THE Next Wave

Vol. 22 | No. 2 | 2019

The National Security Agency's review of emerging technologies

ISSN 2640-1789 (Print)
ISSN 2640-1797 (Online)

MACHINE LEARNING

[Photo credit: Cale Atkinson | www.cale.ca]

THE Next Wave

The National Security Agency's review of emerging technologies

GUEST Editors' column

Joe McCloskey & David J. Mountain

The previous issue of *The Next Wave (TNW)* focused on the exciting area of machine learning (ML), often described as artificial intelligence (AI) or deep learning (DL). This topic has generated a lot of interest in recent years throughout government, academia, and industry; at times, it has been overhyped and misunderstood, but often, it has proven to be wildly successful. Our ML goals at NSA are to understand its current limitations, apply it to important national defense applications, and extend the theory as required by our hardest mission problems.

In this second special issue of *TNW* on ML, we highlight additional advances in ML by NSA researchers and collaborators. In the first article of this second issue, Lawrence Livermore National Laboratory researchers Barry Chen and Nathan Mundhenk and In-Q-Tel researcher Karl Ni develop computer-assisted algorithms for the multimodal analysis and retrieval of large collections of unlabeled images, video, audio, and text. The lack of labeled data, combined with the required multimodal analysis of large amounts of heterogeneous data, makes this a truly daunting technical task.

The US government relies on the use of ML analytics to help inform decisions made in defense of national security. Often this requires that analytics be publicly deployed, where an adversary can study their behavior, discover weaknesses, and then alter their expected behavior in several ways. Sandia National Laboratory researcher Philip Kegelmeyer's article on adversarial ML presents a statistical framework to assess the vulnerability of deployed analytics and then harden their performance against attacks.

In the modern era, data science can take advantage of massive computational power to deal with large data sets that are contaminated with noise. As data science expands into new applications, ad hoc algorithms are often developed—these take advantage of this available computational power but often lack principled motivation or justification. As a result, these algorithms may perform poorly when used on real data sets in support of applications. In his article, NSA researcher Mark Jacobson has developed rigorous statistical methods to address these issues for an important class of recommender system problems.

Another technology benefiting from AI is the smart digital assistant, with which the intelligence community has tremendous potential to revolutionize analysis by helping analysts find the data and analytics they need, fostering collaborations, and making workflow recommendations. Laboratory for Analytical Sciences (LAS) researchers Paul Jones and S. Lynch, along with North Carolina State University researcher Kathleen M. Vogel, describe an example of ML/AI research at LAS that provides digital assistance for the intelligence analyst trying to search through a deluge of data to discover information of value.

Apple's iPhone X has a "neural engine" that enables it to use facial recognition to unlock itself and also to transfer facial expressions onto an animated emoji. Huawei's Kirin 970, Google's Pixel 2, and other smartphones have similar capabilities. Eventually, companies will start using specialized chips to run radio frequency (RF) DL applications for 5G and Internet of Things (IoT). The article by researchers Rob Miller, Marc Lichtman, and Edward Laird begins to explore the potential to apply DL models for the classification of wideband RF data.

Recent advances in ML technology have resulted in the development and ready access of sophisticated software tools than can be readily consumed by users for a variety of applications. The column near the end of this issue discusses In-Q-Tel's partnership

About the cover: Cover features artist Cale Atkinson's interpretation of a 2017 viral video, in which a little girl mistakes a discarded water heater for a robot, that was popular within the artificial intelligence community.

with start-up company Algorithmia, and the current capabilities of the platform that Algorithmia is developing for this purpose.

NSA needs to establish partnerships with industry, academia, and other government agencies to address its staggering future data science needs, requiring a combination of research, innovation, and collaboration. As one of its initiatives, the NSA Technology Transfer Program has a Cooperative Research and Development Agreement with the University of Texas System to explore joint challenges in ML, innovation, and IoT. The Lab to Market article in this issue provides insight into one of the current areas of exploration, anomaly detection and insider threat activity inside high-performance computing systems.

This second special issue of *TNW* on ML highlights additional facets of NSA research and collaboration in this growing and dynamic field. ML offers great promise for national defense, but also a threat in the hands of our adversaries, who will use ML against us with growing sophistication. Hopefully these two special issues of *TNW* taken together convey both the importance of this topic to NSA now and far into the future, and also provide insight into the complexity and richness of the research and applications of ML for national defense.

Joe McCloskey
Deputy Technical Director
Research Directorate, NSA

David J. Mountain
Technical Director, Advanced Computing Systems
Research Program
Research Directorate, NSA

Contents

The Next Wave is published to disseminate technical advancements and research activities in telecommunications and information technologies. Mentions of company names or commercial products do not imply endorsement by the US Government. The views and opinions expressed herein are those of the authors and do not necessarily reflect those of the NSA/CSS.

This publication is available online at http://www.nsa.gov/thenextwave. For more information, please contact us at TNW@tycho.ncsc.mil.

ISSN 2640-1789 (Print)
ISSN 2640-1797 (Online)

Vol. 22 | No. 2 | 2019

Toward a deep learning system for making sense of unlabeled multimodal data

Barry Chen | T. Nathan Mundhenk | Karl Ni

[Photo credit: scyther5/iStock]

Analysts and decision makers are constantly charged with making sense of large amounts of disparate data. With the proliferation of smartphones and the ever-growing array of new sensor technologies, the old adage of swimming in sensors but drowning in data remains especially relevant. As collections of images, video, audio, and text continue to grow, the need for computer-assisted analysis likewise increases. The "data deluge" presents both daunting challenges as well as remarkable opportunities. In this article, we describe deep learning approaches for learning representations of data from vast amounts of unlabeled or weakly labeled training sets. These feature representations enable the computer-assisted retrieval and tagging that can help analysts find patterns in the vast ocean of collected data.

As an analyst, imagine that you're given a hard drive of collected data and told to find all the data related to your area of expertise by the end of the day. Such a task would be easy if every file on the hard drive was already labeled with a set of keywords describing its content. If you knew all the categories of things that you wanted to find, and if you had a lot of examples of these things, you could use your favorite machine learning (ML) system to automatically find them. Unfortunately, keyword tags are often unavailable and difficult to create. Moreover, getting large labeled training sets may be too costly, and the set of categories you're looking for may change in the future.

On the other hand, if there was a way to map all the unlabeled data in the hard drive to a feature space where *multimodal data* (i.e., text, images, videos, etc.) that are conceptually related to each other are proximal, then you could retrieve relevant data by looking for those that are close in feature space to the things you are looking for (see figure 1). For example, if you wanted to find a biplane, you could look for all the data that are mapped to locations in the multimodal feature space where images, video, audio, and text descriptions of biplanes reside. Such a general-purpose feature space would also allow you to retrieve new categories of interest.

In our research, we use deep neural networks (DNNs) to learn these multimodal feature spaces. Figure 1 depicts our DNNs mapping text descriptions, an image, an audio waveform, and a video of biplanes to proximal locations in a multimodal feature space. The creation of this system of DNNs involves two main steps: First, train DNNs to learn high-quality feature representations of unimodal data, and second, learn mappings from these unimodal feature spaces to the multimodal feature space.

There are many challenges in training a system of multimodal neural networks. First, there are not a lot of labeled data sets to train on. To reduce the need for large amounts of labeled training data, we learn unimodal feature representations with enhanced *self-supervised* deep learning approaches that allow us to learn from vast quantities of unlabeled data. Such representations are designed to capture higher order information so that downstream analytics can operate with more compact and meaningful features.

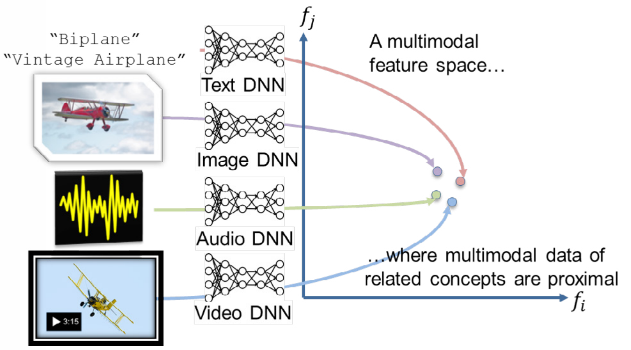

"Biplane"
"Vintage Airplane"
Text DNN
Image DNN
Audio DNN
Video DNN
3:15
f_j
f_i
A multimodal feature space...
...where multimodal data of related concepts are proximal

FIGURE 1. We envision a system of deep neural networks (DNNs) mapping text, imagery, audio, and video into a multimodal feature space where conceptually related data are proximal. In this example, text tags, an image, an audio waveform, and a video of biplanes are mapped to nearby points in the multimodal feature space by a system of DNNs.

Second, word sense ambiguity may pose retrieval challenges. One could argue that the words *up* and *down* are very similar in the sense that they're both directional semantics, and that both are very different from a tangible object like a *peacock*. Then again, a *peacock* is a bird and birds fly, so the word *up* may not be very far off in meaning. So, how would we account for the semantic senses of a single word for truth labels?

Finally, some data sets do have lots of labels, but they can be quite noisy. In the Yahoo Flickr Creative Commons 100 Million (YFCC100M) data set [1] of user-generated content, users can tag images and video in any way that they choose, using any language, with any keyboard (different text coding schemes). This can lead to some creative, but not semantically relevant labels. Our hope is that by training with massive amounts of such weakly labeled data, the noise would cancel out. Indeed, this is the case as we will show below: Training over millions of weakly labeled images leads to more generalizable results than training on much smaller well-labeled training sets.

In the following few sections, we'll take you through the research we've conducted at the Lawrence Livermore National Laboratory and Lab41 to build the beginnings of the multimodal joint embedding space in figure 1. We start from unimodal representations with unsupervised deep learning contributions in the next section, the basic building blocks of higher order deep reasoning. Then, in the following section, we glue the representations together with a joint optimization of the feature spaces to form a multimodal representation, which can serve as a powerful organizing principle to enable better data queries. We demonstrate all of the contributions on large-scale, noisy data sets.

Unimodal unsupervised deep learning

In the past few years, DNNs have exceeded human performance levels on tasks once considered to be quite difficult for machines. In image classification challenges, such as the ImageNet Large Scale Visual Recognition Challenge [2], state-of-the-art DNN systems attain error rates as low as 2.25% [3, 4] compared to 5.1% for a human expert, and in the conversational telephone speech recognition task [5], DNNs now attain word error rates of 5.1%—beating human word error rates of 5.9% [6]. To achieve these impressive results, millions of labeled examples are required for training the DNNs. What sets deep learning apart from other ML techniques today is its increasing effectiveness at improving performance as the amount of labeled training data grows. Hardware acceleration provided by graphics processing units (GPUs) coupled with improved optimization algorithms [7, 8, 9] and new neural network architectures [10, 4] now make it possible to simultaneously reduce bias and variance with large and deep networks (reduces bias) on massive amounts of training data (reduces variance). With enough labeled training data, it is possible to achieve better-than-human performance with supervised deep learning.

Unfortunately, not all data collections are easily annotated. In many commercial applications, harvesting massive amounts of labeled data with crowd sourcing may be feasible. However, for many government or scientific data sets, the data may have sensitivities or require expertise that prevents many people from providing annotations. In these cases, unsupervised algorithms are required to automatically learn important patterns or features from massive amounts of *unlabeled* training data.

Unsupervised learning for neural networks has traditionally been based on a network called an autoencoder, which learns to project input data into a compressed feature space and then from that compressed feature space reconstruct the original data [11]. It is this compressed feature space, which is optimized with the mean squared error between the inputs and reconstructed outputs, that can serve as the unimodal representation we seek. Another family of unsupervised learning algorithms, Generative Adversarial Networks (GANs) [12], changes the cost function in a clever manner to improve the realism of the reconstructed inputs. In GANs, two networks compete against each other: The first one learns features effective for generating input data realistic enough to fool the second one.

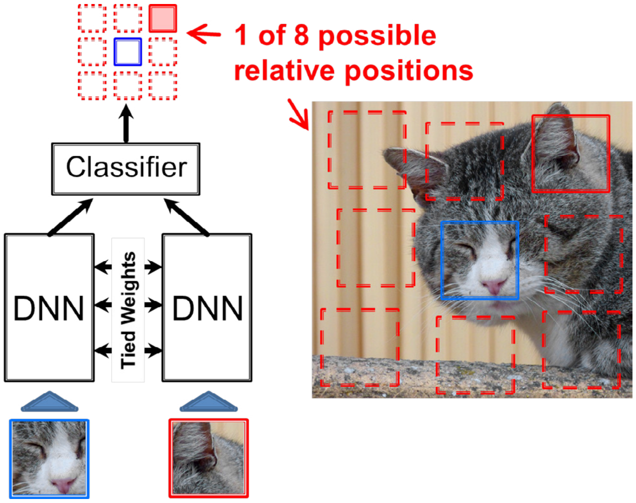


FIGURE 2. Two identical DNNs whose weights are tied (i.e., forced to be the same) automatically learn representations useful for classifying the relative positions of two randomly selected image patches. [Figure adapted from [13]].

A promising new family of unsupervised learning approaches is based on developing meaningful training tasks for which the computer can provide its own training labels. These so-called self-supervised learning approaches, create pretext learning tasks that force the neural networks to learn useful features that can then be recycled for other tasks. An example of self-supervised learning is the work of Doersch [13], where the pretext task is to classify the relative position of one randomly selected image patch to another. As depicted in figure 2 adapted from [13], the neural network is taught to predict that the patch with the cat's ear is northeast of the nose patch. The reason this pretext learning problem works is that it forces the DNN to learn something about what makes a set of pixels ear-like or nose-like and that ears are typically northeast or northwest of noses. Because the computer controls where to sample the two image patches, it knows their relative positions and can thus self-supervise the training of the DNN on massive amounts of unlabeled images.

Once trained, self-supervised DNN features are subsequently used to build classifiers that perform quite well on image classification tasks [13, 14]. We have extended Doersch's original work in several ways to significantly improve its performance. The first extension varies the number and visible area of the patches to increase the probability that meaningful object parts are sampled and sampled over various scales from coarse to fine (see figure 3, triple patch and borders). We typically sample three patches at different configurations: along straight lines, right angles, and at mixed scales. To vary the visible area of the patches, we randomly place gray borders of varying sizes around the patches. This focuses the DNN to learn finer scale pattern and leads to better classification and detection performance.

The second extension is an improved approach for preventing the DNN from "cheating" on chromatic aberrations. Many inexpensive camera systems induce chromatic aberrations that manifest in pushing certain colors radially outward more than others. Our approach blends magenta and green together in L*a*b* color space to prevent the DNNs from using chromatic aberration while keeping the intensity channel crisp. Finally, we train our self-supervised DNNs for many more iterations than is typical for supervised DNNs.

With these extensions, our self-supervised feature learning outperforms the best self-supervised techniques on image classification and object detection tasks. Table 1 compares the mean average precision of our self-supervised approach called multiscale triple patch (MSTP) to two leading self-supervised systems: Doersch's [13] method and the Split Brain autoencoder method [14] for classifying PASCAL VOC imagery (i.e., Pattern Analysis, Statistical Modeling and Computational Learning Visual Object Classes) [15]. Table 2 shows the effectiveness of MSTP for detecting objects in the PASCAL VOC data set. The self-supervised systems are first trained using only the images in the ImageNet training set (i.e., unlabeled data) and then "fine-tuned" using the smaller labeled PASCAL VOC training set. Fine-tuning involves replacing the last layer of a DNN that was pretrained on another task (like the self-supervised learning task) with a brand-new layer that maps the pretrained network to the outputs of the new task. Then all the parameters are adjusted or "fine-tuned" to minimize the cost function for the new task.

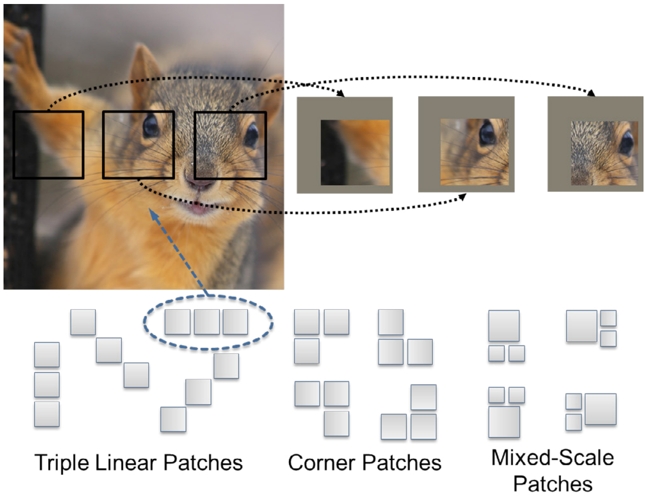


FIGURE 3. In our multiscale triple patch system, our patch orientation classes come from three patches arranged in various configurations with different scales and various visible areas.

TABLE 1. Mean average precision (mAP) performance on the PASCAL VOC *image classification* problem of various systems pretrained on the ImageNet training set and fine-tuned on the PASCAL VOC training set. The Supervised Pretraining system uses the ImageNet training labels, while the other systems are self-supervised methods that do not require labels for pretraining.

Model	mAP (%)
Supervised Pretraining	79.9
Doersch et al.	65.3
Split Brain Autoencoder	67.1
Multiscale Triple Patch	69.6

TABLE 2. Mean average precision (mAP) performance on the PASCAL VOC *object detection* problem of various systems pretrained on the ImageNet training set and fine-tuned on the PASCAL VOC training set. The Supervised Pretraining system uses the ImageNet training labels, while the other systems are self-supervised methods that do not require labels for pretraining.

Model	mAP (%)
Supervised Pretraining	56.8
Doersch et al.	51.1
Split Brain Autoencoder	46.7
Multiscale Triple Patch	55.8

In tables 1 and 2, we also compare the performance of the self-supervised techniques with a system that pretrains on the labeled ImageNet training data. This Supervised Pretraining system provides a measure of how much performance gain we can obtain from having large amounts of *labeled* data for pretraining. Even though the performance for MSTP is 12.9% and 1.8% below that of the Supervised Pretraining system on PASCAL VOC classification and detection respectively, it is remarkable that it can get so close without the luxury of millions of labeled pretraining data.

Multimodal deep learning

Unsupervised learning and self-supervised learning provide an adequate solution for deep unimodal feature extraction in the absence of labels. These features can produce meaningful representations of complex concepts, but without applying labels, deploying these systems would still need a semantic reference point to ground their application to analyst problems. One way is to gain context about targets and objects of interest by relating features from one modality to another modality where deep feature extraction has produced equally powerful representations. By creating a *joint vector embedding* of both modalities, we have a representation that can be queried from both modalities.

