# A Walkthrough for the Principle of Logit Separation[*]

**Gil Keren**[1] , **Sivan Sabato**[2] and **Björn Schuller**[1,3]
[1]ZD.B Chair of Embedded Intelligence for Health Care and Wellbeing, University of Augsburg, Germany
[2]Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel
[3]GLAM – Group on Language, Audio & Music, Imperial College London, UK
cruvadom@gmail.com

## Abstract

We consider neural network training, in applications in which there are many possible classes, but at test-time, the task is a binary classification task of determining whether the given example belongs to a specific class. We define the *Single Logit Classification* (SLC) task: training the network so that at test-time, it would be possible to accurately identify whether the example belongs to a given class in a computationally efficient manner, based only on the output logit for this class. We propose a natural principle, the *Principle of Logit Separation*, as a guideline for choosing and designing loss functions that are suitable for SLC. We show that the Principle of Logit Separation is a crucial ingredient for success in the SLC task, and that SLC results in considerable speedups when the number of classes is large.

## 1 Introduction

When using neural network classifiers over a very large number of classes, a high computational burden at test-time occurs. Indeed, in standard neural networks, using a softmax layer and the cross-entropy loss, the computation needed for finding the logits of the classes (the pre-normalized outputs of the top network layer) is linear in the number of classes [Grave *et al.*, 2017], and can be prohibitively slow for high-load systems, such as search engines and real-time machine-translation systems.

In some applications, the task at test-time is not full classification of each example into one of the many possible classes. Instead, the task, each time the trained classifier is used, is to identify whether the current example should be classified into one of a small subset of the possible classes, or even a single class. This class can be different every time the classifier is used. Consider for example the case of real-time image search [Maturana and Scherer, 2015; Redmon *et al.*, 2016] from a live feed from multiple cameras. When the user queries for images of object A, the classifier has to process a large number of images, and decide whether each image contains an instance of object A or not. The classifier is then activated for the second time, this time with a query to find images of object B. The classifier now processes new images, to determine which ones contain an instance of object B. This setting has various applications, such as identifying a person of interest in a live security feed and finding a specific road sign from a camera of an autonomous car.

In the setting that we consider, while every use of the classifier at test-time tests for a single class (or a small number of classes), the classifier itself must support queries on *any* of the classes, since it will be used again and again, each time with a different class as a query. As the number of classes may be large, it is not reasonable to train a separate model for every possible class that might be queried at test time. Instead, our goal is to have a single model which supports *all* possible class-queries.

For this type of applications, one would ideally like to have a test-time computation that does not depend on the total number of possible classes. A natural approach is to calculate only the logit of the class of interest, and use this value alone to infer whether this is the true class of the example. However, the logit of a single class might only be meaningful

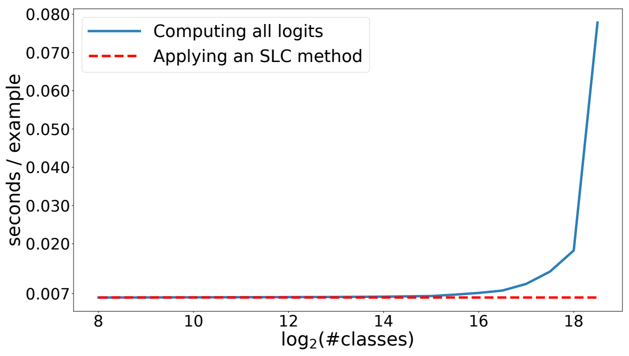

Figure 1: Computation time of the logits from network inputs, using an inception-V3 image classification architecture where the topmost layer is replaced according to the appropriate number of classes. When applying SLC, computation cost is fixed regardless of the number of classes, which can lead to considerable speedups when the number of classes is large.

[*]This work is an extended abstract based on the publication "Fast Single-Class Classification and the Principle of Logit Separation", winner of the Best Student Paper Award in the IEEE International Conference on Data Mining 2019 [Keren *et al.*, 2018c].

