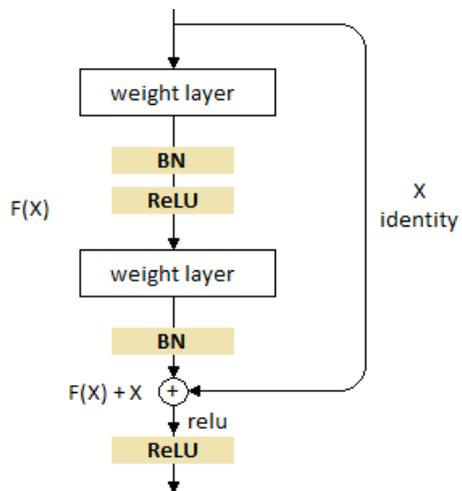


Figure 1: Residual Block

3 Experiments and Results

3.1 CIFAR10 Experiment

The first experiments to measure the effect of using balanced batches with BN were done using the CIFAR10 dataset. Table (1) shows the structure of the two deep residual networks with depths equal to 20 and 44 layers used in this experiment, and they are very similar to the ones used by He et al. [He et al., 2016a] for CIFAR10. The same treatment of the CIFAR10 dataset by enlarging the images with zero-padding from 32×32 to 40×40 pixels is used here. The main stream approach in dealing with the CIFAR10 dataset uses cropping without scaling because the images are very small, 32×32 pixels. However, we found that using scaling has improved the results despite the images being very small. The weight initialization method in [He et al., 2015], and the standard color augmentation method in [Krizhevsky et al., 2012] are used. Simple gradient descent with weight decay is used to update the network parameters.

output size	20 Layers	44 Layers
36 × 36	<i>conv</i> , 3 × 3, 64	
	$\begin{bmatrix} \text{conv}, 3 \times 3, 64 \\ \text{conv}, 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} \text{conv}, 3 \times 3, 64 \\ \text{conv}, 3 \times 3, 64 \end{bmatrix} \times 7$
18 × 18	<i>max pool</i> 3 × 3, stride 2	
	$\begin{bmatrix} \text{conv}, 3 \times 3, 128 \\ \text{conv}, 3 \times 3, 128 \end{bmatrix} \times 3$	$\begin{bmatrix} \text{conv}, 3 \times 3, 128 \\ \text{conv}, 3 \times 3, 128 \end{bmatrix} \times 7$
9 × 9	<i>max pool</i> 3 × 3, stride 2	
	$\begin{bmatrix} \text{conv}, 3 \times 3, 256 \\ \text{conv}, 3 \times 3, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} \text{conv}, 3 \times 3, 256 \\ \text{conv}, 3 \times 3, 256 \end{bmatrix} \times 7$
1 × 1	<i>global avg pool</i> 9 × 9, 10-d <i>fc</i> , <i>softmax</i>	

Table 1: Network Structure

In the first part of the experiment, both training and inference were performed in the standard way, where training batches were created randomly, and inference were carried out on individual images using a fixed set of means and variances computed using the entire training data after training was completed. In the second part of the experiment, training was done using balanced batches, and inference was carried out twice: First, inference was done in the standard way, where images are tested individually using fixed means and variances. Second, to measure the effect of batch structure on the performance of the network, test images were arranged the same way as training images; test images were arranged as balanced batches, and the means and variances of the current test batch itself were used in the inference process. Table (2) shows the results for both experiments. It is clear that balancing the training batches doesn’t change the results, if the inference process is carried out in a standard way. However, if the performance of the network trained using balanced batches is also tested using balanced test batches, the error rate is reduced by about 80% for both network models. The error rate was almost eliminated for the non-trivial CIFAR10.

Training	Inference	20 Layers	44 Layers
standard	standard	4.45%	3.89%
Balanced Batches	standard	4.47%	3.9%
Balanced Batches	Balanced Batches	0.97%	0.69%

Table 2: CIFAR10 Results, when training is done using random vs balanced batches, and inference is done on individual images vs balanced batches.

Figure (2) shows the error curves measured on the central crop of the test set images as training progresses for the 44-layer network. The red curve was obtained by training the network on balanced batches, and the blue curve was obtained by training the network on randomly constructed batches, but because both were measured on individual test

images using the running averages of the means and variances, the results were very similar. However, if the progress of the network trained using balanced batches was also measured on balanced test batches using the current means and variances of the batch itself, then a big reduction in the test error can be measured from the start to the end of the training process as the black curve shows. These three curves agree with the final results shown in the three rows in table (2).

