Interclass Loss: Learning from Class Relationships

Albert Haque
Stanford University
ahaque@cs.stanford.edu

Abstract

Existing machine learning models are often trained as N-way classifiers to distinguish between classes. Loss is computed without referencing the graphical class structure. The class hierarchy is rarely used outside of test time performance evaluation. We propose the use of the interclass loss during the training phase. By introducing an interclass scaling penalty, we are able to leverage additional information during learning, namely class relationships. Our results show that we are able to beat the standard multiclass softmax loss using a coarse accuracy metric. Our method beats the standard multiclass loss by an F1 score of 2% and 10% for the training and test set, respectively.

1. Introduction

With the advent of "big data," the size of image and video data has dramatically increased in past years. The size and number of classes in image classification datasets has steadily increased. PASCAL VOC contains 20 classes [4], CIFAR-10 and CIFAR-100 contain 10 and 100 classes, respectively [13]. Caltech 101 [6] and Caltech 256 [26] contain 101 and 256 classes, respectively. Recently, ImageNet [3, 21] was launched with 1000 object classes. We can expect future datasets to grow in image size, training examples, and number of classes.

Image classification has traditionally attempted to predict an image class from a flat structure of class labels. However, there is one piece of information these classifiers are not utilizing: the class structure. It has been found that humans construct semantic meaning from visual images in a hierarchical manner [19].

In this paper, we propose a method to capture the hierarchical class structure and leverage this information during model training through the use of an *interclass loss* function. Our contributions are as follows: (i) we formulate an interclass loss function which is used during model training, (ii) we evaluate our method on a dataset containing phylogenetic relationships in the form of a tree, (iii) we generalize our method to work on relationships represented by

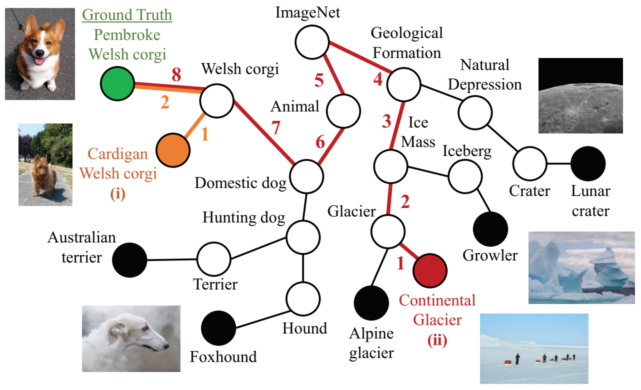


Figure 1: Two misclassifications for the same image. Prediction (i) has a smaller interclass distance to the ground truth than prediction (ii) and therefore should contribute less to the loss. In this example, interclass distance as the tree-distance between the predicted and ground truth label.

any arbitrary graphical structure, and (iv) we propose further studies and areas for future researchers to explore.

2. Related Work

We briefly review related work in hierarchical image classification in Section 2.1. We then survey recent advancements in convolutional networks in Section 2.2.

2.1. Hierarchical Image Classification

Related work has investigated hierarchical CNNs for coarse and fine class labeling. The authors in [29] adopt an approach that leverages an ensemble of CNNs - some of which are trained for coarse classification and some are trained for nuanced classification. There has also been prior work that investigated the use of inter- and intra-class distance distributions by Vailaya et al. in [26]. However, they employ a k-NN classifier.

Due to the computation complexity of many classification methods, there have been several studies that attempt to learn the class hierarchy to accelerate test-time prediction. In [17], the authors propose a method for learning class hierarchies but do not use class hierarchy as a learning tool. The authors in [8] propose a method to automatically learn the class taxonomy and thus reduce computational overhead for new query images.

Kim et al. leverage the hierarchical class information by constructing a path from the query image to the root [12]. The authors results are promising, as they are able to improve classification accuracy, even on images with multiple objects. However, the focus of their work is a sparse representation of each image from a reference image database. In this paper, we differ in several ways: (i) we leverage convolutional networks to learn feature representations instead of using bag of words histograms used in [12], and (ii) we use hierarchical class structure as part of the learning process as opposed to only during test-time. Similar to [12], the authors in [5] explore the use of "semantic gaps" between images using a pre-built concept ontology.

There have been significant research efforts to classify images in a hierarchical manner. This typically involves several classifiers trained for recognition at a specific level of granularity [32]. This has recently been extended to convolutional networks in [30] with positive results.