The creation of a joint vector embedding to unite two modalities is nontrivial. As previously mentioned, the challenges of word sense ambiguity and paucity of semantically labeled training data make it difficult to build high quality embeddings. To address these challenges, we jointly optimize both neural networks for image and word embeddings so that ambiguities in word senses can be moderated by visual cues and vice versa, and use negative sampling and noise contrastive estimation [16] to train on enormous amounts of weakly labeled data. Specifically, we use a traditional cross-entropy cost function and provide an analytical comparison to ranking cost functions [17, 18], to which we also apply the proposed sampling methods for fair comparison. In doing so, we train against the user-generated corpus currently available via open source: the YFCC100M corpus [1], and demonstrate that despite the issues, automated tagging using a sampled cost function can produce considerably more useful information than the original user-generated tags. More importantly, we allow users to search for relevant images using an almost unlimited vocabulary.

The proposed approach makes use of unnormalized cost functions from the natural language processing domain with optimization strategies rooted in unsupervised and embedding approaches. Most notably, Restricted Boltzmann Machines and word embeddings like [19] rely on some variant of noise contrastive estimation, where the distribution of foreground (e.g., the surrounding context of a word) is separated from the distribution of the background (e.g., the probability distribution over all words in the corpora). In our case, the context is the set of tags for each image, and negative samples can be obtained by sampling from the entire tag distribution absent the positive image tags. We represent this in the below cost function:

$$\mathcal{L}(\{W_i\}, \boldsymbol{v}_{p,n}) = \sum_{p}^{P} \log \mathbb{E}_p[\sigma(\boldsymbol{f}_{\{W_i\}}^{T} \boldsymbol{v}_p)] + \sum_{n}^{N} \log \mathbb{E}_n[\sigma(-\boldsymbol{f}_{\{W_i\}}^{T} \boldsymbol{v}_n)] \\ + \alpha \left(\sum_{p,p'} \sigma(\boldsymbol{v}_p^T \boldsymbol{v}_{p'}) + \sum_{p,n} \sigma(-\boldsymbol{v}_p^T \boldsymbol{v}_n) \right)$$

Here, the v_p are positively sampled vectors coming from words the image has been tagged with; v_n are the negatively sampled vectors from the probability distribution over all possible tags; P is the number of positive samples; N is the number of negative samples; $\mathbb{E}$is the expectation value; α is a tuning parameter; and the image feature vector f is parameterized by the set of weights W from a neural network.

To generate the vector space, the first term positively correlates the feature vector with the metadata tags, pulling the image closer to the context through backpropagation over W_i. The second term pushes them away from the background distribution. These two terms in the cost function promote the optimization of the image content. In the same manner, the final two terms serve to promote the analogous situation with the words themselves. We also optimize the word vectors (some of which are not in YFCC100M) with the New York Times corpus and Wikipedia. The entire optimization can be seen in a word2vec-like diagram in figure 4.

Positive and negative sampling is important because it allows us to efficiently train on large vocabulary sizes and massive amounts of training data. It lets us avoid a large matrix multiply when correlating images to words. Other practical issues stem from holding word-embedding matrices for large vocabulary sizes in memory. Because most of the word embedding work [19, 20] is done on multithreaded

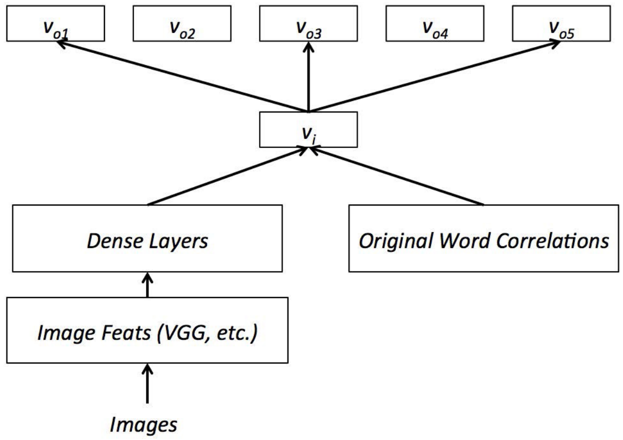


FIGURE 4. Diagram shows jointly optimizing word vectors (right) with image weights (left).

computer processing units (CPUs) due to its capacity, our architecture places the deep portion of the neural network on the GPU, and the wide portion (the input word vector list and the last matrix operation) is on the CPU.

While the primary objective is to train on user-generated content, we compute quantitative metrics on curated, traditional corpora to compare against the state of the art. These include the International Association of Pattern Recognition Technical Committee benchmark (IAPR TC-12) [21], the Extrasensory Perception game (ESP game) [22], and the YFCC100M data set with University of Oxford's Visual Geometry Group (VGG) neural network features.

To assess image-tagging capability, we train, validate, and test against proper splits from a single corpus. To assess generalization capability for a variety of content and word tags, we perform cross-corpus evaluation: train on a single data set, then test on a different data set. For example, Train IAPR TC-12 → Test ESP is trained and validated on the IAPR TC-12 training/validation splits and tested on the ESP game test split. For YFCC100M, we only test on the ESP game because the Train YFCC100M → Test YFCC100M evaluation is not meaningful due to noisy truth data. In table 3, we compare our approach with the state-of-the-art Fast0Tag image-tagging system. Our approach effectively uses the large amounts of weakly labeled training data in YFCC100M, significantly improving performance over the system trained on a smaller amount of well-labeled IAPR TC-12 training data.

However, more important is the real-world demonstration with image retrieval on the YFCC100M data using our bimodal text+image feature learning system. Using a query system that takes less than a second to return from a search of over millions of images, we can accurately retrieve relevant images relating to a search query. Figure 5 (on the following page) shows search results for three different queries from a subset of YFCC100M consisting over 6.7 million images.

Conclusion

We have described a deep learning framework for mapping data of disparate modalities into a joint multimodal feature space in which conceptually related data are proximal. This feature space enables generalized multimodal retrieval, such as "find other images like this image," "find textual descriptions related to this image," "find videos related to these sentences," etc. The multimodal feature space can help analysts quickly find relevant data even if the data are not tagged with keywords. We developed a set of learning algorithms that allows our DNN models to be well trained while minimizing the dependence on massive amounts of high-quality, human-labeled training data. Our self-supervised learning method is a promising approach for learning unimodal feature

TABLE 3. A comparison of image-tagging performance. Each system is first trained on either the IAPR TC-12 (which has 291 unique tags, ~20,000 images) or YFCC100M (which has 432,000 unique tags, ~95,000,000 images) tagged training set and then tested on the ESP game (which has 268 unique tags, ~20,000 images) data set. Compared with a leading zero-shot image-tagging algorithm, our approach for learning a joint image+text feature space greatly improves performance when going from a small well-labeled training set (IAPR TC-12) to a much larger weakly labeled training set (YFCC100M).

Method	Train IAPR TC-12 → Test ESP			Train YFCC100M → Test ESP		
	Precision (%)	Recall (%)	F1	Precision (%)	Recall (%)	F1
Sampled Fast0Tag [19]	5.9	8.3	6.9	5.1	3.9	4.4
Our Approach	13.3	10.2	12.1	21.9	15.1	17.9

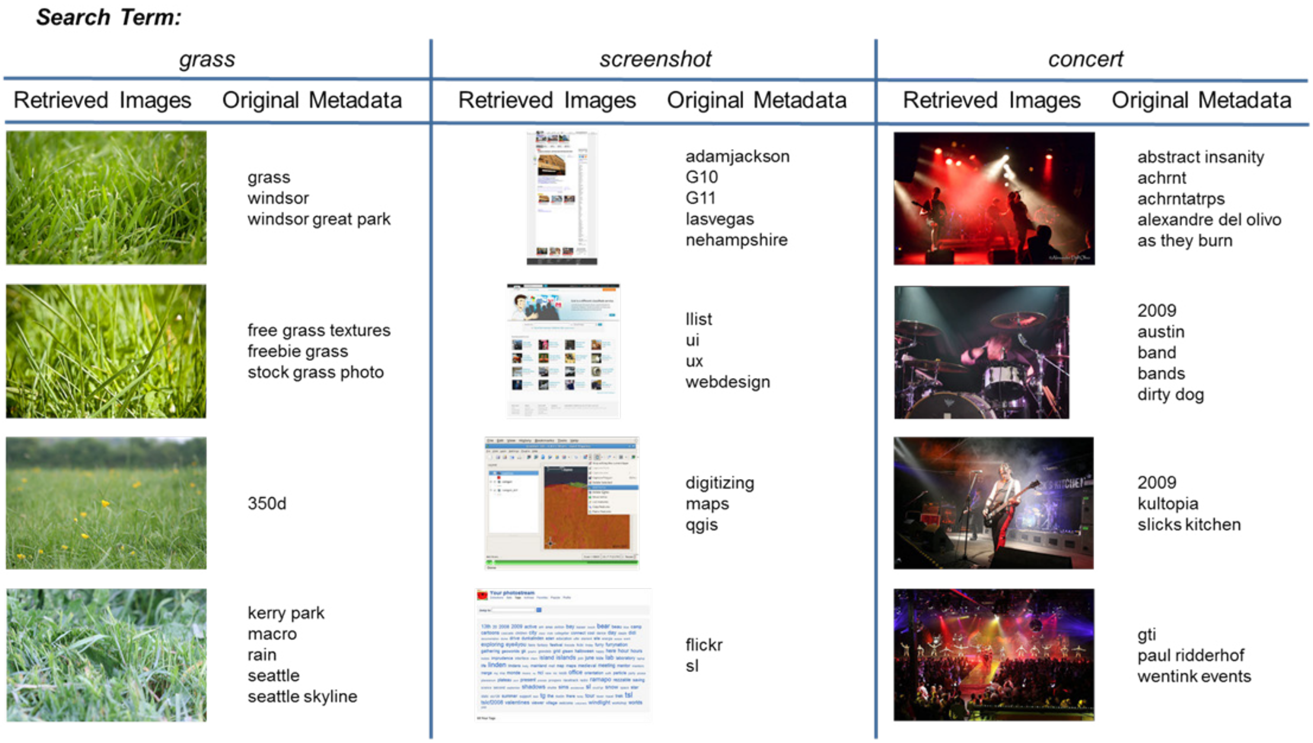


FIGURE 5. A sampling of image retrieval results on the YFCC100M data using our bimodal text+image feature learning system. Each pair of columns shows the top four search results for a single search term returned by our bimodal DNN system. To demonstrate how much our system relies on the bimodal features for retrieval (and not on keywords), we show the retrieved images along with their corresponding metadata tags. Photo credit detailed in [23].

representations, while our joint multimodal feature learning algorithm is excellent for learning from large amounts of weakly labeled data. Together, these approaches make it possible to train DNNs that map data into a joint multimodal feature space and helps us to take a step toward making sense of unlabeled data.

References

[1] Thomee B, Shamma DA, Friedland G, Elizalde B, Ni K, Poland D, Borth D, Li LJ. "YFCC100M: The new data and new challenges in multimedia research." *Communications of the ACM*. 2016;59(2):64–73. Available at: http://cacm.acm.org/magazines/2016/2/197425-yfcc100m/fulltext.

[2] Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma Sean, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg AC, Fei-Fei L. "ImageNet large scale visual recognition challenge." 2014. Cornell University Library. arXiv:1409.0575.

[3] Hu J, Shen L, Sun G, "Squeeze-and-excitation networks." 2017. Cornell University Library. arXiv:1709.01507.

[4] He K, Zhang X, Ren S, Sun J. "Deep residual learning for image recognition." 2015. Cornell University Library. arXiv:1512.03385.

[5] Godfrey J, Holliman E, McDaniel J. "Switchboard: Telephone speech corpus for research and development." In: *IEEE International Conference on Acoustics, Speech, and Signal Processing*; 1992 Mar 23–26; San Francisco, CA. doi: 10.1109/ICASSP.1992.225858.

[6] Xiong W, Wu L, Alleva F, Droppo J, Huang X, Stolcke A. "The Microsoft 2017 conversational speech recognition system," 2017. MSR-TR-2017-039.

[7] Sutskever I, Martens J, Dahl G, Hinton G. "On the importance of initialization and momentum in deep learning." *Proceedings of Machine Learning Research.* 2013;28(3):1139–1147. Available at: http://proceedings.Mlr.press/v28/sutskever13.html.

Return to CONTENTS

[8] Duchi J, Hazan E, Singer Y. "Adaptive sub gradient methods for online learning and stochastic optimization." *Journal of Machine Learning Research*. 2011;12:2121–2159.

[9] Kingma DP, Ba J, "ADAM: A method for stochastic optimization." 2015. Cornell University Library. arXiv:1412.6980.

[10] Nair V, Hinton G. "Rectified linear units improve restricted boltzmann machines." In: *Proceedings of the 2010 27th International Conference on Machine Learning*; 2010 Jun 21–24; Haifa, Israel: pp. 807–814.

[11] Baldi P. "Autoencoders, unsupervised learning, and deep architectures." *Journal of Machine Learning Research: Workshop and Conference Proceedings*. 2012;27:37–50.

[12] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y. "Generative adversarial nets." 2014. Cornell University Library. arXiv:14062661.

[13] Doersch C, Gupta A, Efros A. "Unsupervised visual representation learning by context prediction." In: *2015 IEEE International Conference on Computer Vision (ICCV)*; 2015 Dec 7–13; Santiago, Chile: pp. 1422–1430. doi: 10.1109/ICCV.2015.167.

[14] Zhang R, Isola P, Efros A. "Split-brain autoencoders: Unsupervised learning by cross-channel prediction." In: *Proceedings of the Conference on Computer Vision and Pattern Recognition*; 2017 Jul 21-26; Honolulu, HI. doi:10.1109/CVPR.2017.76.

[15] Everingham M, Van Gool L, Williams CKI, Winn J, Zisserman A. "The PASCAL visual object classes challenge 2007 (VOC2007) results."

[16] Gutmann MU. "Noise contrastive estimation of unnormalized statistical models, with applications to natural image statistics." *Journal of Machine Learning Research*. 2012;13(1)307–361.

[17] Zhang Y, Gong B, Shah M. "Fast zero-shot image tagging." In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*; 2016 Jun 27–30; Las Vegas, NV. doi: 10.1109/CPVR.2016.644.

[18] Burges C, Shaked T, Renshaw E, Lazier A, Deeds M, Hamilton N, Hullender G. "Learning to rank using gradient descent." In: *Proceedings of the 22nd International Conference on Machine Learning*; 2005 Aug 7–11; Bonn, Germany. doi: 10.1145/1102351.1102363.

[19] Mikolov T, Sutskever I, Chen K, Corrado G, Dean J "Distributed representations of words and phrases and their compositionality." In: *Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS '13);* 2013 Dec 5–10; Lake Tahoe, NV: pp. 3111–3119.

[20] Pennington J, Socher R, Manning C. "Glove: Global vectors for word representation." In: *Proceedings on Empirical Methods in Natural Language Processing (EMNLP)*; 2014; Doha, Qatar: pp. 1532–1543.

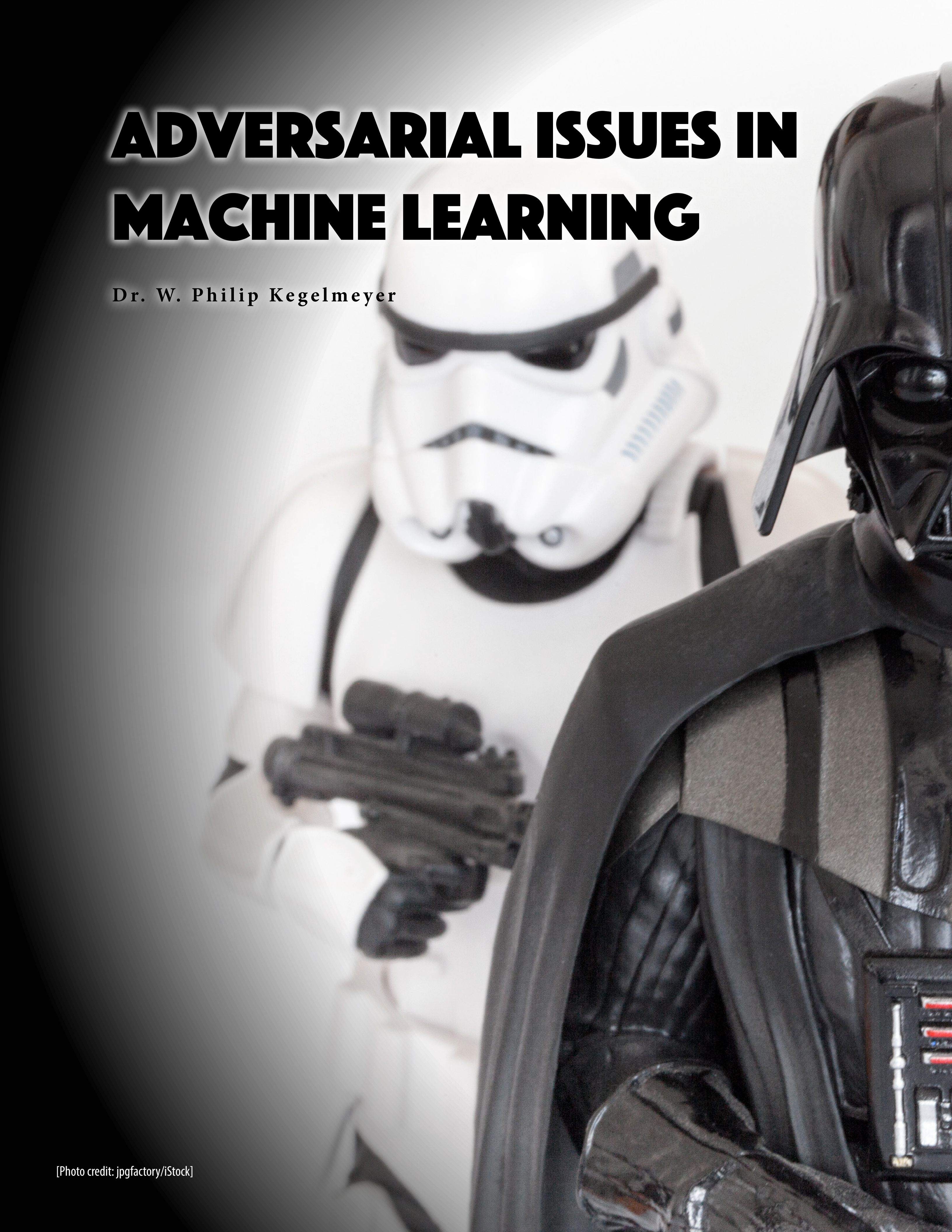
[21] Grubinger M, Clough P, Muller H, Deselaers T. "The IAPR TC-12 benchmark: A new evaluation resource for visual information systems." In: *Proceedings of the International Workshop OntoImage 2006 Language Resources for Content-Based Image Retrieval, held in conjunction with LREC 2006,* Genoa, Italy. pp. 13–23.

