A New Benchmark and Progress Toward Improved Weakly Supervised Learning

Jason Ramapuram
jason@ramapuram.net

University of Geneva & University of Western Sciences
Geneva, Switzerland

Russ Webb
rwebb@apple.com

Apple Inc
Cupertino, California, USA

Abstract

Knowledge Matters: Importance of Prior Information for Optimization [6], by Gülçehre et. al., sought to establish the limits of current black-box, deep learning techniques by posing problems which are difficult to learn without engineering knowledge into the model or training procedure. In our work, we solve the previous *Knowledge Matters* problem with 100% accuracy using a generic model, pose a more difficult and scalable problem, All-Pairs, and advance this new problem by introducing a new learned, spatially-varying histogram model called TypeNet which outperforms conventional models on the problem. We present results on All-Pairs where our model achieves 100% test accuracy while the best ResNet models achieve 79% accuracy. In addition, our model is more than an order of magnitude smaller than Resnet-34. The challenge of solving larger-scale All-Pairs problems with high accuracy is presented to the community for investigation.

1 Introduction

Deep neural networks are powerful functional approximators, allowing for the learning of complex tasks that were not solvable by traditional machine learning methods. Recently, [6] suggested that there exist problems that neural networks would not be able to solve without the guidance of human insight; they define and study the Pentomino problem as an example of this class of problems. For the Pentomino problem, we demonstrate that extra knowledge is not necessary by solving the problem with a small, deep neural network (DNN). Having found a solution to Pentomino, we introduce a new, scalable problem and present progress toward its solution.

To understand the limits of weakly supervised learning applied to generic models, we divide the task of solving a problem into the application of known techniques and the engineering of the system (the model plus training data and procedure). The palette of known techniques is constantly improving and is what enables solving the Pentomino problem with current techniques. The incorporation of explicit knowledge into an engineered solution can be estimated by how many problem specifics can be inferred/discovered by an inspection of the model architecture and the training procedure. Common deep-learning techniques are a codification of knowledge into reusable components which require minimal insight to select. For instance, batch norm [10] speeds convergence and reduces hyper-parameter sensitivity, jump connections [7, 27] enable deeper models, convolutions [13, 14] are useful in spatially-invariant vision problems, and sparse activations [25, 29] reduce overfitting by restricting the flow of information. Even simple observations, like these, allow the practitioner to select components suitable for the problem. These techniques are all excellent examples of knowledge refined into heuristically selectable, generic techniques.

We lack ready-to-apply techniques for some problems and much of the research in the field moves us toward more turn-key application of learning algorithms; for example, the latest AlphaGo [23] is trained without expert human examples. Some examples of the work done to engineer problems with human knowledge are engineering model sub-components to include problem details and adding sub-goal labels or objective functions. Successful engineering of a solution for a particular problem can lead to either a specific solution only applicable to the problem studied or, more usefully, to broadly reusable techniques or insights. The later outcome is our goal in presenting the following contributions:

1. demonstration of solving the Pentomino problem from *Knowledge Matters* [6] with conventional techniques (both model and training)

2. new, scalable challenge problem, All-Pairs, with (effectively) infinite data [9]

3. sampling of existing techniques' performance on All-Pairs as baselines

4. new, generic model, TypeNet [9], which out-performs the baselines on All-Pairs.

The All-Pairs dataset generator and TypeNet reference code are available:
https://github.com/apple/ml-all-pairs.

2 Related Work

Our work spans two distinct areas of machine learning: learning under weak supervision and extracting relational information from high-dimensional data. By *weak supervision*, we mean that our model is required to solve a high-level task such as the binary classification proposed on the left of Figure 1 by observing only raw pixels.

Prior work in weak supervised learning (WSL) in the image domain has focused on image segmentation by classifying them with a standard multi-class loss objective [18] or by utilizing an alternate loss such as a score-based [3] objective. Unsupervised representation learning can also be used to aid the model in learning the end objective. The recent work of *Learning to Count* [17] proposed a method for representation learning in an unsupervised setting by using a pre-trained network to learn counting of visual primitives. This method works well when the features extracted from the pre-trained network are semantically relevant to the current learning objective. Our work differs from this and the WSL objective in the amount of supervision provided to the model. We focus on supervised tasks where a model is not provided with sub-problem class labels (or any other structured, supervised information) and needs to learn a high-level representation of the visual scene using few binary labels for each whole image.