2.2. Convolutional Networks

Convolutional networks have had much success at image classification tasks. Although convolutional networks were proposed in the past, due to the availability of commodity computing and parallel processing techniques (*i.e.* GPUs), convolutional networks were revitalized by Alex Krizhevsky in [14]. Since then, numerous studies have reported state-of-the-art image classification results using convolutional networks [9, 22, 23, 28, 31].

Since it's inception, Dropout [25] has inspired several techniques to improve convolutional network performance. Goodfellow et al. propose the use of the maxout activation function [7]. A probabilistic maxout variant was then introduced in [24].

3. Interclass Relationships

Multiclass softmax loss is commonly used in image classification tasks. We briefly described the notion of interclass distance with examples from selected domains in Section 3.1 followed by our modification of the multiclass softmax in Section 3.2. We then formally define the interclass loss in Section 3.3.

3.1. Interclass Distance

Interclass distance, or in a graphical setting, the distance between two nodes, serves as the foundation for this paper's contributions. For datasets with a large number of labels, there is often a hierarchy associated with the dataset. In biology, phylogenetic trees are commonly used to measure similarity between organisms (see Figure 4). Edge weights are real valued numbers. For datasets lacking this fine-grained information, it is sufficient to use the tree-distance (*i.e.* length of the shortest path) between two nodes.

Given the graphical representation, with or without edge weights, it is possible to precompute the pairwise distance between all nodes (or classes). These distances can be cached in a matrix to provide $O(1)$ lookup time for future operations. This is important for computing the interclass scaling penalty, which we describe next.

3.2. Interclass Scaling Penalty

The intuition that not all errors are equal motivates our use of interclass distances while training. We modify the multiclass softmax loss function to account for interclass distances by using the *interclass scaling penalty*. The interclass scaling penalty uses the precomputed pairwise distance matrix described previously.

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K}\left[\mathbb{1}\{j = y_i\}\log(p_{ij})\left(1 + \frac{d(j, y_i)}{2D}\right)\right] \quad (1)$$

where $d(a, b)$ denotes the distance between class label a and class label b. D denotes the max depth of the tree and $\mathbb{1}\{\bullet\}$ is the indicator function. p_{ij} is the normalized class probability (from the softmax function) for class j on training example i. y_i denotes the ground truth label for training example i.

Our goal is to penalize more distant or "incorrect" predictions. We call the term $d(i, y_i)/2D$ the *interclass scaling penalty*. The interclass scaling penalty is a real valued number between 0 and 1. If the prediction is correct, the interclass loss becomes the standard multiclass softmax loss since the interclass scaling penalty is 1. Note that the indicator function forces the loss to be zero for correct predictions. If the prediction is incorrect, the loss for a specific training example i, is scaled linearly according to it's distance from the ground truth, as defined by the interclass scaling penalty.

3.3. Interclass Loss

We can generalize our interclass scaling penalty to accommodate non-linear scaling. For applications where very distant or "extremely incorrect" misclassifications have large negative costs, it may be preferable to unbound the interclass scaling penalty by using a monotonically increasing interclass scaling penalty such as $\log(\bullet)$ or $\exp(\bullet)$. In Equation 2, we reformulate the interclass loss as a function of a general interclass scaling function, denoted by $\Gamma(a, b)$ where a, b are class labels.

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K}\left[\mathbb{1}\{j = y_i\}\log(p_{ij})\Gamma(j, y_i)\right] \quad (2)$$

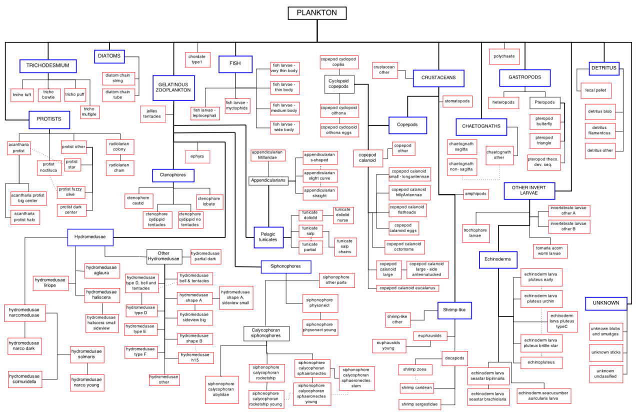


Figure 2: Class hierarchy of the Kaggle plankton dataset [1]. Plankton are grouped according to their lineage.

