EMBEDDING SEMANTIC RELATIONSHIPS IN HIDDEN REPRESENTATIONS VIA LABEL SMOOTHING

Anonymous authors
Paper under double-blind review

ABSTRACT

In recent years, neural networks have demonstrated their ability to learn previously intractable problems, particularly in the field of computer vision. Although classification accuracy continues to increase for the most challenging benchmark datasets, model efficacy evaluations typically focus on raw accuracy results without a consideration for the types of errors being made. Further, most networks are trained such that classes are treated as separate, unrelated categories without any notion of higher order semantic relationships. This work shows a simple approach for embedding semantic relationships into learned representations via category-aware label smoothing. Using MNIST experiments, we demonstrate that preferable error profiles can be enforced this way and that underparameterized networks without the capacity to achieve reasonable raw accuracy are still capable of learning an effective model when relative error cost is taken into consideration. Additionally, we embed hierarchical information into CIFAR-10 target vectors to show that it is possible to enforce arbitrary class hierarchies using this method. Further, we use a new method for analyzing class hierarchy in hidden representations, Neurodynamical Agglomerative Analyisis (NAA), to show that latent class relationships in this analysis model tend toward the target vector relationships as the data is projected deeper into the network.

1 INTRODUCTION

Neural networks perform extremely well across many classification tasks, many of which were previously untractable, which is why they have become ubiquitous in many industry and research applications. An infallible classification model is, however, not possible. Therefore, the type of errors a model is likely to make are often as important as how many errors the model makes - e.g. mistaking one breed of dog for another will have a much less significant impact on a model outcome than mistaking a dog for a pylon in the case of an autonomous vehicle.

From the perspective of a human, such errors seem arbitrary. We see the world through a semantically-driven framework, grouping objects into categories on different hierarchical levels. We have even made attempts to formalize these object categories into formal ontological data bases such as Wordnet (Miller (1995)). A general problem in artificial intelligence today is how this structured knowledge about the world can be incorporated into machine learning models in a meaningful and practical way. This way, in addition to optimizing the class accuracy we would also be able to optimize the error profile such that the types of errors made are preferrable in terms of the practical outcomes of the system. This would enable autonomous systems to be designed with routines that consider error profiles in how the agent behaves, thus adding a layer of error-correcting redundancy in the system.

Classification models are typically trained in a supervised fashion, where input data is mapped onto one-hot vectors identifying class labels in the output. The simplicity of the approach, and success up to this point, explain its popularity. However, this approach implicitly dictates that every class label in its vector encoding is orthonogal and equidistant to all others. We argue that incorporating semantic relations between labels leads to hidden representations that are more analogous to the way we view the world, in addition to enabling the enforcement of preferable error profiles.

Our key contribution is to use a smooth target vector representation that adheres to predefined relations between label classes. Using a notion of class distance based on the cross-correlation matrices of class-sorted network activations developed in Marino et al. (2020), we demonstrate that semantically meaningful representations can be learned without loss of classification performance on the MNIST (Lecun et al. (1999)) and CIFAR-10 (Krizhevsky et al. (2009)) benchmark datasets. More generally, we show that representation similarity in the hidden representation tend towards the spatial orientations in the target space. This also means that what the "semantic relationship" is exactly can be decided by the designer at training time.

2 Related Work

Label smoothing has recently been shown to provide gains in various tasks. In Pereyra et al. (2017), label smoothing in combination with confidence penalties are shown to have a regularizing effect across multiple tasks, and show that in some cases label smoothing is equivalent to a confidence penalty. DisturbLabel, introduced by Xie et al. (2016), uses incorrect labels as a form of regularization by randomly changing labels to the wrong class with a small probability. This was found to improve generalization by introducing noise into the loss function. Recently, Müller et al. (2019) demonstrated that label smoothing improves model calibration but hurts in the case of model distillation.

The idea of exploiting semantic hierarchies for various reasons has been explored recently by several works. Peterson et al. (2019) uses uncertainty in human categorization in order to generate labels that reflect the types of errors a human would make. This approach is shown to have attractive properties in terms of model generalization and robustness against adversarial attacks. In contrast, we allow for application-specific design of class hierarchy rather than attempting to match human performance. Hierarchical multi-class labeling is also a relatively well-explored topic (Vens et al. (2008), Zhang et al. (2017), Wehrmann et al. (2018), Ren et al. (2014)). Most analogous to our approach, Peterson et al. (2018) explicitly use multiple labels on image data in order to generate hierarchical visual representations that embed a hierarchical structure into the latent representation. However, their approach relies on a more complex training routine that trains on each label separately and thus introduces a computational cost this work avoids by simply embedding the desired relationships into one set of labels. We simply augment single target vectors rather than introducing multi-labels.