[22] Robertson S, Vojnovic M, Weber I. "Rethinking the ESP game." In: *Proceedings of the 27th International Conference on Human Factors in Computing Systems Extended Abstracts on Human Factors in Computing Systems*; 2009 Apr 4–9; Boston, MA. doi: 10.1145/1520340.1520597.

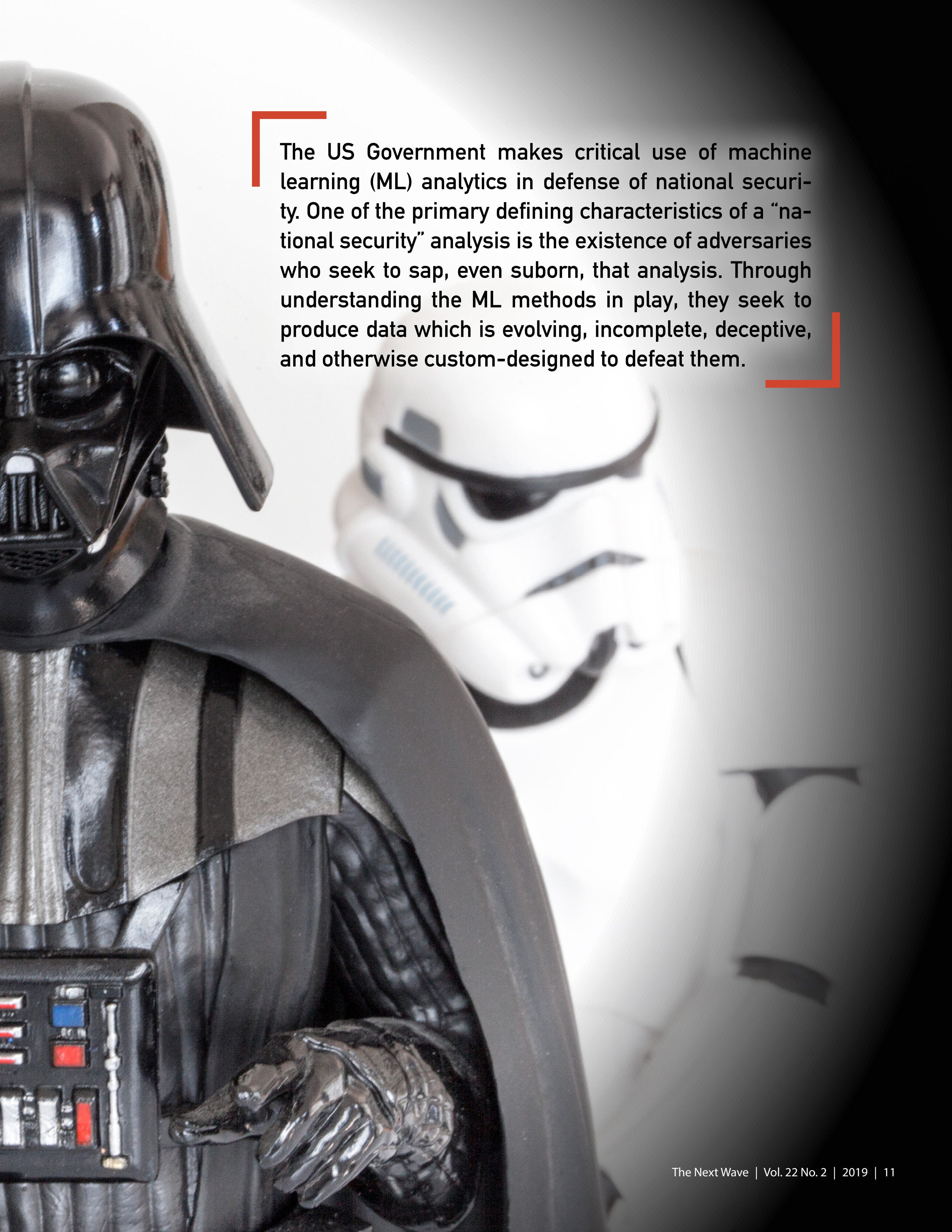
[23] Photos from YFCC100M database. Photographers for individual images include: Adam Dimmick, Andrew Taylor, Pete Warner, Colby Perry, Adam Chandler86, Eston Bond, esambale, Daniel Voyager, Alexandre Dell'Olivo, MarkScottAustinTX, labormaus_69/Andrea Jaeckel-Dobschat, Paul and Menno Ridderhof.

ADVERSARIAL ISSUES IN MACHINE LEARNING

Dr. W. Philip Kegelmeyer



[Photo credit: jpgfactory/iStock]



The US Government makes critical use of machine learning (ML) analytics in defense of national security. One of the primary defining characteristics of a “national security” analysis is the existence of adversaries who seek to sap, even suborn, that analysis. Through understanding the ML methods in play, they seek to produce data which is evolving, incomplete, deceptive, and otherwise custom-designed to defeat them.

This cannot be easily prevented. Recent work [1, 2] has shown that if an ML model is publicly deployed, it can itself be easily modeled, even duplicated, and then studied in private to discover its weaknesses. Even a privately held model might be sufficiently well deduced through reverse engineering or network compromise. And once a model is understood, there are typically many avenues of attack, as the training data, test data, or both are generally uncontrolled and can be modified by an adversary.

Adversarial ML addresses these issues, spanning developing attacks against ML, assessing defenses, detecting whether an attack is in progress, quantitative assessment of worst-case scenarios, considerations around if, when, and how to deploy an ML model, and so on. Adversarial ML tradecraft is essentially applying vulnerability assessment methods at the algorithm level, rather than to software or hardware. The end goal is to harden the ML methods in use, and in any case, to regard their outputs with an informed, wary eye. That is, to become the top middle sheep in figure 1—the one that doesn't quite buy into the "IF (white AND fuzzy) THEN <Harmless>" analytic.

A taxonomy of adversary goals

Though adversarial aspects of ML have been discussed for more than a decade [3], there is no broadly adopted consensus as to how to categorize an adversary's goals. One possibility is to think in terms of quality, confidence, or evasion attacks.

- **Quality attack:** The adversary's goal is to drive down the overall effectiveness of ML as assessed on the training data, regardless of whether test performance is unaffected. The idea might be to convince the defender not to deploy an actually useful analytic or to cause the defender to waste time attempting to improve it.
- **Confidence attack:** The adversary's goal is to drive down the overall effectiveness of ML as assessed on the test data, without necessarily affecting accuracy on the training data. The idea here is to convince the defender to confidently deploy an ineffective analytic.
- **Evasion attack:** The adversary's goal is to engineer a specific desired outcome for a specific future test sample or samples. Thus, the idea is to appropriately shape a specific part of the ML decision surface, or to understand the existing decision surfaces well enough to be able to move evasively within them.

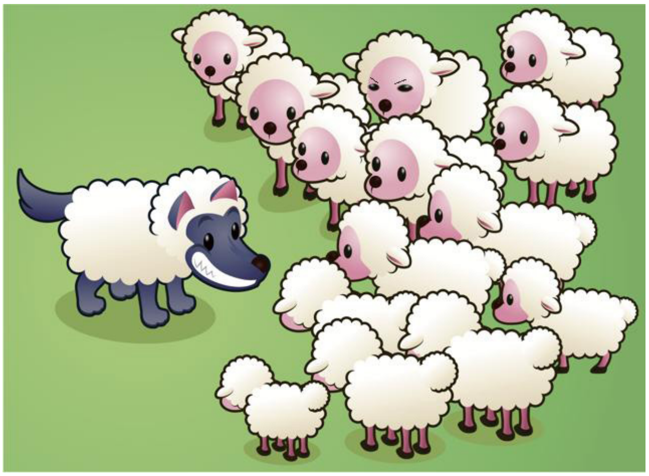


FIGURE 1. IF (white AND fuzzy) THEN <Harmless>

What makes machine learning vulnerable?

What might make an ML algorithm vulnerable to such attacks? Classic supervised ML methods depend on two fundamental assumptions; violating either of them creates exploitable weaknesses.

The first assumption is that the test data is essentially similar to the training data. It has long been well-understood that this is often an unreliable expectation. For instance, data often changes slowly and naturally over time. Therefore, much research has been focused on building ML models that can gracefully handle dissimilar test data; examples are methods for handling concept drift [4] or for using transfer learning [5] to explicitly extend an ML model beyond its original training data.

This test set similarity assumption is also the basis for most of the currently popular attacks against deep learning on image data [6]. Deep learning methods typically overfit their training data, generating ML models which are indeed very accurate if the test data is similar to the training data, but which are easily led astray by minutely altered test data. This vulnerability has created its own subfield, Generative Adversarial Networks [7], in which one ML model is explicitly trained to generate images designed to fool a competing ML model, which is in turn trying to learn how not to be fooled.

The second assumption, perhaps less well appreciated, is that the "ground-truth" labels in the training data used to build the model are accurate. Undermining this assumption by tampering with the labels exposes particularly pernicious algorithmic vulnerabilities.

An example label-tampering vulnerability

As one example, consider ensembles of bagged decision trees [8] as the ML method. For ML in general, it is standard to assume that self-assessment on the training data via cross-validation is a useful, if mildly optimistic, estimate of accuracy on an eventual test set. Further, for ensemble methods in particular, it is standard to assume that ensemble accuracy will be higher than the average accuracy of the individual trees. These assumptions have been correct so consistently that they are rarely examined.

With that in mind, consider figure 2, which depicts the results of tampering with ground-truth labels. The underlying data is a product inspection data set with roughly balanced "Pass/Fail" labels. The curves indicate what happens to three measures of accuracy (on the y-axis) as we flip a certain number of the ground-truth labels (the x-axis) before we build the ensemble model. Here the adversary is choosing which labels to flip in a purely random fashion.

In that plot, the red curve is the test set accuracy, the accuracy we care about. Happily, nearly 40% of the training samples must be corrupted before there is a noted drop in accuracy, which is a reassuring testament to the robustness of ensemble decision tree methods. Also, at first, the ensemble accuracy (in red) outperforms the average single tree accuracy (in black), as we would hope.

The blue curve depicts the cross-validated training set accuracy. That curve does decrease nearly linearly with the amount of tampering (until a full half of the data is flipped) and thereby illustrates a mild example of a *quality attack*. That is, a defender who looked only at training set accuracy might incorrectly conclude that the test set accuracy would not be high enough to be useful.

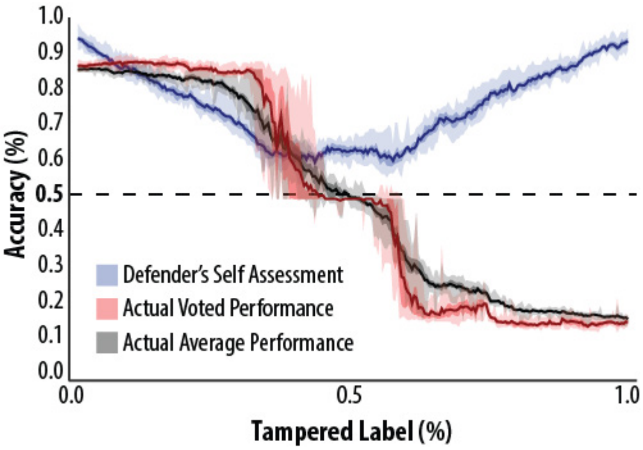


FIGURE 2. A random label-flipping attack is effective but obvious.

Unfortunately, figure 2 is a best-case scenario of a particularly lazy adversarial attack. Now consider figure 3 (on the following page), which illustrates an effective *confidence attack*. Here the adversary has been slightly smarter and has clustered all of the training data, randomly ordered the clusters, and then randomly attacked all members of a cluster before going on to the next one. This small change dramatically improves matters for the adversary. Now the test set ensemble accuracy (in red) decreases nearly linearly with the amount of tampering. It is also essentially no better than the average tree accuracy (in black), which means the extra computation required by ensembles is accomplishing nothing.

Most worrisome, however, is the fact that the training set accuracy (in blue) stays relatively flat regardless of the degree of tampering. This means, for instance, that if the adversary can tamper with 20% of the training points, the actual real-world accuracy will decrease to about 75%, but the defender won't know this! They'll expect the accuracy to be around 90%, because that's what the training set validation indicates.

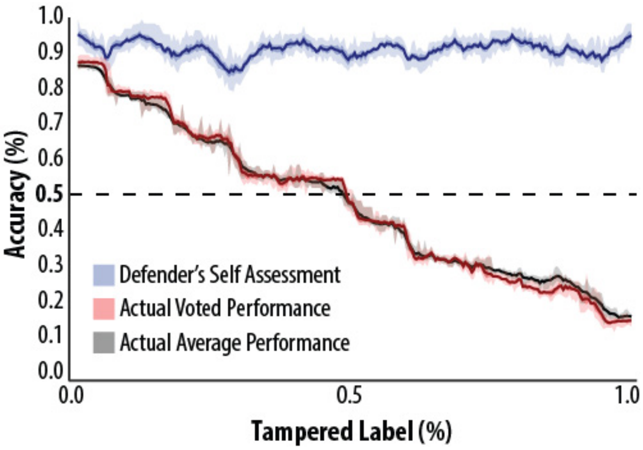


FIGURE 3. A slightly smarter label-flipping attack is effective but not obvious.

Conclusion

Adversarial ML is a new and rapidly developing field, and so this article was able only to introduce some of its ideas, along with a single example of an unnervingly effective attack.

Still, we can't stop using these methods, so perhaps we can learn to consider them with a useful sense of watchful paranoia. G.K. Chesterton famously said "We must learn to love life without ever quite trusting it" [9]; that seems the right perspective to take with ML as well.

Acknowledgments

This work was supported by the Laboratory Directed Research and Development program at Sandia National Laboratories, a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the US Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

References

[1] Tramèr F, Zhang F, Juels A, Reiter MK, Ristenpart T. "Stealing machine learning models via prediction APIs." In: *Proceedings of the 25th USENIX Security Symposium;* 2016 Aug 10-12; Austin, TX: pp. 601–618. arXiv:1609.02943.

[2] Papernot N, McDaniel P, Goodfellow I, Jha S, Celik ZB, Swami A. "Practical black-box attacks against machine learning." In: *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*; Abu Dhabi, United Arab Emirates: pp. 506–519. doi: 10.1145/3052973.

[3] Globerson A, Roweis S. "Nightmare at test time: robust learning by feature deletion." In ICML 2006: *Proceedings of the 23rd international conference on Machine learning*; 2006 Jun 25–29; New York, NY: pp. 353–360. doi: 10.1145/1143844.1143889.

[4] Webb GI, Hyde R, Cao H, Nguyen HL, Petitjean F. "Characterizing concept drift." *Data Mining and Knowledge Discovery*. 2016; 30(4):964–994. doi: 10.1007/s10618-015-0448-4.

[5] Pan SJ, Yang Q. "A survey on transfer learning." *IEEE Transactions on Knowledge and Data Engineering*; 2010; 22(10):1345–1359. doi: 10.1109/TKDE.2009.191.

[6] Evtimov I, Eykholt K, Fernandes E, Kohno Ti, Li B, Prakash A, Rahmati A, Song D. "Robust physical-world attacks on machine learning models." 2017. Cornell University Library. arXiv:1707.08945.

[7] Goodfellow IJ. "NIPS 2016 tutorial: generative adversarial networks." 2017. Cornell University Library. arXiv:1701.00160.

[8] Banfield RE, Hall LO, Bowyer KW, Kegelmeyer P. "A comparison of decision tree ensemble creation techniques." *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2007; 29(1):173–180. doi: 10.1109/TPAMI.2007.2.

[9] Chesterton GK. *The Man Who Was Thursday: A Nightmare.* Jovian Press; 1908.

Return to CONTENTS

ADVANCES IN recommender-system theory

Mark T. Jacobson

Reconciling "now" and "then" in data science

Modern data science contends with big data sets in which *signal* (helpful, illuminating) is contaminated by *noise* (annoying, inexplicable).

Historically, the problem of how to deal with data in the presence of noise—how to infer, understand, predict—has been the province of statistics, with mathematical statisticians developing model-based theory to blaze the trails. *What sorts of things go wrong?* The theoretician forgets that the model was a first approximation and falls in love with it; the front-line statistician embraces a technique but disregards the premises underlying it. Or the real-life data are complicated in ways that preclude robust handling by idealized, textbook techniques.

[Photo credit: mphillips007/iStock]

Now, in the modern era, we have massive computational power, hordes of experts capable of harnessing it, and mechanisms for producing massive amounts of data that can be fed into it. *What sorts of things go wrong?* Machine learning algorithms can be impressive—but inscrutable. The computational expert has no training to distinguish noise from signal, but data *will* be analyzed—with consequences. Our stockholders may see profits—but no one knows whether we could have done better. And understanding theory seems like a waste when we know that the models are simplistic and our data don't meet their assumptions…

All models are wrong,
so let's not bother with any of them.

What? No! The famous aphorism is

All models are wrong, but some are useful [1]**.**

So what might "useful" mean in the modern era of big data and ad hoc algorithms? As NSA researchers, we offer our experience with *recommender systems* (RSs). Seeing opportunities for their beneficial use at NSA, we examined published RS algorithms. We noticed the following pattern:

1. We'd see some component of the algorithm—offered without motivation or justification—and realize that it would resemble the "right answer" to the problem if we introduced a particular model.
2. Guided by this model, we tackled the problem and gained new insights about the problem that led to potentially advantageous modifications to the algorithm.
3. Cognizant that "all models are wrong," we approached experts with our ideas, expecting to hear practical reasons for why they hadn't worked or wouldn't work. Instead, we were encouraged by reactions like "we never thought of that."

This sort of model-based discipline encourages clear, coherent thinking—even if some of the details won't survive the eventual reality check. Furthermore, recommender applications have myriad variations; the coherent, idealistic treatment of basic flavors can lead to insight when it's time to confront trickier, more nuanced situations. In the sections ahead, we'll describe a substantial NSA advance in recommender-system theory, and offer ideas we're currently exploring.