Note that the interclass scaling function $\Gamma(a,b)$ returns a constant. Therefore the backpropagation gradient computation is minimally affected. The remainder of this paper attempts to evaluate the interclass loss function on an image dataset where the class hierarchy is known.

4. Experiments

We introduce our dataset in Section 4.1 followed by a discussion of our convolutional network architecture in Section 4.3. We discuss our on-the-fly data augmentations in Section 4.2.

4.1. Dataset

We use labeled images from the 2015 National Data Science Bowl [1] collected by the Hatfield Marine Science Center at Oregon State University. The training and test set consists of 30K and 130K grayscale images, respectively. There are 121 distinct class labels. We are fortunate to have the class hierarchy for the dataset as shown in Figure 2. Images are rectangular and range in height and width from 40 pixels to 400 pixels. A sample of these images are shown in Figure 3.

Using the class hierarchy, we analyzed the tree depth (see Table 1). Most classes have a depth between 3 and 5. The max depth of the tree is 7.

4.2. Data Augmentation

The training set consists of approximately 30,000 labeled images ranging from 40 to 400 pixels in width/height. We resize each image (while preserving aspect ratio) to 256x256 pixels. The data then undergoes several affine transformations. Because plankton are spatially invariant, we can perform the following transformations:

1. Rotation: random between 0° and 360°

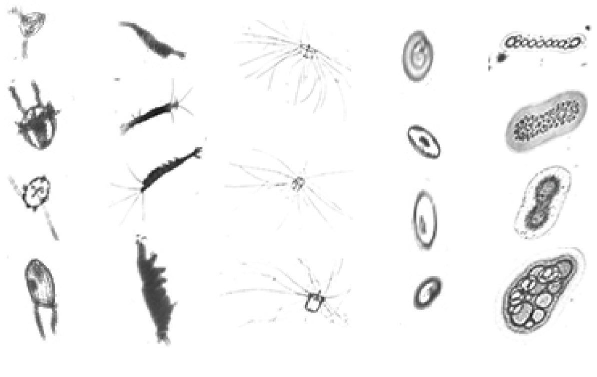


Figure 3: Example images from the dataset. Images belonging to the same column are part of the same class.

2. Translation: random shift between -20 and +20 pixels in the x and y direction
3. Scaling: random scale factor between 1/1.25 and 1.25
4. Flip: Bernoulli trial for a horizontal flip and another for the vertical flip

For each point $p \in P$ in the original image, we transform the point to $p' \in P'$ where P' is the transformed image. Let t_x, t_y denote the translation amount, s_x, s_y denote the scale factor, θ the angle of rotation, and x, y the point in the original image.

$$P' = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (3)$$

Unless otherwise specified, random refers to a uniform distribution. Because these data augmentations can be done in a single step (see Equation 3), we perform the transformations in real-time during training. These "on-the-fly" affine transformations are done at the data layer before the network forward propagates a new input. This also reduces overfitting due to the diversity of input images (*i.e.* the same image is never seen twice).

Table 1: Dataset Class Structure. Data includes parent nodes which are not actual class labels. There are 121 class labels and 54 intermediate (parent) nodes.

Depth	# Classes	Percent
0	1	0.8%
1	4	2.0%
2	16	9.1%
3	45	25.7%
4	36	20.6%
5	40	22.9%
6	26	14.9%
7	7	4.0%

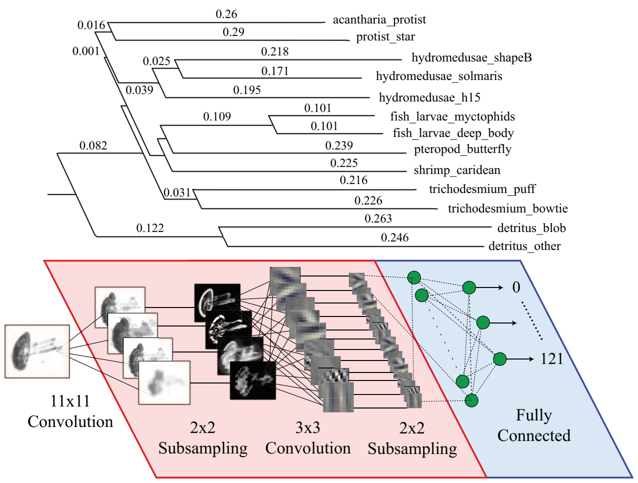


Figure 4: Our model. We employ standard CNN architectures which reference the class relationships for loss computation.