3 Methods

3.1 Semantic Target Smoothing

Generating target vectors for semantic classes is essentially a mapping between a semantic category and a point in N-dimensional space, where N is the dimensionality of the classification layer. The validity of one-hot encoding from a probabilistic perspective is obvious - a thing is itself with a probability of one and something else with a probability of zero. However, the goal of a particular machine learning system is practical, not theoretical, and so whether or not the target vectors are consistent with a particular interpretation as a probability distribution is secondary in practice. What we are interested in is how spatial relationships in the target space impact class relationships in the hidden space. In order to do this, we defined semantically-meaningful target vectors for the MNIST and CIFAR-10 datasets. We define a separate component for each class target that embeds our preferred class relationships. The target vector used for training is then calculated as a weighted sum of the one-hot vector and the relational component vector, with the amount removed from the true class category considered as a smoothing parameter. The training label is then calculated according to the formula

$$\mathbf{y} = \sigma\hat{\mathbf{y}} + (1 - \sigma)\mathbf{y_0} \tag{1}$$

where $\hat{\mathbf{y}}$ is the relational component of the target defined according to (2) and $\mathbf{y_0}$ is the familiar one-hot target vector.

3.1.1 MNIST Targets

The MNIST dataset is convenient in that there is an obvious way to consider the cost of an error. If a handwritten digit classifier were to be used in an application, it would be likely that mistaking a

Table 1: Hierarchies used to define targets in for the CIFAR-10 dataset using equation (4) - class structures for the first three groupings were built by randomly sampling subgroups of cardinality two or three from the 10 class categories . Networks were trained varying the smoothing factor, σ, in order to demonstrate the ability to embed information about root class as well as superclass into the smoothed labels. Grouping four was chosen as a semantically-meaningful hierarchy while still forcing some texturally dissimilar classes together.

	cluster 1	*cluster 2*	*cluster 3*	*cluster 4*
grouping$_0$	frog	cat	ship	horse
	truck	dog	bird	automobile
	airplane	deer		
grouping$_1$	bird	airplane	truck	dog
	automobile	frog	horse	cat
	deer	ship		
grouping$_2$	airplane	deer	frog	bird
	automobile	cat	horse	dog
	ship	truck		
	mass transport	**personal transport**	**domestic animals**	**other animals**
grouping$_4$ (semantic)	airplane	automobile	cat	bird
	ship	horse	dog	deer
	truck			frog

'4' for a '3' would be preferable to mistaking it for a '9'. However, training on one-hot vectors with a cost function based on categorical cross-entropy treats these two errors as equivalent. In order to account for this in the target space, we embedded the desired class relationships using the relational components of the target vector for digit i according to -

$$\hat{y}_{ij} = \begin{cases} \lambda_i |i-j|^{-1} & i \neq j \\ 0 & i = j \end{cases} \tag{2}$$

where $\hat{y}_{ij}$ is the value at index j for target vector i, i and j being values from zero to nine. Intuitively, we are making the amount in each error class inversely proportional to the distance between the error class value and the true class value. The entry for $i = j$ is set to zero is because this value is managed by the one-hot component, which is summed with the relational component to form the target at training time. The scaling factor

$$\lambda_i = \frac{1}{\sum_{j=1}^{9} \hat{y}'_{ij}} \tag{3}$$

is a normalizing factor enforcing that the sum of the values in the relational component vector sum to one, with $\hat{y}'_{ij}$ referring to the non-normalized entries. This is important since we are still using a softmax activation function on the prediction layer. Since both target components sum to one, the smoothing parameter σ can be used to generate the relational targets according to equation (1). The smoothing parameter can be $\sigma \in [0, 1]$ in theory. However, once erroneous class values are comparable to the true class value accuracy results begin to drop off. Therefore in this case σ is varied on the interval $[0, 0.8]$. Note that $\sigma = 0$ corresponds to the traditional one-hot case.

3.1.2 CIFAR-10 TARGETS

The CIFAR-10 dataset is composed of 10 classes - *airplane, automobile, bird, cat, deer, dog, frog, horse, ship truck*. In order to demonstrate the ability to learn arbitrary class hierarchies, four different hierarchies were generated randomly consisting of four superclasses each composed of two or three subclasses as shown in Table 1.

The reason for using randomized hierarchies is that, unlike other works which mostly focus on human-minded hierarchies, our focus is the ability to enforce arbitrary hierarchies. Although the notion of human-minded hierarchy is of central important in psychology and cognitive science, in any particular system the optimal hierarchy will be application-specific. Therefore, we generated three hierarchies by randomly sampling the object classes (*grouping*$_0$-*grouping*$_2$) and also hand-crafted one hierarchy (*grouping*$_3$) to be semantically meaningful in the human sense while still including at least one group with object categories that are texturally distinct. The CIFAR-10 relational components analogous to (2) were defined according to

$$\hat{y}_{ij} = \begin{cases} (n-1)^{-1} & j \in C_i \text{ and } i \neq j \\ 0 & \text{else} \end{cases} \tag{4}$$

where C_i corresponds to the superclass associated with sublass i and n is the number of subclasses within that superclass. That is, we are simply taking an amount from the one-hot component dictated by σ and distributing it evenly to the other classes in the cluster. The final target vector is then calculated as a weighted sum of the vector defined by (4) and the one-hot target as shown (1).