⟵ items ⟶

⟵ users ⟶	6			4	4	7	4			
		1	4		4	3				
							4		4	
		4	5		4			5		5
	7			5	4				5	5
					3			6		
						5	5		6	
	9	8				8		8		

FIGURE 1. Example of a partial array of ratings (here, 1-9) that users assign to items they've purchased.

Collaborative filtering via latent factors

Suppose you run an online marketplace, and you get to see feedback—numerical ratings—from your users about (some of the) items they buy. If you knew what feedback would be provided for the unrated user-item pairs, you could make savvy recommendations for future purchases. Representing ratings as entries in a partial array (see figure 1), we could imagine trying to estimate what we'd see in the empty cells (which typically are massively prevalent). What we're seeking here is an RS that takes the information we have about users, items, and their interactions, and helps make predictions related to further interactions. Note the crucial two-dimensional nature of this problem: Feedback depends on item *and* user (two *aspects*).

Adopting flexible notions of "user" and "item" allows us to cast many problems this way: online media providers try to match subscribers with content; employment agencies try to match employers with job seekers; dating services try to match clients with each other; a baseball manager might seek a batter expected to perform well against a particular opposing pitcher (or vice versa). NSA has its own range of applications that could benefit from RSs.

The most basic flavor of recommender problem is *collaborative filtering* (CF). Here you're given only a partial feedback array, as in figure 1, without any further information about the users and items. The foremost example of CF was provided by the *Netflix challenge* [2], in which the internet movie-rental company sought a better system for recommending movies to its users. It published a large *training set* of users' ratings of movies (1–5 "stars"), solicited algorithms, and offered a $1 million prize for the first to outperform its own proprietary RS by a specified margin on a *test set* of user movie ratings withheld from publication. The contest spawned myriad algorithms and literature, but careful thought was often an unaffordable luxury amid the marketplace scramble for the prize. There *was* crowdsourcing and aggregation: The eventual winner was a mash-up of more than 800 component techniques, as competing entrants formed teams and threw their ideas together.

Our focus is a core technique [3] that uses *latent factors* (LF). Let us index the users by $i = 1, 2, \ldots m$ and the items by $j = 1, 2, \ldots n$, and denote a rating in the array by $r_{i,j}$. The LF approach explains/predicts the rating as

$$r_{i,j} = \mu + \alpha_i + \beta_j + \mathbf{p}_i'\mathbf{q}_j + \text{noise}, \quad (1)$$

representing it in terms of unknown quantities to be estimated from the training data:

- a "baseline" μ, adjusted additively by
- "main effects" α_i and β_j attributed to the distinctive natures of user i and item j;
- for some postulated number f of *factors,* a dot-product of two f-dimensional *factor vectors:* $\mathbf{p}_i$ for the user, $\mathbf{q}_j$ for the item (prime (′) denoting matrix transpose);
- noise that admits our inability to tell a perfect story about the ratings.

That is, for each user, there is a parameter pair $(\alpha_i, \mathbf{p}_i)$; for each item, there is a pair $(\beta_j, \mathbf{q}_j)$.

Understanding LFs as dimension reduction. Statisticians are used to a two-way *analysis of variance* (ANOVA) model that looks like (1), except with the $\mathbf{p}_i'\mathbf{q}_j$ term replaced by a less restrictive $\gamma_{i,j}$, which represents an interaction effect between user i and item j. (Underneath the noise, the ANOVA model allows each $r_{i,j}$ to have an unconstrained mean.) In CF, we hope for substantial interactions that we can proficiently estimate, leading us to accuracy (and profits) beyond what we'd obtain using only main effects. There's a problem with free γ's, though: For the numerous (i, j)'s that are missing ratings, we have no data to allow us to estimate the cell means; in fact, a typical ANOVA scenario expects multiple observations per cell in order to produce reasonable estimates. But by resorting to factor vectors, we're doing dimension reduction on the interaction effects: The parameter count falls from (effectively) mn to $O((m + n)f)$; the hope is that there will be an f that's large enough to explain the ratings but small enough to allow reasonable estimation of the parameters.

We can think of this in matrix terms: What would have been an $m{\times}n$ matrix of γ's (with rank min(m, n)) turns into a rank-f product of an $m{\times}f$ matrix (the $\mathbf{p}$'s) and an $f{\times}n$ matrix (the $\mathbf{q}$'s). Indeed, if we had the γ-matrix, we could obtain a "best" rank-f approximation using the *singular value decomposition* (SVD). Interestingly, the recommender literature commonly refers to LF approaches as "matrix factorization" or "SVD" algorithms, even though we have no actual *matrix* we can factor (the closest is the partial array of ratings, which falls far short).

The LF approach becomes more intuitive if we imagine that each of the f factors captures a salient "quality" underlying the rating process. In the Netflix example, imagine that such a factor of a movie is "number of Academy Award-winning actors in cast." That is, this number appears as some component $q_{j,k}$ in

$$\mathbf{p}_i'\mathbf{q}_j = \sum_{k=1}^{f} p_{i,k}q_{j,k},$$

and there's a corresponding user factor $p_{i,k}$ that tells us how much user i values Academy Award (AA) winners—think of the contribution as

(AA winners in j) × (stars awarded by i per
AA winner)

(though such bilinearity might be unreasonable; we're also apparently allowing arbitrary real values for stars). The $(f - 1)$ other factors contribute similarly; of course, before fine-tuning for factors, we have main effects: α_i (how liberal is user i with stars?) and β_j (how generally well-liked is movie j?). Now, if indeed we *were* told all the components of $\mathbf{q}$'s (and the β's), then consideration of (1) reveals that, for each user i, we could estimate $(\mu + \alpha_i, \mathbf{p}_i)$ as a *linear regression* exercise (over the items j for which i has provided ratings).

In CF-by-LF, underappreciated symmetries are cause for concern. Alas, no one hands us either the $\mathbf{p}$'s or the $\mathbf{q}$'s; in CF, we must find both. No one's even telling us what their dimension f should be . . . or how we might qualitatively understand any of their components; the goal, simply but vaguely, is to find parameters that perform well. Netflix used a typical figure of (de)merit to evaluate RS algorithm performance: the sum of squared errors between predicted and actual ratings over a test set; for a LF scheme, this would be

$$\sum_{(i,j)}\{r_{i,j} - (\mu + \alpha_i + \beta_j + \mathbf{p}_i'\mathbf{q}_j)\}^2 \tag{2}$$

where the sum is over pairs (i, j) included in the test set. The literature overlooks an important issue: The predictions (1) and squared-error sum (2) are invariant under arbitrary linear transformations of the **p**'s, provided the transformation is inverted on the **q**'s. That is, pick any invertible $f \times f$ matrix **A**; the results won't change in (1) or (2) if we replace each $\mathbf{p}_i$ by $(\mathbf{A}'\mathbf{p}_i)$ and each $\mathbf{q}_j$ by $(\mathbf{A}^{-1}\mathbf{q}_j)$. The group of such transformations effectively partitions our possible solutions into *equivalence classes,* each of which comprises equivalently performing alternatives. In particular, a "best" solution has infinitely many equally good counterparts! Ignored, this symmetry can be hazardous conceptually and computationally; properly understood and dealt with, it can be a blessing. We'll see both sides of this.

Squared-error minimization is maximum-likelihood estimation with a Gaussian backstory. We can't minimize (2) since the sum is over test ratings that we don't get to see, but if we sum instead over the *training* set, finding parameters to minimize *that* function might be a reasonable objective. In fact, if the noise component in (1) is modeled as Gaussian—and independent and identically distributed (IID) across (i, j)—the training-set version of (2) is a scaled version of the (negated) log-likelihood function of the parameters; minimizing it is equivalent to finding *maximum-likelihood estimates* (MLE) for the parameters. RS literature (e.g., [4]) discusses techniques for doing so (e.g., *gradient descent, alternating least squares*), but observes that solution performance is poor on test sets (even for moderate f).

Shrinkage to combat overfitting. Each training set yields parameter estimates (which form a prediction function); if parameters are too numerous, the estimates will be unstable across different (but comparably representative) training sets, any particular one of which is liable to produce bad estimates (which will generalize poorly when making predictions beyond the training set). This single problem has two names/perspectives: *overfitting*—too many parameters to be precisely estimated by the available quantity of training data; *variance*—too little training data to support precise estimation of the specified parameter collection. (We saw the problem in extreme form when we considered the unrestricted ANOVA model.) Variance will decrease if we increase the training-set size—but typically we can't.

How can we improve the situation? RS efforts frequently turn to *shrinkage* (or *regularization*): The objective function [the training-set version of (2)] is augmented by a weighted term that penalizes a putative solution for its distance from a certain fixed point (in parameter space), resulting in an objective function like

$$\sum_{(i,j)}\{r_{i,j} - (\mu + \alpha_i + \beta_j + \mathbf{p}_i'\mathbf{q}_j)\}^2 + \lambda\left\{\sum_{i=1}^{m}(\alpha_i^2 + \mathbf{p}_i'\mathbf{p}_i) + \sum_{j=1}^{n}(\beta_j^2 + \mathbf{q}_j'\mathbf{q}_j)\right\} \tag{3}$$

in which each main effect and factor component pays a penalty—determined by yet another parameter, λ—for its squared distance from 0. The effect is to shrink the parameter estimates toward 0, decreasing variance: If we take $\lambda = 0$, we get the unaltered MLEs, but as we increase λ, the estimates progressively discount the training data (and their vagaries), ultimately all approaching 0. The downside is that penalization increases *bias*—in this case, the tendency to redirect a parameter estimate toward an arbitrary, fixed choice (0), away from an optimal (but unknown) alternative. There are various strategies for navigating this *bias-variance trade-off* (e.g., choosing λ by *cross-validation* involving held-out portions of the training set; see, e.g., [5]).

Shrinkage understood through the Gaussian backstory: Random effects. Suppose we regard our users as samples from some larger (maybe infinite) population of users; specifically, we're interested in their $(\alpha_i, \mathbf{p}_i)$-pairs as samples from a population. Similarly, regard the items' $(\beta_j, \mathbf{q}_j)$-pairs as samples dispensed from some (other) item population. (See figure 2.) These still influence the $r_{i,j}$'s, per (1), but we've introduced a preliminary "tier" of randomness [prior to the random noise that arrives in (1)]. What we formerly regarded as parameters—to be *estimated* from the training data—are now (unobservable) random variables to be *predicted* conditional on observed instances of related random variables (the ratings in the training data).

Return to CONTENTS

In ANOVA, such parameters-now-random-variables are called *random effects;* their postulated distributions amount (in a Bayesian context) to *priors,* but don't disturb frequentists who eschew subjective notions of probability. The priors may themselves involve parameters (called, in this context, *hyperparameters*).

This route leads to insight about the penalty term in (3), which is another (scaled, negated) Gaussian log-likelihood—consistent with IID mean-0 Gaussian priors for all of the components of the α's, β's, $\mathbf{p}$'s, and $\mathbf{q}$'s. Since the initial term relates to the *conditional* distribution of the ratings given the random effects, (3) is now the negative log of the joint probability density of everything random—the observable ratings and the unobservable random effects—with (hyper) parameters μ, λ, and the variance of the noise [in (1)]. From this joint density, *Bayes' Rule* leads us (in theory) to the *posterior* distribution of the random effects, conditional on the observed ratings; this can be thought of as a savvy updating of the prior (in light of pertinent data), and would be the appropriate vehicle for inference about main effects and factor vectors. By minimizing (3), we produce their *maximum a posteriori* (MAP) estimates (collectively, the *mode* of their posterior distribution). We note a standard phenomenon: The center of the joint random-effects prior serves as the "magnet" (here, all zeros) toward which the MLE gets pulled; the force it exerts depends on the variances (prior vs. noise) and the training-set size [the number of terms in the first sum in (3)].

In the literature, there has been hand-wringing over the wisdom of various refinements to (3) (e.g., penalizing the user components differently from the item components). Salakhutdinov and Mnih have proposed an appealing, expansive approach [6], which translates to our story thusly: Each user's random effects are modeled by a multivariate Gaussian distribution having a completely general covariance matrix as a hyperparameter (allowing for correlation between factors); each item's effects are modeled by another (with its own covariance matrix). The authors then proceed along a hierarchical, fully Bayesian route, postulating diffuse priors for the hyperparameters as a further, earlier tier. Random-effect and ratings predictions follow from consideration of the appropriate posteriors: Though intractable, the distributions can be sampled practically using *Markov chain Monte Carlo* (MCMC) methods, with (say) a long-term average of samples serving as an estimate of the sought-after posterior *mean.*

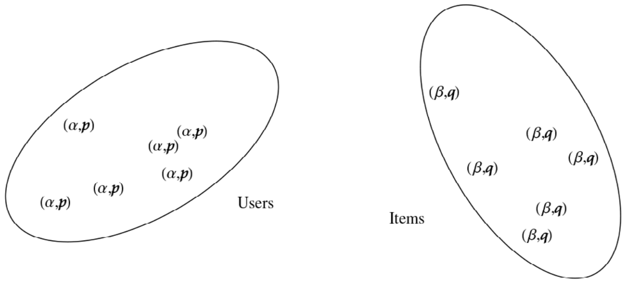


FIGURE 2. Each user's parameters are regarded as a random draw from a population comprising all users; similarly, item parameters are regarded as emissions from an item population.

Pursuing the Gaussian premise leads to wonderful revelations. The multivariate Gaussian (or *normal*) distribution has two parameters: its mean vector and covariance matrix. These appear respectively in our notation for the distribution: We write, for example, $\mathbf{p}_i \sim \mathrm{N}(\boldsymbol{\theta}, \boldsymbol{\Sigma})$ to indicate that $\mathbf{p}_i$ has a particular Gaussian distribution. A convenience of the Gaussian is its closure under affine transformations: For such $\mathbf{p}_i$, we have $\mathbf{A}\mathbf{p}_i + \mathbf{b} \sim \mathrm{N}(\mathbf{A}\boldsymbol{\theta} + \mathbf{b}, \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}')$ for conformable (nonrandom) $\mathbf{A}$ and $\mathbf{b}$. Now, once we've bought the multivariate-Gaussian premise for our random effects, we can shift the user- and item-factor scales to be centered on 0 by absorbing the consequences of the shifts into the main effects; we can further center the main-effect scales on 0 by absorbing the necessary shifts into the baseline term μ; that is, we lose no generality by taking all of our random-effect means to be zero. We therefore postulate that

user vectors $\mathbf{p}_i$ are IID draws from $\mathrm{N}(\mathbf{0}_f, \boldsymbol{\Sigma})$; similarly, itemwise, $\mathbf{q}_j \sim \mathrm{N}(\mathbf{0}_f, \boldsymbol{\Phi})$

for some (unknown) $f \times f$ covariance-matrix hyperparameters $\boldsymbol{\Sigma}$ and $\boldsymbol{\Phi}$. Now, consider:

- our ratings model uses the $\mathbf{p}_i$'s and $\mathbf{q}_j$'s only in the $\mathbf{p}_i'\mathbf{q}_j$ terms;
- the $\mathbf{p}_i$'s and $\mathbf{q}_j$'s are latent constructs, actual observations of which we'll never be blessed (or stuck) with—so we're free to transform them, even in ways that involve the unknown hyperparameters;

- $\mathbf{\Sigma}$ and $\mathbf{\Phi}$, being symmetric positive definite (SPD) matrices, have (SPD) square roots.

We can thus write

$$\mathbf{p}_i'\mathbf{q}_j = (\mathbf{\Sigma}^{-1/2}\mathbf{p}_i)' \, (\mathbf{\Sigma}^{1/2}\mathbf{\Phi}^{1/2}) \, (\mathbf{\Phi}^{-1/2}\mathbf{q}_j),$$

in which transformed factor vectors now are both distributed as $\mathrm{N}(\mathbf{0}_f, \mathbf{I}_f)$—free of hyperparameters, which appear in between as a "geometric mean" diagonalizable via the SVD:

$$\mathbf{\Sigma}^{1/2}\mathbf{\Phi}^{1/2} = \mathbf{U\Lambda V}',$$

where $f \times f$ orthogonal matrices sandwich a diagonal matrix $\mathbf{\Lambda}$ whose diagonal (comprising the singular values) is decreasing and positive. This allows us to write

$$\mathbf{p}_i'\mathbf{q}_j = (\mathbf{U}'\mathbf{\Sigma}^{-1/2}\mathbf{p}_i)' \, \mathbf{\Lambda} \, (\mathbf{V}'\mathbf{\Phi}^{-1/2}\mathbf{q}_j),$$

and absorbing $\mathbf{U}'$ and $\mathbf{V}'$ into the flanking factor vectors does not alter their $\mathrm{N}(\mathbf{0}_f, \mathbf{I}_f)$ distribution. Here's what we've accomplished:

- We've revealed—and eliminated—considerable model overspecification: Without loss of generality, we can reexpress

  $$\mathbf{p}_i'\mathbf{q}_j \quad \text{with } \mathbf{p}_i \sim \mathrm{N}(\mathbf{0}_f, \mathbf{\Sigma}),\ \mathbf{q}_j \sim \mathrm{N}(\mathbf{0}_f, \mathbf{\Phi}),$$

  involving effectively $f(f + 1)$ dimensions' worth of covariance hyperparameters—as

  $$\mathbf{p}_i'\mathbf{\Lambda}\mathbf{q}_j \quad \text{with } \mathbf{p}_i \sim \mathrm{N}(\mathbf{0}_f, \mathbf{I}_f),\ \mathbf{q}_j \sim \mathrm{N}(\mathbf{0}_f, \mathbf{I}_f),$$

  involving a single diagonal matrix—merely f dimensions' worth of hyperparameters.

- Because $\mathbf{p}_i'\mathbf{\Lambda}\mathbf{q}_j = \sum_{k=1}^{f} \lambda_k p_{i,k} q_{j,k}$, with the p's and q's all IID N(0, 1) (standard normal), the diagonal elements of $\mathbf{\Lambda}$ are revealed as weights that scale otherwise similarly behaving products $p_{i,k}\, q_{j,k}$. And, thanks to the sorting of the singular values, λ_k can be thought of as the *importance* of the kth most important LF.
- That is, we've settled on a particular, appealing representation of the interactions between users and items: one in which the f contributing factors—whether on the user side or the item side—are orthogonal to each other (uncorrelated, independent), and are distinguished by their relative importance (provided the singular values are distinct).

