Approaches for Deep Learning in Data Sparse Environments

Joshua Haley, Ross Hoehn PhD, Jeremiah Folsom-Kovarik PhD, Robert Wray PhD, Richard Pazda, Brian Stensrud PhD

Soar Technology Inc.
12124 High Tech Ave 350
Orlando, FL 32826
first.last@soartech.com

ABSTRACT

Deep Learning (DL) techniques offer innovative solutions advancing DOD training, such as planning training paths or inferring training needs based on performance observations. However, effective DL requires labor-intensive collection and labeling of large amounts of data. When operationally relevant datasets are difficult to acquire, DL offers reduced benefit in data sparse environments. We examine the feasibility of leveraging domain knowledge and machine transfer learning to make initial progress in reducing the initial data required for a representative DL algorithm. Easily generated synthetic data provides a useful starting point while data is being collected, or provides a valuable supplement to real data in extremely low-data environments. Transfer learning over synthetic data accelerates and enhances DL performance on real data that exhibits regularities describable through domain knowledge.

Using the Sudoku domain, we show that transfer from random puzzles, which are easy to generate, makes DL on published puzzles efficient after adding relatively few published examples. Furthermore, DL performance increases as published puzzles are added. The findings suggest that transfer learning can make use of synthetic data to produce initial results and to reduce the real data requirement. The benefits are hypothesized to support delivering instruction in settings with new and emerging tactics, and in any circumstance where only minimal data is available to DL.

Index Terms
Deep Learning, Machine Learning, Machine Transfer Learning

ABOUT THE AUTHORS

Joshua Haley is a Software Engineer in Soar Technology's Intelligent Training Division. He holds both Bachelor and Master degrees of Computer Engineering from the University of Central Florida (UCF) focusing in Machine Learning and Intelligent Systems. At SoarTech, Josh uses his Machine Learning experience to develop systems and scenarios for Intelligent Training Systems as well as Cognitive agents who play a role in such training systems. His academic and research interests include Artificial Intelligence, Machine Learning, Non-Declarative Knowledge Representation, and Computational Intelligence. Josh is currently a Doctoral student at UCF.

Ross Hoehn, PhD is a research scientist in Soar Technology's Intelligent Training division. He earned his Ph.D. in Theoretical Chemistry from Purdue University in 2014, and continued research in chemical physics, quantum information and artificial intelligence until 2018. His research areas include: adaptive artificial intelligence, generative AI techniques, team learning and training, pedagogy, biological-based agent simulations, swarm mechanics, quantum information science, quantum computing and quantum mechanically-driven biophysical phenomenon. He was an active researcher and manager of the NSF Center for Chemical Innovation: Quantum Information for Quantum Chemistry, a multi-million-dollar multi-year research effort centered at Purdue University to utilize quantum computing for quantum mechanical calculations

J.T. Folsom-Kovarik, PhD is the lead scientist at Soar Technology, Inc. for adaptation and assessment within intelligent training. His research combines modern data science and machine learning with SoarTech's trusted expert models of human cognition. Together, the approaches yield intelligent decisions in real-world training settings when available data is small, concepts evolve over time, or nontechnical users need to control the training.

Richard Pazda is a Software Engineer in SoarTech's Intelligent Training division. He earned his B.S in Computer Science from the University of Central Florida in 2018. His development interests include simulation/training environments, user interfaces, web applications, and data visualization.

Robert Wray, PhD is Senior Scientist at Soar Technology. He received a Ph.D. in computer science and engineering from the University of Michigan. At Soar Technology, he leads or has led R&D projects for the Air Force and Navy research offices, and the Defense Advanced Research Projects Agency. Dr. Wray's research encompasses many areas of artificial intelligence research including agent-based systems and agent architectures, machine learning, cognitive science, intelligent tutoring systems, and knowledge representation and ontology.

Brian Stensrud, PhD is a senior research scientist at Soar Technology and currently serves on the executive management team for its Intelligent Training division. On staff since 2003, Brian has provided scientific and technical leadership for over $25M of DoD research efforts in the areas of interactive human behavior models for simulation, serious games, adaptive training, intelligent user interfaces and robotic platforms. Brian received a Ph.D. in Computer Engineering from the University of Central Florida (2005) and holds bachelor's degrees in Electrical Engineering and Mathematics from the University of Florida (2001).