Extracting relational information with neural networks has been studied in many settings from text-based relationships [2, 30] to visual query answer (VQA) models such as the recent work of Relational Networks [21] and Show-Tell-Attend [32]. Relational Networks have been used to learn relationships between objects in a scene given a rich textual query, such as the

CLEVR dataset [11], which provides input in the form of an image coupled with a textual query. Despite having a pair-wise structure that we intuitively think is useful for our All-Pairs problem, a Relational Network [21] does not solve our proposed dataset (Section 4). To solve our problem, we introduce a model called TypeNet which aggregates channel-wise statistics and solves the overall task by combining these statistics.

Our TypeNet model takes inspiration from winner-take-all (WTA) strategies [15, 26] and can build a set of problem-related, local statistics to combine for predicting the end objective. We demonstrate empirically that TypeNet outperforms state-of-the-art models such as ResNet18, Resnet34 [7], VGG19 & VGG16 with batch norm [24], and InceptionV3 [28] on the proposed All-Pairs problem.

3 Solving the Pentomino Problem

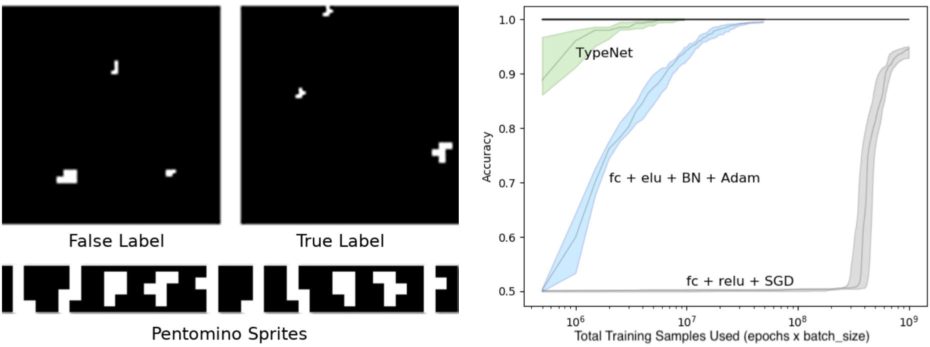

Figure 1: *Left*: The Pentomino sprites and two examples illustrating the $true$ and $false$ classes. *Right*: Test accuracy (median and inner quartiles, 10 trials) on the Pentomino problem with and without modern training advances. Note, log-scale of x-axis.

Knowledge Matters [6] explores the extent to which neural networks are able to learn problems given minimal supervised information. Their formulation has a fully defined loss function; however, the gradient of the loss with respect to the parameters provides no direct information about potentially useful subtasks such as segmentation, object classification, or counting. They concluded that the networks and training methods they tested converged to a local minima.

The *Knowledge Matters* demonstration utilized the Pentomino dataset, which is formed from a set of sprites [5] shown in Figure 1. The dataset is generated by placing three sprites onto a canvas $C \in \mathbb{R}^{64\times64}$. Each sprite undergoes a random rotation (0°, 90°, 180°, or 270°) and integer scaling ($1\times$ or $2\times$). The goal of the neural network is to predict a 1 if the rotated and scaled sprites in an image are the same and 0 otherwise. One possible solution to the Pentomino problem is to learn to segment, classify, and count the number of underlying objects in the image. The challenge (claimed impossible in [5]) is to find a solution using a generic network given only the binary label for each image.

