A Statistician Teaches Deep Learning

G. Jogesh Babu1 · David Banks2 · Hyunsoon Cho3 · David Han4 · Hailin Sang5 · Shouyi Wang6

Accepted: 5 March 2021 / Published online: 23 March 2021
© Grace Scientific Publishing 2021

Abstract
Deep learning (DL) has gained much attention and become increasingly popular in modern data science. Computer scientists led the way in developing deep learning techniques, so the ideas and perspectives can seem alien to statisticians. Nonetheless, it is important that statisticians become involved—many of our students need this expertise for their careers. In this paper, developed as part of a program on DL held at the Statistical and Applied Mathematical Sciences Institute, we address this culture gap and provide tips on how to teach deep learning to statistics graduate students. After some background, we list ways in which DL and statistical perspectives differ, provide a recommended syllabus that evolved from teaching two iterations of a DL graduate course, offer examples of suggested homework assignments, give an annotated list of teaching resources, and discuss DL in the context of two research areas.

Keywords Deep learning · Neural networks · Statistics · Teaching

1 Introduction

Deep learning (DL) has become an essential methodology in many industries, but it is not yet part of the standard statistical curriculum. We believe it is important to bring DL into the purview of statisticians. But there are challenges—aspectsof DL

---

1 Pennsylvania State University, Pennsylvania, USA
2 Duke University, Durham, USA
3 National Cancer Center of Korea, Goyang, South Korea
4 University of Texas San Antonio, San Antonio, USA
5 University of Mississippi, Oxford, USA
6 University of Texas Arlington, Arlington, USA
do not align with conventional statistical wisdom, and there are practical and theoretical barriers that arise when trying to teach this material.

Nonetheless, DL has achieved tremendous successes in the last decade, which make it a critical statistical tool and object of study. It is the foundation for autonomous vehicles [29, 50], which have the potential for revolutionary advances in safety, fuel economy, and congestion reduction [53, 65, 69, 88]. DL can read X-rays more accurately than trained radiologists [1, 67, 94]; it outperforms humans in complex strategy games [75, 86] and has set the standard for automatic language translation [51, 52, 73]. Looking ahead, DL is poised to control robotic sorting of waste [11, 71], coordinate complex medical treatment, and reimbursement management in the US healthcare system [3, 17, 49], and optimize dynamical systems in high-tech manufacturing [78]. DL techniques are especially popular in image classification problems, image synthesis, and speech recognition [13, 25, 37–39, 60, 61, 74, 79, 89, 92].

To remain viable and relevant, statisticians must master DL. That goal means that some of us need to learn how to teach DL. But there are practical, pedagogic, and technical difficulties. This paper will address all three hurdles.

A DL course needs to balance practice and theory, but (for now) the main emphasis should be practice. There is a growing body of DL theory that we could and probably should teach, but the appetite and drive among the students is to do DL, not to prove things about DL. But statisticians generally teach theory before practice, and so we need to adapt. One result of that choice is that there are many platforms for DL programming, and it is problematic to manage a class in which students are programming in different environments.

Our advice is to either let students use whatever software tools they choose, or to enforce a single environment—if the students are from the same department and have a programming lingua franca, we recommend the latter. But in many situations that homogeneity is unrealistic—graduate courses in DL are popular and attract students from many different disciplines with many different backgrounds.

The major open-source environments for DL are TensorFlow, Keras, and PyTorch, but it is a fluid situation and other contenders include Caffe, theano, mxnet (Apache), CNTK (Microsoft), DL4J, Chainer, and fast.ai. When picking an environment, one should consider the learning curve, the rate of new development, the commitment of the user community, the likelihood it will be stable and sustained, and the ecosystem of tools. How one weights these different aspects depends upon one’s purpose, applications, and background.

If we had to choose, we would select TensorFlow, partly because it is strongly supported by Google [34, 43]. The Google Brain team developed it, and they designed it to take advantage of Google’s powerful tensor processing units, or TPU. TensorFlow has a vigorous community of open-source developers, notably H2O, and it has the top ratings from Zacharias et al. [91] and from Hale (http://bit.ly/2GBa3tU). Keras and PyTorch are both good choices, and PyTorch seems to be gaining in popularity faster than Keras, but we have no specific data that confirms that impression. PyTorch was developed at Facebook and is also open source; its Python interface is popular. PyTorch is the basis for the Tesla Autopilot [70] and Uber’s Pyro [64]. Keras is an open-source DL library that can run on top of
TensorFlow, and it is also supported by Google. Keras is written in Python and was originally conceived as an interface rather than a standalone DL platform—the intent was to provide high-level abstractions that enable developers to work fairly independently of the underlying software system. But those high-level abstractions may obscure important pedagogical lessons by sweeping aside practical complexities and enforcing certain default choices.

