Yuan-Ting Hu and Alexander G. Schwing
University of Illinois at Urbana-Champaign (UIUC)

Editors: Romit Roy Choudhury, Haitham Hassanieh

AN ELEVATOR PITCH ON DEEP LEARNING

Machine learning improves all aspects of our life and very few days pass where we don't interact with a machine learning driven application. For example, we may use fitness trackers that monitor the quality of our sleep. Before leaving to work, we may check the weather forecast. To get to work, we use a navigation system that predicts the fastest route to our first meeting. On the road, we use voice commands to interact with our phone. Web searches and e-commerce recommendations are based on machine learning. Cameras detect faces when we take a picture and our email inboxes are protected by spam filters, which are designed using machine learning tools.

At a very high level, all those applications recognize patterns. Predictions for today's data are based on patterns that have been deemed useful in the past. For example, a specific sequence of words in an email, *i.e.*, a pattern, may lead the algorithm to tag the incoming email as spam. Concretely, a spam filter parses an incoming email, looking for a number of textual patterns. Based on the result, a score is computed, which determines the decision about whether or not to flag this email as spam.

Classical machine learning techniques require careful engineering of those patterns, also referred to as features of the data. Designing features is a time consuming and labor intensive process, which often requires a significant amount of domain expertise. As a result, the approach is hard to scale.

Deep learning, in contrast, subsumes machine learning tools, which use the raw data as input, and automatically infer the patterns that are needed to achieve good results for the considered task. To this end, deep learning algorithms extract multiple levels of abstractions from the raw data by successively transforming the provided input [20]. Consecutive layers transform the data into an increasingly abstract representation, which is semantically meaningful. Importantly, while the type of computations performed in each layer are specified by humans to be operations, such as matrix multiplications or convolutions, their parameters and hence the extracted features are not predetermined.

As a consequence, compared to classical machine learning algorithms, deep learning is more scalable when considering human labor, and does not require significant domain expertise for the considered task beyond understanding input and desired output. It has hence lead to a flurry of applications that can now

Photo, istockphoto.com

be addressed by machine learning wizards joining forces with domain experts.

However, deep learning methods are known to be very data hungry, often requiring thousands of data points. In addition, they are also known to being computationally very expensive, requiring computational resources that can only be provided by recent accelerators, such as graphics processing units (GPUs).

In the following, we first provide a high-level introduction to machine learning in general and deep learning specifically. We then discuss a series of applications that have been addressed using the presented techniques, before we conclude with remarks on future directions for deep learning.

DEEP LEARNING PRIMER

To describe the machine learning concept, we consider the spam filtering task as an example. Slightly more formally, we use x to refer to a data point, *i.e.*, an email in our case. Note that x may encompass all kinds of data that we possess beyond the text itself, *e.g.*, header information etc. Given data about an email, x, we want to predict whether it is spam or not. We use a variable y to indicate this choice. Since y is the output of the system, it is often referred to as an output space variable. For our example, $y = 1$ indicates the email is spam, and $y = 0$ refers to a regular email.

For every incoming email, our system may produce two real-valued numbers, *i.e.*, two scores, assessing the degree to which the system thinks an email is spam or regular. In general, we refer to the score using $F(x, y, w)$, which provides the assessment for regular mail, $F(x, 0, w) \in \mathrm{R}$, and the assessment for spam mail, $F(x, 1, w) \in \mathrm{R}$. Based on the magnitude of the score we flag an email x to be spam if $F(x, 1, w) > F(x, 0, w)$. Beyond the data, the score also depends on some parameters w, often called weights.

Machine learning refers to techniques for adjusting the parameters w, *i.e.*, for using machines to learn the weights w. This process is called training or supervised learning. To train the parameters we construct a dataset $\mathcal{D}$, which contains pairs of input data x, *i.e.*, emails, and its corresponding desired true output y_T, informing us about whether the email should be considered spam. Formally, the dataset contains pairs (x, y_T) which constitute the supervision that we provide to the learning algorithm. Note that both spam and regular emails are required in our dataset $\mathcal{D}$.