3.2 Neurodynamical Agglomerative Analysis

Neurodynamical Agglomerative Analysis (NAA) was introduced by Marino et al. (2020) as a way of modeling class relationships in neural networks. The method uses the cross-correlation matrix of class-sorted network activations as a lower-dimensional approximation of how the class is being represented by the network. That is, given a matrix of n-dimensional vectorized activation data for a particular class, Z, NAA considers the cross-correlation matrix R, estimated according to

$$R = \frac{1}{n} Z Z^T. \tag{5}$$

as a low-dimensional estimation of how that class is being modeled, giving one cross-correlation matrix for each class. In order to compare how the different classes are represented, a generalized cosine is taken between the cross-correlation matrices for each class, defining a similarity matrix, S with each row/column representing a class in the network. This similarity matrix is then subtracted from one to generate a distance matrix, D, which can be used to generate a semantic hierarchy via agglomerative clustering. It is shown in Marino et al. (2020) that the generalized cosine from Jaeger (2014) can be equivalently calculated using the formula

$$\cos^2(R_i, R_j) = \frac{\text{Tr}\{R_i R_j\}}{\sqrt{\text{Tr}\{R_i R_i\}\text{Tr}\{R_j R_j\}}} \tag{6}$$

which brings the computational complexity of the matrix cosine from $\mathcal{O}(n^3)$ down to $\mathcal{O}(n^2)$. The proof of this formula is included in Appendix D for reference. A class distance matrix, D, is then defined according to

$$d_{ij} = 1 - \cos^2(R_i, R_j) \tag{7}$$

The agglomerative step of NAA consists of taking the class distance matrix and performing agglomerative clustering on the class distances in order to form an agglomerative tree that can be used to analyze the semantic relationships of the network. However, for this work we are only interested in how spatial relationships in the target space translate into class relationships in the hidden space and so our focus is to use the class distance metric defined in (7) in order to evaluate latent class relationships. An example of the resulting distances for MNIST can be found in Figure 3.

4 Experiments & Discussion

4.1 MNIST

In order to evaluate how the semantic targets defined in (2) would impact class relationships as defined in the NAA model as well as error relationships on a simple dataset, vanilla CNNs were trained on the MNIST task with varying levels of parameterization as shown in Table 4. The MNIST

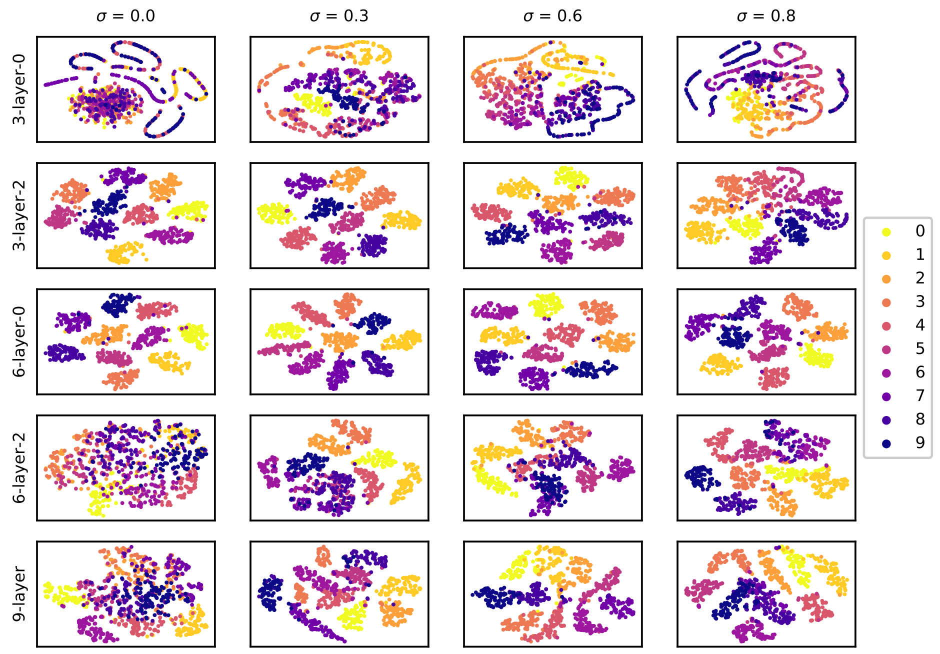


Figure 1: t-SNE visualizations (Maaten & Hinton (2008)) for the penultimate layer of 100 validation images for each class in five models from Table 4 - Class data is plotted with a heat map with yellow representing digit 0 and dark violet representing digit 9. Level of parameterization increases from top to bottom and the smoothing factor (σ) increases from left to right. Similar color tones of the heat map tend to cluster closer together as σ increases. For layers with greater than 50 nodes t-sne was run after PCA was used to lower the dimensionality of the data to 50.