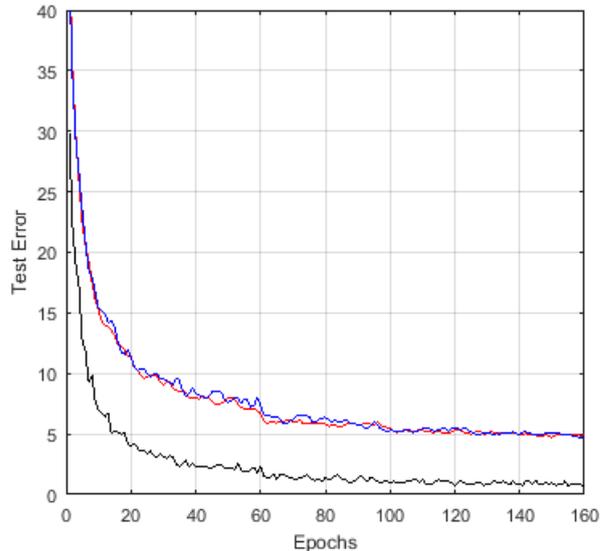


Figure 2: validation (test) error curves as training progresses for the CIFAR10 dataset. **blue curve**: standard training and inference, **red curve**: training using balanced batches, **black curve**: both training and inference using balanced batches.

3.2 Experiment Two

Three datasets with 10, 50, and 100 classes were randomly sampled from ImageNet, and will be used here. The reason for not using the entire ImageNet dataset is because it has 1000 classes, and using balanced batches will require a batch size of 1000 image, which cannot be supported by our hardware. The reason for sampling three datasets instead of just one is to measure the impact of batch size on the results obtained using balanced batches. All sampled classes have 1300 training images, and 50 test images. Table (4) shows the structure of a 34-layer deep residual network used in this experiment for all three datasets, which is similar to that used by [He et al., 2016a] for ImageNet. Data augmentation in [Szegedy et al., 2015], weight initialization in [He et al., 2015], and color augmentation in [Krizhevsky et al., 2012] were used. The RMSProp optimization method is used instead of gradient decent with momentum to update the network parameters, using a decay value of 0.999 to calculate the running average per parameter.

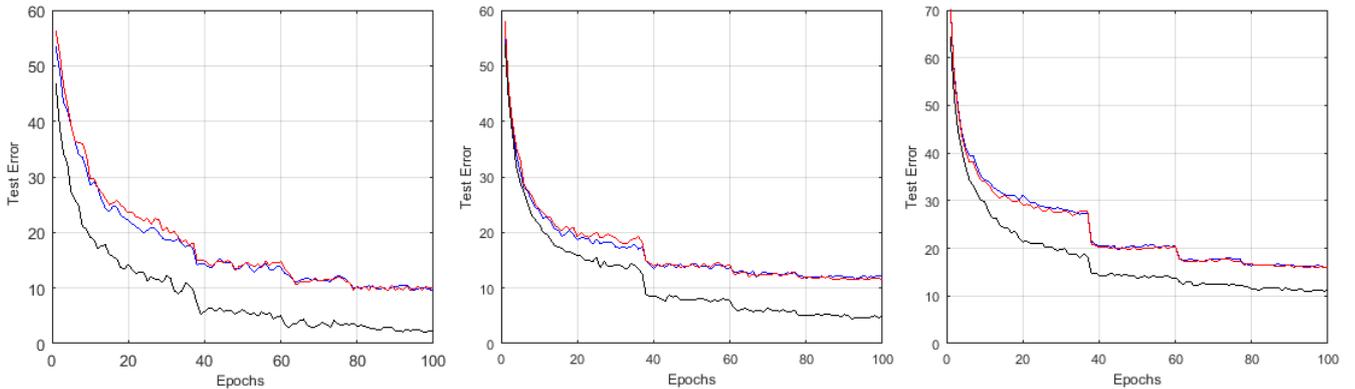


Figure 3: the validation (test) error curves as training progresses, blue curve using standard training and inference, red curve training using balanced batches, black curve both training and inference using balanced batches. **left:** for the 10-classes dataset, **middle:** for the 50-classes dataset, **right:** for the 100-classes dataset.