During training, we find the parameters w by comparing our current prediction $y^\star$ with the provided desired output, *i.e.*, the groundtruth. If the current prediction agrees with the groundtruth for a pair (x, y_T), the parameters w of the scoring function are not modified. If, in contrast, the current prediction does not agree with the groundtruth, we adjust the parameters in a direction that could correct this error. Note that this adjustment can be computed easily for many useful types of scoring functions $F(x, y, w)$.

In practice, instead of using a single datapoint (x, y_T), we often use a set of examples at a time, average their output error, and adjust the parameters according to the average. We repeat this process for many small subsets of examples extracted from the entire training set $\mathcal{D}$, until the error does not change significantly. This technique is referred to as stochastic gradient descent with mini-batches and performs surprisingly well in general.

Upon having completed training, it is useful to test the model parameters w by assessing their performance on examples that we haven't used for training, *i.e.*, the test set. This quality control ensures that our approach generalizes to data that the algorithm has never observed before. Multiple cases should be differentiated. First, the model may perform well on the training set and slightly worse on the test set, which is the desired behavior. Second, the model may perform well on the training set and poorly on the test set. Third, the model may perform poorly on both the training and test set. In the second case the model does not generalize, which may be caused by overfitting to the specifics of the training data. Possible solutions are to increase the training data or to simplify the model complexity. The third case could be addressed by increasing the model complexity.

Some of the least complex models are linear models, such as logistic regression [8, 29] or support vector machines [7]. Hereby, the score function $F(x, y, w)$ depends linearly on the parameters w and a vector of features $\phi(x, y)$, *i.e.*, $F(x, y, w) = w^T \phi(x, y)$. This vector of features used to be hand-crafted, *e.g.*, in case of email spam filters, each entry in the vector $\phi(x, y)$ refers to a specific textual pattern. For example, a feature indicates the number of times the word "drug" occurs minus 0.5 if $y = 1$, or 0.5 minus the number of times the word "drug" occurs if $y = 0$. Each of those natural language patterns is then combined linearly to compute a score. In the following, let n denote the number of times the word "drug" occurs in an email. Intuitively, if the discussed feature is the only feature, and its corresponding weight w is positive, the algorithm predicts an email to be spam if the word "drug" occurs at least once, since $F(x, 1, w) = n - 0.5 > F(x, 0, w) = 0.5 - n$ if the number of times the word drug occurs is larger than 0.5, *i.e.*, if $n > 0.5$.

Interpretability as an advantage of a linear scoring function is immediately obvious from this example. Moreover, assuming all features to be roughly normalized, the scale of the weight determines the importance of the feature. However, a linear score function is also very simple. Intuitively, the decision boundary between the two classes 'spam' ($y = 1$) and 'not spam' ($y = 0$) is linear as shown in Figure 1(a). It is hence the job of the domain expert to find features that

