Proceedings of Machine Learning Research 74:141–151, 2017 Poster - LIDTA 2017

Controlling Imbalanced Error in Deep Learning with the Log Bilinear Loss

Yehezkel S. Resheff YEHEZKEL.RESHEFF@MAIL.HUJI.AC.IL
Edmond and Lily Safra Center for Brain Sciences, The Hebrew University of Jerusalem

Amit Mandelbom AMIT.MANDELBOM@CS.HUJI.AC.IL
School of Computer Science and Engineering, The Hebrew University of Jerusalem

Daphna Weinshall WEINSHALL@GMAIL.COM
School of Computer Science and Engineering, The Hebrew University of Jerusalem

Editors: Luís Torgo, Bartosz Krawczyk, Paula Branco and Nuno Moniz.

Abstract

Deep learning has become the method of choice for many machine learning tasks in recent years, and especially for multi-class classification. The most common loss function used in this context is the cross-entropy loss. While this function is insensitive to the identity of the assigned class in the case of misclassification, in practice it very common to have imbalanced sensitivity to error, meaning some wrong assignments are much worse than others. Here we present the bilinear-loss (and related log-bilinear-loss) which differentially penalizes the different wrong assignments of the model. We thoroughly test the proposed method using standard models and benchmark image datasets.

Keywords: Unbalanced Classes, Deep Learning, Bilinear Loss

1. Introduction

Classification may well be the most studied machine learning problem, from both the theoretical and the practical aspects. The problem is often phrased as an optimization problem; we seek to minimize some loss function associated with correct classification (Kotsiantis et al., 2007). Initially we may phrase the problem using the 0-1 loss, which effectively counts the number of misclassified points. In the multi-class case, most methods replace this function by a surrogate loss due to various computational reasons (or just to achieve a tractable problem), or design a more specific goal-oriented or application motivated loss.

In this paper we go back to basics, and question the use of the 0-1 loss function in multi-class classification. Already in the binary scenario, we see the need for a more subtle loss function, which distinguishes between the 2 different types of possible errors: false negative - when a point of class 1 is wrongly classified as class 2, and false positive - when a point of class 2 is wrongly classified as class 1. When class 1 symbolizes the diagnosis of some serious illness while class 2 refers to the lack of findings, differential treatment of the two errors may be a matter of life and death. In most applications this distinction doesn't make it into the definition of the loss function (but see review of related work below), and it is usually approached post-hoc by such means as the ROC curve, which essentially describes the behavior of the classifier as a trade-off between the two types of error.

These days the field of applied machine learning is swept by deep learning (LeCun et al., 2015), and we customary solve very large multi-class classification problems with a growing number of classes. Now the question becomes more acute, and the 0-1 loss function does not seem to provide a good model for our problem anymore. In most application domains, different misclassification errors are likely to have different detrimental implications, and should be penalized accordingly to achieve the desired classifier. For example, the consequence of diagnosing a certain illness wrongly by coming up with a related illness may be considered less harmful to the patient than missing it altogether.

Unlike binary classification, with multi-class problems we don't have only two types of errors to worry about, but rather $O(k^2)$ errors if there are k labels to choose from. We can no longer postpone the resolution of the problem to some post-hoc stage, and the choice of a single threshold. Furthermore, when considering deep learning, encoding the desired trade-off between types of errors will hopefully lead to a representation more suitable to make the necessary distinctions.

Yet in practice, the basic loss function one uses in the training of neural network classifiers has changed little, revealing its deep roots in the world of binary classification and the 0-1 loss function. Specifically, the categorical cross-entropy (which when using a single correct response reduces simply to log-loss), has the property of invariance to the division of the output weight among the erroneous labels, and therefore it is also invariant to the identity of the class an example is assigned to, in case of misclassification.

This is the problem addressed here. We propose two loss functions, which provide the means for distinguishing between different misclassification errors by penalizing each error differently, while maintaining good overall accuracy. Specifically, in Section 2 we develop the bilinear loss (and related log-bilinear loss) which directly penalize a deep learning model differentially for weights assigned in the output layer, depending on both the correct label and the identity of the wrong labels which have weight assigned to them.

