Comparing Cross-Entropy, Hinge, and Squared Hinge Loss Functions in Image Classification

Stefan Dierauf, Margaret Stutz, Miquel Marti

DD2417 - KTH Royal Institute of Technology

Abstract. Convolutional networks have become increasingly important in the field of deep learning. They have been successful on many applications and therefore have become one of the most widely used tools for many different tasks such as speech recognition, natural language processing, and image recognition [1], [2]. Most deep learning models minimize the cross-entropy loss function for image classification tasks [1], [2], however studies imply that using hinge loss may lead to better results. We compare how loss functions affect the final accuracy and the convergence time of networks in the context of image classification on the CIFAR-10 dataset. The experimental results, with default training hyper-parameters, show a much faster convergence of the classification accuracy when using the cross-entropy loss, but do not necessarily show a better performance in terms of classification accuracy since the validation accuracy for the network trained with squared hinge loss seems not to have plateaued before the last epoch.

Keywords: ConvNet, Loss Function, Cross-entropy, Softmax, Hinge, Squared Hinge, CIFAR-10

1 Introduction

A convolutional network consists of convolutional layers, activation layers, and a loss function at the last layer (and possibly some regularization layers). The loss function measures how close the learned network is to the ground truth labels and is therefore a vital part of the network [3]. We found three studies that experiment with deep convolutional networks using cross entropy, squared hinge loss, and hinge loss for image classification tasks. All studies discovered that squared hinge loss and/or hinge loss performed better than cross-entropy on all datasets. We were intrigued by this and thought it would be interesting to see if we would get similar results on the CIFAR-10 dataset using a deep convolutional network that is known to perform well on CIFAR-10 with cross entropy loss.

In this study different loss functions will be studied in the context of deep convolutional neural networks for multi-class image classification. The task of the convolutional network is thus to assign a label from a predefined set of labels, namely the image class, to the input data, an image. The model built by training the convolutional network is a discriminative model in the sense that it computes

a score for each class that can be loosely linked to the probability of the given image belonging to the respective class, in the sense that a higher score means a higher probability of belonging to that class.

The data used to train the model is a set of equally-sized RGB images with labels where each RGB channel is represented as a 2D array of 32 bit floats from 0 to 1. The label is just the class number, which is transformed to a one-shot vector pointing to the class that the image belongs to.

The performance is measured with the accuracy rate of the classification (the number of correctly classified images divided by the total number of images in the set) both in the training set and in a validation set, which consists of novel data not used for training but with the same properties. No cross-validation is performed to choose the hyper-parameters (learning rate, weight decay, etc.) so there is a lot of room for improvement in this sense.

The loss functions studied - the hinge loss, the squared hinge loss and the cross-entropy loss - are defined next and depicted in Figure 1. Equation 1 defines the hinge loss. y_k is the score of class k, which when training for a sample of class i will be 1 if $k = i$ and -1 otherwise, and $\hat{y}_i$ is the estimated score value of class i and N the total number of classes.

$$L = \frac{1}{N} \sum_k \max(0, 1 - y_k \hat{y}_i) \tag{1}$$

Notice that this implementation of the hinge loss is a one-vs-all implementation.[4] The squared hinge loss is just the previously defined loss squared so that it is differentiable. Equation 2 defines it. The fact that it is differentiable allows faster training with larger learning rates and less epochs. [2]

$$L = \frac{1}{N} \sum_k \max(0, 1 - y_k \hat{y}_i)^2 \tag{2}$$

Finally, the cross-entropy loss is defined as in Equation 3. It gives more intuitive outputs as they can be interpreted as the probability of the input image to be classified as a given class.

$$L = -\frac{1}{N} \sum_i y_i \log(\hat{y}_i) \tag{3}$$

Here, $\hat{y}_i$ is the *softmax* function, defined in Equation 4, which takes the output scores of the network and normalizes them to have positive values so they can be used in the cross-entropy loss.

$$\hat{y}_i = \frac{e^{f_i}}{\sum_k e^{f_{ik}}} \tag{4}$$

Finally, the complete loss incorporates a *regularization loss* that helps to prevent overfitting to the training data by penalizing large weights and the *data*