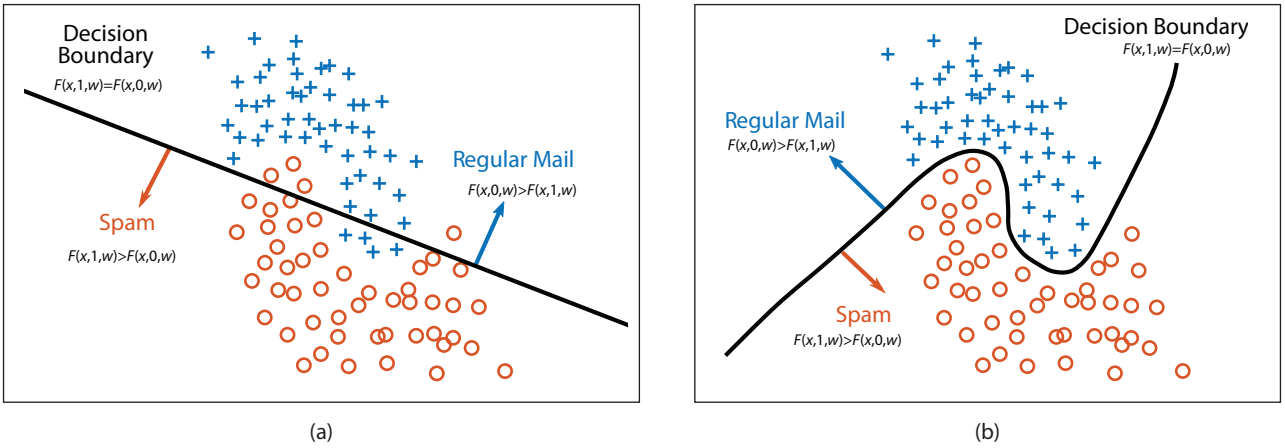

FIGURE 1. The decision boundary of (a) a linear model and (b) a non-linear model. The scoring function $F(x, y, w)$ used in (a) is linear, *i.e.*, $F(x, y, w) = w^T . \phi(x, y)$. A more complex model is more flexible and hence can separate the data as shown in (b).

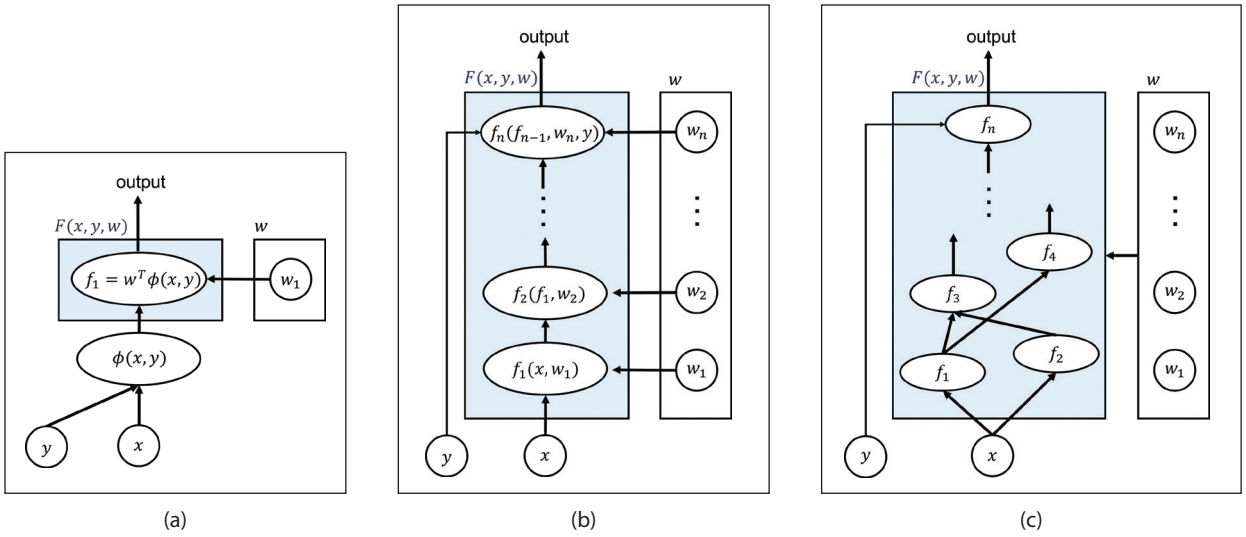

FIGURE 2. The computation graphs of three different models. (a) A single-layer model where the scoring function $F(x, y, w)$ is a linear function. (b) A deep model where the scoring function $F(x, y, w)$ is composed out of n serialized computations. (c) A general form of a deep model where $F(x, y, w)$ is a composite function. Commonly used operations for functions f are convolution and matrix multiplication followed by non-linear operations, such as a rectified linear unit, a hyperbolic tangent or a sigmoid function.

ensure almost linearly separable data points and hence good classification performance. A significant amount of domain expertise is required to design suitable features such that a linear decision boundary achieves a good performance on a given dataset $\mathcal{D}$.

However, it is difficult to design features such that the data is linearly separable. One possible solution is to increase the complexity of the model, *e.g.*, by using a non-linear score function. Given the same training data points, the decision boundary of a non-linear scoring function as shown in Figure 1(b) can better separate the training data than the linear one.