Gülçehre et. al [6] observed that "black-box machine learning algorithms could not perform better than chance on [the Pentomino problem]." Decomposing the problem into two stages however, made the task easily solvable. The first stage in the decomposition was a classification step, where extra label information was provided to the model. Given the predicted classes, the second stage projected this output to the Bernouili log-likelihood objective. Using some of the recent advances in DNN training, we are able to solve the original problem with 100% accuracy as demonstrated in *Knowledge Matters*; we do so without the requirement of an intermediary model or the addition of extra information. We also experimented with a reproduction of the model proposed in the paper and found that given enough time (over 1000 epochs) the model does make progress on the Pentomino problem, as shown in Figure 1 in gray. This observation is in line with recent insights of [8] that discuss the effects of training duration and batch size.

The fully-connected (fc) model presented in [6] was composed of layer sizes [2050, 11, 1024] and trained with ADADelta [33] and weight regularization. The 11-unit layer served as a bottleneck to bring structural information into the network. We leverage four recent advances to solve the Pentomino problem: Batch Normalization (BN) [10], Exponential Linear Units [1], the Adam optimizer [12], and Xavier initializations [4]. In constrast to the large model employed in [6], we use a fully-connected network with layer sizing of $[32,64,12,32,8]$; this translates to a 98.5% reduction of the total number of model parameters. Comparable in size to the largest training sets used in [6], 486k samples were used for training and 54k samples were held out for testing.

Gülçehre et. al [6] were only able to train black-box (generic), fully-connected models to achieve 50% accuracy on the Pentomino dataset. Their best model, after significant hyper-parameter search, resulted in a 5.3% training and 6.7% test error on the 80k Pentomino training dataset. This performance was achieved via a two-stage network that induced structural information into the neural network. On the same training set, we achieved a 1% error using a black-box neural network with the 5-layer network described above. Figure 1 shows the training accuracy for the original *Knowledge Matters* network (gray), our modification (blue), and our TypeNet model (green, see Section 5.1) on the Pentomino problem (note the log scale on the x-axis).

4 The All-Pairs Problem

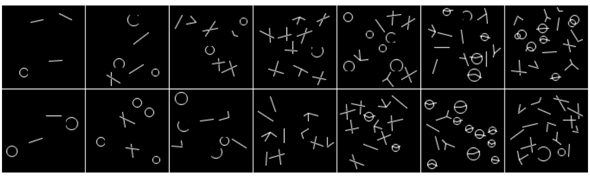

Figure 2: All-Pairs examples from 2-2 on the left to 8-8 on the right. The bottom row is $true$ and the top row is $false$.

4.1 Definition and Examples

Extending the ideas in the Pentomino problem, we use anti-aliased white symbols on a black background to construct the following new problem. The *N-K* All-Pairs problem contains

$2N$ symbols from an alphabet of K choices. Each example is *true* if each of its symbols pairs with a symbol of the same type without reuse, and *false* otherwise. Symbols are positioned randomly with no overlap. Symbols are of similar scales, ranging from 10–18 pixels across, and have differing symmetries (for instance, some are rotationally invariant, while others are not). The exact structure and variations of each symbol are given by the generator code supplied online [9].

Each symbol is shown below with the number of unique ways it can appear, as configured in our experiments. In contrast, the Pentomino problem used 8 variations for each symbol. The symbols are used in the order given, so the 4-4 All-Pairs problem will use **circle**, **line**, **cross**, and **angle**. For this work a 76×76 image is used for $N < 6$ and a 96×96 image is used for larger N.

id	name	examples	cardinality	id	name	examples	cardinality
1	**circle**		165	10	**box**		480
2	**line**		174	11	**box-diagonal**		518
3	**cross**		45.3k	12	**barbell**		78
4	**angle**		39k	13	**dot-line**		156
5	**3-star**		1.43M	14	**z**		518
6	**theta**		20k	15	**triangle-lid**		1036
7	**phi**		20k	16	**dot-mid-line**		78
8	**2-circle**		7k	17	**hourglass**		518
9	**circle-3star**		7.15M	18	**triangle**		11.8k

Figure 2 shows a *true* and a *false* example for the 2-2 to 8-8 All-Pairs problem. A data generator for All-Pairs is used to generate on-demand, unique training examples (the 4-4 All-Pairs problem has approximately 10^{28} unique images), and a fixed validation set is generated at the start of training. The separability of the eighteen symbols was confirmed by training a simple conv-net to 100% test accuracy in 350k training samples.