Extending the Gaussian premise to the main effects and conducting suitable transformations (details omitted) leads us to the ratings model [cf. (1)]

$$r_{i,j} = \begin{bmatrix} 1 & \alpha_i & \mathbf{p}_i' \end{bmatrix} \begin{bmatrix} \mu & \sigma_\beta & \boldsymbol{\tau}_\mathrm{q}' \\ \sigma_\alpha & 0 & \mathbf{0} \\ \boldsymbol{\tau}_\mathrm{p} & \mathbf{0} & \mathbf{\Lambda} \end{bmatrix} \begin{bmatrix} 1 \\ \beta_j \\ \mathbf{q}_j \end{bmatrix} + \text{noise}, \tag{4}$$

where this (block-matrix) bilinear form organizes the hyperparameters into the middle matrix, and the random variables (factors and main effects) into the flanking matrices; here,

- $\mathbf{\Lambda}$ is (still) the $f \times f$ diagonal matrix of (positive, decreasing) *importances*;
- the user effects (α_i, $\mathbf{p}_i$) and item effects (β_j, $\mathbf{q}_j$) are all IID $\mathrm{N}(\mathbf{0}_{f+1}, \mathbf{I}_{f+1})$;
- σ_α^2 and σ_β^2 are main-effect variances;
- $\boldsymbol{\tau}_\mathrm{p}$ and $\boldsymbol{\tau}_\mathrm{q}$ are free, real $f \times 1$ vectors that allow correlation between a user's (or item's) main effect and factor vector.

Apart from the noise variance, our model has $3(f + 1)$ dimensions' worth of hyperparameters—far more parsimonious than the overspecified models in the literature.

From here, we have charted a fully Bayesian, hierarchical route, identifying expedient prior distributions for the hyperparameters and constructing a (MCMC) *Gibbs sampler* for drawing samples from the (ratings-informed) posterior distributions of the random effects and hyperparameters. Early implementation efforts [7] show promise.

More on symmetries and factor interpretation. Without an appropriate appreciation for the symmetries lurking in (1), LF efforts have incurred various consequences. When random-effect collections are to be sampled from a posterior distribution, such symmetries will manifest themselves as *aliasing*: equivalence classes on which the probability density is constant. We regard such a density as having "lobes" that are copies of each other, symmetrically arranged—in our case, around an unfortunate mean: the all-zero-effect collection. Our sample average, then, is destined to misleadingly approach this zero-collection. In [6], the fact that this seems not to happen is evidence of sampling from a *poorly mixing* Markov chain: The chain likely lands in a lobe from which it then fails to escape. (We've seen evidence of this in other settings: for example, when multiple chains are run in parallel and each lands—and stays—in a different region [8].)

Return to CONTENTS

Ironically, this mimics the proper remedy of restricting sampling to a single lobe (any lobe), but it's happening accidentally; eliminating symmetries ensures that it will happen properly.

LF efforts often seek qualitative understanding of the factor vectors they produce; for example, Bell & Koren's winning Netflix team was able to look at scatterplots of factor-subvectors and realize that two prominent factors they'd found were "seriousness" (of user or movie) and "gender [appeal]" [3]. Such sleuthing is undoubtedly complicated by symmetries (a solution is high-dimensional and may not have landed in a particularly interpretation-friendly lobe); perhaps our canonical, orthogonal representation facilitates interpretation: Factors are uncorrelated, so we gain nothing by considering them in tandem. ("Orthogonality isn't realistic," we've heard, but this doesn't seem like a legitimate complaint when any lobe is as admissible as any other; why must we rule out the "orthogonal" one?) Now, there may well be correlation between "seriousness" and "gender"; our representation would redraw the axes to define two alternative, uncorrelated factors. Does this make the interpretation less clear? Or does it reveal deeper truths?

Finally, we note that our representation (4) is not yet free of symmetries: $\boldsymbol{\Lambda}$ will be unchanged if pre- and post-multiplied by a diagonal matrix of arbitrary signs (±1), the two copies of which can be algebraically moved out to the flanking factor vectors (changing some signs). This amounts, for example, to deciding whether we're measuring "seriousness" or its opposite ("whimsy"?). A remedy is to choose a *reference* user or item and stipulate that all of its factors are to be positive (e.g., "we're estimating either seriousness or whimsy, whichever *Groundhog Day* has a positive amount of").

Variations, and ideas for tackling them

What we've learned suggests new ways to grapple with common RS complications.

Multivariate response. We might observe a vector of feedback from the (i, j) experience (instead of the univariate $r_{i,j}$). Extending our earlier perspective, we imagine this as dimension reduction within *multivariate* analysis of variance (MANOVA). We pursue our univariate approach across the feedback vector, with each component having its own factor vectors and hyperparameters; correlation between components is modeled naturally by postulating correlation between factor vectors across components. (Details remain to be worked out.)

Nonignorable missingness. Imagine that even when we don't get to observe $r_{i,j}$, it "lurks." There may be correlation between $r_{i,j}$ and *whether* we get to see it (e.g., i tries j only if expecting to like it, and then rates j only if extremely [un]happy). Such *nonignorable missingness* (NM) is a concern since we use observed ratings to predict unobserved ones. Broadly, the successful statistical treatment of NM seems to rest on strong modeling assumption(s) about how what's seen relates to what isn't seen. The standard text [9] devotes only its last chapter to this challenge, but describes a *normal selection* model we could adopt: Supplement $r_{i,j}$ with a latent "gatekeeper" response $\tilde{r}_{i,j}$ whose sign (positive or negative) controls whether we observe $r_{i,j}$. That is, we now have a bivariate response $(r_{i,j}, \tilde{r}_{i,j})$—but $\tilde{r}_{i,j}$ is never observable; its *sign* always is, and, (only) if that sign is positive, $r_{i,j}$ is observed. [10] and [11] pursue schemes like this; we expect progress from a more careful, modern treatment of the issues.

Observables are not continuous, and sometimes it matters. Our Gaussian premises have led to insight, but ratings—even if quantitative—tend to be discrete: low-medium-high, or numbers of "stars." [10] and [11] have attempted to model such *ordinal* recommendations, using a latent continuous rating and cutpoints that "bin" it. (A related example is the binarization of the NM gatekeeper $\tilde{r}_{i,j}$, although this isn't a rating.) We suspect that a thoughtful study of ordinal-data literature (e.g., [12]) would yield dividends.

Implicit feedback. In most scenarios, we get no explicit rating—just some measurement that we regard as a reasonable surrogate (e.g., number of visits to a website versus an actual rating of the website). The gatekeeper device helps us understand how things change: When $\tilde{r}_{i,j} > 0$, we see a meaningful measurement ($r_{i,j}$, now likely noisier than if feedback were explicit). But when $\tilde{r}_{i,j} < 0$, we can no longer detect this; instead of a recognized nonresponse, we see a measured value whose meaning is ambiguous. [Does 0 visits mean the user intentionally avoided the site (a de facto rating), or was the user unaware of the site (a nonresponse)?] A careful multivariate-response formulation (comprising the always-observable measurement and suitable, related latent components) should yield insight and progress beyond present ad hoc approaches.

Inclusion of user, item features. RS literature generally draws a line between CF and *content-based* methods that take advantage of observable user/item characteristics (*feature vectors*) to help with recommendations. We motivate a different perspective: If you're planning to do CF-by-LF, and then you're handed vectors of (user) features, shouldn't these give you a head start on the (user) factor vectors? From the position that LFs capture the users' salient qualities, we regard the observable user feature vector $\mathbf{x}_i$ as some (unknown, noisy) function of the latent $(\alpha_i, \mathbf{p}_i)$. Assuming the function is affine and the noise is Gaussian, we discover that when we condition on the $\mathbf{x}_i$'s, the effect is to "personalize" the prior mean of $(\alpha_i, \mathbf{p}_i)$ away from $\mathbf{0}$ to an affine function of $\mathbf{x}_i$ (involving new hyperparameters). (A symmetric analog relates item features to their factor vectors.) Remarkably, this coincides with a basic version of the *regression-based latent factor model*, an ad hoc proposal [13] for contending with the classic *cold-start* problem of making recommendations for a new user/item that shows up without any ratings; we've now supplied a theoretical justification.

Acknowledgments

Acknowledgments to Chris Dearing and Bob Bell for many helpful conversations.

References

[1] Box G, Hunter J, Hunter W. *Statistics for Experimenters, 2nd ed.* Hoboken (NJ): John Wiley & Sons; 2005. ISBN-13: 978-0-471-71813-0.

[2] Feurverger A, He Y, Khatri S. "Statistical significance of the Netflix challenge." *Statistical Science.* 2012;27(2):202–231. doi: 10.1214/11-STS368.

[3] Koren Y, Bell R, Volinsky C. "Matrix factorization techniques for recommender systems." *Computer.* 2009;42(8):30–37. doi: 10.1109/MC.2009.263.

[4] Koren Y, Bell R. "Advances in collaborative filtering." In: Ricci F, Shapira B, editors. *Recommender Systems Handbook, 2nd ed.* New York (NY): Springer; 2015. pp. 77–118.

[5] Hastie T, Tibshirani R, Friedman J. *The Elements of Statistical Learning, 2nd ed.* New York (NY): Springer; 2009. ISBN-13: 978-0-387-848-570.

[6] Salakhutdinov R, Mnih A. "Bayesian probabilistic matrix factorization using MCMC." In: *Proceedings of the 25th International Conference on Machine Learning;* 2008 July 5–9; New York, NY.

[7] Jacobs J, Maissen J, personal communication, September 2017.

[8] Dearing C., personal communication, December 2016.

[9] Little R, Rubin D. *Statistical Analysis with Missing Data, 2nd ed.* Hoboken (NJ): John Wiley & Sons; 2002. ISBN-978-0-471-18386-0.

[10] Ying Y, Feinberg F, Wedel M. "Leveraging missing ratings to improve online recommendation systems." *Journal of Marketing Research.* 2006;43:355–365. doi: 10.1509/jmkr.43.3.355.

[11] Hernandez-Lobato J, Houlsby N, Ghahramani Z. "Probabilistic matrix factorization with non-random missing data." In: *Proceedings of the 31st International Conference on Machine Learning;* 2014 Jun 22–26; Beijing, China.

[12] Agresti A. *Analysis of Ordinal Categorical Data, 2nd ed.* Hoboken (NJ): John Wiley & Sons; 2010. ISBN-10: 047-008-2895.

[13] Agarwal D, Chen B-C. *Statistical Methods for Recommender Systems.* New York (NY): Cambridge University Press; 2016. ISBN-13: 978-1-107-03607-9.

Applying deep learning to implement a wideband radio frequency classifier for dynamic spectrum access

Dr. Rob Miller | Dr. Marc Lichtman | Edward Laird

In 2017, Apple released the iPhone X equipped with an A11 processor that contains what Apple calls a "neural engine." Huawei announced a similar "neural processing unit (NPU)" available in one of their Kirin 970 system-on-chip (SoC) for mobile devices [1]. Google also released the Pixel 2 which contains an image processing unit (IPU) on their SoC. In addition, Google announced TensorFlow Lite, their version of the TensorFlow framework targeting Android and iOS application development. Intel acquired Movidius, a company that makes the neural compute stick (NCS) [2]. The NCS is a similar low-power neural processing-specific hardware for Internet of Things (IoT) development and prototyping. It is only a matter of time before companies start using specialized chips to run radio frequency (RF) deep learning applications for 5G and IoT dynamic spectrum access (DSA) applications.

[Photo credit: jamesteohart/iStock]

Introduction

An overarching goal of radio frequency (RF) deep learning research is to perform spectral use awareness for dynamic spectrum access (DSA) applications. Spectrum sharing is believed to be an important evolution moving from 4G LTE to future 5G and Internet of Things (IoT) technologies. RF signal classification from base-banded raw in-phase and quadrature (I/Q) signal data is an acceptable approach to engineering a solution, but what if the complex process of signal detection, tuning, filtering, and decimating could be removed from the process entirely? Another approach is to design an RF classifier that consumes the entire RF band of interest. A convolutional network could be trained on the wideband data, where minimal digital signal processing takes place after analogue-to-digital sampling.

The wideband RF classifier approach is to build a semisynthetic data set consisting of wideband raw I/Q needed to explore deep learning spectrum classification approaches. The simplest approach to classifying known signals in the industrial, scientific, and medical (ISM) radio band is to train a neural network over the entire static bandwidth and sample duration relative to a common receiver center frequency. The neural network assesses the entire RF bandwidth at once, relying on minimal digital signal processing in front of the classifier. This is a major change from traditional signal detection, filter, tune, decimate, and classify approaches. While the traditional approach works, the signal processing chain, latency, and system-on-chip (SoC) complexity could be greatly reduced by using neural networks to classify raw wideband spectrum.

We explored this theory using the 100 megahertz (MHz)-wide 2.4 gigahertz (GHz) ISM band for its manageable propagation distance and the widely available generative equipment that operates in the band. We chose the Ettus X310 universal software radio peripheral (USRP) [3], equipped with a UBX160 [4] daughter board, because of its cost, bandwidth, and future RF network-on-chip (RFNoC [5]) enhancement potential. We developed and tested several convolutional models using TensorFlow. We trained the models using an Nvidia 1080 Ti graphics processing unit (GPU), and ran them from inference using the Ettus X310 USRP connected to an Nvidia Jetson TX2 development board. We used GNU Radio [6] RFNoC blocks to subsample the streaming 100 MHz raw I/Q data to a rate the Jetson TX2 development board could handle.

Technical approach

Our first step in developing a wideband RF classifier was to generate a wideband data set. We used an RF isolation tent as the base recording environment. While a Ramsey RF isolation box is suitable for recording some of the devices, we preferred the larger tent so we could experiment with recording multidevice networks in a larger space. We placed the Ettus X310 in the RF isolation tent and configured it with a VERT2450 omnidirectional vertical antenna. We calibrated the X310 USRP to account for receiver I/Q imbalance and directly connected it to a server via 10 gigabit Ethernet (GbE). We took all of the recordings with the same gain, centered at 2.45 GHz. We used the UBX160 card in the X310. We made 16-bit scalar complex (SC16) recordings at a sample rate of 100 MHz to allow for coverage of the full 2.4 GHz ISM band. We called the Universal Software Radio Peripheral (USRP) hardware driver using the C language application program interface (API) wrapper to ensure minimal buffer sample drop issues. We did not off tune the low-frequency oscillator (LO) because we assumed the final application would be using lower cost receivers that do not have that option. The LO leakage introduced phase noise and distortion in the recordings; noise and distortion are common to most low-cost receivers. We left the LO leakage in the data set to help target practical real-world applications. Our data recordings lasted roughly three minutes, written directly to an Ubuntu 16.04 server random-access memory (RAM) disk. Three minutes is an arbitrary time; our main concern was to capture enough diverse activity from some of the lower duty cycle signals. The recording length just had to be long enough to contain enough active energy bursts to provide 6,000 unique examples of 1,024 sample slices. 6,000 samples per class is an initial number to start with based on the Canadian Institute for Advanced Research (CIFAR)-10 data set. The more important factor is that the 6,000 samples contain a balanced sampling of each RF channel and mode within a signal class. The United States Federal Communications Commission (FCC) allows Wi-Fi to have 11 channels in the 2.4 GHz ISM band. Bluetooth has 79 channels in the 2.4 GHz ISM

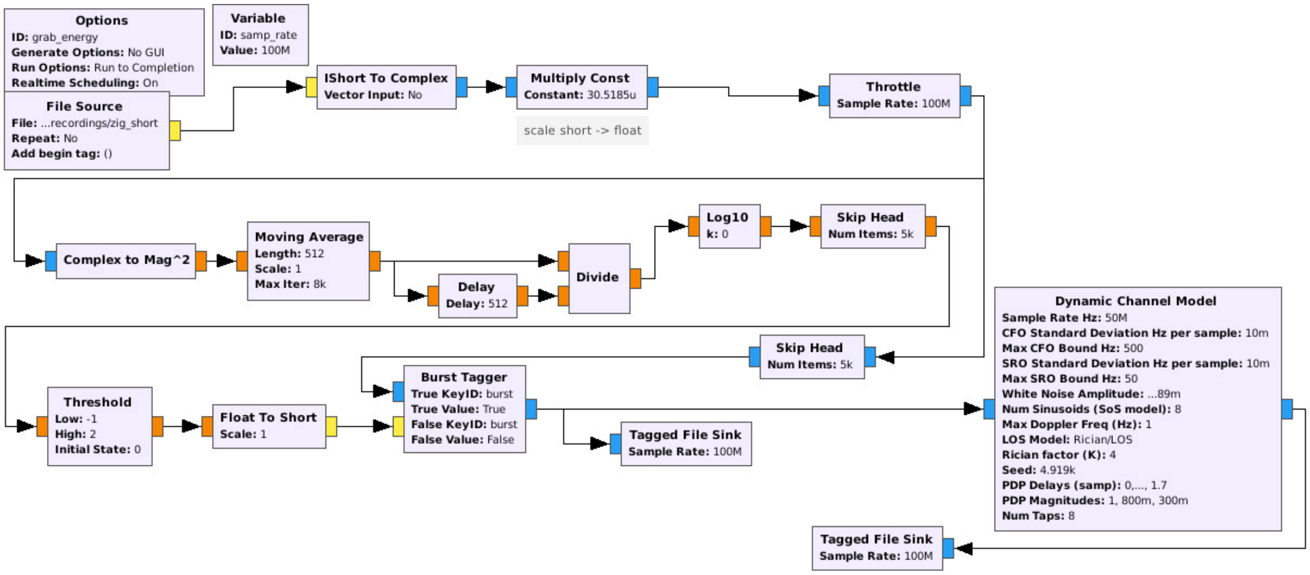


FIGURE 1. This GNU Radio active energy sampling flow graph is the basis behind the automated Python script that created files with active energy. It also applies a scriptable dynamic channel model to the energy, creating synthetic controllable signal-to-noise levels that add diversity to our data set.

band. This sampling process to cover all channels and modes needed to vary from signal to signal.

Our data set consisted of four classes: 1) Bluetooth, 2) nRF24L01+, 3) Wi-Fi, and 4) 802.15.4. We recorded Wi-Fi using a Buffalo ac router configured to 2.4 GHz single band 20 MHz channel width. This is essentially 802.11n with possible 802.11b/g legacy mode operation depending on client device capabilities. We connected a laptop and two tablets to the Wi-Fi router configured to stream videos. Multiple operational channels were captured in the recording; however, not all possible Wi-Fi channels were included in the data set. Bluetooth was recorded using a laptop connected to a Bose SoundLink Mini speaker streaming music. We believe the Bluetooth version was 2.1 + EDR, but the Bose product documentation did not confirm this. We generated 802.15.4 using a special testing device that transmits, continuously hopping through all possible channels in sequential order. We generated nRF24L01+ using a Microsoft wireless keyboard and mouse connected to a laptop.