Table 2: Results from models trained on mnist dataset for the model with the best mse as well as its corresponding validation accuracy and the corresponding smoothing factor

Model	**CNN Layers**	**FC Layers**	**accuracy**	**top mse**	σ	**top accuracy**	σ
3-layer-0	2 - 3x3x2	1x2	.465	1.62	0.7	.782	0.1
3-layer-1	2 - 3x3x4	1x4	.848	.734	0.7	.959	0.0
3-layer-2	2 - 3x3x8	1x8	.981	.356	0.1	.981	0.1,0.2
3-layer-3	2 - 3x3x16	1x16	.986	.271	0.3	.986	0.3
6-layer-0	3 - 3x3x16	3x16	.985	.260	0.4	.986	0.2
6-layer-1	3 - 3x3x32	3x32	.986	.232	0.5	.987	0.2,0.3,0.4
6-layer-2	3 - 3x3x32	3x64	.988	.197	0.3	.988	0.3
9-layer	3 - 3x3x32	3x64, 3x128	.984	.262	0.3	.986	0.1

dataset was also chosen because there is a straightforward way to consider error cost - i.e. the mean squared error between the true value and the predicted class. It is typically the case that only raw class accuracy is considered in a model's performance. However, in most real-world applications involving number classification errors closer to the true value would likely be preferable. For each network architecture, the smoothing parameter was varied from 0.0, the one-hot case, to 0.8, which is the point at which the neighboring target classes have values near the true class and accuracy begins to drop off significantly. All networks were trained for 150 epochs with standard SGD with a constant learning rate of 10^{-2}. The input layer was normalized but otherwise no regularization was used. All layers prior to the classification layer used ReLu activation functions, with a standard softmax layer at the output. All analysis was done using the validation dataset.

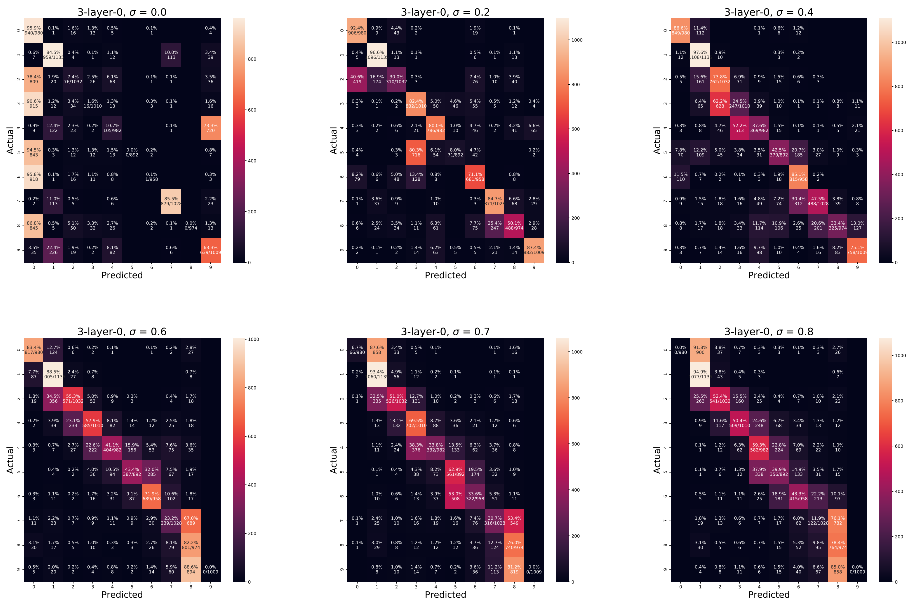


Figure 2: Confusion matrices for varied σ for model 3-layer-0 from Table 4, σ increasing from left to right and top to bottom. Top-left corresponds to $\sigma = 0$, the one-hot case, and the bottom right corresponds to $\sigma = 0.8$, the point at which raw accuracy drops off significantly. Note the model does not have the capacity to learn a reasonable raw accuracy, but it does have the capacity to minimize the mse once the targets are smoothed.

Table 3: Architectures used for CIFAR-10 experiments - each consists of standard Residual Modules (insert citation) with batch normalization applied before each activation function. Detailed descriptions of architectures are included in Appendix A.

Model	**# Modules**	**# Parameters**
ResNet-3-0	3	9,718
ResNet-3-1	3	133,462
ResNet-9	9	321,622
ResNet-18	18	603,862

Table 4 shows the parameterization of each architecture with the top mean-squared error achieved for each architecture, corresponding accuracy achieved, and the corresponding smoothing factor. In every model the best mean-squared error was achieved with smoothing factors greater than zero, the standard one-hot case. Additionally, top accuracy was achieved by models with σ greater than zero in all but one case. Particularly interesting is the case of the smallest network in which the accuracy in the one-hot case was lowest and the network appeared to only be capable of training with $\sigma > 0$.