4.2 Comparison with Conventional Results

Conventional algorithms from the literature have difficulty with the 4-4 All-Pairs problem, as shown in the following table. Clearly, of the hundreds of conventional, valuable DNN algorithms, there may exist some that can solve the 4-4 problem. One open challenge is to identify them and extend training techniques to efficiently solve these types of problems. Of the runs of each algorithm summarized below, none achieved more than 92% test accuracy after training on 100M samples. An expert human made one mistake in 100 samples for each of the All-Pairs problem from difficulty 4-4 to 7-7, taking 8-9 seconds to classify each image. Humans use sequential attention and working memory to do the All-Pairs task, suggesting the task as a benchmark for building sequential models. TypeNet consistently achieves 100% test accuracy in the 4-4 All-Pairs problem using 20k test samples.

algorithm	model size	normalized size	accuracy	std deviation
TypeNet [×10]	918k	1.0	**1.000**	**0.000**
Expert Human [×1]	–	–	**0.990**	–
Relational Net [×10]	**630k**	**0.7**	0.867	0.078
Inception v3 [×10]	22M	24	0.803	0.079
Resnet-34 [×10]	21M	23	0.788	0.068
Resnet-18 [×10]	11M	12	0.711	0.157
Vgg19 [×6]	139M	151	0.509	0.002
Vgg16 [×3]	134M	146	0.506	0.002

5 Toward an All-Pairs Solution

Algorithm 1: TypeNet algorithm

Data:

- Number of layers, N_c and N_f.
- Number of type branches, N_t, and spatial branches, N_s.
- Activations, $\textsc{Ac}$, and convolutions, $\textsc{Conv}$, for feature extraction layers.
- Activations, A, and n 1x1 convolutions, $\textsc{Conv}1{\times}1$, for type matching.
- Spatial diversity operations, $\textsc{Spatial}$.
- Activations, $\textsc{Afc}$, weights, W, and biases, B, for fully-connected layers.

$C = \textsc{Image}$ `# convolution block`
for $i = [1 \rightarrow N_c)$ **do**
  $C = \textsc{Ac}_i(\textsc{Conv}_i(C))$
  $C = \textsc{BatchNorm}(C)$
$\mathrm{T} = \sum_{i=0}^{N_t} \mathrm{A}_i(\textsc{Conv}1{\times}1_i(C))$
$\mathrm{Y} = \textsc{Concatenate}\Big(\big[\, \sum_{w,h} \textsc{Spatial}_i(T) \text{ for } i = [0 \rightarrow N_s) \,\big]\Big)$
for $i = [0 \rightarrow N_f)$ **do** `# fully-connected layers`
  $Y = \textsc{Afc}_i(W_i Y + B_i)$
  $Y = \textsc{BatchNorm}(Y)$
return $\textsc{SoftMax}(Y)$

5.1 Type-Net Model

After verifying that a fully-connected model can easily solve the 4-4 All-Pairs problem from the histogram of symbols in each image, we designed and tested a generic model capable of learning a similar, whole-image statistic. The resulting model was created using insights derived from the All-Pairs problem, but does not make use of explicit problem details or enhanced training data.

We refer to the resulting network as a TypeNet because it estimates the affinity of each receptive field to n ideal types (via a dot-product) and then aggregates those type-affinities over the spatial extent. This spatial summation is global for solving the All-Pairs problem, but could be spatially restricted to produced learned features similar to histogram of gradients (HOG) found in [16]. A learned attention mask could also generalize the summation to salient areas of each image.

Model details can be found in the supplementary material and in the online sample code found at [9]. The general algorithm for TypeNet is presented in Algorithm 1. The algorithm begins and ends conventionally with a convolution stack and fully-connected layers, respectively. Lines 5 and 6 show the key steps for the algorithm:

- line 5, the 1×1 convolution implements a dot-product similarity with a learned kernel, these are the "types" of TypeNet.
- line 5, the activation, A_i, applied was experimentally studied:
  - A_i = SOFTMAX in the feature dimension, gives a soft N_t-hot representation here which was seen to reduce variance in training times).
  - A_i = IDENTITY was the most versatile activation and can be seen as creating a "type" difference operator
- line 5, superposition (via summation) of learned template matching
- line 6, diversify spatially with non-linear operators such as MAXPOOL.