We used a generic wideband energy detection technique to transform the recordings into only active energy. The signals we chose for the data set were all bursting signals that were not continuously transmitting. It was important that we capture only the times that signals were active in order to train the neural network. Typically, a matched filter technique or something specific to each signal type is used for best detection results. For example, a traditional 802.15.4 receiver detects the signal using a matched filter to correlate on a sequence at the start of each energy burst. To properly correlate, the filter must be coarsely tuned to the center frequency of the burst, allowing only a narrow bandwidth view of the 100 MHz spectrum. A generic wideband energy detection process, however, is designed to work reasonably well over different modulations, bandwidths, and channel hopping. We chose this technique also because of the possibility of adapting it to RFNoC. We used GNU Radio to pull out only areas of active duty cycle for each recording. (See figure 1.) Our initial approach was to train on labeled data consisting of only active energy.

We used the GNU Radio dynamic channel model block to synthetically adapt the isolation tent recordings to simulated signal-to-noise ratios. We configured the dynamic channel model block similarly to how it was configured in the publication "Convolutional radio modulation recognition networks" [7]. The only dynamic channel model block variable that we changed was the noise amplitude, which controls the standard deviation of the additive white Gaussian

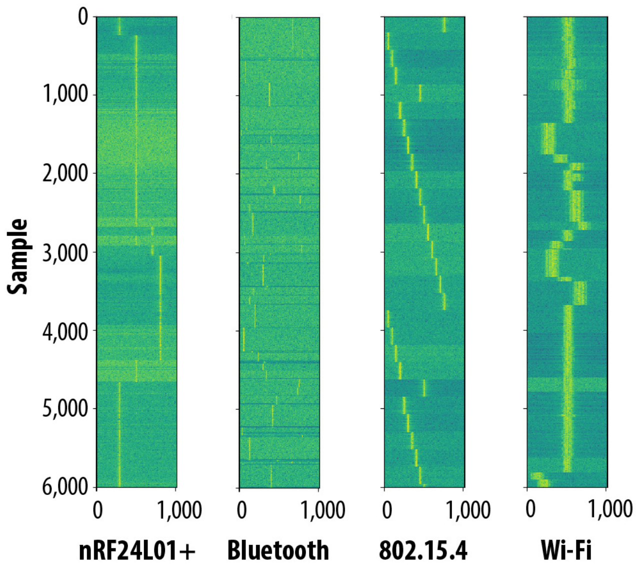


FIGURE 2. The following 100 MHz-wide waterfall plots show all 6,000 data set examples of 1,024 sample, 18 decibel (dB) data concatenated together. From left to right, the classes are nRF24L01+, Bluetooth, 802.15.4, and Wi-Fi.

noise (AWGN) process [6]. The data set refers to this label value as signal-to-noise ratio; however, that is not accurate. The recordings initially contain some noise. The isolated bursting signals also vary in signal to noise and power level over time, and are not necessarily placed equal distances from the recording antenna within the RF isolation tent. The signal-to-noise label is really just an estimate to examine the effects of noise and distortion on the classifier. In addition to energy sampling and dynamic channel modeling, we also used GNU Radio to synthetically combine the energy-sampled data into all possible combinations of the signal classes. Having combined classes on which to train a neural network allows the classifier to use categorical one-hot encoding for each class and class combination.

After noise was added to the active energy samples, the output consisted of various energy burst captures of different lengths stored in individual files. We then imported the files into a Python script that cut them up into 1,024 sample NumPy vectors. A Pandas data frame-based data set was generated that contains 1,024 samples of 100 MHz bandwidth (0.0001024 seconds) data. The vector gain was normalized, and 6,000 samples per class for each signal-to-noise ratio were created. (See figures 2 and 3.)

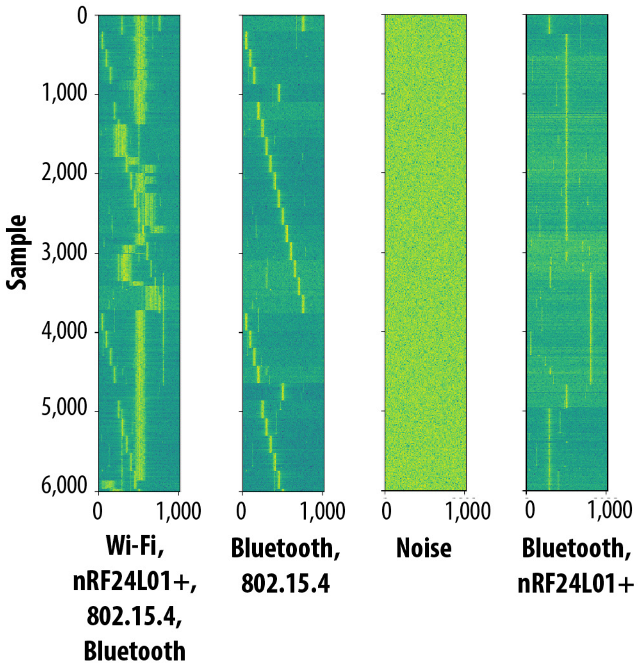


FIGURE 3. The following 100 MHz-wide waterfall plots show all 6,000 data set examples of 1,024 sample, 18 dB data concatenated together. From left to right, the classes are: Wi-Fi, nRF24L01+, 802.15.4, and Bluetooth combined; Bluetooth and 802.15.4 combined; Noise; Bluetooth and nRF24L01+ combined.

We developed and explored several different deep learning models using TensorFlow and TensorBoard. The best performing model on the wideband 1,024 vector length data used a 2,048 x 2 shaped Fast Fourier transform (FFT) as the input features. We applied a padded 2,048 FFT to the 1,024 vector length complex I/Q data. We cast the real and imaginary parts to float 32 values and separate into two dimensions. Four one-dimensional convolutional layers were used in conjunction with max pooling and batch normalization layers. We flattened the output and fed it into three dense layers using elu activation and a dropout layer. (See table 1.)

We generated figure 4 (on page 27) and figure 5 (on page 28) by training on 15 classes of data with an 18 dB signal-to-noise ratio for 20 epochs. There are 6,000 examples per class. The train and test data were randomly shuffled and split in half using the TensorFlow estimator class. As the epochs increase over the x-axis, the model begins to overfit on the data. As the train (gray) and test (green) data begin to diverge, the model overfits on the training data. This means the model

TABLE 1. Keras model summary for 2,048 x 2 FFT-based convolutional model classifier

Layer Type	Output Shape	Parameters
Input Layer	(None, 2,048, 2)	0
Convolutional 1-Dimensional	(None, 1,024, 24)	264
Max Pooling 1-Dimensional	(None, 255, 24)	0
Convolutional 1-Dimensional	(None, 128, 48)	6,960
Max Pooling 1-Dimensional	(None, 31, 48)	0
Batch Normalization	(None, 31, 48)	192
Convolutional 1-Dimensional	(None, 16, 128)	30,848
Max Pooling 1-Dimensional	(None, 3, 128)	0
Convolutional 1-Dimensional	(None, 3, 256)	98,560
Batch Normalization	(None, 3, 256)	1,024
Flatten	(None, 768)	0
Dense	(None, 256)	196,864
Dense	(None, 128)	32,896
Dense	(None, 64)	8,256
Dropout	(None, 64)	0
Dense	(None, 15)	975
Total Parameters		376,839
Trainable Parameters		376,231
Nontrainable Parameters		608

is providing a higher accuracy on training data than the test data. Overfitting starts to noticeably occur at 82.5% accuracy. At 86.2% test accuracy, the model overfits by 1.8%. The highest test accuracy achieved with the model is 89% at epoch 34, not shown in figures 4 and 5 (which stop at epoch 20).

One of our goals of experimenting with wideband RF classification and deep learning was to develop some working implementations on commercial off-the-shelf (COTS) equipment. We investigated two edge devices: 1) the Nvidia Jetson TX2 development board and 2) an Ettus X310 USRP running the classifier implemented in RFNoC. (See figure 6 on page 28) The Ettus X310 USRP is far from an edge device but, at the time, was the only COTS equipment available that supported GNU Radio with RFNoC and bandwidths of 100 MHz. Another approach is to create a new wideband RF data set partitioned into smaller bandwidth chunks. An approach to spectrum classification using parallel classifiers to break up the 100 MHz, 2.4 GHz ISM bandwidth can be found in the publication "Wireless interference identification with convolutional neural networks" [8]. Parallel classifiers could allow some of the Universal Serial Bus (USB) 3.0 digitizers or the EPIQ Solutions Sidekiq [9] to be used. Possibly these same experiments could also be done with the RTL-SDR [10] or LimeSDR Mini [11] as well. The narrow bandwidth would most likely require major data set modifications. We were successful in using the Ettus URSP X310 connected to the Nvidia Jetson TX2 [12] over the network interface. RFNoC blocks were used to throttle the wideband data to a rate the Jetson GPU could ingest. We purposely dropped RF data before the classifier by subsampling. An FFT RFNoC block was used for the FFT-based classifiers. This helped to remove some of the central processing unit (CPU) load on the Jetson TX2. The GNU Radio flow graph was connected to the Python

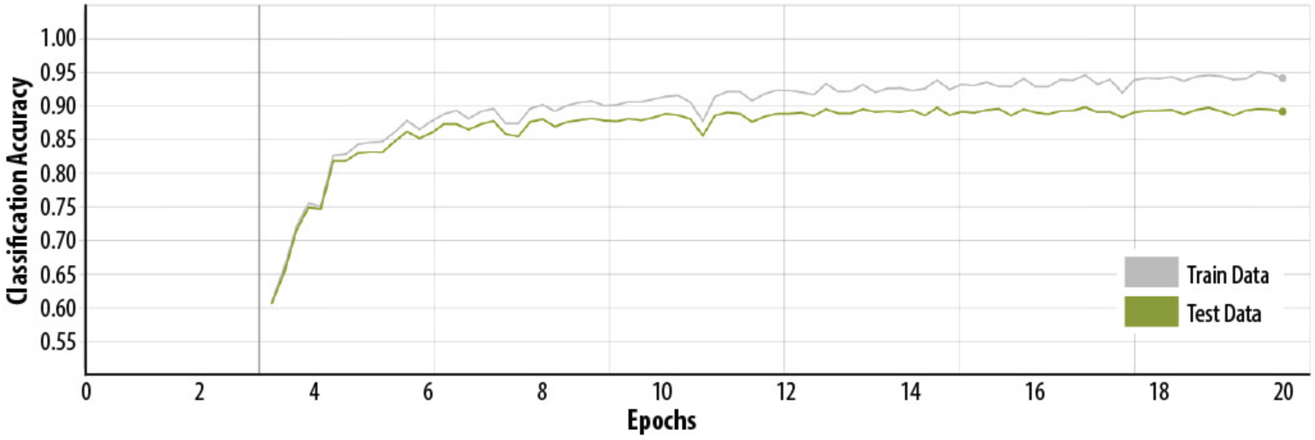


FIGURE 4. The classification accuracy for train data (gray) and test data (green) on 18 dB, 2.4GHz ISM band data for 20 epochs shows that as the epochs increase over the x-axis, the model begins to overfit on the data.

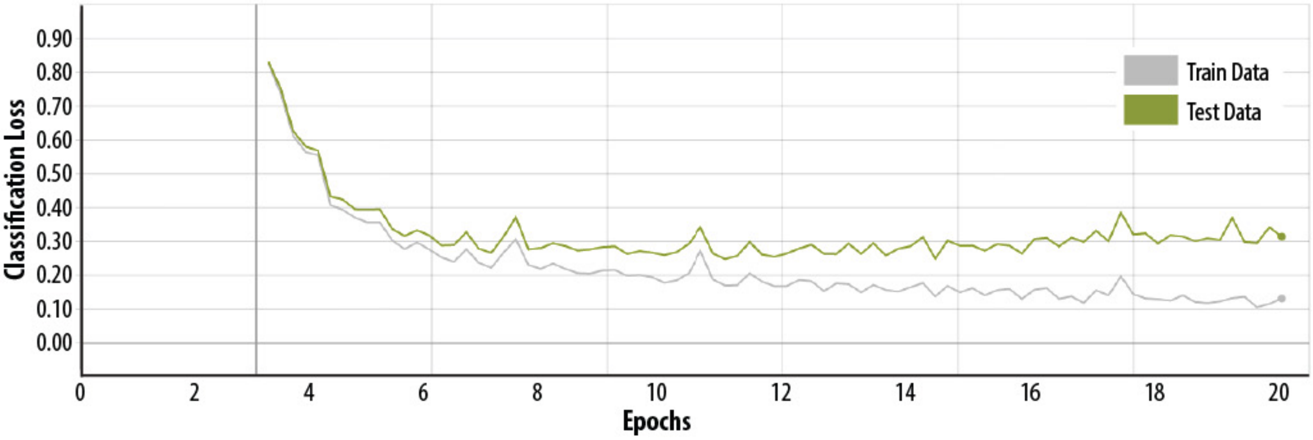


FIGURE 5. The classification loss for train data (gray) and test data (green) on 18 dB, 2.4GHz ISM band data for 20 epochs shows that as the epochs increase over the x-axis, the model begins to overfit on the data.

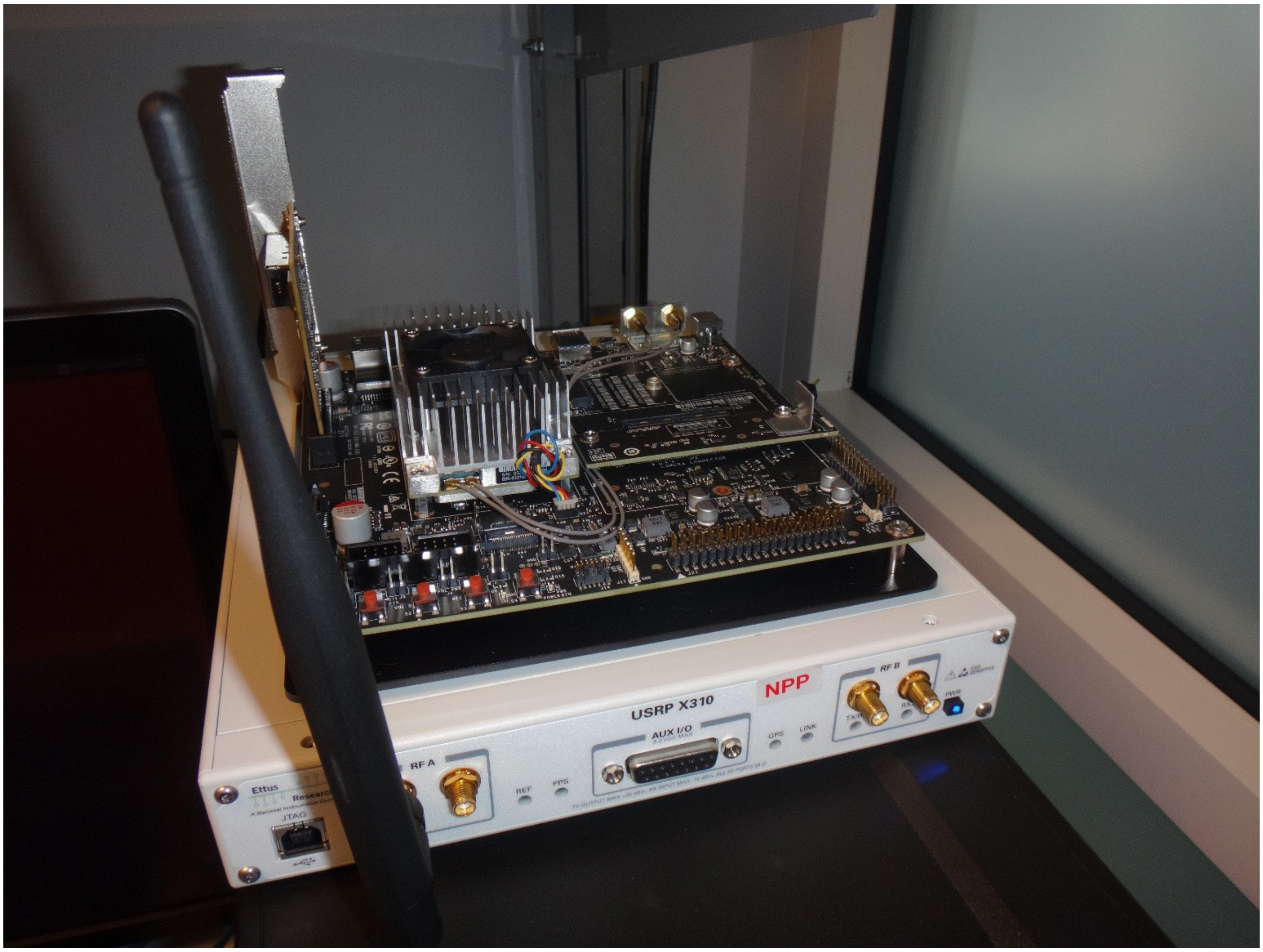


FIGURE 6. Ettus X310 USRP connected to Nvidia Jetson TX2 development board.

script tasking the GPU using ZeroMQ. We tested the 2,048-point FFT model on the Jetson. We used the RF isolation tent with the same devices to generate the data set. We obtained mixed results; the 2,048-point FFT model on the Jetson seemed to be very sensitive to the gain setting on the X310. The samples in the data set were normalized per sample; possibly there is a more realistic approach that would make the neural network classification accuracy more RF-gain invariant. One of the issues is that the data set did not cover all the RF channels of each technology. In the future, GNU Radio could be used to synthetically manipulate the recordings to cover all channel frequencies. We are conducting ongoing work in these areas.

Conclusion

A great deal of more research needs to be done in the area of deep learning models to apply to wideband RF data. The convolutional models here are relatively simple, looking at just FFT data. Residual and recurrent networks should additionally be developed. There is a great deal of room here to explore different types of classifiers.

The most challenging aspect of our research was creating a labeled curated wideband RF data set. Creating a semisynthetic data set that accounts for all aspects of each RF signal is time-consuming and complex. It requires deep knowledge of the standards to cover all operational modes and channels. GNU Radio proved to be a great tool for synthetically expanding the recordings using the dynamic channel model and combining signal classes. Further work needs to be done exploring additional GNU Radio methods for augmenting curated data set creation.

COTS devices are reasonably available to explore wideband RF classification on edge devices; however, there is a great deal of room for the commercial market to expand, enabling a more realistic 5G implementation on a handset or IoT device. Neural enabling-SoC and edge-GPU devices are still commercially focused on image-based classifiers. It would be beneficial to see some future Jetson-like device with an Analog Devices AD9361 on board instead of a camera. There is a long way to go before neural-network-based spectrum sharing implementations can be used efficiently and effectively in commercial handsets and IoT devices.