Training	Inference	10 Classes dataset	50 Classes dataset	100 Classes dataset
standard	standard	5.97%	7.3%	10.1%
Balanced Batches	standard	5.9%	7.34%	10.14%
Balanced Batches	Balanced Batches	1.1%	3.32%	6.74%

Table 3: results for 3 datasets, when training is done using random vs balanced batches, and inference is done on individual images vs balanced batches.

output size	34 Layers
112×112	<i>conv</i> , 5×5 , 96
56×56	<i>max pool</i> 3×3 , stride 2
	$\begin{bmatrix} \text{conv}, 3 \times 3, 96 \\ \text{conv}, 3 \times 3, 96 \end{bmatrix} \times 3$
28×28	<i>max pool</i> 3×3 , stride 2
	$\begin{bmatrix} \text{conv}, 3 \times 3, 192 \\ \text{conv}, 3 \times 3, 192 \end{bmatrix} \times 4$
14×14	<i>max pool</i> 3×3 , stride 2
	$\begin{bmatrix} \text{conv}, 3 \times 3, 384 \\ \text{conv}, 3 \times 3, 384 \end{bmatrix} \times 6$
7×7	<i>max pool</i> 3×3 , stride 2
	$\begin{bmatrix} \text{conv}, 3 \times 3, 768 \\ \text{conv}, 3 \times 3, 768 \end{bmatrix} \times 3$
1×1	<i>global avg pool</i> 9×9 <i>10-d fc, softmax</i>

Table 4: Network Structure

Table (3) shows the results for the three datasets, and compares the results when training and inference are done in the standard way versus using balanced batches. As with the CIFAR10 dataset, training the network using balanced batches doesn’t change the results if inference is carried out on individual images using fixed means and variances. However, if the network trained on balanced batches, is also tested using balanced test batches, the error rate is

reduced considerably. When using balanced batches, the batch size is equal to the number of classes, and therefore, the batch sizes used here are 10, 50, and 100 images, for datasets with 10, 50, and 100 classes respectively. From table (3) The relative reduction in the error when training and inference were done using balanced batches, is very dependent on the batch size. For the 10-classes dataset the reduction was 81%, for the 50-classes dataset it was 54%, and for the 100-classes dataset it was 33%. This is not surprising because balancing the test batches means that the real task here is not really classifying the images but rather identifying the identity of each image in a balanced batch. And it makes sense that this task gets harder as the size of the batch gets bigger, and as the batch size (thus the number of classes) goes to infinity (or a large number) the performance measured using balanced batches converges to the performance measured on individual images. It is interesting to notice that the 81% conditional gains for the 10-classes dataset is very close to the 80% conditional gains obtained with the CIFAR10 data sets, which also has 10 classes.

Figure (3) shows the error curves measured on the central crop for all three datasets as training progresses. Again, the red and blue curves show that training the network on balanced batches will produce similar results to training it on randomly constructed batches if the performance is measured on individual test images using the running averages of the means and variances. The black curves show the gains when both training and inference are done on balanced batches using the batch’s own means and variances. These learning curves agree with the final results shown in the three rows of table (3).

3.3 Shuffling the balanced test batches

In the training phase shuffling the training images improves the results if BN is implemented in the network. The results also show that if balanced test batches are shuffled, the results will improve

0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9	9

4	0	0	0	0	0	0	0	0	0	0
1	4	1	1	1	1	1	1	1	1	1
2	2	4	2	2	2	2	2	2	2	2
3	3	3	4	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	4	5	5	5	5	5
6	6	6	6	6	6	4	6	6	6	6
7	7	7	7	7	7	7	4	7	7	7
8	8	8	8	8	8	8	8	4	8	8
9	9	9	9	9	9	9	9	9	4	9

1	0	0	0	0	0	0	0	0	0	9
0	1	1	1	1	1	1	1	1	1	1
2	2	3	2	2	2	2	2	2	2	2
3	3	2	4	3	3	3	3	3	3	3
4	4	4	3	5	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	7	6	6	6	6
7	7	7	7	7	7	6	8	7	7	7
8	8	8	8	8	8	8	7	8	8	8
9	9	9	9	9	9	9	9	9	9	0

Figure 4: a balanced test batch made of images classified with high confidence, **left**: a misclassified test image made to circulate through the batch, **middle**: correctly classified test image (belong to class no. 4) made to circulate through the batch, **right**: 2 misclassified test images made to circulate through the batch.

even further. It seems that there is an advantage in presenting the different crops of a test image with different batches rather than presenting all of them with the same batch. One possible explanation is that if images are not shuffled, the current balanced batch may contain multiple similar images that belong to different classes, and that may confuse the network. By shuffling the images at inference time, the chances of presenting all the crops of an image with another similar image from a different class will be reduced. Table (5) shows the error rate measured on shuffled and balanced test batches for the three datasets sampled from ImageNet.