The goal of introducing TypeNet is to expand the palette of techniques available to solve similar types of problems and decrease the problem specific reasoning required in similar domains (such as parity, counting, holistic scene understanding, and visual query answer), which can be solved from a histogram-like summary of local statistics.

5.2 Contrast to Relational Methods

Relational neural learning generally accomplishes it's goal by learning a functional over (i, j) tuples in a latent feature space f. In Relational Networks [21] for example, the model learns two functionals $[h, g]$ (parameterized by deep-neural networks) that **exhaustively** operate over **all** (i, j) pairs in the latent feature space of a deep-convolutional network as shown in the table below. Memory Networks [30] on the other hand learn a probabilistic relationship between the input query (embedded into a feature representation) f_i and an associated set of memory vectors $M = \{m_1, ...m_i, m_N\}$, followed by a smoothed weighting against an embedded query vector c_i.

Relational Networks	Memory Networks
$g(\sum_i \sum_j h(f_i, f_j))$	$p_i = \text{softmax}(f_i^T, m_i) \qquad o_i = \sum_i p_i c_i$

Relational Networks [21] have high computational complexity when the dimensionality of the feature-space f is large. Memory-networks on the other hand scale proportionate to the number of embedded memories dim(M). Our objective with TypeNet is twofold: relax computational constraints compared to these relational models and incorporate the probabilistic smoothing of Memory Networks [30].

We reduce the computational complexity by forcing the model to divide the input representation through a set of N_t branches. This division allows the model to learn a disparate feature representation per branch. Rather than learning over every (i, j) as in Relational Networks [21] we approximate this with a spatial sum after our branch-divide strategy.

During our branching strategy we do a sum across an activated feature space; this can be interpreted as a probabilistic weighting of the features of each individual branch against each other. Training TypeNet to convergence is **8-10 times faster** than training a comparable Relational Network and produces **more accurate results** in the weak-supervised learning scenario of All-Pairs.

5.3 All-Pairs Result

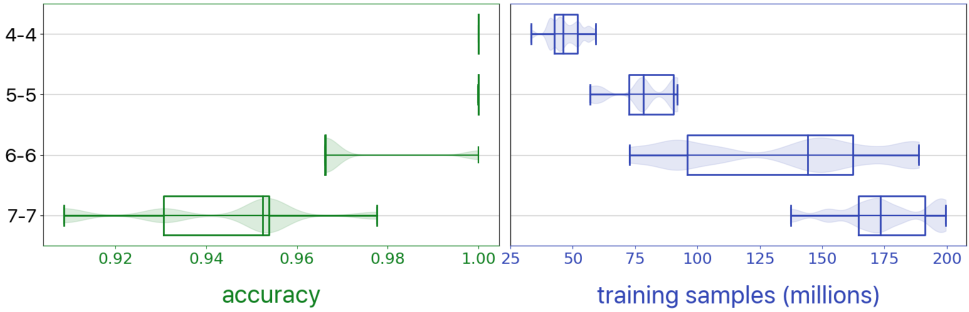

Figure 3: Training results, showing validation accuracy and total number of training sample for TypeNet on increasingly difficult versions of All-Pairs, from 4-4 to 7-7. Shading shows the distribution over 10 trials. Note, conventional DNN models cannot solve the 4-4 problem.

As described, the TypeNet for the All-Pairs problem has 1.04M trainable parameters (many times smaller than the baseline models). Unless otherwise noted, we used the following training setup for the TypeNet results on the All-Pairs problem: 4 GPUs, batch size 600, Adam with learning rate = 0.001 and no weight decay, cross-entropy loss, test results reported every 50k training samples, and 100M total training samples. A 100M sample training run typically takes 20 hours on 4×P100 GPUs.