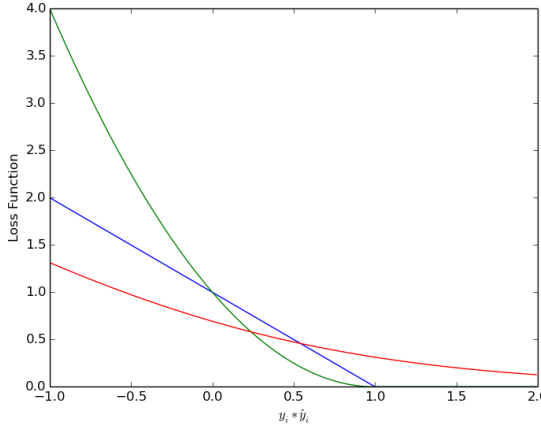

Fig. 1. Loss functions compared in this study. *Hinge loss* (blue), *Squared hinge loss* (green), *Cross-entropy loss* (red)

loss as defined in Equation 5. Parameter λ is usually chosen via cross-validation and defines the trade-off between both losses. [3]

$$L_{tot} = L + \lambda R(W) \tag{5}$$

2 Background

We came across two notable works comparing the performance of different loss functions in deep convolutional networks for various multi-class image classification tasks. The first study done by Yichuan Tang at the University of Toronto compares one-vs-all squared hinge loss function (L2-SVM) and cross-entropy (soft-max) loss function on 3 different datasets. The first dataset was for facial expression recognition with 7 classes, the second was MNIST, a handwritten digit classification dataset, and the third was the CIFAR-10 dataset. The study used different, yet simple, deep learning architectures for each dataset, which are explained below, comparing performance of the two loss functions on each.[1]

- Facial Recognition Architecture: simple convolutional network trained with stochastic gradient descent and momentum. The number of epochs used is not stated.[1]
- MINST Architecture: Fully connected mode by first performing PCA from 784 dimensions down to 70 dimensions, followed by 2 hidden layers of 512 units each followed by loss function" [1]. Trained with stochastic gradient descent and momentum for 400 epochs.
- CIFAR-10: The Convolutional Net part of both the model is fairly standard, the first C layer had 32 55 filters with Relu hidden units, the second C layer

has 64 5 5 filters. Both pooling layers used max pooling and downsampled by a factor of 2. The penultimate layer has 3072 hidden nodes and uses Deep Learning using Linear Support Vector Machines Relu activation with a dropout rate of 0.2. The Number of epochs not stated. [1]

The study found that for each dataset one-vs-all L2-SVM reduced the error more than cross-entropy. This study did not test hinge loss (L1-SVM) because it claimed to have found that L2-SVM performs better. [1]

A second study done at the University of California San Diegeo also studied, among other things, the effects of five different loss functions in deep nets on CIFAR-10 dataset. The five loss functions they investigated were cross entropy, one-vs-all squared hinge loss (L2-SVM), one-vs-all hinge loss (L1-SVM), one-vs-one hinge loss (L1-SVM), and rank loss. Figure 2 shows the architecture of the ConvNet used for this study. [5]

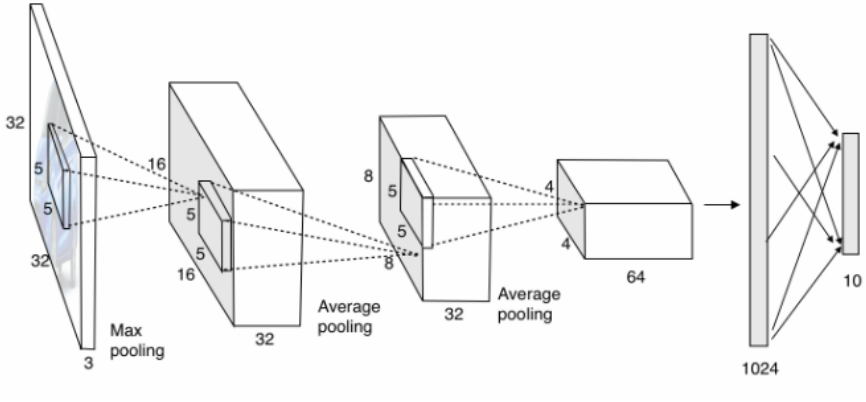

Fig. 2. Architecture of the network used in [5]