4.2 CIFAR10

In order to explore an alternative method of smoothing the targets, experiments were done training residual networks (He et al. (2016)) on the relational targets defined in (4). With a set of more general semantic classes, it is not quite as simple to evaluate error cost as was the case using the mse in the mnist case. However, the case of more general semantic classes allows us to more easily define a semantic hierarchy as in Table 1. In order to demonstrate another method of embedding higher order relationships into the target space with CIAR-10, we trained networks that can be used both to classify images at the base-class level while also being able to effectively delineate the superclusters

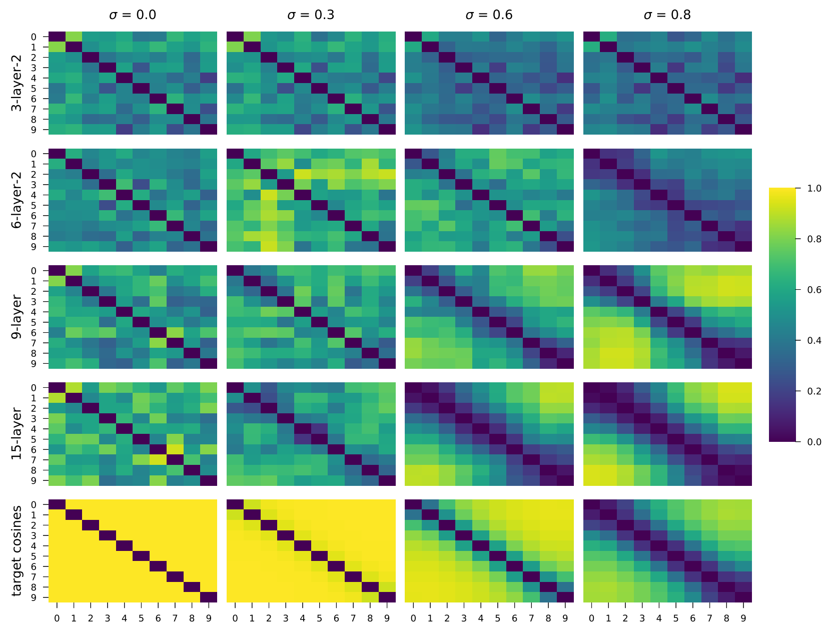
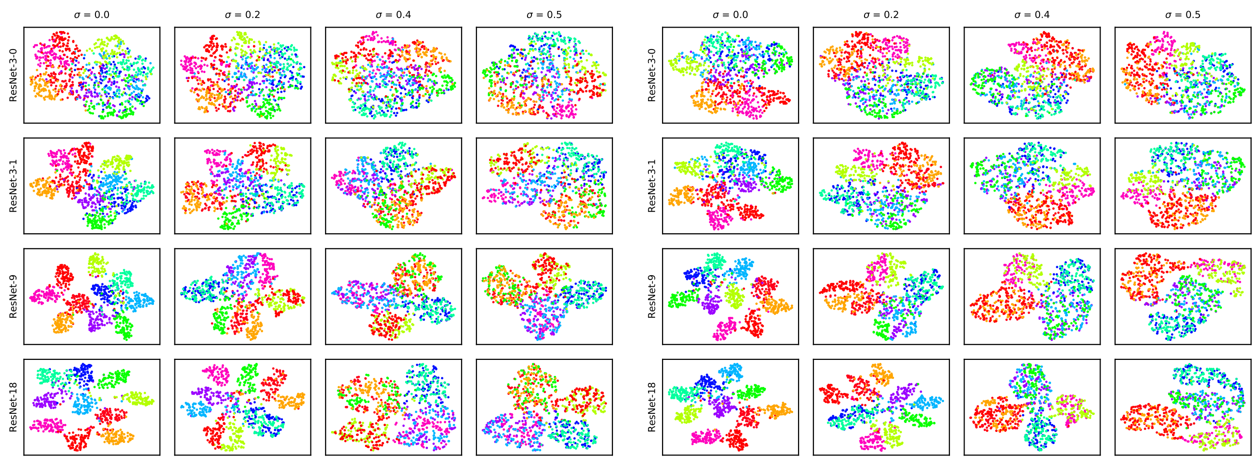


Figure 3: Class distances in the penultimate layer according to the the NAA model - class distances according to the cosine similarity metric defined in (6) were calculated for the final hidden layer in models trained on the MNIST dataset. The bottom row shows $\cos^2(y_i, y_j)$ subtracted from one for each of the target vectors, which could be considered a sort of ground truth orientations. Note the similarity between the cosine relationship in the target space and the class distance relationships in the NAA model, particularly as both parameterization and σ increases. This in addition to findings in Figure 2 give evidence that the smoothed relationships in the target space are easier to learn than the more orthogonal relationships such as in the one-hot case.