Deep nets simplify the process of designing the features by operating directly on the raw data [20]. Repeated parametric non-linear transformations attempt to extract successively more abstract representations that lead to the desired score $F(x, y, w)$. In the deep net case, the parameter vector w denotes the concatenation of all transformation parameters, which we aim to adjust during training of the model.

Mathematically, we can denote repeated parametric transformations via a composite function. In its most general form, composite functions are illustrated using a computation graph, as shown in Figure 2, composed out of base transformations, such as linear functions, convolutions, etc. The computation graph of a simple linear model is shown in Figure 2(a). The computation graph of a deep net which is composed out of n serialized components is illustrated in Figure 2(b). Note that deep nets automatically transform the raw data into feature representations so there is no need to ex-

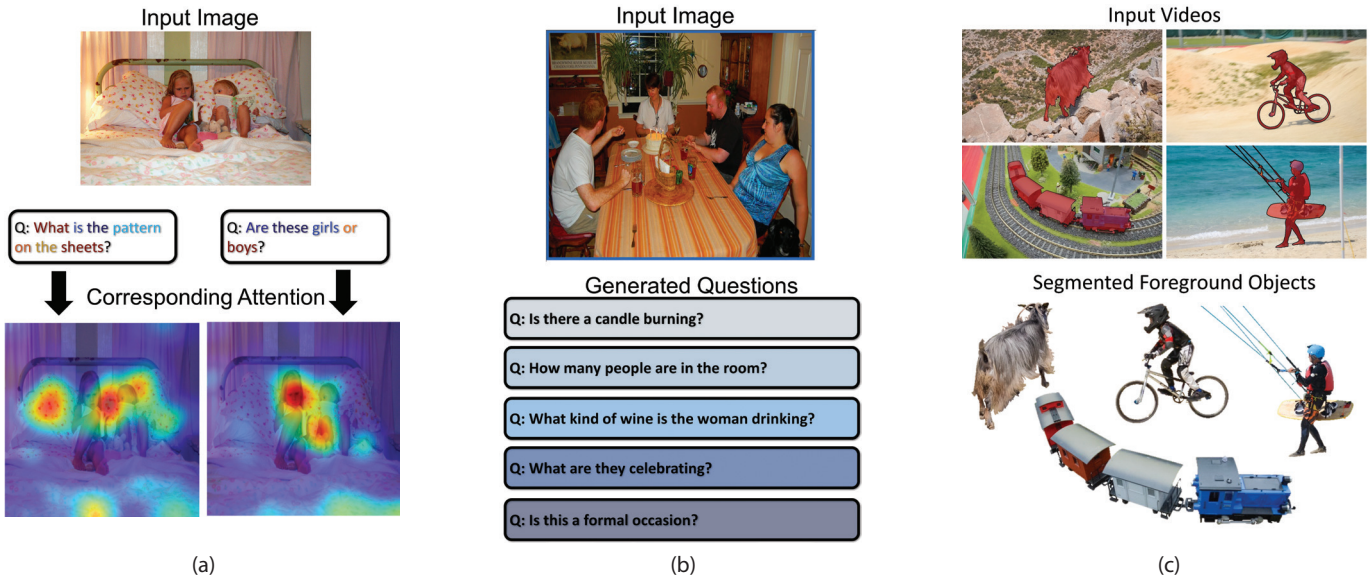

FIGURE 3. Applications of deep learning. (a) Visual question answering. Given an input image and a question, the learned model can answer the question based on its attention which highly depends on the question. The red region is the region with higher attention. (b) Visual question generation. Given an input image, the model is able to generate a diverse set of questions. (c) Video object segmentation. The goal for video object segmentation is to separate the foreground objects (bottom) from the video sequences (top).

plicitly find feature vectors $\phi(x, y)$. A more general form of deep net is illustrated in Figure 2(c). Roughly speaking, the depth of this computation graph corresponds to the complexity of the scoring function.

Nowadays, a variety of software tools [6, 15, 4, 26, 1, 27][1] in different programming languages are available to specify computation graphs and train their parameters on collected training data $\mathcal{D}$.