In Section 3 we describe the empirical evaluation of training deep networks with these loss functions, using standard deep learning models and benchmark multi-class datasets. Thus we empirically show the ability of these methods to maintain good overall performance while redistributing the error differently between the possible wrong assignments.

The main contribution of this is in introducing the bilinear/log-bilinear loss functions and showing their utility for controlling error location in deep learning models.

Related Work

Asymmetric loss functions have been studied extensively in the context of binary classification, both theoretically and algorithmically, see e.g. (Bach et al., 2006; Lin et al., 2002), and have been shown to improve classification performance with asymmetrical error costs or imbalanced data.

In the context of multiclass classification, proper rescaling methods are discussed in Zhou and Liu (2010) in order to address the issue of imbalanced datasets; rescaling methods, however, typically penalize error based on the identify of the wrong label only. In Domingos (1999) a wrapper method is described to transform every multi-class classifier to a cost-sensitive one, while (Margineantu et al., 1999) shows how multi-class decision trees can be trained to approximate a general loss matrix. A truly asymmetric cost-sensitive loss function is used in (Lee et al., 2004), employing a *generalized cost matrix* somewhat similar

to matrix A defined in Section 2, while (Zhang and Zhou, 2010) offers a formulation which is based on the Bayes-decision theory and the k-nearest neighbor classifier.

In the context of deep learning however, the purpose of a cost sensitive objective is not only to change the output of the model, but first and foremost to learn a representation in intermediate layers of the network that is able to better capture the aspects of the data that are important according to the relative cost of the different types of error.

2. The [Log] Bilinear-loss

We assume a model with k non-negative output units, such that the output for the $i-th$ example is:

$$\hat{y}^{(i)} = \hat{y}_1^{(i)}, \ldots, \hat{y}_k^{(i)}$$

and further assume a per-example normalized output, i.e.: $\forall i : \sum_{j=1}^{k} \hat{y}_j^{(i)} = 1$

In most cases in practice, the training set is labeled with a single correct response per example. Thus, the common cross-entropy loss reduces to $-log(y_{l_i}^{(i)})$, where l_i is the correct label for the i'th example. This loss essentially represents an implicit policy of rewarding for weight placed on the correct answer, while being indifferent to the identity of the wrong labels which have weight assigned to them.

Arguably, this common practice is often in direct opposition to the goal of the process of learning; in many real world scenarios, some mistakes are extremely costly while others are of little consequence. In this section we develop alternative loss functions which address this issue directly.

We start by modifying the loss function used for training the classifier. We augment the usual loss function with a term where a non-negative cost is assigned to any wrong classification, and where the cost is based on *both* the correct label and the identity of the misclassified label in an asymmetrical manner. Specifically, let $a_{i,j}$ denote the relative cost associated with assigning the label j to an example whose correct label is i, and let $A = \{a_{i,j}\} \in \mathcal{R}^{k \times k}$ denote the penalty matrix.

Using A we define two related loss functions: The Bilinear loss is defined as:

$$L_B = y^T A \hat{y} \tag{1}$$

where y denotes the correct output (y is a probability vector). Similarly, the log-Bilinear loss is defined as:

$$L_{LB} = -y^T A log(1 - \hat{y}) \tag{2}$$

where $log(\cdot)$ operates element-wise.

Finally, we combine the regular cross-entropy loss with (1) and (2) to achieve a loss function with the known benefits of cross-entropy, and which also provides the deferential treatment of errors:

$$L_{CE+B} = (1 - \alpha) L_{CE} + \alpha y^T A \hat{y} \tag{3}$$

$$L_{CE+LB} = (1 - \alpha) L_{CE} - \alpha y^T A log(1 - \hat{y}) \tag{4}$$

where $L_{CE} = -\sum y_i log(\hat{y}_i)$ is the regular cross-entropy loss.

The fundamental difference between the bilinear and log-bilinear loss formulations is the implied view on what in the output of the model is to be penalized (and thus controlled). The bilinear formulation adds a constant cost $a_{ij}\Delta p_j$ for each Δp_j increase in the j'th normalized output unit, for a training example from the i'th class. The log-bilinear formulation on the other hand is insensitive to this sort of increase as long as the overall value in the j'th output is small, but picks up sharply as p_j approaches 1.