References

[1] Simonite T. "Apple's 'neural engine' infuses the iPhone with AI SMARTS." *Wired Business.* 2017. Available at: https://www.wired.com/story/apples-neural-engine-infuses-the-iphone-with-ai-smarts/.

[2] Intel Movidius Neural Compute Stick. Available at: https://developer.movidius.com/.

[3] Ettus Research. Product USRP X310. More information available at: https://www.ettus.com/product/details/X310-KIT.

[4] Ettus Research. Product UBX 10-6000. More information available at: https://www.ettus.com/product/details/UBX160.

[5] Ettus Research. RF Network on Chip. More information available at: https://www.ettus.com/sdr-software/detail/rf-network-on-chip.

[6] GNU Radio. Available at: https://www.gnuradio.org/.

[7] O'Shea TJ, Corgan J, Clancy CT. "Convolutional radio modulation recognition networks." 2016. Cornell University Library, arXiv:1602.04105.

[8] Schmidt M, Block D, Meier U. "Wireless interference identification with convolutional neural networks." 2017. Cornell University Library, arXiv:1703.00737.

[9] EPIQ RADIO. Available at: https://epiqsolutions.com/sidekiq/.

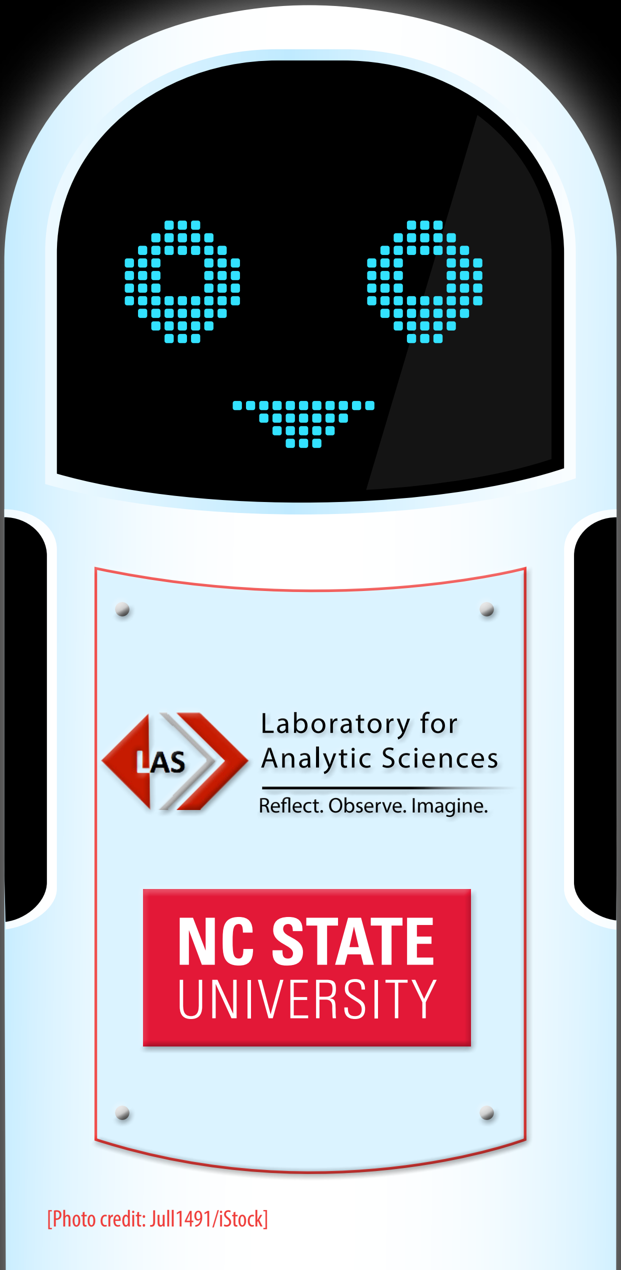
[10] RTL-SDR. Available at: https://www.rtl-sdr.com/about-rtl-sdr/.

[11] Lime Microsystems. Available at: https://limemicro.com/news/99-limesdr-mini-software-defined-radio-launches-crowd-supply/.

[12] NVIDIA JETSON. Product Jetson TX2 Module. More information available at: http://www.nvidia.com/object/embedded-systems-dev-kits-modules.html.

Smart digital assistance for intelligence analysis

Dr. Paul Jones | Dr. S. Lynch | Dr. Kathleen M. Vogel



[Photo credit: Jull1491/iStock]

The promises of machine learning (ML) and artificial intelligence (AI) are beginning to be realized. Technology sector and public research and development investments over the past decades have produced massive improvements in processing power, algorithmic design, and data management. These have in turn fueled a rise in AI capabilities, sparking further investment and creating a positive feedback loop. Private investments alone in 2016 were between $26–$39 billion and have been increasing at an average annual rate of over 40% since 2009 [1]. One technology benefiting from AI advances is digital assistants (DAs). For example, Siri, Cortana, and Alexa now service billions of user requests per week, a testament to how far we've come from Clippy (i.e., Microsoft's famously failed, yet pioneering, DA introduced in Office 97). For the Intelligence Community, enormous potential exists for DA technologies to transform the tradecraft of analysis by helping analysts find the data and analytics they need, facilitating collaboration, and offering workflow recommendations. This is one area under investigation at the Laboratory for Analytic Sciences (LAS), which is one of more than 70 highly integrated industrial and governmental partnerships at North Carolina State University (NCSU).

Machine learning at the LAS

The LAS is a multidisciplinary collaboration of researchers anticipating the future of intelligence analysis and envisioning the technologies and tradecraft that it will require. The model of the LAS is itself an ambitious experiment [2], where immersive collaboration toward mutually beneficial outcomes is a primary goal. This differs from traditional contractual models in that it encourages more open engagement among collaborators and enables experimentation with new research methodologies. Among the LAS-relevant topic areas that benefit from ML are:

- New anticipatory intelligence capabilities,
- Technology and tradecraft efficiency,
- Next-generation user experience, and
- Data triage and discovery assistance.

Specific LAS projects in these areas include:

- The design of recommender algorithms for workflow model construction,
- A system to deliver ML as a service to analysts,
- A software library for enhancing exploratory data analysis via data pipeline test development (software is available, see [3]), and
- System-level multiquery optimization methods to improve efficiency of map-reduce analytics.

However, given the open-ended nature of intelligence analysis, DA research possesses the greatest potential to revolutionize intelligence tradecraft. Below we describe ongoing ML/AI research at the LAS investigating data modeling for DA technology to enhance intelligence analysis.

Mission problem

Modern intelligence analysts sift through a deluge of information in various formats to discover information of value. Unfortunately, most analysis tools are primitive, allowing only one-dimensional analysis of complex, multifaceted problems. Moreover, analysts often work in teams and must consider several different analysis strands simultaneously. The question becomes: How can we empower intelligence analysts to better access, keep track of, and process analytic artifacts in these data rich, collaborative, multitasking environments? We envision future DA technologies meeting this challenge by offering a range of proactive assistance to users including real-time recommendations of relevant data sources or analytic workflows and, in special cases, autonomous analysis operations.

Data acquisition

Training a DA requires quantitative data on analyst workflows, which is not trivially acquired, so our first task was to devise a minimally intrusive means of allowing analysts to document and track their workflows in a continuous and objective manner. We could then use this data to create a user-focused, user-friendly predictive DA for intelligence.

To start, we created two *instrumentation agents* [4]—one for the macOS operating system, and the other for the Google Chrome web browser. Our ***macOSinstrumenter*** captures high-level details of all activity on the system, including all applications opened, documents accessed, and screenshots. Our ***chromeInstrumenter*** captures finer-grained information on web browsing activity across all operating systems. This data alone was sufficient to start training sequence prediction models to infer what an analyst might do or need next [5]. However, initial sequence prediction results showed that our models were getting confused by multitasking behavior. To distinguish individual tasks, we added a feature to allow analysts to optionally tag their workflows with task and goal labels using a *journaling* interface [6]. Facilitating such tagging in a minimally intrusive manner is a research challenge in itself [7]. We adopted a simple *task-tree* approach triggered by indications that in-use documents are relevant (i.e., tag worthy). A particularly effective indicator was the scroll-trigger, which causes the task-tree interface to appear when a user scrolls down a web page or document. We tried to ensure that the journaling interface provided benefits to users (such as reminding them of other tasks and identifying other users working on the same tasks) to encourage adoption with the underlying goal of obtaining high-quality labeled data sets for training supervised ML algorithms capable of predicting task labels.

However, error-prone and disruptive prototype tools are not well suited to high-pressure operational analysis environments! Fortunately, the unique model of the LAS facilitated data capture experiments with proxies that were not too dissimilar to intelligence work, including a class of 10 NCSU graduate political science students participating in an international security course. The students were eager to learn about analysis tradecraft, and we were able to structure their course assignments and a journaling task tree in a manner that was consistent with recent research on generic analysis workflows [8]. Throughout a full semester, labeled data was captured from the students in a mutually beneficial experience. This experiment offered many technical and human factor lessons [9] while enabling continued design improvements to the instrumentation and journaling prototypes. We also captured data from an NCSU computer science class (based on a homework-oriented task tree) and from a small group of intelligence analysts at the LAS.

Model development

With three unique corpora of labeled analysis workflows in hand, we set a goal of designing an ML model to predict a task label for each document a user accessed, and to do so in real time so that we could organize their documents for them, both temporally and, critically, by task. After attempting many standard ML methods, it became clear that we needed a new way to model workflows that would represent users, tasks, and documents in a common manner while accommodating the many different types of associations between them. Associations include similarities between users (e.g., coworkers on a team), relationships between tasks (e.g., subtasks of a common goal), similarities between documents (e.g., topics), as well as temporal proximity between these user, task, and document entities. With this new workflow representation, we could then make probabilistic associations between entities, and do so in a scalable and streaming manner.

We chose a multilayer graph representation of our workflows (see figure 1) containing different types of edges, both between and inside each of the node layers, to represent the various types of associations. The problem of guessing task labels for new documents then became one of link prediction between new document nodes and existing task nodes. The field of link prediction is well developed [10]; one common approach is to factor the graph's adjacency matrix (or tensor for multilayer graphs) to uncover latent features useful for prediction. However, such methods can be difficult to scale for large graphs and to adapt for use in streaming environments. Instead, we created a *vector representation* of each node so that distances between nodes could be calculated naturally, providing an alternative approach to link prediction. To do this, we built upon recent research in natural language processing that succeeded in creating vector representations of words in a sentence, ***word2vec*** [11]. ***Word2vec*** turns words into vectors based on their context, drawing from a concept known as the *distributional hypothesis* [12]: "You shall know a word by the company it keeps." Given workflow graphs rather than sentences, the distributional hypothesis could be applied in spirit by defining a context for a graph node.

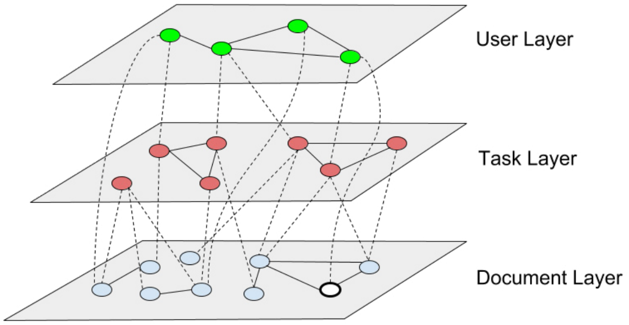


FIGURE 1. Workflows are represented here as multilayer graphs with User, Task, and Document layers. Different types of associations (edges) between nodes within and between layers are permitted. Given a new Document Layer node (white), task association for that document is then equivalent to link prediction between the new node and the Task Layer nodes. Image reproduced with permission from [13]. ©ACM

We generated context data sets for nodes using random walks across the multilayer graph, a technique that had been used on social network graphs [14] but never on workflow graphs. To better explore graph communities, we used guided walks [15] rather than purely random walks. With this collection of walks comprising the context space, we could then train vector representations of each node using ***Skipgram*** [11], a simple neural net for predicting context words given a target word. Skipgram learns a vector embedding of words that maximizes the product of conditional probabilities of context words given target words. With enough training data, Skipgram can learn high-quality representations even for rare words, and in our case, we could produce as much training data as required simply by increasing the number and length of random walks. Skipgram's output layer

Return to CONTENTS

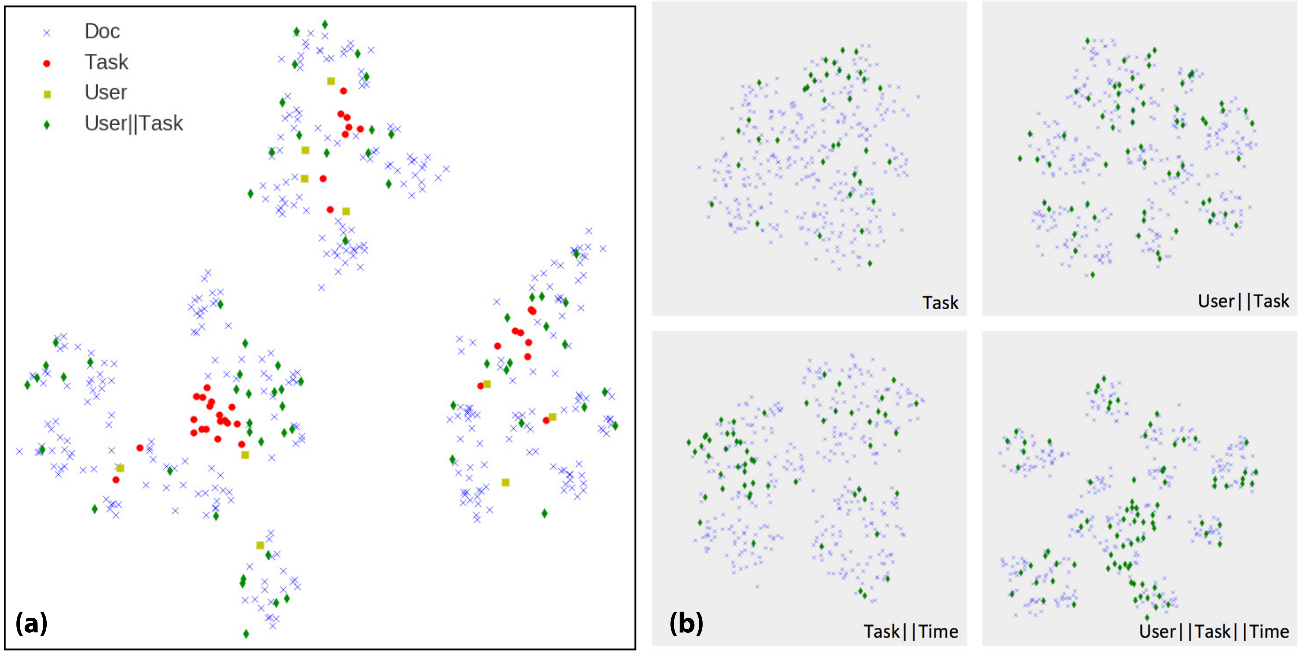


FIGURE 2. Two-dimensional representations of Document, Task, User, and concatenated nodes from the political science workflow data set via t-distributed Stochastic Neighbor Embedding [19]. **(a)** Representation of Document, Task, User, and concatenated User || Task nodes. **(b)** Representations of Document (blue) nodes with Task, User || Task, Task || Time, and User || Task || Time concatenated (green) nodes. Note the increased separation and document coverage resulting from introduction of User||Task and User||Task||Time concatenated nodes (Task||Time nodes showed less clear improvement). Image reproduced with permission from [13]. ©ACM

employs a softmax function to produce a probability vector across all context walks, which trains slowly but speeds up dramatically using hierarchical softmax [16] or negative sampling [17]. Our desired vector embeddings are obtained directly from the converged weights on hidden layer units, and can be visualized with dimension reduction techniques (see figure 2a). Finally, for each new document node that appears, we use its Euclidean or Cosine *distance* from each candidate task node to define document-task association measures. It should be noted that research on methods to create vector embeddings for graph nodes has been moving forward apace in the past couple of years, and there are now a variety of other approaches [18] that we could have used instead.

Model refinement and deployment

The above approach showed promise, which was particularly intriguing since we had neither designed a custom objective function, nor incorporated temporal features of workflows. Intuition and empirical measures suggested that the most recently used documents were more likely to be used again, indicating that temporal information could improve our model. Our solution was to create additional nodes in the multilayer graph that represented *tasks during a particular time period*, the thought being that the artifacts needed for a given task are likely to shift over time. We could also create such concatenated nodes to represent users doing a particular task and users doing a particular task at a particular time. Concatenated nodes end up covering the document space much more thoroughly and evenly, allowing the distance metric to produce a clearer ranking of “close” task nodes (see figure 2b). Indeed, their introduction allowed documents to be classified much more accurately.

Overall accuracy now looks practical for a real system even when as little as 10% of labeled workflow data is supplied by the analyst (see figure 3). We created a real-time version of the algorithm, which updates the underlying multilayer graph when analysts accept or reject task labels, and incorporated it into a

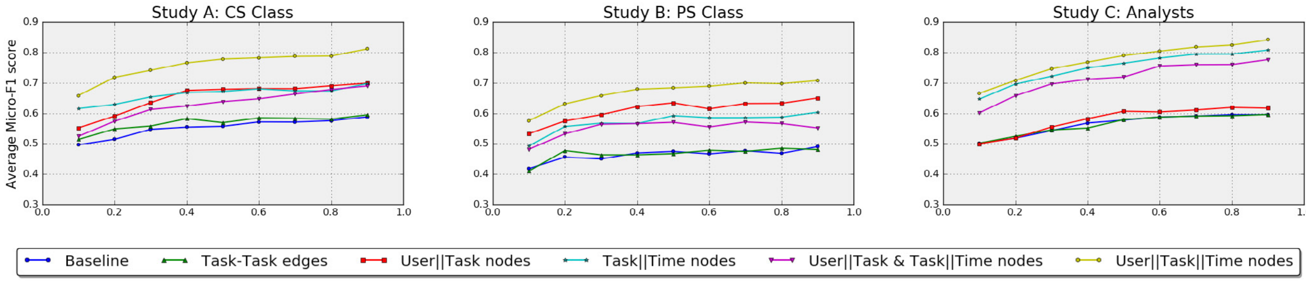


FIGURE 3. The micro-F1 scores versus the fraction of labeled data for each of the three data sets [computer science students (Study A), political science students (Study B), and intelligence analysts (Study C)] shows fairly practical accuracy even with as little as 10% of workflow data being labeled by the analyst. Image reproduced in modified form with permission from [13]. ©ACM

recent-work dashboard prototype [13]. The dashboard organizes documents into reverse time-ordered task blocks and allows other algorithms (such as document summaries) to be incorporated (see figure 4). This serves as an initial DA interface to help individual analysts, and groups thereof, find, file, and *discover* documents relevant to their current task. This saves time, facilitates information sharing, and, crucially, is largely achieved with minimal disruption. Using the instrumentation software discussed earlier, we are now working to quantify the time savings in practice.