The main hyper-parameters and architecture-variations explored are the feature activation, number of branches (k), and number of features (n). Details of those studies can be found in the supplementary material. We concluded that $k = 2$ and $n = 64$ performed well on the All-Pairs problem. In the case of the 4-4 problem (see Figure 7.2), we observed that the utilizing only 1 branch resulted in requiring approximately 10M more samples to reach convergence. Increasing the number of features to 96 results in slightly lower training times, at a cost of a larger model. All options explored reached 100% accuracy.

The TypeNet approach cannot easily solve every All-Pairs problem; Figure 3 shows results for the 4-4 to 7-7 All-Pairs problem. We see an increase in the magnitude and variance of the number of samples needed for convergence. The plot shows the results of 10 training runs for each difficulty level; TypeNet can solve the first 3 of these challenges to 100% validation accuracy. No model and training methodology has been found that solves the 7-7 problem to 100% accuracy. An inspection of the errors made by the best 7-7 solution shows that they are systematic, unambiguous errors.

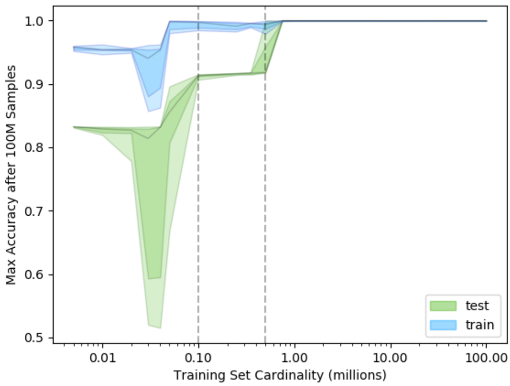

Figure 4: Training the TypeNet on 4-4 All Pairs with 100M samples drawn from a fixed-size training set.

5.4 Training Set Size

In Figure 4, we show the effect of reducing the cardinality of the training data from effectively infinite to sizes smaller than the total number of training samples presented. A training set cardinality of 100k is minimal for 90% test accuracy and 500k is minimal for some trials to reach 100% test accuracy. The increased variance in both train and test accuracy at cardinality 30k is interesting. We hypothesize this is due to sampling noise for these small sizes leading to significantly different train and test distributions. For larger cardinality, both sets consistently represent the same distribution; for smaller sets, learning is limited enough that distribution differences are not apparent.

To avoid the overhead of datasets on disk, varying the training set cardinality is accomplished using an array of seeds for the data generator. Each seed is used to generate 1k samples. When each seed in the list has been used once, the list is shuffled and the process starts back at the beginning of the list.

5.5 Other Applications

TypeNet was evaluated on other datasets to determine its applicability to common classification problems. The following table presents results for the test accuracy from four training runs. For training, each dataset was augmented by random original-size crops (padding of 4), random rotations from $0°$ to $4°$, and normalized by subtracting 0.5. CIFAR10 and Fashion MNIST [31] were also augmented with random horizontal flips. A detailed discussion and comparison with a simple convolutional net can be found in Supplement 7.4.

	ConvNet		TypeNet	
dataset	accuracy	# parameters	accuracy	# parameters
MNIST	0.9953 ± 0.0002	2M	$\mathbf{0.9971 \pm 0.0006}$	**1M**
Fashion-MNIST	$\mathbf{0.9409 \pm 0.0005}$	2M	0.9346 ± 0.0011	**1M**
CIFAR10	0.7773 ± 0.0013	2.5M	$\mathbf{0.8820 \pm 0.0080}$	**1M**
4-4 All-Pairs	0.8080 ± 0.0925	9.9M	$\mathbf{1.0000 \pm 0.0000}$	**1M**

For these classification tasks, adding more spatial information via two parallel pathways branching from the **similarity** step (algorithm line 5) and joining at the **concatenation** step (line 6) was useful. One of these extra pathways has a MAXPOOL3X3 and the other has AVGPOOL3X3 after the **similarity** step. This enhanced model also solves the All-Pairs problem and has 10% more parameters than the simpler TypeNet presented as a minimal version for All-Pairs.