This training procedure iterates four steps: (i) scores are evaluated on a subset of data-points using the current parameters, which is known as the 'forward pass'; (ii) the difference between the obtained scores and the expected ground-truth is computed; (iii) this difference is then used to scale the derivative of the scoring function w.r.t. its parameters, which is referred to as the 'backward pass'; (iv) the gradient obtained from the preceding step is used to update the models parameters w.

For many models, forward pass and backward pass are the most expensive operations. Depending on the computation graph, matrix multiplications and convolutions are frequent operations. To enable computational efficiency of those operations which are executed thousands of times during training, all of the aforementioned software tools take advantage of GPU accelerators. Even with GPU acceleration, training of deep nets may take up to a few days. Without those accelerators, training, and often also inference, is hardly feasible. To process a useful number of data points at a time, GPUs are required to have a reasonable amount of memory, too.

[1] http://torch.ch, http://caffe.berkeleyvision.org, http://mxnet.io, http://deeplearning.net/software/theano/, http://tensorflow.org, http://www.vlfeat.org/matconvnet/

APPLICATIONS

Because of the general concept, deep learning has found widespread use in many different application domains. It is well beyond the scope of this article to review applications extensively. We will focus on a small subjective selection of computer vision and natural language processing applications in the following.

Making sense of image data is the main focus of computer vision [13, 25]. One of its efforts was organized in the ImageNet challenge [9], a competition requiring to predict the dominant object from 1000 possibilities, given a single input image. First organized in 2010, methods traditionally used linearly weighted hand-crafted features up until 2012 when the University of Toronto team outperformed competitors by a significant margin using a deep net based technique [18]. This result marked a turning point in computer vision and beyond, making deep learning one of the core technologies in many applications.

Among those applications that received a significant amount of attention are image captioning [28, 16, 17, 11] and visual question answering [2, 21], two tasks that were very hard prior to the deep nets [3, 12, 19]. We illustrate some results of our group in Figure 3(a). In recent work [14], our group reversed the task and investigated deep nets for generations of questions given an image. This is challenging since there is not a single question that fits the content of an image. Hence we require mechanisms that generate a diverse set of possible questions. See Figure 3(b) for an example.

Beyond images, deep nets are also applied on videos, *e.g.*, for video object segmentation. Video object segmentation aims at separating the foreground objects from the background region in videos as illustrated in Figure 3(c). It attracts attention because of its various applications, such as video editing, automatic

background removal and replacement, and video compression, which are particularly appealing for mobile applications and mobile augmented reality. The DAVIS challenge for video object segmentation [22] is organized to encourage progress in this area.

Other applications of deep nets include dermatologist-level cancer classification accuracy using a deep net [10]. Deep nets are also used to power autonomous navigation applications, and they are used in AlphaGo, a Go engine which achieved expert level game playing performance to win against Lee Sedol, a Go professional of highest rank [24].

FUTURE CHALLENGES

Deep learning is the beginning of a more data driven era, and applications like visual question answering, question generation and video segmentation will transform how we interact with mobile devices. Going forward, significant challenges remain to be addressed. For example, computationally more effective techniques, faster communication, and hardware is in need to support larger deep nets in edge devices such as phones. Improved training will lead to larger models but it will also lead to algorithms, which automatically search for appropriate composite functions, an approach that is not scalable at this point in time. Deep learning for joint prediction of multiple variables is a recent research topic as well [5, 23].

Applications will continue to address more and more complicated tasks. For example, while natural language processing is fairly common in virtual assistants, their reasoning abilities are still very limited. Finding mechanisms to combine extracted information and attach semantics is an open challenge. ■

Yuan-Ting Hu is a PhD student in the ECE department of the University of Illinois at Urbana-Champaign (UIUC). She received her master's degree and bachelor's degree from National Taiwan University. Her research interests include computer vision and machine learning.

Alexander G. Schwing is an assistant professor in the ECE department of the University of Illinois at Urbana-Champaign (UIUC). Before joining UIUC, he was a postdoctoral fellow at the University of Toronto after receiving a PhD from ETH Zurich. His research interests are in the general areas of machine learning and computer vision.