4.3. Models

We use the standard Caffe Reference network [11]. We create a new Caffe layer, namely the interclass loss, which scales the loss contribution of each softmax output node j by the inter-class distance between class j and the true class y_i. This allows us to accumulate larger losses for incorrect, biologically distant predictions. An overview of our convolutional network is shown in Figure 4.

In addition to the Caffe Reference network, we test three other network architectures. These are listed in Table 2. We experiment with the input image size as well as the kernel size of the first convolutional layer.

Leaky Rectified Linear Units (LReLU). We use a leaky rectified linear unit with $a = 0.01$ [10]. These have been shown to outperform the standard ReLU nonlinearity in [18, 16] by avoiding zero gradients.

Dropout & Dropconnect. We employ dropout as described in [25] and DropConnect as described in [27]. We use a dropout probability of 0.4

Loss Function. We use the softmax layer to generate normalized class probabilities. These are then into one of two loss functions: (i) standard softmax loss and alternatively, (ii) our interclass loss function as described in Section 3.

4.4. Performance Metrics

We use two metrics to evaluate our model performance: (i) the top-1 classification accuracy and (ii) the average predicted interclass distance. The average interclass distance is defined in Equation 4 where $\hat{y}_i$ is the predicted label for training example i and y_i is the ground truth:

$$\frac{1}{N}\sum_{i=1}^{N} d(\hat{y}_i, y_i) \tag{4}$$

We examine metric (i) due to it's prevalence in image classification tasks. However, we do not expect our model to excel in this metric (see Section 5). We evaluate metric (ii) as this more accurately measures the effect of interclass loss.

Table 2: Network Architectures. All convolution layers are followed by a leaky rectified linear unit with a negative slope of 0.01.

A	B	C	D
img64	img64	**img128**	**img128**
conv3-128	**conv5-128**	**conv10-128**	**conv15-128**
maxpool			
conv5-256	conv5-256	conv5-256	conv5-256
maxpool			
conv3-384	conv3-384	conv3-384	conv3-384
conv3-384	conv3-384	conv3-384	conv3-384
conv3-256	conv3-256	conv3-256	conv3-256
maxpool			
FC-4096			
dropout(0.4)			
FC-121			
softmax			

4.5. Hardware and Software

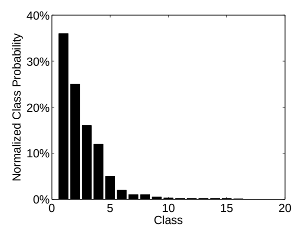
Software. We use Caffe [11], GraphLab [15], and NVIDIA's cuDNN engine [2] for convolutional network implementations. Offline affine transformations were done using a single transformation with scikit-learn [20] and the Python Imaging Library (PIL)[1]. Online "on-the-fly" transformations are applied using a slight modification to Caffe's data blob layer. All class distances are computed before training and are stored in main memory as a pairwise distance matrix $D \in \mathbb{R}^{K\times K}$ where K is the number of class labels.

Hardware. We use Amazon EC2 g2.2xlarge instances equipped with one NVIDIA Kepler GK104 containing 1536 cores and 4 GiB of GDDR5 memory for final model training and testing. Configuration and preliminary results were run on a single workstation with a single GTX 750 Ti (2 GiB GDDR5 memory).

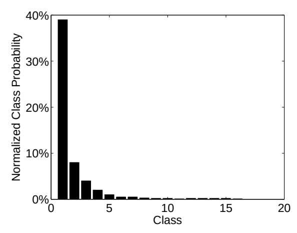
5. Results & Discussion

Before moving to a discussion into the results, we propose a few hypotheses. First, we expect that using the class

[1]`http://www.pythonware.com/products/pil/`



(a) Multiclass Softmax



(b) Interclass Loss

Figure 5: Illustration of shifted probability mass. For the same input image, the interclass loss moves the probability mass away from "distant" classes. We sort the probabilities in decreasing order. Both networks were trained with the same number of epochs.

distance will improve performance if accuracy is based on top-5 hit rate. However, by definition, accuracy only takes into account the top-1 hit rate. We hypothesize that our model will improve the average interclass distance. That is, we expect the interclass loss, on average, to produce predictions "closer" to the ground truth than the standard multiclass softmax loss.