Approaches for Deep Learning in Data Sparse Environments

Joshua Haley, Ross Hoehn PhD, Jeremiah Folsom-Kovarik PhD, Robert Wray PhD, Richard Pazda, Brian Stensrud PhD

Soar Technology Inc.
12124 High Tech Ave 350
Orlando, FL 32826
first.last@soartech.com

INTRODUCTION

Deep Learning – as introduced by (LeCun et al, 2015) – is a form of Machine Learning (ML) model inspired by neural structures observed in the brain. Deep Learning (DL) has had a transformative impact on tasks such as image recognition, content generation, machine translation and strategic game AI (Silver et al, 2017)(Girshick et al, 2018) While artificial neural networks have existed for quite some time, recent advances in computational hardware – especially the availability of powerful graphics processing units (GPUs) – have made larger network architectures possible on cheaper commercial hardware. This additional hardware power has been accompanied by an increase in the number of layers engineered into neural network models.

The problem is that as model complexity is increased, data requirements to achieve accuracy and prevent overfitting increase as well. Too often machine learning approaches are treated as a panacea to alleviate the cost of accumulating the totality of definitions and knowledge to define and describe a domain, referred to as semantic knowledge. However, we believe that situations exist where exploiting this semantic knowledge can help alleviate the data volume required. Our hypothesis is that semantic information can be synergistically leveraged to help bootstrap data intensive DL approaches by facilitating the generation of training datasets. This bootstrapping process is effective instructing the system on a similar, yet distinct, problem with the intent that this gleaned knowledge can be transferred to the new (real) task marked by real data.

This ML Transfer Learning is accomplished by presenting a similar training problem that is data rich in order to train the feature extraction capabilities of intermediate layers. As an example (Figure 1 provides a visual representation of this example), take a simple case of two-dimensional regression with only a couple data points of a complex function in the form of $\sin(x) + \frac{1}{2}x$, shown by a blue line in Figure 1. If we only use the data observed to fit, we could conclude that the correct relation is an inverted parabola which perfectly fits our data, but does not match the real manifold at all. However, if domain knowledge informs us that there exists a linear component to the function, we can seed artificial observations. We will not perfectly fit the function, but our fit becomes closer for initial performance and the sinusoidal nuances adapted overtime as more real observations are collected