Enhancing analytic tradecraft

The graph structure and algorithm defined in the previous section enables document and user predictions to be made for new tasks as well, as long as they are related to existing tasks or users. Mature versions of this prototype will enable the following hypothetical. Suppose a user wants to create an annual report on civil unrest in Brazil. To begin, they create a new task via the dashboard titled “2017 Civil Unrest Brazil” (and organize it near other civil unrest- or Brazil-related tasks in the task tree). Immediately after doing so, the DA will discover relevant documents the user may wish to access and it will autonomously place copies or links to those documents in a folder. These could include recent Brazilian civil unrest reports, previous annual reports, reports on neighboring countries, etc. The DA would also discover users who may require access to the shared folder (e.g., colleagues with regional expertise or supervisors) and suggest that “2017 Civil Unrest Brazil” be added to their task trees. Future versions could also recommend and preload workflow actions such as particular queries into available crime databases.

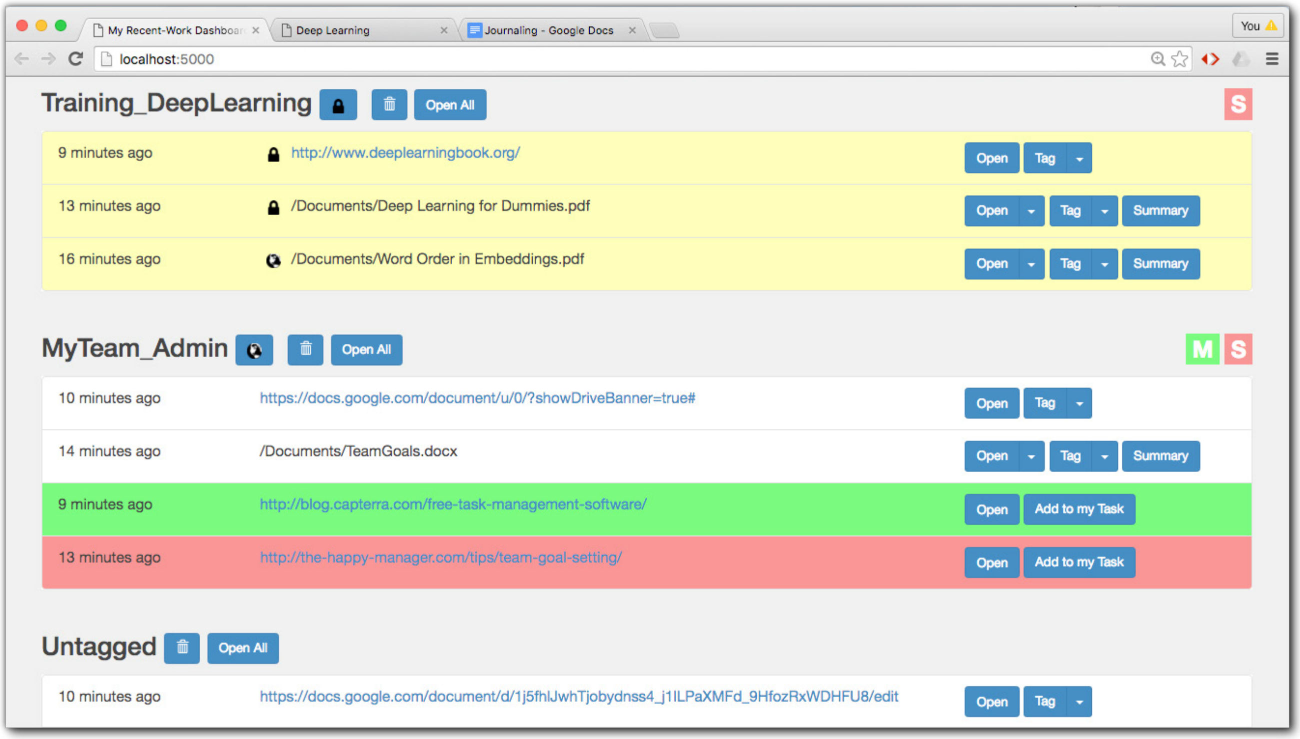
Methods to effectively identify user-task-document associations in real time, such as those presented in previous sections, are key to enabling DAs for intelligence analysis. Smart software prefetching, goal-driven search engines, training of workflow recommendation algorithms, query recommendation and preprocessing, and many other technical innovations derive from this capability. These and other technical innovations will significantly enhance analysts’ abilities to find the data and analytics they need, triage the deluge of data, efficiently utilize their time, and effectively collaborate with other analysts. Even basic user/task/document statistics, which are available as a byproduct of this model, can offer intelligence leaders greater insight to inform resource allocation decisions. ML research underpins these technical advancements, and in our case, this was enabled by the unique blending of academia, industry, and government resources at the LAS.

References

[1] Bughin J, Hazan E, Ramaswamy S, Chui M, Allas T, Dahlström P, Henke N, Trench M. “Artificial Intelligence: The next digital frontier.” *McKinsey Global Institute* (discussion paper). 2017 June.

[2] Vogel K, Jameson J, Tyler B, Joines S, Evans B, Rendon H. “The importance of organizational innovation and adaptation in building academic-industry-intelligence collaboration: Observations from the Laboratory for Analytic Sciences.” *The International Journal of Intelligence, Security, and Public Affairs*. 2017;19(3):171–196. doi: 10.1080/23800992.2017.1384676.

[3] Campbell J, Gong A. Laboratory for Analytic Sciences. Software and documentation available on GitHub at: github.com/great-expectations/great_expectations.



Training_DeepLearning

9 minutes ago http://www.deeplearningbook.org/

13 minutes ago /Documents/Deep Learning for Dummies.pdf

16 minutes ago /Documents/Word Order in Embeddings.pdf

MyTeam_Admin

10 minutes ago https://docs.google.com/document/u/0/?showDriveBanner=true#

14 minutes ago /Documents/TeamGoals.docx

9 minutes ago http://blog.capterra.com/free-task-management-software/

13 minutes ago http://the-happy-manager.com/tips/team-goal-setting/

Untagged

10 minutes ago https://docs.google.com/document/d/1j5fhlJwhTjobydnss4_j1ILPaXMFd_9HfozRxWDHFU8/edit

FIGURE 4. An initial prototype DA interface to help individual analysts, and groups thereof, find, file, and discover documents relevant to their current task. Document summarization software is easily integrated.

[4] Jones P, Thakur S, Cox S, Matthews M. "A versatile platform for instrumentation of knowledge worker's computers to improve information analysis." In: *2016 IEEE Second International Conference on Big Data Computing Service and Applications (BigDataService)*; 2016 Mar 29–Apr 1; Oxford, UK: pp. 185–194. doi: 10.1109/BigDataService.2016.47.

[5] Moon C, Medd D, Jones P, Harenberg S, Oxbury W, Samatova NF. "Online prediction of user actions through an ensemble vote from vector representation and frequency analysis models." In: *Proceedings of the 2016 SIAM International Conference on Data Mining*; 2016 May 5–7; Miami, FL: pp. 90–98. doi: 10.1137/1.9781611974348.11.

[6] Jones P, Thakur S, Matthews M, Cox S, Streck S, Kampe C, Srinath P, Samatova N. "Journaling interfaces to support knowledge workers in their collaborative tasks and goals." In: *2016 International Conference on Collaboration Technologies and Systems (CTS)*; 2016 Oct 31–Nov 4; Orlando, FL: pp. 310–318. doi: 10.1109/CTS.2016.0064.

[7] Shen J, Irvine J, Bao X, Goodman M, Kolibaba S, Tran A, Carl F, Kirschner B, Stumpf S, Dietterich TG. "Detecting and correcting user activity switches: Algorithms and interfaces." In: *Proceedings of the 14th International Conference on Intelligent User Interfaces*; 2009 Feb 8–11; Sanibel Island, FL: pp. 117–126.

[8] Dhami M, Careless K. "Ordinal structure of the generic analytic workflow: A survey of intelligence analysts." In: 2015 *European Intelligence and Security Informatics Conference (EISIC);* 2016 Jan 14; Manchester, UK. doi: 10.1109/EISIC.2015.37.

[9] Kampe C, Reid G, Jones P, Colleen S, Sean S, Vogel K. "Bringing the National Security Agency into the classroom: Ethical reflections on academia-intelligence agency partnerships." *Science and Engineering Ethics*. 2018 January. doi: 10.1007/s11948-017-9938-7.

[10] Al Hasan M, Zaki MJ. "A survey of link prediction in social networks." In: Aggarwal C, editor. *Social Network Data Analytics*. Boston (MA): Springer; 2011. pp. 243–275. doi: 10.1007/978-1-4419-8462-3_9.

[11] Mikolov T, Chen K, Corrado G, Dean J. "Efficient estimation of word representations in vector space." 2013 Sep 7. Cornell University Library. arXiv:1301.3781 [cs.CL].

[12] Harris ZS. "Distributional structure." *Word*. 1954;10(2-3):146–162. doi: 10.1007/978-94-009-8467-7_1.

[13] Jones P, Sharma S, Moon C, Samatova N. "A network-fusion guided dashboard interface for task-centric document curation." In: *Proceedings of the 22nd International Conference on Intelligent User Interfaces;* 2017 Mar 13–16; Limassol, Cyprus: pp. 481–491. doi: 10.1145/3025171.3025177.

[14] Perozzi B, Al-Rfou R, Skiena S. "Deepwalk: Online learning of social representations." In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining;* 2014 Aug 24–27; New York, NY: pp. 701–710. doi: 10.1145/2623330.2623732.

[15] Grover A, Leskovec J. "Node2vec: Scalable feature learning for networks." In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining;* 2016 Aug 13–17; San Francisco, CA: pp. 855–864.doi: 10.1145/2939672.2939754.

[16] Mnih A, Hinton GE. "A scalable hierarchical distributed language model." In: Koller D, Schuurmans, Bengio Y, editors. *Advances Neural Information Processing Systems 21 (NIPS 2008)*. Neural Information Processing Systems Foundation; 2008. pp. 1081–1088.

[17] Mikolov T, Dean J. "Distributed representations of words and phrases and their compositionality." In: Burges CJC, Bottou L, Welling M, Ghahramani Z, Weinberger KQ, editors. *Advances in Neural Information Processing Systems 26 (NIPS 2013)*. Neural Information Processing Systems Foundation; 2013.

[18] Hamilton WL, Ying R, Leskovec J. "Representation learning on graphs: Methods and applications." 2017 Sept 17. Cornell University Library. arXiv preprint: 1709.05584.

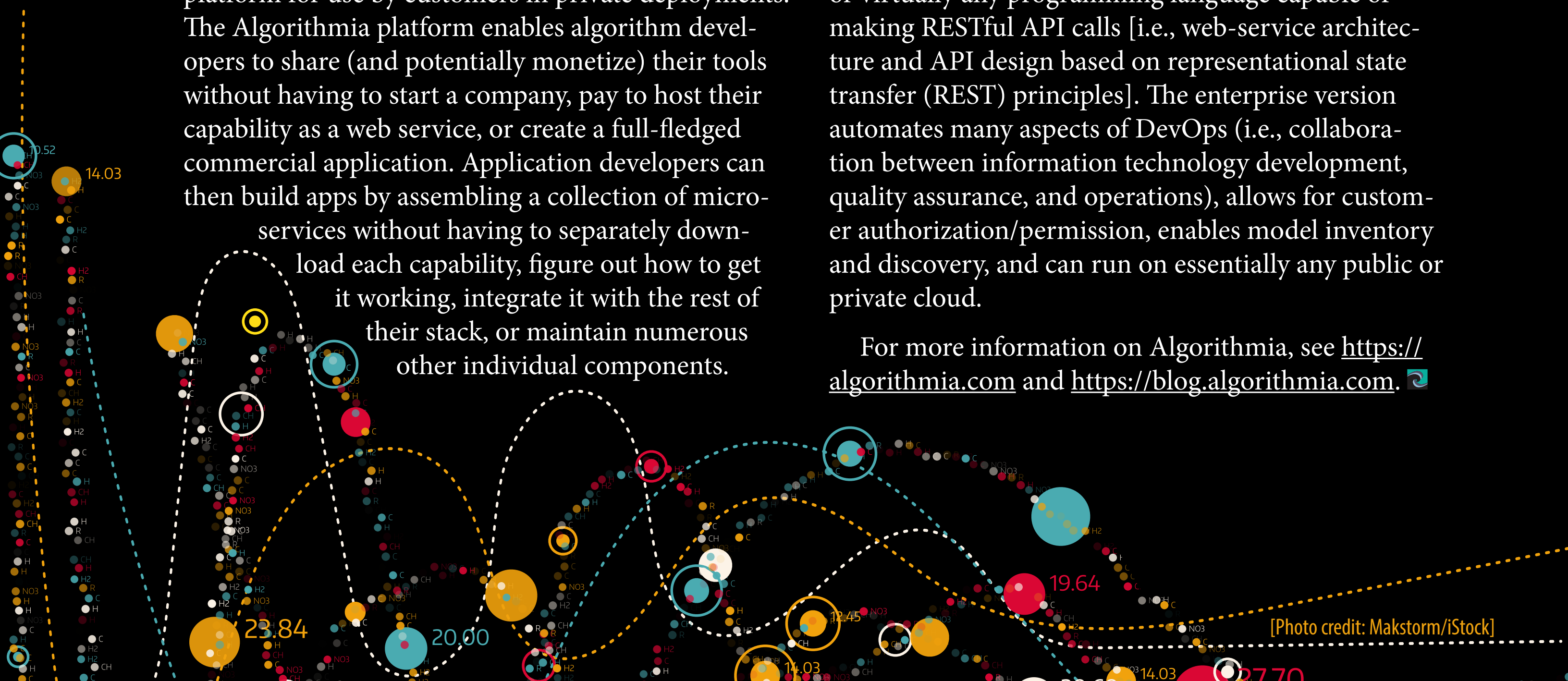
[19] Maaten LVD, Hinton G. "Visualizing data using t-SNE." *Journal of Machine Learning Research.* 2008;9:2579–2605.

In-Q-Tel and Algorithmia partner to make ML algorithms and tools more accessible

In-Q-Tel's recent partnership with start-up company Algorithmia seeks to offer easier access to artificial intelligence (AI) and machine learning (ML) algorithms and tools. Algorithmia provides infrastructure supporting the development and operation of AI/ML capabilities. The two main products' offerings are: a public marketplace providing application programming interface (API) access to thousands of algorithmic microservices and AI/ML models, and an enterprise version of the underlying microservices platform for use by customers in private deployments. The Algorithmia platform enables algorithm developers to share (and potentially monetize) their tools without having to start a company, pay to host their capability as a web service, or create a full-fledged commercial application. Application developers can then build apps by assembling a collection of microservices without having to separately download each capability, figure out how to get it working, integrate it with the rest of their stack, or maintain numerous other individual components.

To implement, algorithm developers simply upload their source code or AI/ML model, and the platform instantly deploys it as a live API. Algorithmia supports uploading code in seven different languages (i.e., Java, JavaScript, Python, R, Ruby, Rust, and Scala) and can host models built on 18 different AI/ML frameworks (e.g., CNTK, Caffe, Keras, PyTorch, Scikit-learn, TensorFlow, and Theano). Application developers can utilize easy-to-use Algorithmia client libraries for many different development platforms, or virtually any programming language capable of making RESTful API calls [i.e., web-service architecture and API design based on representational state transfer (REST) principles]. The enterprise version automates many aspects of DevOps (i.e., collaboration between information technology development, quality assurance, and operations), allows for customer authorization/permission, enables model inventory and discovery, and can run on essentially any public or private cloud.

For more information on Algorithmia, see https://algorithmia.com and https://blog.algorithmia.com.



[Photo credit: Makstorm/iStock]

NSA and University of Texas: Joining forces in machine learning

It takes a combination of research, innovation, and collaboration to invent the future. NSA, through a partnership with the University of Texas System (UT System), plans to use this approach to address national challenges in the field of machine learning.

The NSA Technology Transfer Program (TTP) recently signed a five-year agreement between NSA and the UT System to jointly address challenges in the areas of machine learning, innovation capability development, and Internet of Things. The Cooperative Research and Development Agreement (CRADA) provides a flexible framework for both parties to explore these areas in order to solve specific problems at their respective organizations. This research partnership will complement and accelerate ongoing research efforts at NSA, potentially resulting in development breakthroughs for mission.

Movement has already started on many of the CRADA's work plans, including a collaboration with NSA in Texas' cyber office and the University of Texas at San Antonio (UTSA)'s Center for Security Enabled Cloud Computing. The two-year effort focuses on anomaly detection and insider threat activity inside high performance computing (HPC) systems. Though there are existing methods of detecting anomalies in data, the methods are primarily restricted to stable, non-complex application models and require significant computing resources. This partnership provides NSA researchers with direct access to UTSA faculty, their research, and the supercomputing resources of the Texas Advanced Computing Center, housing one of the largest supercomputers in the world. The discoveries resulting from this engagement will benefit both parties.

The success of this research collaboration may lead to computationally efficient techniques that significantly improve the defensive posture of critical infrastructure computing systems across the nation. Initial plans for the collaboration will focus on the detection of insider threat indicators using low-cost network data, which will serve as a benchmark to evaluate the inclusion of high-cost, host-based data in later stages of the project. Additionally, this partnership will serve as a baseline for further collaborations on mission critical efforts concerning cybersecurity, cloud computing, analytics, machine learning, and data visualization. The partnering teams are working to develop a source of data that can be shared between the UTSA and NSA ecosystems.

[Photo credit: antoniokhr/iStock]

The NSA TTP, located within the Research Directorate, establishes partnerships with industry, academia, and other government agencies to help accelerate mission goals, advance science, foster innovation, and promote technology commercialization. For more information about the NSA TTP, visit www.nsa.gov/what-we-do/research/technology-transfer/.

National Security Agency Central Security Service

Defending Our Nation. Securing The Future.