Furthermore, for two equally penalized mistakes j, k for an example from class i (meaning $a_{ij} = a_{ik}$), the bilinear loss is insensitive to the division of error between the two classes, since $\alpha_1 a_{ij} + \alpha_2 a_{ik} = (\alpha_1 + \alpha_2)a_{ij}$, so only the sum of the assignments to these two classes matters. The log-bilinear loss however amounts to $\alpha_1 log(1 - a_{ij}) + \alpha_2 log(1 - a_{ik})$, which for a constant sum is maximized when all the weight is placed on one of the errors.

The meaning of this property is that the bilinear loss is penalizing for the total weight placed on erroneous classes weighted by the relative importance (as measured by the penalty matrix A). The log-bilinear loss is penalizing peakiness of the wrong assignment, again weighted by importance. Thus, the bilinear loss is a more likely candidate when we would like to control the locations where erroneous weights concentrate, and the log-bilinear loss when we would like to control which errors the model is confident about.

3. Empirical evaluation of the loss functions

In this section we evaluate and compare the efficacy of the two loss functions defined in (3) and (4), in terms of two measures and the trade-off between them: (i) the total unweighted error in the original classification task; (ii) the error distribution as measured by the different elements of the confusion matrix at train and test time.

Specifically, the purpose of the following experiments is: (a) to evaluate and compare the ability of the proposed formulation to control the location of error, using standard deep learning methods and benchmark datasets; (b) test the influence of the trade-off parameter α in the above formulations (3) and (4); and (c) evaluate the trade-off between control of error location, and overall accuracy of the model.

3.1. Methods

In order to test the ability to control the location of error, we randomly select a varying number n of specific errors to be avoided. Each such error is defined by the identity of both the correct label and the wrong label (such as for example: "don't mistake a 2 for an 8"). In practice, this is done by sampling a random mask (Boolean matrix) of size k^2, with exactly n positive off-diagonal elements, where k is the number of classes. These n locations are henceforth called *the masked zone*.

The datasets we use for empirical evaluation in this section are MNIST and CIFAR10, where $k = 10$. For each combination of n in $\{10, 20, 30, 40, 50\}$ and the trade-off parameter α in $\{0, .1, .5, .9, .95, .99\}$, we trained 50 and 10 models (MNIST and CIFAR10 respectively). Thus we trained a total of 1500 MNIST models, and 300 CIFAR10 models. This design was repeated for the Bilinear and Log-Bilinear loss functions, leading to a total of 3600 models.

All models were trained using the open source Keras and TensorFLow (Abadi et al., 2015) software packages. In order to facilitate the training of a large number of models,

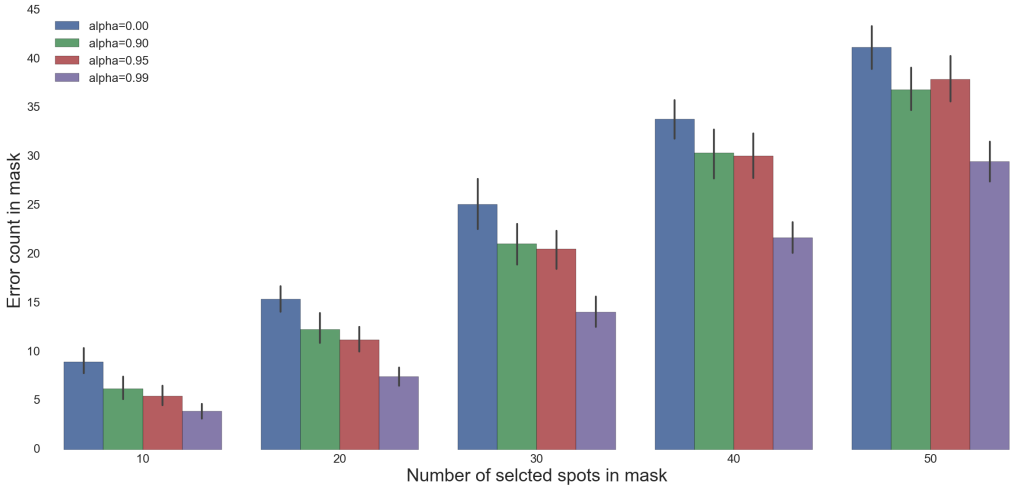

Figure 1: MNIST, Bilinear loss. Number of erroneous test examples in the masked zone (y-axis) as a function of the zone's size (x-axis), mean over the 50 repetitions per configuration with error bars showing the 95% confidence interval.