This study found that with 60,000 epochs one-vs-all L2-SVM and one-vs-all L1-SVM performed the best with almost identical test accuracy and one-vs-one L1-SVM and cross-entropy has lower test accuracies which were almost identical. One-vs-all L2-SVM and one-vs-all L1-SVM also converged the quickest. They also mentioned that although one-vs-all hinge loss does well on CIFAR-10, they found by some preliminary experiments that the One-Versus-All hinge loss does not work on CIFAR100, which is a 100 class extension of CIFAR10. [5]

Another study done at Florida State University compares the performance of squared hinge loss (L2-SVM) and cross-entropy (soft-max/logistic) loss functions, along with the Lorenz and Tangent loss functions, for binary image classification (rather than multi-class like the we are doing) in convolutional networks. The task was using CNNS for computer-aided abdominal and mediastinal lymph node detection and diagnosis (CAD) using CT images [2]. The study used the same CNN architecture with each loss function and 420 epochs and found that

soft-max loss and tangent had the worst performance, with squared hinge loss performing slightly better and Lorenz performing the best. It also mentions that squared hinge overfits the least. [2]

Regarding other computer vision tasks besides image classification, we came across a study on the impact of loss functions in the context of image processing. This study compared multiple loss functions, including hinge and squared hinge loss. This study found that training a CNN with hinge loss instead of squared hinge loss leads to improved image quality. [6]

3 Approach

We implemented the convolutional neural network in Python 2.7 using Keras [7], a library for succinctly describing and evaluating neural networks. Keras requires a tensor library to run the neural network, but allows the user to choose either Tensorflow [8] or Theano [9]. For hardware, we rented an Amazon Web Services g2.x2 server with an Nvidia GRID K520 GPU. This choice of hardware forced us to use Theano, as Tensorflow did not support GPU acceleration with the older version of CUDA that the GRID K520 supports. We were also not able to get cuDNN support working on AWS because of the older version of CUDA installed, but were able to get it working locally.

For our data, we used CIFAR-10. CIFAR-10 consists of 10 classes with 5,000 training and 1,000 test images each, for a total of 50,000 training and 10,000 test images. Each image is an RGB triplet of size 32x32. Each pixel in the image takes a 32-bit float value in the range 0 to 1. Data augmentation was used in the form of shifting images horizontally and vertically by at most 10%, and random horizontal flipping. Keras provides this functionality out of the box with the ImageDataGenerator class.

The model used has been adopted from University of Freiburg's "The All Convolutional Net" in order to have a known model that performed well on CIFAR-10. This model achieved a test error of 9.08% and 7.25% with and without data augmentation respectively.[10] The model is shown in Figure 3 and summarized in Listing 1.1.

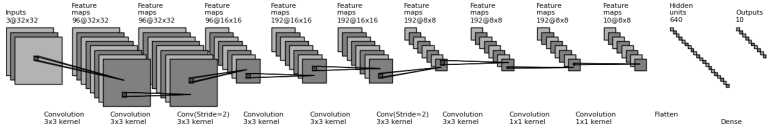

Fig. 3. Architecture of the network used. Model obtained from [10]

Listing 1.1. Model summary table with parameters per layer

Layer (type)	Output Shape	Param #
convolution2d_1 (Convolution2D)	(None, 96, 32, 32)	2688
activation_1 (Activation)	(None, 96, 32, 32)	0
convolution2d_2 (Convolution2D)	(None, 96, 32, 32)	83040
activation_2 (Activation)	(None, 96, 32, 32)	0
convolution2d_3 (Convolution2D)	(None, 96, 16, 16)	83040
activation_3 (Activation)	(None, 96, 16, 16)	0
convolution2d_4 (Convolution2D)	(None, 192, 16, 16)	166080
activation_4 (Activation)	(None, 192, 16, 16)	0
convolution2d_5 (Convolution2D)	(None, 192, 16, 16)	331968
activation_5 (Activation)	(None, 192, 16, 16)	0
convolution2d_6 (Convolution2D)	(None, 192, 8, 8)	331968
activation_6 (Activation)	(None, 192, 8, 8)	0
convolution2d_7 (Convolution2D)	(None, 192, 8, 8)	331968
activation_7 (Activation)	(None, 192, 8, 8)	0
convolution2d_8 (Convolution2D)	(None, 192, 8, 8)	37056
activation_8 (Activation)	(None, 192, 8, 8)	0
convolution2d_9 (Convolution2D)	(None, 10, 8, 8)	1930
activation_9 (Activation)	(None, 10, 8, 8)	0
flatten_1 (Flatten)	(None, 640)	0
dense_1 (Dense)	(None, 10)	6410
activation_10 (Activation)	(None, 10)	0