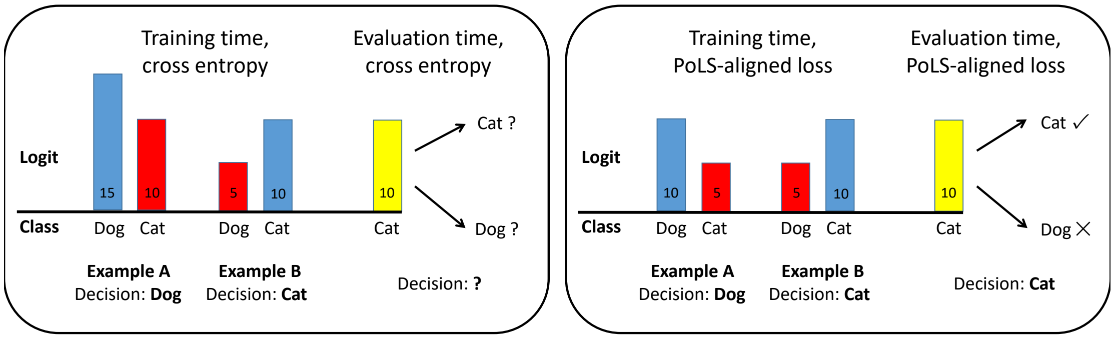

Figure 2: The Principle of Logit Separation. Left: when training with the cross-entropy loss, the logit values for the class ‘Cat’ can be the same for two examples, one where it is the true class (blue) and one where it is not (red). Therefore, at test-time, a logit with the same value for the class ‘Cat’ does not indicate whether the example belongs to this class. Right: With a loss function that is aligned with the Principle of Logit Separation, all true logits are greater than all false logits at training time. Hence, at test time, a single logit can indicate the correctness of its respective class.

in comparison to logits of other classes, in which case, unless the other logits are also computed, it cannot be used to successfully determine whether the test example belongs to the class of interest. We name the goal of inferring class correctness from the logit of that class alone *Single Logit Classification* (SLC). Note that SLC is a binary classification task, stressing the fact that only one logit is computed. In Figure 1 we demonstrate the speedup yielded by SLC, compared to binary classification in the method which computes all logits and uses them for normalization, as the number of classes increases. For instance, computing only a single logit yields a 10x speedup in evaluation time when there are 400,000 possible classes. The speedup increases with the number of possible classes. See Section 6 for more details on the experimental setting and the resulting speedup.

In the next sections, we introduce the Principle of Logit Separation as a guideline for choosing and designing loss functions that are appropriate for the SLC task. The extended version of the work [Keren *et al.*, 2018c] contains full proofs for alignment of each of the seven considered loss functions with the Principle of Logit Separation as well as full experiment details and extended results.

## 2 The Principle of Logit Separation

In the SLC task, the only information about an example is the output logit of the model for the single class of interest. Therefore, a natural approach to classifying whether the class matches the example is to set a threshold: if the logit is above the threshold, classify the example as belonging to this class, otherwise, classify it as not belonging to the class. We refer to logits that belong to the true classes of their respective training examples as *true logits* and to other logits as *false logits*. For the threshold approach to work well, the values of all true logits should be larger than the value of all false logits across the training sample (in fact, it is enough to separate true and false logits on a class level, but we stick to the stronger assumption in this work). This is illustrated in Figure 2. The Principle of Logit Separation (PoLS) captures this requirement. We formalize this principle below.

Let $[k] := \{1, \dots, k\}$ be the possible class labels. Assume that the training sample is $S = ((x_1, y_1), \dots, (x_n, y_n))$, where $x_i \in \mathbb{R}^d$ are the training examples, and $y_i \in [k]$ are the labels of these examples. For a neural network model parametrized by $\theta$, we denote by $z_y^\theta(x)$ the value of the logit assigned by the model to example $x$ for class $y$. The Principle of Logit Separation (PoLS) can be formally stated as follows:

**Definition 1** (The Principle of Logit Separation)**.** *The* Principle of Logit Separation *holds for a labeled set $S$ and a model $\theta$, if for any $(x, y), (x', y') \in S$ (including the case $x = x', y = y'$) and any $y'' \neq y'$, we have $z_y^\theta(x) > z_{y''}^\theta(x')$.*

The definition assures that every true logit $z_y^\theta(x)$ is larger than every false logit $z_{y''}^\theta(x')$. If this simple principle holds for all train and test examples, it guarantees perfect accuracy in the SLC task, since all true logits are larger than all false logits. Thus, a good approach for a training objective for SLC is to attempt to optimize for this principle on the training set. A loss $\ell$ is aligned with the Principle of Logit Separation if for any training sample $S$, minimizing $\ell$ on $S$ ensures that the requirement in Definition 1 holds for the resulting model $\theta$.