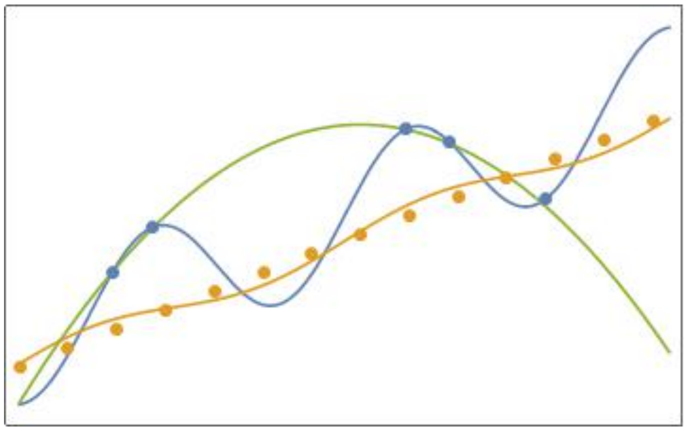


Figure 1 2D Regression Example

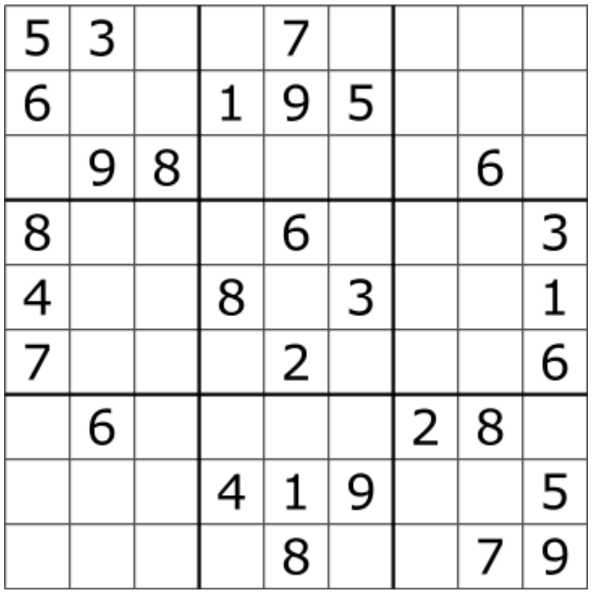
Operationally Relevant Domain

DL algorithms are ideal tools for many applications reliant on identification, learning over repeated instances, and pattern recognition. DL algorithms have found utility in target recognition, self-driving vehicles, prevention of cyber-attacks and the dynamic construction of new training scenarios (Bojarski et al, 2016; Loukas et al, 2017; Haley et al, 2018). There exist many applications where the number of valid training instances are unavailable, either due to the lack of observed instances or the rarity of an event, making the collecting and curating of a sufficient training library for DL difficult. An operationally relevant example where synthetic dataset generation would directly benefit the armed services is Battle Damage Assessment (BDA). BDA is the inventorying of damage on facilities, equipment, and capabilities inflicted by stand-off weaponry, and is a vital task in during conflicts. A specific instance of BDA is accounting for ordnance damage to airport runways on embattled military installations. The United States Air Force Corps of Engineers is developing the capability of small UAVs to remotely detect ordnance damage to runways, and is automating the recognition and alert processes using computer imagery and machine learning. To train the system, ordnance craters on runways were photographed, craters which the Air Force, themselves, installed in their own runways. This is a costly practice, which is apparent in the lack of image data and the hundreds of photographs of a single crater instance (with varying lighting, altitude, and angle). Employing the principle of synthetic dataset generation to this problem, we would identify some semantics observations of the imagery (i.e. Craters are mostly round discolorations). We could then artificially combine these features with existing imagery to produce synthetic examples of new craters, with variability in depth \& size. While these reproductions will be lacking the nuance and noise features of the real crater photographs, they will be much cheaper to produce. The expanded dataset would generally benefit the training of the system, while the variability within the new instances would increase performance.

In order to explore the hypothesis that limited domain knowledge can bootstrap data intensive DL approaches, a case study was performed on the efficacy of using synthetic data in building a DL solver for Sudoku Puzzles with limited published puzzles.

CASE STUDY DOMAIN - SUDOKU

A Sudoku is a puzzle embedded onto an nxn grid (where the typical value of n is 9); the goal is to completely populate any unoccupied spaces within the grid such that each column, row and 3x3 sub-grid contains the integer values 1-9 exactly once, disregarding canonical order (Sudoku, 2019). A well-formed Sudoku puzzle, such as the one shown in Figure 2 will provide a partially completed grid with exactly one correct solution to complete the blank spaces that does not require guessing. While a 9x9 grid can be exhaustively searched via any common search method, in general Sudoku puzzles are NP-complete and are equivalent to a satisfaction of constraints to be met simultaneously (Lynce & Ouaknine, 2006). These competing constraints lead to several reoccurring patterns and multi-step logical inferences that people use to incrementally solve the puzzle in an engaging manner (Sudoku Wiki, 2019). These emerging patterns – the inferred doubles pattern can be seen in Figure 2 for an example – are analogous to the operational nuances encountered in real observations in operational domains.



5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Figure 2 Example of an Authored Sudoku Puzzle (Sudoku Wiki, 2019)

Generating engaging published Sudoku puzzles requires an explicit modeling of these patterns, but it is far simpler to generate poorly formed random puzzles by starting with a completed puzzle and then randomly deleting digits. These poorly formed puzzles do not have a guarantee of unique solutions, nor that they can be solved without guessing. However, they do represent data that can be generated with limited knowledge of the domain.

METHODOLOGY

Synthetic Training Data Generation

To simulate a data sparse environment, formally authored Sudoku puzzles were harvested and a population of synthetic puzzles were generated. Formally authored Sudoku were scraped from an online source, Extreme Sudoku© (Sudoku Too Easy?, 2019). Scraped puzzles covered a spread of difficulties with labels including 'easy', 'hard', egregious' and 'evil'. The generation of synthetic puzzles was performed by generating permutations of entirely solved Sudoku boards, and randomly removing a specified number of entries. A series of puzzles were produced by employing the above procedure; these synthetically derived puzzles lacked some basic qualitative characteristics present in the formally authored puzzles. This quality difference in synthetic puzzles defines a population of puzzles that are both easier to solve and distinct from their formally authored counterparts.