MNIST models were each trained for only 10 epochs, and CIFAR10 models for 300 epochs with an early stopping criterion.

3.2. MNIST dataset: results

The MNIST dataset (LeCun et al., 1998) is a benchmark dataset of small images (28×28 pixels) of hand-written digits. The data is divided into $55,000$ images in the training set, $5,000$ for validation, and $10,000$ images in the test set. In recent years, deep learning methods have been used successfully to reach almost perfect classification when using the MNIST dataset (see for example (Ciregan et al., 2012; Jarrett et al., 2009)).

The model we use in our experiments is a typical small model with two convolutional layers, followed by a single fully-connected layer. Max-pooling and dropout (Srivastava et al., 2014) are used following each convolutional layer. This model was selected for the current experiments primarily because of the short training time required to achieve satisfactory results (approx. 99.2% correct after 10 epochs), which allowed us to generate a very large number of models, as discussed in the methods section above.

Bilinear Loss.

Results are shown in Fig. 1. Clearly the number of test errors in the selected mask spots (the 'masked zone') is dramatically reduced when using our proposed loss function (3) with $\alpha > 0$, as compared to the baseline models ($\alpha = 0$). As expected, the total number of errors increases linearly with the size of the mask. At the same time the reduction in error count in the masked zone is attenuated by the value of the trade-off parameter α, with higher values of α leading to fewer errors in this zone. Noticeably, as the number n of selected

spots in the mask is increased, a larger value of α is necessary in order to retain the 82 reduction in the number of errors relative to the baseline.

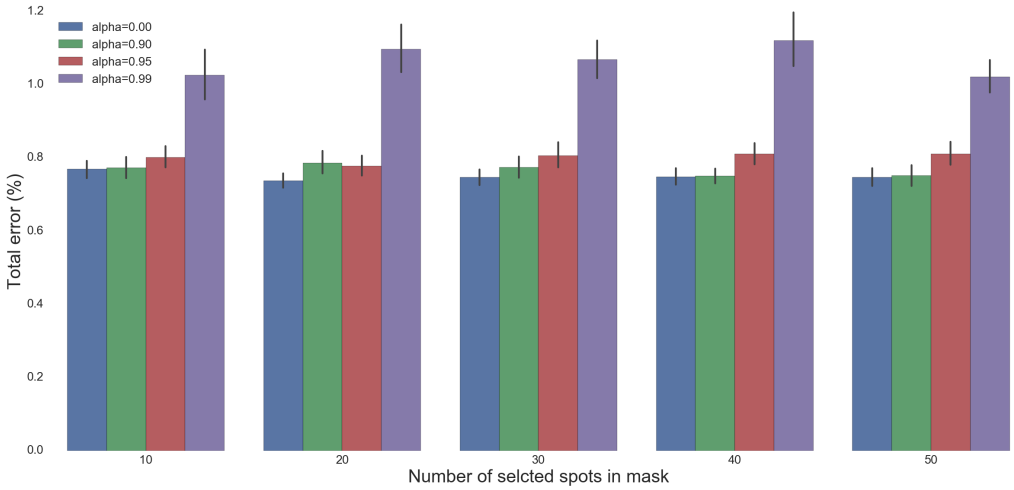

Figure 2: MNIST, Bilinear loss. Percent of overall model error, mean over the 50 repetitions per configuration with error bars showing the 95% confidence interval.

The overall model accuracy (Fig. 2) changes very little for all but the largest value of α. However, for $\alpha = .99$ we see a substantial increase in overall error. This value of α was also responsible for the most dramatic decrease in errors in the masked zone. Thus in this parameter value regime the network achieves an inferior overall solution, but succeeds in pushing most of the errors outside the masked zone.