## 3 Standard Objectives in View of the PoLS

**The Cross-Entropy Loss** The cross-entropy loss, which is the standard loss function for neural network classifiers (e.g., [Krizhevsky *et al.*, 2012]), is defined on a single example as

$$\ell(z, y) = -\log(p_y), \tag{1}$$

where

$$p_y := \frac{e^{z_y}}{\sum_{j=1}^{k} e^{z_j}} = \Big(\sum_{j=1}^{k} e^{z_j - z_y}\Big)^{-1}.$$

It can be shown that the cross-entropy loss does not satisfy the PoLS. Indeed, as the loss depends only on the difference between logits for every example separately, minimizing it guarantees a certain difference between the true and false logits for every example separately, but does not guarantee that all true logits are larger than all false logits in the training set.

| Dataset | Method | 1-AUPRC | 1-Precision@0.9 | 1-Precision@0.99 |
|---|---|---|---|---|
| MNIST | Mean non-PoLS | 0.010 | 0.011 | 0.232 |
| | Mean PoLS | 0.002 | 0.001 | 0.027 |
| | Relative improvement | 80.0% | 90.9% | 88.4% |
| SVHN | Mean non-PoLS | 0.022 | 0.026 | 0.538 |
| | Mean PoLS | 0.017 | 0.016 | 0.319 |
| | Relative improvement | 23.6% | 39.6% | 40.8% |
| CIFAR-10 | Mean non-PoLS | 0.101 | 0.305 | 0.704 |
| | Mean PoLS | 0.074 | 0.211 | 0.608 |
| | Relative improvement | 26.9% | 30.9% | 13.7% |
| CIFAR-100 | Mean non-PoLS | 0.487 | 0.879 | 0.975 |
| | Mean PoLS | 0.405 | 0.834 | 0.971 |
| | Relative improvement | 16.8% | 5.2% | 0.4% |
| Imagenet (1000 classes) ($6 \cdot 10^6$ iterations) | CE | 0.366 | 0.739 | 0.932 |
| | batch CE | 0.245 | 0.563 | 0.865 |
| | Relative improvement | 33.1% | 23.8% | 7.2% |

Table 1: Results on Single Logit classification (SLC), averaged according to alignment with the PoLS.

**The Max-Margin Loss** Max-margin training objectives, most widely known for their role in training Support Vector Machines, are used in some cases for training neural networks [Socher *et al.*, 2011; Janocha and Czarnecki, 2017]. Here we consider the multiclass max-margin loss suggested by [Crammer and Singer, 2001], defined as

$$\ell(z, y) = \max(0, \gamma - z_y + \max_{j \neq y} z_j), \tag{2}$$

where $\gamma > 0$ is a hyperparameter that controls the separation margin between the true logit and the false logits of the example. It can be shown that this loss too does not satisfy the PoLS, since minimizing it again guarantees only a certain difference between the true and false logits for every example separately, and not across the entire training sample.

## 4 Objectives that Satisfy the PoLS

In this section we discuss training objectives that have been previously proposed in the literature, and we show that these objectives indeed satisfy the PoLS.