Deep Learning Model

In any DL model, design decisions of how to represent the input to the network, neural network architecture, output representation and output interpretation are engineering design decisions. Input and Output representations are commonly referred to as feature engineering, a process to encode semantic information to the underlying model that often is domain dependent. The specific neural network architecture is often dependent both upon the domain which dictates the types of layers (ex: Convolutions for Image Recognition vs LSTMs for Machine Translation) while the specific configuration (depth of the network, width of the layers) is determined using a parameter optimization process. The increasing depth of layers in a DL model extract higher level features and patterns for later layers to reason upon, thus the complexity of the problem often dictates the number of layers required. For the Sudoku task, we present the puzzle to the model in its entirety and receive as output weightings for particular digit labels.

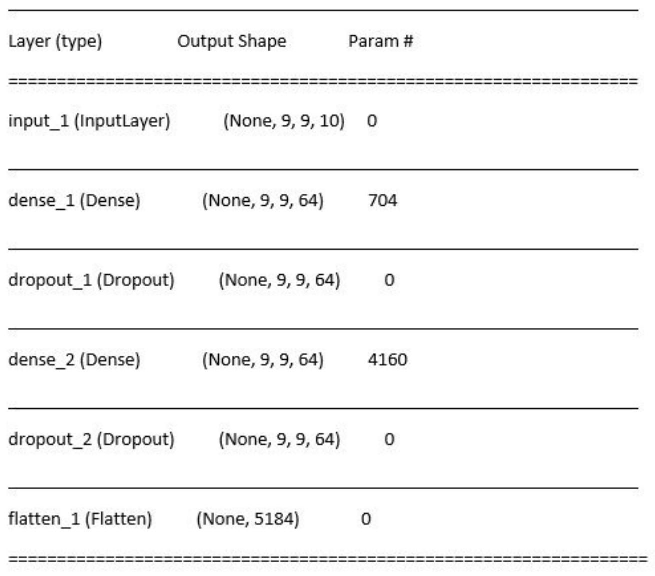
Input Representation

While it would be tempting to represent each digit's input as a nominal value, there is not a quantitative relationship between the numbers. It is in fact a categorical problem, where the numbers 1-9 could be replaced by any unique value. To represent the categorical nature of inputs, each digit was transformed into a one-hot encoding for the labels of 'unknown' and the 9 possible digit values. This has an input tensor shape of (9,9,10) in which a binary array of length 10 represents each digit.

Output Representation and Interpretation

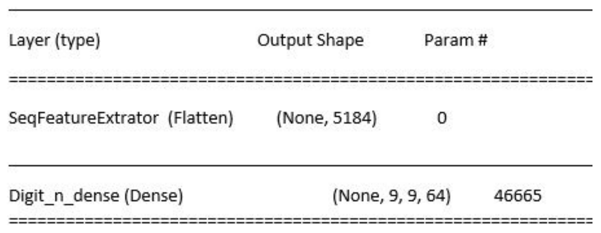
Similar to the input representation, the output representation is encoded as a categorical output vector. While there are 10 possible inputs, the outputs of the network should include no unknown digits and thus has a shape of (9,9,9) with a [0,1) valued output for each digit.

For each digit, the 9 valued vector is interpreted to be a weighting toward a specific label. Rather than fill in all unknown puzzle digits at once, we instead us infer the value of the highest weighted label across all digits as recommended in (Kyubyong, 2016). We then represent this more complete Sudoku back to the neural network before inferring the next digit and continue the process until the entire puzzle has been completed.



```
_________________________________________________________________
Layer (type)            Output Shape          Param #
=================================================================
input_1 (InputLayer)        (None, 9, 9, 10)    0
_________________________________________________________________
dense_1 (Dense)          (None, 9, 9, 64)       704
_________________________________________________________________
dropout_1 (Dropout)       (None, 9, 9, 64)       0
_________________________________________________________________
dense_2 (Dense)          (None, 9, 9, 64)       4160
_________________________________________________________________
dropout_2 (Dropout)       (None, 9, 9, 64)       0
_________________________________________________________________
flatten_1 (Flatten)       (None, 5184)          0
=================================================================
```