Table 4: Results showing best argmax and k-means accuracy for models trained on CIFAR-10 dataset - Extended tabulated results for CIFAR-10 experiments can be found in appendix

Model	**grouping**	**top argmax**	σ	**top k-means**	σ
ResNet-3-0	$grouping_0$	.742	0.0	.636	0.3
	$grouping_1$	.732	0.1	.681	0.5
	$grouping_2$	.738	0.0	.663	0.3
	$grouping_3$	.746	0.0	.719	0.4
ResNet-3-1	$grouping_0$	.873	0.0	.746	0.2
	$grouping_1$	.870	0.0	.752	0.2
	$grouping_2$	.864	0.0	.722	0.4
	$grouping_3$	.869	0.0	.797	0.4
ResNet-9	$grouping_0$	.924	0.1	.881	0.4
	$grouping_1$	.926	0.1	.854	0.5
	$grouping_2$	.918	0.0,0.1	.838	0.2
	$grouping_3$	.927	0.1	.870	0.4
ResNet-12	$grouping_0$	.938	0.1	.909	0.1
	$grouping_1$	.935	0.1	.888	0.5
	$grouping_2$	.936	0.0	.879	0.2
	$grouping_3$	.937	0.1	.905	0.3

via k-means clustering. That is, each target can be used to classify on two different levels. When an $argmax$ function is applied to the layer, this will classify the image in one of the ten standard categories. In addition, a k-means classifier can be trained on the output layer in order to determine which cluster in Table 1 the image belongs to. Models consisted of standard residual modules



(a) (b)

Figure 4: t-SNE visualizations of the penultimate layer for two groupings from Table 1 on CIFAR-10 models - (left) t-SNE for $grouping_1$ and (right) $grouping_3$ (semantic grouping) - note as σ increases the points tend to cluster into their superclusters while the differentation between subclasses decreases.

with batch normalization applied before a ReLu activation function after each convolutional layer. Basic information regarding the models can be found in Table 3 with more detailed descriptions in Appendix A. All models were trained for 200 epochs with a batch size of 128 and an initial learning rate of 10^-1, applying linear learning rate decay at each epoch. Training parameters and model architectures were largely adapted from Rosebrock (2017).

Results for nest class and categorical accuracy of each model can be found in Table 4 with extended results in Appendix C. In the most shallow networks, the one-hot case gives the highest class accuracy in all but one case, while architectures trained with higher σ tended to perform better on the k-means classification task. As the parameterization increases in the deeper networks, this ceases to be the case. In the two deepest networks, $\sigma = 0.1$ appears to often perform best on both tasks, only being outperformed by one-hot label training in one out of eight trials, and was only outperformed by 0.1% in that case. Visualizations using t-SNE on the penultimate layer can be found in Figure 4. Note that as σ increases, the clusters tend to spread out into supercluster formations compared with the separate class groupings in the one-hot case.

5 Summary

This work introduced a simple approach to embed semantic hierarchies into deep models via label smoothing that adds no additional computational cost on standard training approaches and minimal additional effort on part of the model designer. We showed using MNIST training experiments and two different methods of analysis - t-SNE and NAA - that class relationships in models trained on such targets tend toward the spatial relationships of the targets themselves. Through confusion analysis, we showed that even a model without sufficient capacity to learn when training on one-hot vectors can be capable of learning effective representations using our categorically-aware target vectors. Using CIFAR-10 experiments, we also showed the network's ability to learn arbitrary class hierarchies that can be used explicitly to cluster points on multiple hierarchical levels. The main result we show is that, in addition to label smoothing having attractive properties in terms of training and generalization as has been shown by other label smoothing approaches, it is also possible to embed multiple layers of information into class labels without loss of accuracy.

Acknowledgments

References

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

Herbert Jaeger. Controlling recurrent neural networks by conceptors. *arXiv preprint arXiv:1403.3369*, 2014.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Y Lecun, C Cortes, and CJC Burges. The mnist dataset of handwritten digits(images), 1999.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

Michael Marino, Georg Schröter, Gunther Heidemann, and Joachim Hertzberg. Hierarchical modeling with neurodynamical agglomerative analysis. In *ICANN 2020*, volume 12396 of *Lecture Notes in Computer Science*, 2020.

George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11): 39–41, 1995.

Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? In *Advances in Neural Information Processing Systems*, pp. 4694–4703, 2019.

Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017.

Joshua C Peterson, Paul Soulos, Aida Nematzadeh, and Thomas L Griffiths. Learning hierarchical visual representations in deep neural networks using hierarchical linguistic labels. *CoRR, abs/1805.07647*, 2018.

Joshua C Peterson, Ruairidh M Battleday, Thomas L Griffiths, and Olga Russakovsky. Human uncertainty makes classification more robust. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9617–9626, 2019.

Zhaochun Ren, Maria-Hendrike Peetz, Shangsong Liang, Willemijn Van Dolen, and Maarten De Rijke. Hierarchical multi-label classification of social text streams. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pp. 213–222, 2014.

Adrian Rosebrock. *Deep Learning for Computer Vision with Python: Practitioner Bundle*. PyImageSearch, 2017.