We consider the binary cross-entropy loss, that is often used in multilabel classification settings. In multilabel settings, each example can belong to several classes, and the goal is to identify the set of classes an example belongs to. A common approach [Wang *et al.*, 2016; Huang *et al.*, 2013] is to try to solve $k$ binary classification problems of the form "Does $x$ belong to class $j$?" using a single neural network model, by minimizing the sum of the cross-entropy losses that correspond to these binary problems. In this setting, the label of each example is a binary vector $(r_1, \ldots, r_k)$, where $r_j = 1$ if $x$ belongs to class $j$ and 0 otherwise. The loss for a single training example with logits $z$ and label-vector $r$ is

$$\ell(z, (r_1, \ldots, r_k)) = \\ - \sum_{j=1}^{n} r_j \log(\sigma(z_j)) + (1 - r_j) \log(1 - \sigma(z_j)),$$

where $\sigma(z) = (1 + e^{-z})^{-1}$ is the sigmoid function. This loss can also be used for our setting of multiclass problems, by defining $r_j := \mathbf{1}_{j=y}$ for an example $(x, y)$. This gives the multiclass loss

$$\ell(z, y) = -\log(\sigma(z_y)) + \sum_{j \neq y} \log(1 - \sigma(z_j)).$$

The binary cross-entropy is also aligned with the PoLS. Indeed, similarly to case of the NCE loss, it is easy to see that when the term above is minimized for one example, the value of true logit $z_y$ converges to infinity, and the values of all false logits converge to negative infinity. When the above term is minimized for the entire training set, all true logits are larger than all false logits across the training set.

In the extended version of this work [Keren *et al.*, 2018c], we consider two additional existing loss function, namely the Self-normalization loss [Devlin *et al.*, 2014] and the Noise Contrastive Estimation (NCE) loss [Gutmann and Hyvärinen, 2010; Mnih and Teh, 2012]. Both loss function were previously considered in the context of natural language learning, and we show that they are aligned with the PoLS.

## 5 New Training Objectives for the SLC Task

We propose new training objectives for the SLC task, designed to satisfy the PoLS[1]. We define a batch version of the

[1] `Tensorflow` code for optimizing the new batch losses is publicly available at https://github.com/cruvadom/Logit_Separation.

cross-entropy loss, using the KL-divergence between distributions over batches. Note that the standard cross-entropy loss can be defined using the KL-divergence between the model posterior probability distribution and the one-hot target distribution. Recall that the $i$'th example in a batch $B$ is denoted $(x_i, y_i)$. Let $P_B$ be the distribution over $[m] \times [k]$ defined by

$$P_B(i,j) := \begin{cases} \frac{1}{m} & j = y_i, \\ 0 & \text{otherwise.} \end{cases}$$

Let $Q_B$ be the distribution defined by the softmax normalized logits over the entire batch $B$. Formally, denote $Z(B) := \sum_{i=1}^{m} \sum_{j=1}^{k} e^{z_j(x_i)}$. Then $Q_B(i,j) := e^{z_j(x_i)}/Z(B)$. We then define the batch cross-entropy loss as follows.

**Definition 2** (The batch cross-entropy loss)**.** *Let* $m > 1$ *be an integer, and let* $B$ *be a uniformly random batch of size* $m$ *from* $S$. *The* batch cross-entropy loss *of a training sample* $S$ *is*

$$\ell(S) := \mathbb{E}_B[L_c(B)], \quad \textit{where} \quad L_c(B) := \mathrm{KL}(P_B || Q_B).$$

This batch version of the cross-entropy loss is aligned with the PoLS. Indeed, when this loss is minimized for one training batch, all true logits converge to some positive value (as a normalized exponentiated true logit converges to $1/m$), while all false logits converge to negative infinity (as a normalized exponentiated false logit converges to zero). Therefore, when minimizing this loss across the whole training set, all true logits are larger than all false logits in the training set. Similarly, we introduce the batch version of the max-margin loss, which is described in the extended version of this work [Keren *et al.*, 2018c].

## 6 Experiments

We tested the SLC tasks on neural networks trained with each of the seven loss functions we considered above, five of which are aligned with the PoLS. To evaluate the success of a learned model in the SLC task, we measured, for each class $j$ and each threshold $T$, the precision and recall in identifying examples from class $j$ using the test $z_j > T$, and calculated the Area Under the Precision-Recall curve (AUPRC) defined by the entire range of possible thresholds. We also measured the precision at fixed recall values (with dictate the threshold $T$ to use) 0.9 (Precision@0.9) and 0.99 (Precision@0.99). We report the averages of these values over all the classes in the dataset.