Figure 3 Feature Extraction Layers



```
_____________________________________________________________
Layer (type)                     Output Shape         Param #
=============================================================
SeqFeatureExtrator  (Flatten)      (None, 5184)          0
_____________________________________________________________
Digit_n_dense (Dense)                    (None, 9, 9, 64)       46665
=============================================================
```

Figure 4 Softmax Digit Classifier Layers

The Neural Network architecture was developed during initial testing on the ability to solve Sudoku puzzles on the synthetic data. There are two functional layer groupings to our model. A fully connected component which takes in input and has several fully connected layers as outlined in Figure 3 acts as a feature extractor to identify the interdependence between different input locations as per (LeCun et al, 2015). We use two fully connected dense layers with sigmoidal activation functions followed by Dropout layers which act to prevent over fitting to the data (Srivastava et al, 2014). This output, representative of discovered computational features, is then flattened into a single vector of size 5184. This flattened output is then input into a fully connected dense layer as shown in Figure 4 for each digit with a softmax activation function to predict a probability between [0,1] that a specific value should fill this digit. Given that there are 81 such classifiers, one for each digit in the Puzzle, and the architecture of the feature extractor component, we have 3,784,729 parameters that need to be tuned by our training process.

Training Process

Supervised learning is the process by which known inputs and expected output pairs are presented to the network and then the difference between expected and actual outputs are used to adjust the model weights toward a more optimal configuration. The different between expected and actual outputs are characterized by a loss function to be minimized.

Rather than expect our model to train to a full difficulty Sudoku, we instead train incrementally by constructing progressively more difficult puzzles from our training data. We accomplish this by filling in a puzzle with digits from the solution until we have a specified number of blanks remaining. We then train the model iterative on a progressively increasing number of blanks within the puzzle. Training in both incremental and non-incremental manners were tested with the incremental approach outperforming the non-incremental approach by a factor of $2x$. This is another form of Transfer Learning whereby pre-training the model on a simpler puzzle, we are able to incrementally fit the search space before trying to optimize for a more complex. Given the described iterative training process, we for each training pass use an Adam optimization process with categorical cross entropy as the loss function measure we are minimizing (Kingma & Ba, 2014). During initial passes, our validation loss continues to decrease after several Epochs, but after several passes have occurred the number of epochs before over-fitting decreases as can be seen in Figures 5-6. We believe that is due to the amount of new information to the network decreasing during each training pass as the model has already learned many features from the earlier passes. To prevent such overfitting, the number of Epochs is dynamic via the use of an early stopping criteria that ends training after validation loss has not decreased after five Epochs. We continue performing iterative training passes until the model is operating on Sudokus with 64 blanks.