The main problem with allowing students to use whatever software they prefer is that it becomes very difficult to compare their performance on class assignments. In this kind of course, it is important for students to learn which techniques work well in which situations, and the success of those techniques can be confounded with the choice of software environment. Performance also depends upon how long the students train their deep networks, the initialization of the parameters, and numerous technical choices in how the DL training is done. In Sect. 3.2, we provide examples of model homework assignments we constructed to put evaluations on a common footing.

Finding a good textbook is another practical problem. The bible in the field is Deep Learning by Goodfellow et al. [27]. But the book was not written for statisticians, and we find it presents cognitive headwinds. For example, we are concerned about the bias–variance trade-off and the Curse of Dimensionality, which are only lightly touched upon in the text. So our advice is that statistical educators write standalone lecture notes, which may draw heavily upon [27], but which also employ other sources. One reason is that the first five chapters introduce all the mathematical technicalities (e.g., the Moore–Penrose inverse, principal components analysis, overflow and underflow, maximum likelihood, Bayesian inference, and tensors). Later on, when these concepts are used, the text assumes that the reader recalls that material flawlessly. In our experience, it is better for mathematical concepts to be introduced on a just-in-time basis.

Statisticians also face a terminology barrier. For example, the hidden nodes in a deep neural network have activation functions, which are often logistic-like functions of a linear transformation of the data with the familiar form $\beta_0 + x^T \beta$. In the DL community, $\beta_0$ is called a bias. Other terms that are not used in the way a statistician expects are noise robustness, normalization, and hyperparameter. These are minor points, but they can cause confusion.

In this paper, we advise on how to teach DL from a statistical perspective. Further, we urge our colleagues to bring DL into our graduate programs—the students want and need this education. Section 2 lays out ways in which the computer scientist’s view and the statistician’s view do not align. Section 3 provides a detailed syllabus and two model assignments. Section 4 lists more educational resources, and Sect. 5 concludes.

2 Different Worldviews

Statisticians have different training and instincts than computer scientists. This section lists some of the disjunctions that became evident when teaching a DL graduate course in statistics.
2.1 Function Composition

One contrast between statisticians and DL computer scientists is that we generally make predictions and classifications using linear combinations of basis elements—this is the cornerstone of nearly all regression and much classification, including support vector machines, boosting, bagging, and stacking. (Nearest-neighbor methods are a rare exception.) But DL uses compositions of activation functions of weighted linear combinations of inputs. Function composition enables one to train the network through the chain rule for derivatives, which is the heart of the back-propagation algorithm.

The statistical approach generally enables more interpretability, but function composition increases the space of models that can be fit. If the activation functions in the DL networks were linear, then DL in a regression application would be equivalent to multiple linear regression, and DL in a classification application would be equivalent to linear discriminant analysis. In that same spirit, a deep variational autoencoder with one hidden layer and linear activation functions is equivalent to principal components analysis. But the use of nonlinear activation functions enables much more flexible models.

Statisticians are taught to be wary of highly flexible models—these have the potential to interpolate the training data, leading to poor generalizability. Recent work indicates that DL actually does interpolate the training data, but still performs well on test data [9].

2.2 Parameterization

Statisticians worry about overparameterization and issues related to the Curse of Dimensionality (COD), as in Hastie [33]. But DL frequently has millions of parameters. Even though DL typically trains on very large datasets, their size still cannot provide sufficient information to overcome the COD.

Statisticians often try to address overparameterization through variable selection, removing terms that do not contribute significantly to the inference. This generally improves interpretability, but often makes relatively strong use of modeling assumptions.

In contrast, DL wants all the nodes in the network to contribute only slightly to the final output and uses dropout (and other methods) to ensure that result. Part of the justification for that is a biological metaphor—genome-wide association studies find that most traits are controlled by many, many genes, each of which has only small effect.

And, of course, with the exception of Chen et al. [12], DL typically is not concerned with interpretability, which is generally defined as the ability to identify the reason that a particular input leads to a specific output. DL does devote some attention to explainability, which concerns the extent to which the parameters in deep networks can shape the output [72]. To statisticians, this is rather like assessing variable importance.
2.3 Theory

Statistics has its roots in mathematics, so theory has primacy in our community. Computer science comes from more of a laboratory science tradition, so the emphasis is upon performance.