6 Conclusion

In this work, conventional training methods and model features have been demonstrated to solve a previously unsolved task by training a black-box model to solve the Pentomino problem. The All-Pairs problem is introduced as a challenge to the research community by measuring the limits of conventional model performance, introducing a model advancement (TypeNet) to solve such problems, and measuring the limits of TypeNet on the All-Pairs and conventional image classification benchmarks.

Future work will aim at improving sample complexity of learning in WSL environments. Most existing gradient based WSL models (including TypeNet) require millions of samples (see Figure 3 (right)). Humans on the other hand, learn from very few labels; paralleling this is quintessential for the progress in the domain. The following extensions to the TypeNet model and its training may prove useful or generate further insights: (1) filtering the data generator output to study supervised and unsupervised curriculum learning, (2) generating multi-scale statistics before the final **fully-connected** layers, (3) annealing a $softmax$-type activation during training to help the network seek better minima, and (4) using the TypeNet structure in a residual architecture by adding the post-**superposition** block of features back to the **conv** block. The hope is to direct research toward valuable investigations and to promote a methodology of falsifiable scientific claims both by falsifying previous claims and by making further claims which, if we believe Popper [19], are likely to be false.

References

[1] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[2] Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. Chains of reasoning over entities, relations, and text using recurrent neural networks. *arXiv preprint arXiv:1607.01426*, 2016.

[3] Thibaut Durand, Nicolas Thome, and Matthieu Cord. Weldon: Weakly supervised learning of deep convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4743–4752, 2016.

[4] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256, 2010.

[5] Caglar Gulcehre. Arcade-universe, Jun 2015. URL `https://github.com/caglar/Arcade-Universe`.

[6] ÇaÇğlar Gülçehre and Yoshua Bengio. Knowledge matters: Importance of prior information for optimization. *The Journal of Machine Learning Research*, 17(1):226–257, 2016.

[7] 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*, pages 770–778, 2016.

[8] Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. *arXiv preprint arXiv:1705.08741*, 2017.

[9] Apple Inc. All-pairs data generator and sample code. https://github.com/apple/ml-all-pairs, 2018.

[10] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[11] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1988–1997. IEEE, 2017.

[12] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2014.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[14] 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.

[15] Wolfgang Maass. On the computational power of winner-take-all. *Neural computation*, 12(11):2519–2535, 2000.

[16] Robert K McConnell. Method of and apparatus for pattern recognition, January 28 1986. US Patent 4,567,610.

[17] Mehdi Noroozi, Hamed Pirsiavash, and Paolo Favaro. Representation learning by learning to count. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5898–5906, 2017.

[18] Maxime Oquab, Léon Bottou, Ivan Laptev, and Josef Sivic. Is object localization for free?-weakly-supervised learning with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 685–694, 2015.

[19] Karl Popper. *The Logic of Scientific Discovery*, pages 57–74. Hutchinson & Co., 1959. Logik der Forschung first published 1935 by Verlag von Julius Springer, Vienna, Austria.

[20] Reuven Y Rubinstein and Ruth Marcus. Efficiency of multivariate control variates in monte carlo simulation. *Operations Research*, 33(3):661–677, 1985.

[21] Adam Santoro, David Raposo, David GT Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. *arXiv preprint arXiv:1706.01427*, 2017.

[22] Shai Shalev-Shwartz, Ohad Shamir, and Shaked Shammah. Failures of gradient-based deep learning. In *International Conference on Machine Learning*, pages 3067–3075, 2017.

[23] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[24] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

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

[26] Rupesh K Srivastava, Jonathan Masci, Sohrob Kazerounian, Faustino Gomez, and Jürgen Schmidhuber. Compete to compute. In *Advances in neural information processing systems*, pages 2310–2318, 2013.

[27] Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. In *Advances in neural information processing systems*, pages 2377–2385, 2015.

[28] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[29] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *International Conference on Machine Learning*, pages 1058–1066, 2013.

[30] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

[31] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017. URL http://arxiv.org/abs/1708.07747.

[32] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.

[33] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.