Overall, for intermediate values of the trade-off parameter α, the number of errors in the masked zone is reduced dramatically, even for relatively large masks with as many as $40 - 50$ points, without an appreciable increase in overall error. This result demonstrates the feasibility of the proposed bilinear loss (when added to the regular cross-entropy loss), as a means of controlling the location of error without harming overall accuracy in deep learning models.

Log-Bilinear Loss.

The results for the log-bilinear loss are qualitatively similar to the bilinear loss presented above. The reduction in the number of errors in the masked zone (Fig. 3) relative to the baseline is larger than with the bilinear loss for small mask sizes ($n = 10 - 20$), and comparable for the larger masks. The overall model accuracy (Fig. 4) is, however, significantly worse than in the linear case, with small adverse effects showing already for small α values.

Overall, the log-bilinear loss, much like the linear version, is able to reduce the number of errors in an MNIST model in a selected zone. However, the reduction in error in the masked zone when using the log-bilinear loss is more substantial, at a price of a worse model overall; this harmful effect is slight for small values of α.

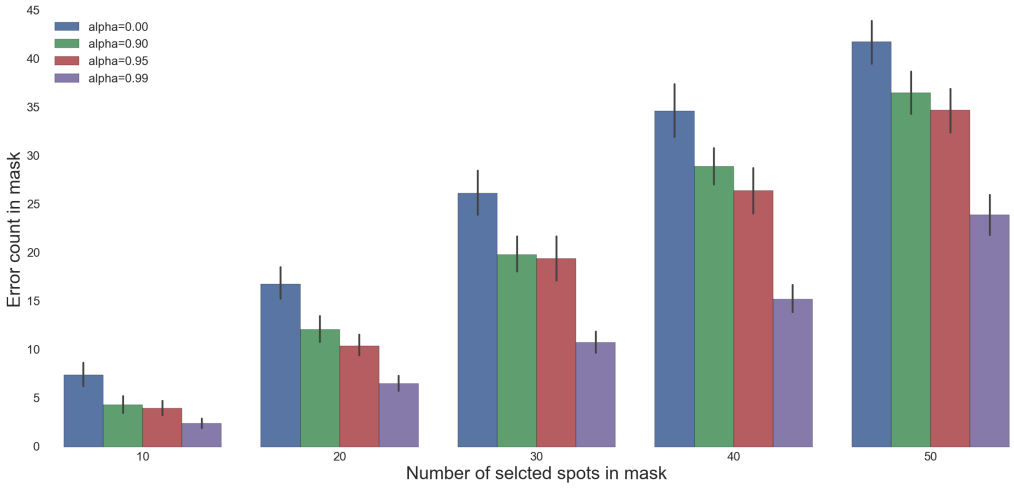

Figure 3: MNIST, Log Bilinear loss; see caption of Fig. 1.

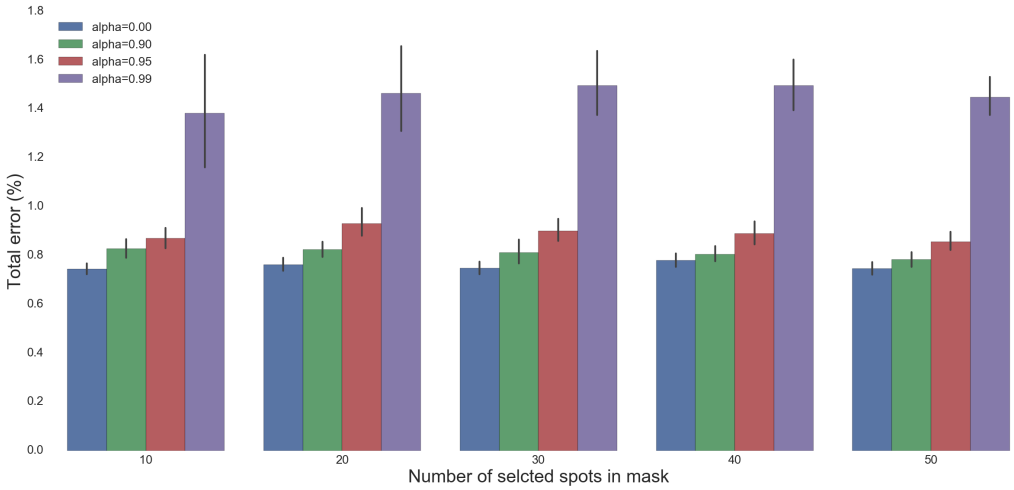

Figure 4: MNIST, Log Bilinear loss; see caption of Fig. 2.