Celine Vens, Jan Struyf, Leander Schietgat, Sašo Džeroski, and Hendrik Blockeel. Decision trees for hierarchical multi-label classification. *Machine learning*, 73(2):185, 2008.

Jonatas Wehrmann, Ricardo Cerri, and Rodrigo Barros. Hierarchical multi-label classification networks. In *International Conference on Machine Learning*, pp. 5075–5084, 2018.

Lingxi Xie, Jingdong Wang, Zhen Wei, Meng Wang, and Qi Tian. Disturblabel: Regularizing cnn on the loss layer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4753–4762, 2016.

Lingfeng Zhang, Shishir K Shah, and Ioannis A Kakadiaris. Hierarchical multi-label classification using fully associative ensemble learning. *Pattern Recognition*, 70:89–103, 2017.

A ResNet architecture details from Section 4.2

Table 5: ResNet architecture details - All architectures consist of residual modules with batch normalization applied before each ReLu activation function, followed by a global average pooling layer before the final classification layer.

	CIFAR-10 Model Architectures			
	ResNet-3-0	ResNet-3-1	ResNet-9	ResNet-18
	3x3, 16	3x3, 64		
Module 1	1x1, 4 3x3, 4 1x1, 16	1x1, 16 3x3, 16 1x1, 64		
Module 2	1x1, 8 3x3, 8 1x1, 32	1x1, 32 3x3, 32 1x1, 128	1x1, 16 3x3, 16 1x1, 64	
Module 3	1x1, 16 3x3, 16 1x1, 64	1x1, 64 3x3, 64 1x1, 256	1x1, 16 3x3, 16 1x1, 64	
Modules 4-6	-	-	1x1, 32 3x3, 32 1x1, 128	1x1, 16 3x3, 16 1x1, 64
Modules 7-9	-	-	1x1, 64 3x3, 64 1x1, 256	1x1, 32 3x3, 32 1x1, 128
Modules 10-12	-	-	-	1x1, 32 3x3, 32 1x1, 128
Modules 13-18	-	-	-	1x1, 64 3x3, 64 1x1, 256
	global average pooling			
	softmax classifier			

B Extended MNIST Results

Table 6: Extended results from Table 4 - Below are the accuracy and mean-squared error results for each architecture at for each value of σ applied, with best results emphasized for each model.

Architecture	σ	accuracy	mse	Architecture	σ	accuracy	mse
3-layer-0	0.0	.362	17.8	3-layer-1	0.0	**.959**	.752
	0.1	**.782**	2.98		0.1	.956	**.606**
	0.2	.697	2.77		0.2	.952	.755
	0.3	.699	1.89		0.3	.941	.758
	0.4	.622	2.67		0.4	.941	.795
	0.5	.613	2.04		0.5	.822	1.28
	0.6	.559	1.88		0.6	.921	.811
	0.7	.465	**1.62**		0.7	.848	.734
	0.8	.437	1.72		0.8	.601	.995
3-layer-2	0.0	.976	.419	6-layer-0	0.0	.980	.349
	0.1	**.981**	**.356**		0.1	.984	.300
	0.2	**.981**	.373		0.2	**.986**	.281
	0.3	.977	.425		0.3	.984	.277
	0.4	.976	.429		0.4	.985	**.260**
	0.5	.974	.489		0.5	.983	.300
	0.6	.975	.393		0.6	.984	.274
	0.7	.942	.494		0.7	.967	.306
	0.8	.398	1.02		0.8	.432	.835
6-layer-1	0.0	.982	.375	6-layer-2	0.0	.979	.401
	0.1	**.987**	.253		0.1	.987	.265
	0.2	**.987**	.253		0.2	.988	.221
	0.3	**.987**	.278		0.3	**.989**	**.197**
	0.4	**.987**	.246		0.4	.988	.229
	0.5	.986	**.232**		0.5	.988	.204
	0.6	.984	.280		0.6	.985	.259
	0.7	.973	.272		0.7	.978	.248
	0.8	.422	.794		0.8	.427	.816
9-layer	0.0	.981	.358	12-layer	0.0	.982	.362
	0.1	.984	.300		0.1	**.986**	.267
	0.2	.985	.267		0.2	.982	.326
	0.3	.986	.269		0.3	.984	**.262**
	0.4	**.988**	**.228**		0.4	.983	.304
	0.5	.985	.248		0.5	.983	.291
	0.6	.985	.240		0.6	.982	.305
	0.7	.978	.246		0.7	.973	.335
	0.8	.427	.752		0.8	.493	.727

C Extended CIFAR-10 Results

Table 7: Results from models trained on CIAFR-10 applying smoothed targets according to the clusters defined in Table 1 and clustering on two different levels. Note that even at $\sigma = 0.1$, with a trivial loss in accuracy, the model has effectively embedded two different levels of labels that can be used to cluster data on 2 different levels of granularity relative to the semantic hierarchy.