REFERENCES

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org.

[2] Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C. L., and Parikh, D. (2015). VQA: Visual question answering. In *Proc.* ICCV.

[3] Barnard, K., Duygulu, P., Forsyth, D., Freitas, N. D., Blei, D. M., and Jordan, M. I. (2003). Matching words and pictures. *JMLR*.

[4] Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C., and Zhang, Z. (2015). MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems. In *https://arxiv.org/abs/1512.01274*.

[5] Chen*, L.-C., Schwing*, A. G., Yuille, A. L., and Urtasun, R. (2015). Learning Deep Structured Models. In *Proc. ICML*. *equal contribution.

[6] Collobert, R., Bengio, S., and Marithoz, J. (2002). Torch: A Modular Machine Learning Software Library.

[7] Cortes, C. and Vapnik, V. N. (1995). Support-Vector Networks. *Machine Learning*.

[8] Cox, D. R. (1958). The regression analysis of binary sequences (with discussion). *Royal Statistical Society*.

[9] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *Proc. CVPR*.

[10] Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., and Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature*.

[11] Fang, H., Gupta, S., Iandola, F., Srivastava, R., Deng, L., Dollár, P., Gao, J., He, X., Mitchell, M., Platt, J. C., Zitnick, C. L., and Zweig, G. (2015). From captions to visual concepts and back. In *Proc. CVPR*.

[12] Farhadi, A., Hejrati, M., Sadeghi, M. A., Young, P., Rashtchian, C., Hockenmaier, J., and Forsyth, D. (2010). Every picture tells a story: Generating sentences from images. In *Proc. ECCV*.

[13] Forsyth, D. A. and Ponce, J. (2011). *Computer Vision: A Modern Approach (2nd Edition)*. Pearson.

[14] Jain*, U., Zhang*, Z., and Schwing, A. G. (2017). Creativity: Generating Diverse Questions using Variational Autoencoders. In *Proc. CVPR*. *equal contribution.

[15] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Cae: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093*.

[16] Karpathy, A. and Fei-Fei, L. (2015). Deep visual-semantic alignments for generating image descriptions. In *Proc. CVPR*.

[17] Kiros, R., Salakhutdinov, R., and Zemel, R. S. (2015). Unifying visual-semantic embeddings with multimodal neural language models. In *TACL*.

[18] Krizhevsky, A., Sutskever, I., , and Hinton, G. E. (2012). Imagenet classication with deep convolutional neural networks. In *Proc. NIPS*.

[19] Kulkarni, G., Premraj, V., Dhar, S., S. Li, Y. C., Berg, A. C., and Berg, T. L. (2011). Baby talk: Understanding and generating simple image descriptions. In *CVPR*.

[20] LeCun, Y., Bengio, Y., and Hinton, G. E. (2015). Deep learning. *Nature*.

[21] Malinowski, M., Rohrbach, M., and Fritz, M. (2015). Ask your neurons: A neural-based approach to answering questions about images. In *Proc. ICCV*.

[22] Perazzi, F., Pont-Tuset, J., McWilliams, B., Gool, L. V., Gross, M., and Sorkine-Hornung, A. (2016). A benchmark dataset and evaluation methodology for video object segmentation. In *Proc. CVPR*.

[23] Schwing, A. G. and Urtasun, R. (2015). Fully Connected Deep Structured Networks. In *https://arxiv.org/abs/1503.02351*.

[24] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*.

[25] Szelisiki, R. (2011). *Computer Vision: Algorithms and Applications*. Springer.

[26] Theano Development Team (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints, abs/1605.02688*.

[27] Vedaldi, A. and Lenc, K. (2014). MatConvNet - Convolutional Neural Networks for MATLAB. In *https://arxiv.org/abs/1412.4564*.

[28] Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). Show and tell: A neural image caption generator. In *Proc. CVPR*.

[29] Walker, S. H. and Duncan, D. B. (1967). Estimation of the probability of an et as a function of several independent variables. *Biometrika*.