3.3. CIFAR-10: results

The CIFAR-10 dataset (Krizhevsky and Hinton, 2009) is made up of $60,000$ small images (32×32 color pixels), each belonging to one of 10 classes (airplane, car, bird, cat, deer, dog, frog, horse, ship, truck). The data is divided into a training set of $50,000$ images, and a test set of the remaining $10,000$ images.

As in the MNIST case, the model we use was selected on the basis of typicality, accuracy, and relatively short training time to allow many models to be computed. Here, the model consists of three blocks of convolutional layers (each containing two convolutional layers, followed by max-pooling and dropout layers), and two fully-connected layers. Overall accuracy is approximately 92% after 300 epochs at most. (300 was set to be the maximal

number of epochs; early stopping was employed when convergence was achieved earlier, which was often the case.)

Bilinear Loss.

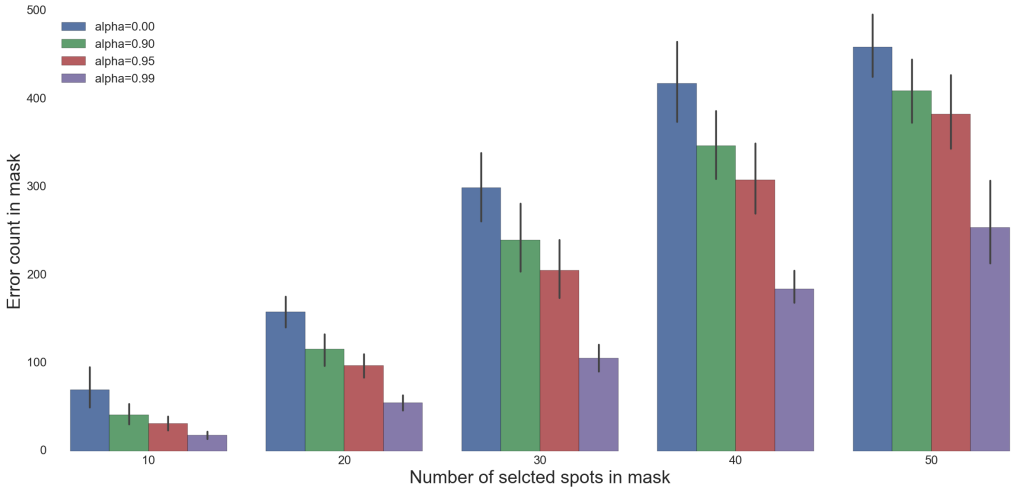

Figure 5: CIFAR-10, Bilinear loss. We plot the number of erroneous test examples in the masked zone (y-axis) as a function of the zone's size (x-axis), mean over the 10 repetitions per configuration with error bars showing the 95% confidence interval.