We used five benchmark datasets for our evaluations. For the MNIST dataset [LeCun *et al.*, 1998], we used a neural network with two fully-connected layers. For the SVHN [Netzer *et al.*, 2011], CIFAR-10 and CIFAR-100 [Krizhevsky and Hinton, 2009] datasets, a neural network with six convolutional layers and one fully-connected layer was used. For the Imagenet dataset, we used the Inception-V3 architecture [Szegedy *et al.*, 2016]. Experiment results are reported in Table 1, and contain averaged results according to PoLS alignment. It can be seen that the mean relative improvement of training objectives that are aligned with the PoLS compared to non-aligned objectives is usually at least 20%, and in many cases considerably more.

| Architecture | Classes | Time [s] | SLC Speedup |
|---|---|---|---|
| Alexnet | 1 (SLC) | $3.6 \cdot 10^{-3}$ | — |
| | $2^{14}$ | $4.0 \cdot 10^{-3}$ | x1.14 |
| | $2^{18}$ | $20.2 \cdot 10^{-3}$ | x5.68 |
| | $2^{18.5}$ | $76.0 \cdot 10^{-3}$ | x21.38 |
| VGG-16 | 1 (SLC) | $9.4 \cdot 10^{-3}$ | — |
| | $2^{14}$ | $9.9 \cdot 10^{-3}$ | x1.05 |
| | $2^{18}$ | $26.4 \cdot 10^{-3}$ | x2.79 |
| | $2^{18.5}$ | $80.5 \cdot 10^{-3}$ | x8.52 |
| Inception-v3 | 1 (SLC) | $6.0 \cdot 10^{-3}$ | — |
| | $2^{14}$ | $6.5 \cdot 10^{-3}$ | x1.09 |
| | $2^{18}$ | $18.7 \cdot 10^{-3}$ | x3.11 |
| | $2^{18.5}$ | $76.6 \cdot 10^{-3}$ | x12.75 |
| Resnet-50 | 1 (SLC) | $6.1 \cdot 10^{-3}$ | — |
| | $2^{14}$ | $6.6 \cdot 10^{-3}$ | x1.08 |
| | $2^{18}$ | $19.1 \cdot 10^{-3}$ | x3.11 |
| | $2^{18.5}$ | $78.0 \cdot 10^{-3}$ | x12.69 |
| Resnet-101 | 1 (SLC) | $8.0 \cdot 10^{-3}$ | — |
| | $2^{14}$ | $8.3 \cdot 10^{-3}$ | x1.04 |
| | $2^{18}$ | $23.5 \cdot 10^{-3}$ | x2.95 |
| | $2^{18.5}$ | $80.4 \cdot 10^{-3}$ | x10.10 |

Table 2: Speedup experiment results. When the number of examples is large, SLC results in a considerable speedup.

In addition, we estimated the speedups gained by performing SLC, compared to methods in which all logits are computed, using five prominent image classification architectures. Since we are interested in test-time performance, we report the time required for computing the forward-pass of a given network. To measure SLC computation time, we replace the top layer by a layer with a single unit and measure the time to compute the single logit given an input to the network. To measure the computation time when computing the logits of $k$ classes, we replace the top layer with a layer containing $k$ units, and again measure the time it takes to compute all logits, given an input to the network.

The timing results are given in Table 2. The results in the table show that for networks with up to $2^{14} = 16384$ classes, the speedup is relatively small, since computation of the network layers other than the logit layer dominates the forward-pass computation time. In contrast, when there are many classes, the computation of logits dominates the forward-pass computation time. Hence, SLC obtains a speedup as high as $x20$, which grows when the number of classes is larger.