Training & Inference	10 Classes dataset	50 Classes dataset	100 Classes dataset
Balanced Batches	0.2%	2.04%	5.5%

Table 5: Results obtained by shuffling balanced test batches

4 Inference using the batch’s own means and variances

The results in the previous sections show that if batches are structured as balanced batches at the training and inference stages, then the results improve considerably. The experiment presented here shows how the network uses the structure of a balanced test batch in the classification process. This experiment is done on the CIFAR10 dataset using a single balanced test batch made of images identified with high confidence by the network. In the first run a misclassified image (using standard inference) is selected and used to circulate through the created test batch by replacing one image at a time. In the second run an image that has been classified by the network with high confidence (using standard inference) is made to circulate through the batch. Because the number of classes for the CIFAR10 dataset is 10, then each time the selected image will replace one of the

10 images that make up the balanced batch, and the classification of all the images in the batch will be reported. Figure (4) shows the classification results, where each column shows a single step, and represents a single batch.

Figure (4, left) shows what happened to the weak (misclassified) image, where each time the network classifies the image to belong to the missing class. Figure (4, middle) shows the results for the strong (classified with high confidence) image, where the image was always classified correctly, and the identity of the image wasn’t changed each time to match the identity of the replaced image. The third run is a generalization of the first one, where two weak images are made to circulate through the balanced batch, replacing two images at a time, and figure (4, right) shows how the network has interpreted their identity to replace the missing classes (or one of them).

The results show that when the network is confident about the identity of the test image, then it ignores the structure of the batch. On the other hand, if the network is not confident about the identity of the image, then it relies mainly on the structure of the batch to classify that image. Based on the results shown in figure (4), the network uses the structure of a balanced batch as follows: if a balanced test batch made up of n images is passed through the network, and the network is confident of the identity of m out of the total n images, then the network will restrict the prediction of the remaining $n - m$ images to the remaining $n - m$ classes.

*(m out of n images identified) && (batch is balanced)
 \Rightarrow (remaining images belong to remaining $n-m$ classes)*

Figure (4, left) showed how the network was only confident of the identity of 9 out of the 10 images, and therefore the tenth image was always interpreted as the missing class. Figure (4, right) showed how the network was only confident of the identity of 8 out of the 10 images, and therefore the identity

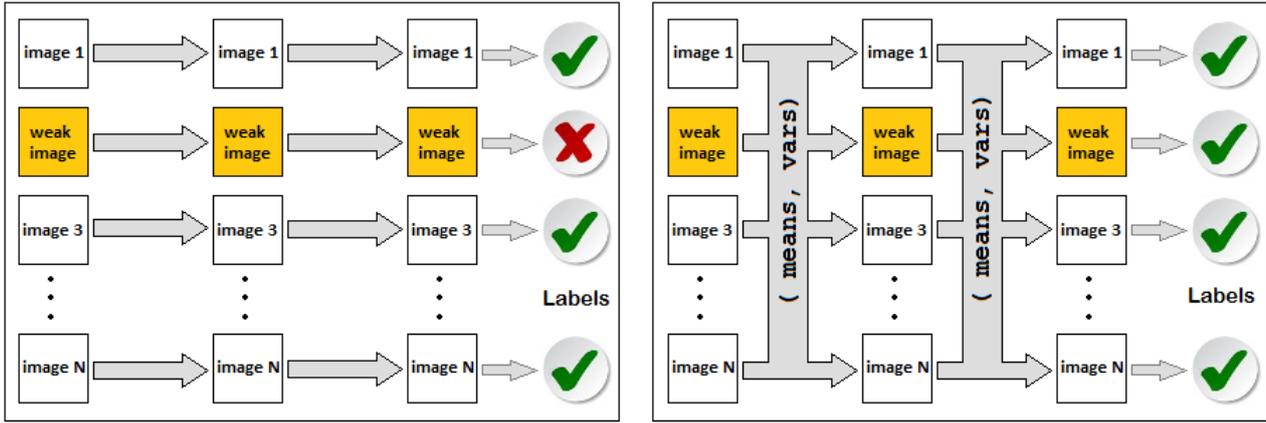


Figure 5: inference for a balanced batch, **left**: with precomputed fixed means and variances, **right**: with current means and variances.

of the 2 remaining images were always restricted to the 2 missing classes. This logic (highlighted in blue) cannot be implemented by the network if the prediction of one image is independent from all the other images in the batch, and therefore it cannot be implemented without BN. The network was able to implement this logic because it was only exposed to balanced batches in the training stage, and because BN gave the network the capacity to learn something based on the structure of the training batches by using the shared means and variances as a communication tool between the images in the batch.