Total params: 1376148

4 Experimental results

We tested three loss functions, categorical cross-entropy, hinge, and hinge squared. Each was run with a stochastic gradient descent optimizer for 100 epochs. Our stochastic gradient descent optimizer had a learning rate of 0.01, a decay weight of 1e-6, and a Nesterov momentum factor of 0.9. With this parameters, it took around 7.5 hours to complete the 100 epochs for each of the cases, at a pace of about 260-280sec per epoch. The values for the hyper-parameters were decided *ad hoc* as they are the default values given in Keras documentation. K-fold cross-validation was not an option due to limited resources and time although it would definitely have helped to improve the performance.

In order to fairly compare the different loss functions we thought it was alright to keep the same values for the different cases. No test results were obtained as the test set was used as the validation set in order to have more data for the training. However, as the validation set has not been used to tune the hyper-parameters it was only used for testing the accuracy at the end of each epoch. The final test accuracy can be said to equal the validation accuracy for the last epoch.

In Keras weight regularization is applied in a per layer basis and was finally not included in our model as it was not clear which weight penalties to choose

and, as mentioned, neither resources nor time were available to determine them using cross-validation, which would have been needed to be done for each of the loss functions studied.

Figure 4 shows the loss score and accuracies for training and validation in the case of using the cross-entropy loss function. It can be seen how it plateaus around epoch 20 to around 85% validation accuracy, where it starts overfitting to the training set that finally achieves a 91% training accuracy, which continues to rise while the validation accuracy remains the same. In the loss plot similar results can be seen and it is more clear that around epoch 10 the loss score for the validation set diverges from the one for the training set, that continues to go down.

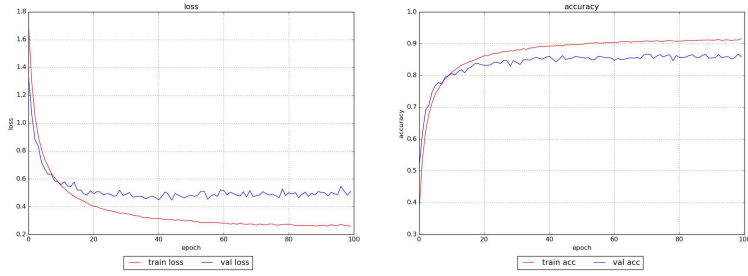

Fig. 4. Loss scores and accuracies for *training* (red) and *validation* (blue) data for the cross-entropy loss

Figure 5 shows the loss score and accuracies for training and validation in the case of using the standard hinge loss function. Both the loss scores and the classification accuracies for both sets vary quite linearly although with some discontinuities until around epoch 60, where it seems to start rising at a slower pace. After 100 epochs both accuracies and losses are at about the same level so there is no overfitting to the training data and the curve seems to keep rising. However, the final accuracy is around 55%, which is much smaller than the one obtained for the cross-entropy loss.

Figure 6 shows the loss score and accuracies for training and validation in the case of using the squared hinge loss function. Both the training accuracy and the validation accuracy rise fast until around epoch 40, when they start diverging, the model is overfitting to the data. However, they keep rising until epoch 100 giving a final accuracy of 87% and 82% for the training and validation sets respectively. The squared hinge loss clearly outperforms the standard hinge loss in terms of convergence speed and final accuracy, at least with the chosen parameters and number of epochs. However, the cross-entropy loss achieves a much faster convergence. It is not clear from our results whether it is also better in terms of the final validation accuracy, as when using the squared hinge loss