Below, we discuss results for both a standard convolutional network with an unmodified softmax output and our method (modified softmax output / interclass loss).

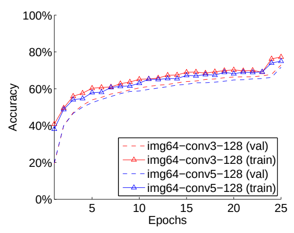
5.1. Shifted Probability Mass

We trained a set of convolutional networks (specific architectures are listed in Table 2) and fixed the number of training epochs to 100. Figure 5 shows the normalized probabilities for the predicted class labels after passing a single test image through our network. We sort the output probabilities in order from largest to smallest. We can see that the model trained using the multiclass softmax loss generates a "fatter" distribution (see Figure 5a) where the probability mass is located in the top few classes. If we look at Figure 5b, our model penalized the classes further away from the ground truth. As a result, most of the probability mass shifted to the ground truth class.

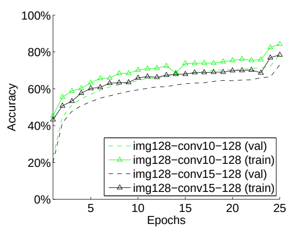
To quantify this phenomenon, we use kurtosis and skewness to analyze the output normalized class probabilities. Kurtosis measures the "peak-ness" of a distribution while skewness measures the asymmetry of the distribution (*i.e.* how long each tail is). It is formally defined in Equation 5.

$$\kappa = \frac{\sum_{j=1}^{K} (p_j - \overline{p})^4}{\left(\sum_{j=1}^{K} (p_j - \overline{p})^2\right)^2} \tag{5}$$

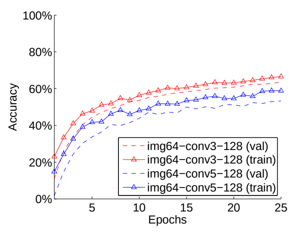
where K is the number of classes and p_j is the normalized class probability j. If we compute the average kurtosis for our method using interclass loss, we have 4.803. The average kurtosis for the standard multiclass softmax loss is 3.96. This confirms that our interclass loss moves the mass to fewer classes.



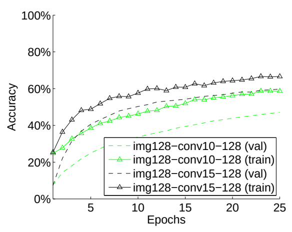
(a) Multiclass Loss (img64)



(b) Multiclass Loss (img128)



(c) Interclass Loss (img64)



(d) Interclass Loss (img128)

Figure 6: Top-1 Classification Accuracy

A large positive skewness value indicates that a small amount of probability is spread over a large number of classes. We expect this to be the case for our method. The multiclass loss generates an average skewness of 11.32 while the average skewness for our method is 28.9.

5.2. Classification Accuracy

We now turn our analysis to the top-1 classification accuracy. We trained a total of 8 networks. We experimented with varying the input image size and kernel size. Results are shown in Figure 6.

Receptive Field Ratio We compute the ratio of the conv1 receptive field to the input image area. The purpose of this analysis is to understand how and why input image size and kernel size affect classification accuracy. We define the receptive field ratio R as:

$$R = \frac{HH \times WW}{H \times W} \tag{6}$$

where $HH = WW$ is the kernel size and H, W are the input image dimensions. We show the results in Table 3.

For the standard multiclass loss, our empirical results show that a larger image with a larger filter size outperform a smaller image with a smaller filter size. We now analyze the ratios of the effective receptive fields for these two experiments. That is,

Table 3: Comparison of receptive field ratios. We use this to "normalize" between differences in image and kernel size.

Input	Kernel	Image Area	Kernel Area	Ratio
64	3	4096	9	0.21%
64	5	4096	25	0.61%
128	10	16384	100	0.61%
128	15	16384	225	1.37%

Returning to Figure 6, we can see that img128-conv15-128 performs the worst among our models. A potential explanation is its receptive field ratio. Each kernel sees over 1% of the image. Compare this to the other models which see 0.22%, 0.61%, and 0.61%. On the other hand, the model with the smallest receptive field ratio is not the best performing.