		$grouping_0$		$grouping_1$		$grouping_2$		$grouping_3$	
		Accuracy Results							
Model	σ	argmax	k-means	argmax	k-means	argmax	k-means	argmax	k-means
ResNet-3-0	0.0	**.742**	.483	.729	.354	**.738**	.443	**.746**	.434
	0.1	.712	.359	**.732**	.570	.736	.489	.714	.632
	0.2	.691	.584	.707	.674	.714	.662	.697	.650
	0.3	.675	**.636**	.696	.678	.695	**.663**	.678	.707
	0.4	.669	.623	.681	.666	.682	.650	.666	**.719**
	0.5	.514	.588	.568	**.681**	.548	.614	.548	.690
ResNet-3-1	0.0	**.873**	.383	**.870**	.521	**.864**	.477	**.869**	.437
	0.1	.858	.700	.856	.714	.855	.620	.859	.769
	0.2	.849	**.746**	.855	**.752**	.850	.714	.842	.751
	0.3	.838	.731	.836	.679	.833	.645	.830	.711
	0.4	.838	.731	.825	.707	.826	**.722**	.805	**.797**
	0.5	.647	.690	.659	.731	.658	.664	.661	.724
ResNet-9	0.0	.920	.471	.923	.447	**.918**	.379	.923	.460
	0.1	**.924**	.846	**.926**	.717	**.918**	.637	**.927**	.731
	0.2	.920	.879	.922	.849	.917	**.838**	.921	.846
	0.3	.913	.855	.908	.846	.909	.832	.911	.858
	0.4	.913	**.881**	.902	.835	.909	.834	.899	**.870**
	0.5	.710	.874	.738	**.854**	.736	.808	.734	.868
ResNet-18	0.0	.933	.489	.933	.485	**.936**	.377	.933	.472
	0.1	**.938**	**.909**	**.935**	.875	.935	.646	**.937**	.897
	0.2	.931	.885	.930	.881	.931	**.879**	.935	.888
	0.3	.932	.887	.928	.878	.930	.862	.928	**.905**
	0.4	.927	.890	.918	.885	.922	.848	.918	.869
	0.5	.728	.891	.746	**.888**	.736	.854	.734	.892

D Proof of generalized cosine formula from equation (6)

As noted in Section 3.2, a generalized cosine can be defined according to (Jaeger (2014))

$$cos^2(R_i, R_j) = \frac{\|\Sigma_i^{1/2} U_i' U_j \Sigma_j^{1/2}\|^2}{\|diag\{\Sigma_i\}\| \|diag\{\Sigma_j\}\|} \tag{8}$$

which is equivalent to the formula

$$cos^2(R_i, R_j) = \frac{Tr\{R_i R_j\}}{\sqrt{Tr\{R_i R_i\} Tr\{R_j R_j\}}}. \tag{9}$$

Below is the proof from Marino et al. (2020), with some intermediate steps included for clarity.

Proof. Considering the following properties of the trace of matrix products, Frobenius norm, and of any unitary matrix U

$$\|A\|^2 = \text{Tr}\{A^T A\} \tag{10}$$

$$\text{Tr}\big\{AB\big\} = \text{Tr}\big\{BA\big\} \tag{11}$$

$$U^T = U^{-1}, \tag{12}$$

the numerator in equation (8) can be simplified by exploiting the fact that the trace of matrix products is invariant with respect to cycling the matrices being multipled as well as the fact that $UU^T = I$, the identity matrix. Cycling the matrices and simplifying gives

$$\begin{aligned}
\big\|\Sigma_i^{1/2}U_i^TU_j\Sigma_j^{1/2}\big\|^2 &= \text{Tr}\big\{\big(\Sigma_i^{1/2}U_i^TU_j\Sigma_j^{1/2}\big)^T\Sigma_i^{1/2}U_i^TU_j\Sigma_j^{1/2}\big\} \\
&= \text{Tr}\big\{\Sigma_j^{1/2}U_j^TU_i\Sigma_i^{1/2}\Sigma_i^{1/2}U_i^TU_j\Sigma_j^{1/2}\big\} \\
&= \text{Tr}\big\{\Sigma_jU_j^TU_i\Sigma_iU_i^TU_j\big\} \\
&= \text{Tr}\big\{(U_j\Sigma_jU_j^T)(U_i\Sigma_iU_i^T)\big\} \\
&= \text{Tr}\big\{R_jR_i\big\} \\
&= \text{Tr}\big\{R_iR_j\big\}
\end{aligned}$$

Since Σ is a diagonal matrix, the denominator can be expressed as

$$\begin{aligned}
\big\|diag\{\Sigma_i\}\big\|^2 &= \big\|\Sigma_i\big\|^2 \\
&= \text{Tr}\big\{\Sigma_i^T\Sigma_i\big\} \\
&= \text{Tr}\big\{U_i\Sigma_iU_i^TU_i\Sigma_iU_i^T\big\} \\
&= \text{Tr}\big\{R_iR_i\big\}
\end{aligned}$$

$\square$

Combining these gives the desired result in (6).