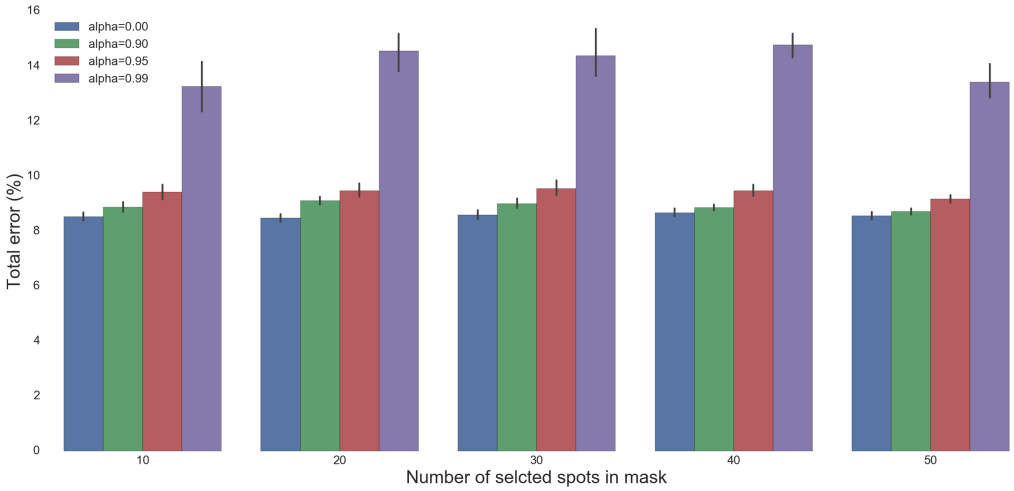

Figure 6: CIFAR-10, Bilinear loss. We plot the percent of overall model error, mean over the 10 repetitions per configuration with error bars showing the 95% confidence interval.

Results are shown in Fig. 5. Like before, the number of test errors in the masked zone is dramatically reduced when using our proposed loss function (3) with $\alpha > 0$, as compared to the baseline models ($\alpha = 0$).

Compared to the MNIST results, for a small mask of 10 locations, the reduction in error in the masked zone is appreciably larger for CIFAR-10. We attribute this to *compulsory errors* – errors which stem from true class overlap, and thus can't be overcome by changing the objective. Arguably, the MNIST dataset has a higher volume of these, explaining the better ability to control the error away from the mask when using the CIFAR-10 dataset.

The overall model accuracy (Fig. 6) changes very little for all but the largest value of α. Unlike the MNIST case, however, we see a real (albeit small) increase in overall error even for lower values of α. For $\alpha = .99$ we again see a substantial increase in overall error. This value of α was also responsible for the most dramatic decrease in errors in the masked zone. Thus in this regime of parameter values the network achieves an inferior overall solution, but succeeds in pushing most of the errors outside the masked zone.

Overall, for intermediate values of the trade-off parameter α, the number of errors in the masked zone is reduced dramatically, even for relatively large masks with as many as $40 - 50$ points, without drastically harming the overall performance of the model.

Log-Bilinear Loss.

The log-bilinear loss results show a large reduction in the error in the masked zone (Fig. 7), but at the same time an increase in overall model error (Fig. 8) already for small masks and smaller values of α.

We show data in this case (Fig. 7 and 8) for masks of size up to 30, and α values of up to 0.95. The log-bilinear loss is less effective than the bilinear loss already in this regime, and is not of practical use for this dataset with larger masks. It would seem that for this dataset the bilinear loss produces better results overall in terms of control of error on the one hand, while maintaining reasonable overall model accuracy on the other.

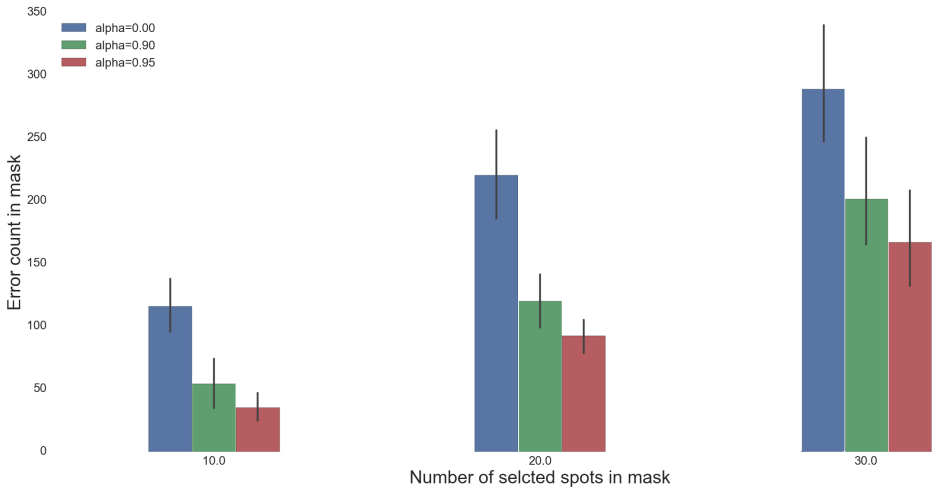

Figure 7: CIFAR-10, Log Bilinear loss; see caption of Fig. 5.