Interclass vs Multiclass Softmax Loss We now turn our attention to the comparison between the interclass loss and standard multiclass softmax loss. As seen in Figure 6, the interclass loss underperforms compared to the standard multiclass loss. However, we expected this result. If we compare the average interclass distance, that is, the distance of all test set predictions to their ground truth values, we find that the interclass loss outperforms the multiclass loss. Using the metric defined in Equation 4, we find our model achieves an average interclass distance of 0.648 while the standard multiclass loss resulted in an average distance of 0.821. This supports the claim that the interclass loss produces errors that are "closer" to the true value. We further quantify this in the next section.

5.3. Accuracy at Multiple Levels

To further validate our claim that the interclass loss produces errors that are "closer" to the true value, we train the two models normally. However, to compute accuracy, we force all classes to be within a certain depth level. That is, we change the test-set labels such that no label is beyond 2 levels away from the root class. This is a relatively coarse label. We select level 2 based off the dataset class structure in Table 1.

If we look at the confusion matrices in Figures 7 and 8, we can see that the standard softmax loss has a "cleaner" confusion matrix. This is expected since the standard softmax has a higher training accuracy. If we analyze the coarse-level confusion matrices, we can see that the interclass loss is not as noisy as the multiclass softmax. We quantify the coarse-level class performance by analyzing the F1-scores for each of the eight confusion matrices. The F1 scores are shown in Table 4

The interclass loss enjoys an increase in F1 score as we move the classification level from tree-depth-7 (fine) to tree-depth-2 (coarse). Although the standard softmax loss also experiences an increase in F1 score as we move from fine to coarse, the interclass F1 score is greater than the standard multiclass softmax in the coarse case, for both the training and test sets.

Table 4: F1 Scores for Multiclass Loss and Interclass Loss. Fine-level is computed on the full class-depth of 7. Coarse is computed on a class-depth of 2.

	Train		Test	
	Fine	Coarse	Fine	Coarse
Standard	0.49	0.76	0.47	0.64
Interclass	0.41	0.78	0.41	0.74

6. Conclusion

In this paper, we proposed the use of interclass distance during the training phase of a convolutional network for image classification. We evaluated the interclass loss function which yields positive results when minimizing total interclass distance. We are able to achieve higher F1 scores for coarse-level accuracy metrics. Future work can explore the use of interclass loss on standardized datasets such as ImageNet. The degrees of separation (or tree-distance) can be used as the interclass distance. For further qualitative analysis, insights may be gleaned by analyzing differences in model weights for the two models. Additionally, a comparison of accuracy and F1 score at different tree-depths (*i.e.* different granularities) would prove useful for future research studies.

References

[1] Inaugural national data science bowl. `http://www.datasciencebowl.com/`. Sponsored by Kaggle and Booz Allen Hamilton. 2015.

[2] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer. cudnn: Efficient primitives for deep learning. *CoRR*, abs/1410.0759, 2014.

[3] J. Deng, K. Li, M. Do, H. Su, and L. Fei-Fei. Construction and Analysis of a Large Scale Image Ontology. Vision Sciences Society, 2009.

[4] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[5] J. Fan, Y. Gao, and H. Luo. Hierarchical classification for automatic image annotation. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 111–118. ACM, 2007.