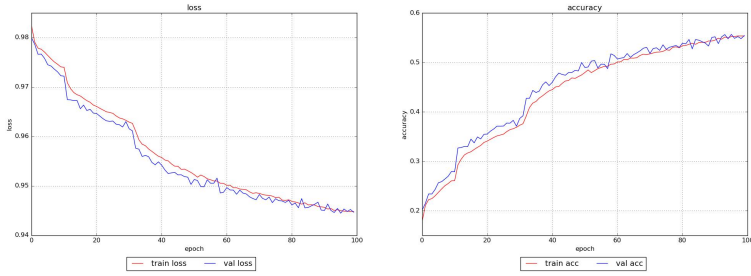

Fig. 5. Loss scores and accuracies for *training* (red) and *validation* (blue) data for the hinge loss

the shape of the curve suggests that it would have kept rising if the network had been trained for a larger number of epochs.

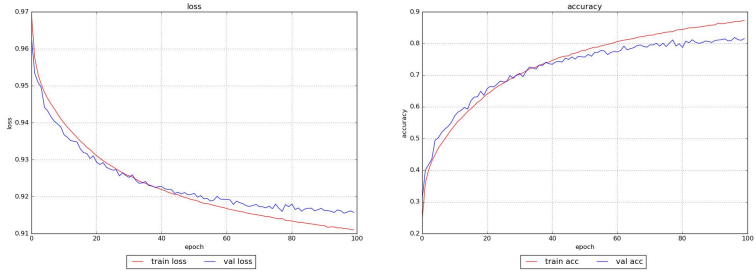

Fig. 6. Loss scores and accuracies for *training* (red) and *validation* (blue) data for the squared hinge loss

5 Conclusions

Our study shows that within a somewhat short amount of training (100 epochs) our deep convolutional neural network using the cross-entropy loss function reaches a higher accuracy on both training and validation set and converges the quickest. Squared hinge loss performed significantly better than standard hinge loss, most likely because standard hinge loss is not differentiable. However, since the validation accuracy of the net trained with squared hinge loss did not plateau within the 100 epochs of training, if we had trained for longer the accuracy could have continued to increase and could potentially have ended up with an accuracy equal to or higher than that of cross-entropy. We were not

able to run our experiments for more epochs because we were limited in terms of time and resources.

Our experiment results differ with the results of similar studies as most similar studies found that both squared hinge loss and standard hinge loss outperform and converge faster than cross-entropy [1] [2] [5]. There are multiple potential reasons for the difference in results such as hyper-paramater values, difference in architectures, absence of dropout (in our experiment), difference in regularization techniques, etc. We are also using significantly less epochs than the other studies. Further experimentation with hyper-parameters and more epochs is needed to make a stronger conclusion as to whether cross-entropy or squared hinge loss definitely leads to higher validation accuracy.

References

1. Tang, Y.: Deep learning using support vector machines. CoRR **abs/1306.0239** (2013)
2. Barbu, A., Lu, L., Roth, H., Seff, A., Summers, R.M.: An analysis of robust cost functions for cnn in computer-aided diagnosis
3. Karpathy, A.: Notes of cs231n: Convolutional neural networks for visual recognition (2016) Notes of the course at Stanford University.
4. Duan, K.B., Keerthi, S.S.: Which Is the Best Multiclass SVM Method? An Empirical Study. In: Multiple Classifier Systems: 6th International Workshop, MCS 2005, Seaside, CA, USA, June 13-15, 2005. Proceedings. Springer Berlin Heidelberg, Berlin, Heidelberg (2005) 278–285
5. Chen, S., Wang, Y.: Convolutional neural network and convex optimization
6. Zhao, H., Gallo, O., Frosio, I., Kautz, J.: Is L2 a Good Loss Function for Neural Networks for Image Processing? ArXiv e-prints (November 2015)
7. Chollet, F.: Keras. https://github.com/fchollet/keras (2015)
8. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015) Software available from tensorflow.org.
9. Theano Development Team: Theano: A Python framework for fast computation of mathematical expressions. arXiv e-prints **abs/1605.02688** (May 2016)
10. Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M.A.: Striving for simplicity: The all convolutional net. CoRR **abs/1412.6806** (2014)