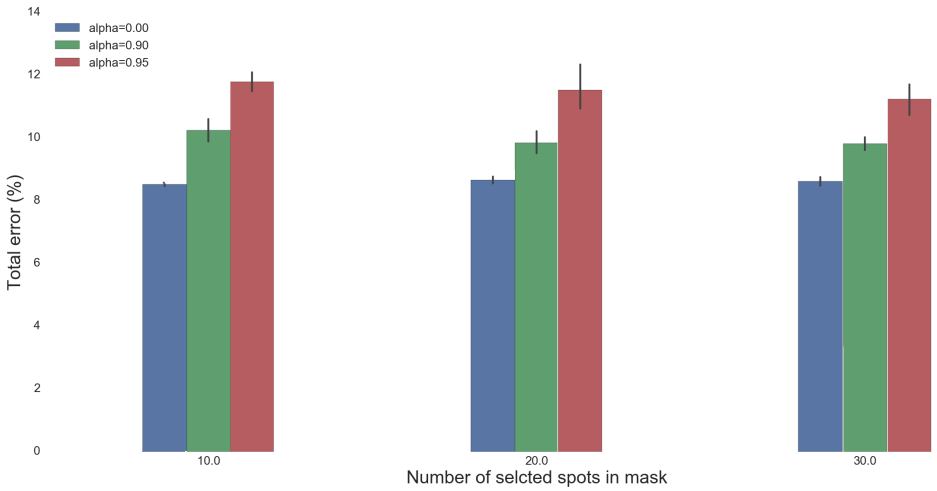

Figure 8: CIFAR-10, Log Bilinear loss; see caption of Fig. 6.

4. Conclusions

Often in real-world classification tasks, the cost of an error depends on the identity of the correct label as well as that of the misclassification target. The cross-entropy objective most commonly used in classification with deep learning models does not accommodate this.

In this paper we present the bilinear/log-bilinear loss functions, which directly add to the regular loss a component that depends on the true and wrongly assigned labels. We evaluate these formulations extensively using standard models and benchmark datasets.

We show that with the MNIST/CIFAR-10 datasets we are able to direct error away from a randomly chosen mask of up to 50 out of the 90 possible errors (the non-diagonal elements of the 10 by 10 confusion matrix), while preserving the overall model accuracy.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, and et. al. Zhifeng Chen. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL http://tensorflow.org/. Software available from tensorflow.org.

Francis R Bach, David Heckerman, and Eric Horvitz. Considering cost asymmetry in learning classifiers. *Journal of Machine Learning Research*, 7(Aug):1713–1741, 2006.

Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3642–3649. IEEE, 2012.

Pedro Domingos. Metacost: A general method for making classifiers cost-sensitive. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 155–164. ACM, 1999.

Kevin Jarrett, Koray Kavukcuoglu, Yann LeCun, et al. What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2146–2153. IEEE, 2009.

Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. Supervised machine learning: A review of classification techniques, 2007.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015.

Yoonkyung Lee, Yi Lin, and Grace Wahba. Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data. *Journal of the American Statistical Association*, 99(465):67–81, 2004.

Yi Lin, Yoonkyung Lee, and Grace Wahba. Support vector machines for classification in nonstandard situations. *Machine learning*, 46(1):191–202, 2002.

Dragos Dorin Margineantu, Thomas G Dietterich, et al. Learning decision trees for loss minimization in multi-class problems. Technical report, Corvallis, OR: Oregon State University, Dept. of Computer Science, 1999.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Yin Zhang and Zhi-Hua Zhou. Cost-sensitive face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(10):1758–1769, 2010.

Zhi-Hua Zhou and Xu-Ying Liu. On multi-class cost-sensitive learning. *Computational Intelligence*, 26(3):232–257, 2010.