[6] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental

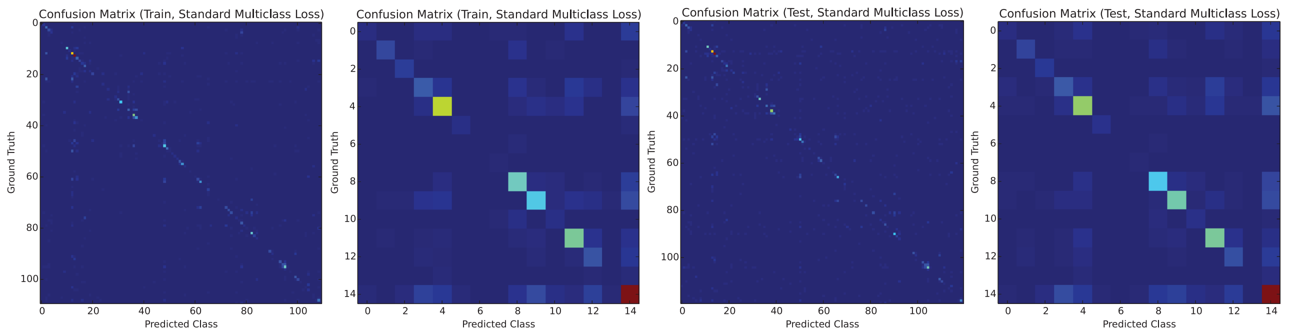


Figure 7: Confusion Matrices for Standard Multiclass Softmax Loss

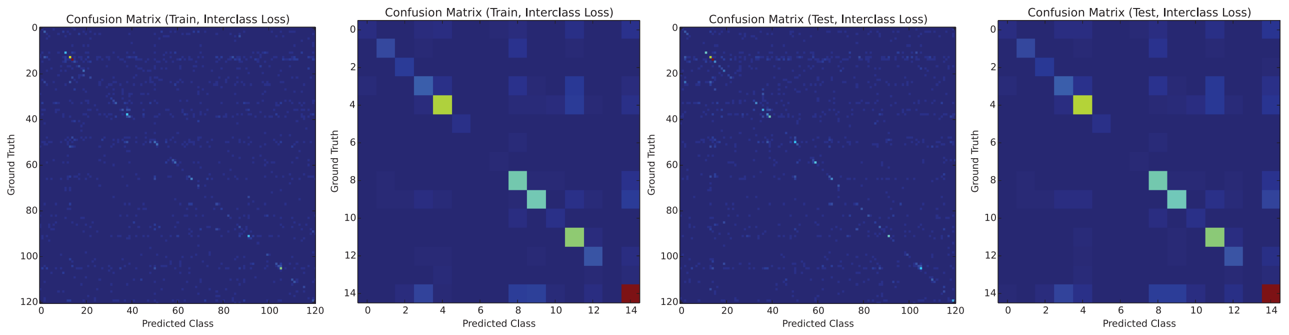


Figure 8: Confusion Matrices for Interclass Loss

bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 106(1):59–70, 2007.

[7] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. *arXiv preprint arXiv:1302.4389*, 2013.

[8] G. Griffin and P. Perona. Learning and using taxonomies for fast visual categorization. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *arXiv preprint arXiv:1406.4729*, 2014.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *arXiv preprint arXiv:1502.01852*, 2015.

[11] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[12] B.-s. Kim, J. Y. Park, A. C. Gilbert, and S. Savarese. Hierarchical classification of images by sparse approximation. *Image and Vision Computing*, 31(12):982–991, 2013.

[13] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Computer Science Department, University of Toronto, Tech. Rep*, 1(4):7, 2009.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[15] Y. Low, J. E. Gonzalez, A. Kyrola, D. Bickson, C. E. Guestrin, and J. Hellerstein. Graphlab: A new framework for parallel machine learning. *arXiv preprint arXiv:1408.2041*, 2014.

[16] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30, 2013.

[17] R. Mittelman, M. Sun, B. Kuipers, and S. Savarese. A bayesian generative model for learning semantic hierarchies. *Frontiers in psychology*, 5, 2014.

[18] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

[19] S. E. Palmer. *Vision science: Photons to phenomenology*, volume 1. MIT press Cambridge, MA, 1999.

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[21] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge, 2014.

[22] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization

and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[23] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[24] J. T. Springenberg and M. Riedmiller. Improving deep neural networks with probabilistic maxout units. *arXiv preprint arXiv:1312.6116*, 2013.

[25] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[26] A. Vailaya, A. Jain, and H. J. Zhang. On image classification: City images vs. landscapes. *Pattern Recognition*, 31(12):1921–1935, 1998.

[27] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1058–1066, 2013.

[28] T. Xiao, J. Zhang, K. Yang, Y. Peng, and Z. Zhang. Error-driven incremental learning in deep convolutional neural network for large-scale image classification. In *Proceedings of the ACM International Conference on Multimedia*, pages 177–186. ACM, 2014.

[29] Z. Yan, V. Jagadeesh, D. DeCoste, W. Di, and R. Piramuthu. HD-CNN: hierarchical deep convolutional neural network for image classification. *CoRR*, abs/1410.0736, 2014.

[30] Z. Yan, V. Jagadeesh, D. DeCoste, W. Di, and R. Piramuthu. Hd-cnn: Hierarchical deep convolutional neural network for image classification. *arXiv preprint arXiv:1410.0736*, 2014.

[31] M. D. Zeiler and R. Fergus. Stochastic pooling for regularization of deep convolutional neural networks. *arXiv preprint arXiv:1301.3557*, 2013.

[32] A. Zweig and D. Weinshall. Exploiting object hierarchy: Combining models from different category levels. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.