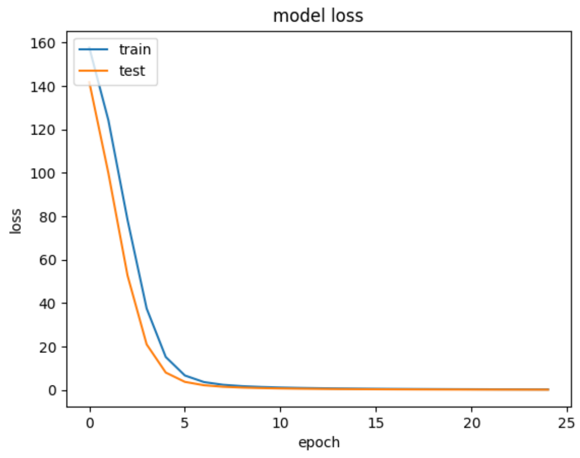


Figure 5 **Pass 1 Training with 1 Blank**

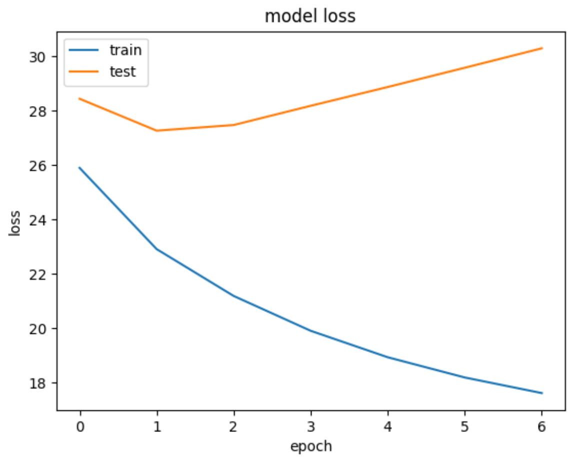


Figure 6 **Pass 40 Training with 40 Blanks**

EXPERIMENT

To validate the proposed data supplementation, we generated two distinct series of model training data. The first series is comprised of data, $data_1$, sets built from a specified number (r) of web-scrapped, formally authored puzzles ($|data_1| = r$ \$). The series is comprised of data sets constructed by combining a fixed number (s) of synthetic data ($\{S\}$) with a specified number (r) of real puzzles($s = 25000$; $data_2 = s \cup data_1$; $|data_2| = r + s$).

Fixed r samples were collected from each of these two populations, acting as the training data for two comparable ML models intended to solve Sudoku puzzles (*e.g.* $model_{800}^{Real}$ and $model_{800+25000}^{Real+Synthetic}$). Each set of authored data, with cardinality r, was pulled from a subset ($\{Data_{Real}\}$) of all scraped puzzles of cardinality 20,000; $\{Data_{Real}\}$ was shuffled between each sampling of r data forming $data_2$. A second subset of authored data ($\{Test_{Real}\}$; $Test_{Real} \cap Data_{Real} = \emptyset$) of cardinality 9,600 was set aside to act as testing data. A single model was trained on each of the r-cardinality data sets, and another series of models were trained over the $(r + s)$-cardinality data sets. Following training, each model was evaluated against $\{Test_{Real}\}$, the number of accurately solved puzzles, the average accuracy of each solution, and the variance of the average accuracy were all determined. This set-up allows us to examine the effect of transfer-learned synthetic information to supplement insufficient real-world data during model training.

RESULTS

Figure 7 gives a list point plot of the average accuracy over all $n = 9600$ puzzles within the $Test_{Real}$ data set for each of the two training schema, real (red) and real supplemented with synthetic (blue). Abscissae marks the number of (cardinality) authored puzzles within the training data set, and ordinate represents the average accuracy over each of the \$n=9600\$ puzzles. The variances are also plotted for each average, providing certitude of the separability of the two training schema.