## 7 Conclusion

We consider the Single Logit Classification (SLC) task, which is important in various applications. We formulate the Principle of Logit Separation (PoLS), and study its alignment with seven loss functions, including the standard cross-entropy loss and two novel loss functions. We established, and corroborated in experiments, that PoLS-aligned loss functions yield class logits that are more useful for binary classification. We further demonstrated that training with a PoLS-aligned loss function and applying SLC leads to considerable speedups when there are many classes. Recent years have seen a constant increase in the number of classes in datasets from various domains, thus we expect SLC and the PoLS to play a key role in various applications. Future work plans include extending the scope of the Principle of Logit Separation by applying it to other training mechanisms that do not involve loss functions [Keren *et al.*, 2017] and to neural network regressors designed as classifiers [Keren *et al.*, 2018a; Keren *et al.*, 2018b].

## Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 826506 (sustAGE). Sivan Sabato was supported in part by the Israel Science Foundation (grant No. 555/15).

## References

[Crammer and Singer, 2001] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

[Devlin *et al.*, 2014] Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard M. Schwartz, and John Makhoul. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL*, pages 1370–1380, Baltimore, MD, 2014.

[Grave *et al.*, 2017] Edouard Grave, Armand Joulin, Moustapha Cissé, David Grangier, and Hervé Jégou. Efficient softmax approximation for GPUs. In *Proceedings of ICML*, pages 1302–1310, Sydney, Australia, 2017.

[Gutmann and Hyvärinen, 2010] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of AISTATS*, pages 297–304, Chia Laguna Resort, Sardinia, Italy, 2010.

[Huang *et al.*, 2013] Yan Huang, Wei Wang, Liang Wang, and Tieniu Tan. Multi-task deep neural network for multi-label learning. In *Procedding of IEEE International Conference on Image Processing, ICIP*, pages 2897–2900, Melbourne, Australia, 2013.

[Janocha and Czarnecki, 2017] Katarzyna Janocha and Wojciech Marian Czarnecki. On loss functions for deep neural networks in classification. *Arxiv*, abs/1702.05659, 2017.

[Keren *et al.*, 2017] Gil Keren, Sivan Sabato, and Björn Schuller. Tunable sensitivity to large errors in neural network training. In *Proceedings of AAAI*, pages 2087–2093, San Francisco, CA, 2017.

[Keren *et al.*, 2018a] Gil Keren, Nicholas Cummins, and Björn W. Schuller. Calibrated prediction intervals for neural network regressors. *IEEE Access*, 6:54033–54041, 2018.

[Keren *et al.*, 2018b] Gil Keren, Jing Han, and Björn Schuller. Scaling speech enhancement in unseen environments with noise embeddings. In *In Proceedings of the CHiME Workshop on Speech Processing in Everyday Environments*, pages 25–29, Hyderabad, India, 2018.

[Keren *et al.*, 2018c] Gil Keren, Sivan Sabato, and Björn Schuller. Fast single-class classification and the principle of logit separation. In *Proceedings of the International Conference on Data Mining (ICDM)*, pages 227–236, Singapore, 2018.

[Krizhevsky and Hinton, 2009] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images, 2009.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of NIPS*, pages 1106–1114, Lake Tahoe, NV, 2012.

[LeCun *et al.*, 1998] 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.

[Maturana and Scherer, 2015] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Proceedings of IROS*, pages 922–928, Hamburg, Germany, 2015.

[Mnih and Teh, 2012] Andriy Mnih and Yee Whye Teh. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings ICML*, Edinburgh, Scotland, UK, 2012.

[Netzer *et al.*, 2011] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*. Granada, Spain, 2011.

[Redmon *et al.*, 2016] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of CVPR*, pages 779–788, Las Vegas, NV, 2016.

[Socher *et al.*, 2011] Richard Socher, Cliff Chiung-Yu Lin, Andrew Y. Ng, and Christopher D. Manning. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of ICML*, pages 129–136, Bellevue, WA, 2011.

[Szegedy *et al.*, 2016] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceddings of CVPR*, pages 2818–2826, Las Vegas, NV, 2016.

[Wang *et al.*, 2016] Jiang Wang, Yi Yang, Junhua Mao, Zhiheng Huang, Chang Huang, and Wei Xu. CNN-RNN: A unified framework for multi-label image classification. In *Proceedings of CVPR*, pages 2285–2294, Las Vegas, NV, 2016.