Figure (5) explains the process of making a decision about a weak image in a balanced batch, where the network snoops on the decisions about the other images in the batch through the shared means and variances to help decide on the weak image. What is interesting is that this process happens in one forward pass, where the network identifies all images in the current batch at the same time. As signals travel forward, the network found a way to use the shared means and variances at each layer to help guide the prediction of all the images in the batch, based on the fact that batches are balanced. The results presented here show that with BN, controlling the structure of the training batches, adds another layer of control over what the network can learn.

5 Difficulty balancing the test batches

It is tempting to try to translate these big conditional gains into actual gains, however these results are very hard to achieve in practice because structuring test images as balanced batches requires the test image labels. One attempt is to use the labels generated by standard inference to balance the test batches, and then use these semi-balanced batches to generate more accurate labels, where the process is repeated until no improvements can be achieved. Figure (6) shows this process repeated 20 times for the CIFAR10 dataset, but unfortunately it didn't

lead to more accurate results (better labels). The error rate stayed almost constant as a horizontal line. From table (2) the error rate using standard inference is 3.89%, which means 3.89% of the images will be misclassified, and those are the toughest images where inference using fully balanced batches made gains to reduce the error to 0.69%. From the previous section, figure (4) shows what happens to misclassified images when placed in semi-balanced batches, the network often misinterprets their identity to replace the missing class. Therefore, standard inference is not adequate to solve the problem of balancing the test batches, because it will misclassify the important images, where gains need to be made.

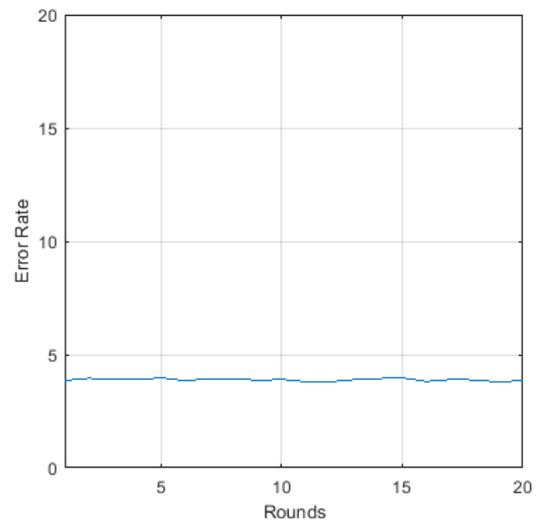


Figure 6: the repeated process of balancing test batches starting from labels generated by standard inference fails.

References

- [Bengio et al., 2009] Bengio, Y. et al. (2009). Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127.
- [Goodfellow et al., 2013] Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A., and Bengio, Y.

- (2013). An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*.
- [He and Sun, 2015] He, K. and Sun, J. (2015). Convolutional neural networks at constrained time cost. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5353–5360.
- [He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.
- [He et al., 2016a] He, K., Zhang, X., Ren, S., and Sun, J. (2016a). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.
- [He et al., 2016b] He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer.
- [Hu et al., 2017] Hu, J., Shen, L., and Sun, G. (2017). Squeeze-and-excitation networks. *arXiv preprint arXiv:1709.01507*.
- [Huang et al., 2016] Huang, G., Sun, Y., Liu, Z., Sedra, D., and Weinberger, K. Q. (2016). Deep networks with stochastic depth. In *European Conference on Computer Vision*, pages 646–661. Springer.
- [Ioffe and Szegedy, 2015] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456.
- [Kingma and Ba, 2014] Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- [Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.
- [Liu et al., 2015] Liu, X., Gao, J., He, X., Deng, L., Duh, K., and Wang, Y.-Y. (2015). Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *HLT-NAACL*, pages 912–921.
- [Shimodaira, 2000] Shimodaira, H. (2000). Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244.
- [Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
- [Srivastava et al., 2015] Srivastava, R. K., Greff, K., and Schmidhuber, J. (2015). Highway networks. *arXiv preprint arXiv:1505.00387*.
- [Szegedy et al., 2017] Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, pages 4278–4284.
- [Szegedy et al., 2015] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.
- [Szegedy et al., 2016] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826.
- [Zagoruyko and Komodakis, 2016] Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. *arXiv preprint arXiv:1605.07146*.