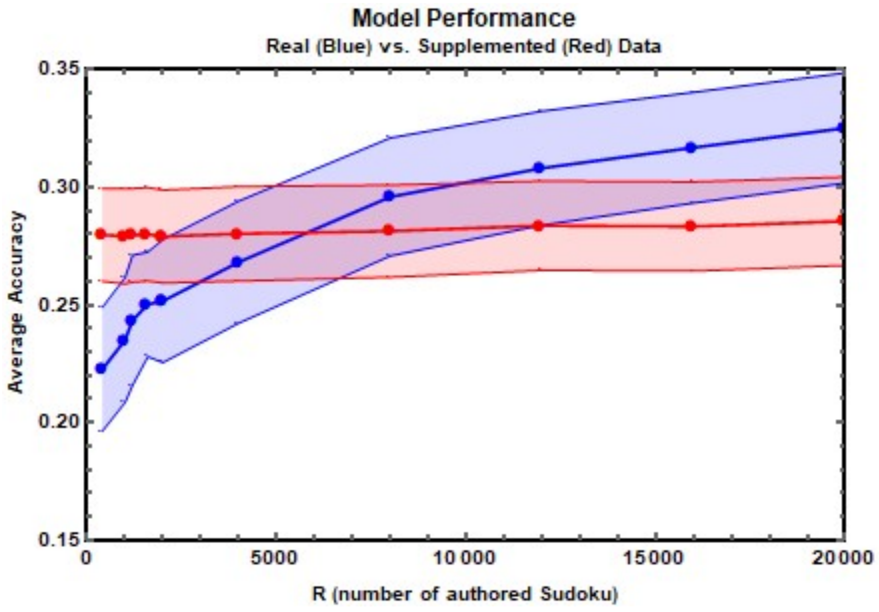


Figure 7 Real (Blue) vs Supplemented (Red) Dataset Performance

It is evident from Fig. 7 that in the region of low r-values, $r < 2000$, there is a large effect on the synthetic-inclusive model due to the additional supplemental training data. Within this low-r range, the supplemental data provides a sufficient bootstrap to the system, and marks a sufficiently high average accuracy. This relatively high average accuracy implies that the synthetic training data contains pertinent information, thus enabling the system to effectively learn analogous concepts and transfer that knowledge to solve authored Sudoku.

Within the large-r range of Fig. 1, the performance (measured by average accuracy) of the supplemented training scheme remains fairly static over the span of r-values. The average accuracy of the real-only scheme eventually outstrips the bootstrapped scheme. This is reasonable as the synthetic training data is qualitatively different – while being similar – from the authored puzzles. As the number of real training instances increases, the qualitative difference begins to be learned by the system, resulting in the real model (though trained on less data) to perform better than the synthetic-inclusive scheme.

CONCLUSIONS

It can be seen from the results that in situations where a very small number of real data instances are available, datasets supplementing real data with well-informed synthetic data can outperform the limited real-world training data. This supports our hypothesis that limited domain knowledge can be leveraged to bootstrap data intensive DL approaches. However, we do see a point at about 7500 real instances where the performance of models train only with real data begin to outperform the supplemented dataset. We believe that this is due qualitative differences inherent between the synthetic and real data; differences engendered by qualities ill-described in our semantic representation of domain knowledge. This large number of partially representative data imparts non-representative information, akin to noise, in the parameters of the neural network. Drawing on our initial analogy (that to the regression function in Figure 1), this is the equivalent to adding large numbers of linear points such that the sinusoidal component is entirely bleached. Further work should be done to ascertain the correct ratio of real-to-synthetic data to be useful during bootstrapping, yet not detrimental to performance as new data becomes available (as in applications of active learning). One approach would be to keep the synthetic data variable such that it is decreased in cardinally as new data points are collected, gaining the benefit of initial performance, but moderating synthetic data impact as more real data is collected. We also hypothesize that another inflection point should exist once the cardinally of real data becomes greater than the synthetic data where performance will begin asymptotically approaching that of the real data only but did not have enough real data to determine this to be true. Given the support of our hypothesis, bootstrapping ML-systems with domain-driven data generation widens the aperture of ML applications in operationally relevant domains to problems once hindered by an insufficient volume of training data.

REFERENCES

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, Karol Zieba. (2016) End to End Leanring for Self-Driving Cars. arXiv https://arxiv.org/abs/1604.07316

Girshick, R., Radosavovic, I., Gkioxari, G., Dollár, P., & He, K. (2018). Detectron.

Haley, J., Hung, V., Bridgman, R., Timpko, N., Wray, R. E. (2018). Low Level Entity State Sequence Mapping to High Level Behavior via a DeepLSTM Model. In *Proceedings on the International Conference on Artificial Intelligence (ICAI)* (pp. 131-136). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Kyubyong Park. Can neural networks crack Sudoku?, 2016. URL https://github.com/ Kyubyong/sudoku.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436.

Loukas, George and Vuong, Tuan and Heartfield, Ryan and Sakellari, Georgia and Yoon, Yongpil and Gan, Diane. (2017) Cloud-based cyber-physical intrusion detection for vehicles using deep learning. IEEE Access, 6, 3491-3508

Lynce, I., & Ouaknine, J. (2006, January). Sudoku as a SAT Problem. In ISAIM.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., ... & Chen, Y. (2017). Mastering the game of go without human knowledge. *Nature*, *550*(7676), 354.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1), 1929-1958.

Sudoku. (2019, June 09). Retrieved from https://en.wikipedia.org/wiki/Sudoku

Sudoku Too Easy? (n.d.). Retrieved June, 2019, from http://extremesudoku.info/sudoku.html

Sudoku Wiki. (2019, June). Retrieved from http://www.sudokuwiki.org/