This is not say that there is no theory behind DL; it is being built out swiftly. But much of it is done after some technique has been found to work, rather than as a guide to discovering new methodology.

One of the major theorems in DL concerns the universal approximation property. Let \( \mathcal{F} \) be the class of functions generated by a network with layer width \( m \) and non-constant, bounded, and continuous activation functions \( \sigma \). This network has the universal approximation property if for any \( \varepsilon > 0 \) and any continuous function \( f(x) \) on \([0, 1]^d\), there exists an integer \( m = m(f, \varepsilon) \), such that

\[
\inf_{g \in \mathcal{F}} \| f - g \|_{L^\infty([0,1]^d)} \leq \varepsilon.
\]

The following result is foundational in DL.

**Theorem 2.1** Neural networks with smooth activation functions that are not polynomials have the universal approximation property.

Cybenko [16] and Hornik et al. [35] independently proved an early version of this theorem for shallow feedforward neural networks with sigmoid activation functions. It was later shown in Leshno et al. [46] that the class of deep neural networks is a universal approximator if and only if the activation function is not polynomial.

2.4 Bounds on Generalization Error

One of the most important theoretical results for DL is upper bounds on its generalization error (also known as expected loss). Such bounds indicate how accurately DL is able to predict outcomes for previously unseen data. Sometimes the bounds are unhelpfully large, but in other cases they are usefully small.

The generalization error for any function \( f(x) \) is the expectation \( L(f) = \mathbb{E} [\ell(f(X), Y)] \), where the function \( \ell \) is the loss function (e.g., squared error for continuous outputs or the indicator loss function for classification). An objective of machine learning is to develop an algorithm to find functions \( \hat{f}_n(x) = \hat{f}_n(x; X_1, Y_1, \ldots, X_n, Y_n) \) based on the training data set \( \{(X_i, Y_i)\}, X \in \mathbb{R}^d, Y \in \mathbb{R} \) that predict well on unseen data. The standard approach to evaluating the generalization error \( L(\hat{f}_n) \) of the algorithm function \( \hat{f}_n(x) \) is to study the quantity \( \sup_{f \in G} |L(f) - \hat{L}(f)| \) for a suitable class of functions \( G \). Here \( \hat{L}(f) = \frac{1}{n} \sum_{i=1}^n \ell(f(X_i), Y_i) \) is the empirical risk of a function \( f \). One of the classical metrics which is used to evaluate \( \sup_{f \in G} |L(f) - \hat{L}(f)| \) and hence the generalization error of learning algorithms is the Rademacher complexity. Rademacher...
complexity for generalization error bounds has been studied by many researchers [5, 6, 26, 40, 41, 90].

Other new theoretical results about DL include Vapnik–Chervonenkis bounds, which partially address concerns previously raised about overly flexible models [76]. And Betancourt et al. [10] use symplectic geometry and dynamical systems theory to set hard limits on the rate at which a DL network can be trained.

2.5 The Double Descent Curve

In statistics, we worry about the bias–variance trade-off and know that overtraining causes performance on the test sample to deteriorate after a certain point, so the error rate on the test data is U-shaped as a function of the amount of training. But in DL, there is a double descent curve; the test error decreases, rises, then decreases again, possibly finding a lower minimum before increasing again. This is thought to be because the DL network is learning to interpolate the data.

The double descent phenomenon is one of the most striking results in modern DL research in recent years. It occurs in CNNs, ResNets, and transformers: as the number of parameters increases in a neural network, the test error initially decreases, then increases, and, just as the model approaches interpolation with approximately zero training error, the test error starts a second descent. The first switch happens in the under-parameterized regime which is consistent with the classical bias–variance trade-off in the standard statistical machine learning theory. In the over-parameterized regime, where the model complexity is large compared to the sample size, DL practitioners often claim that “bigger models are better”; c.f., Krizhevsky et al. [42], Szegedy et al. [83], Huang et al. [36]. This model-wise double descent was first proposed as a general phenomenon by Belkin et al. [9]. They demonstrated this phenomenon in simple machine learning methods as well as simple neural networks. This phenomenon was observed for different models in Opper (1995; 2001), Advani and Saxe [2], Spigler [80], and Geiger et al. [22].

Recently, Nakkiran et al. [57] show that double descent is a robust phenomenon that occurs in a variety of datasets, architectures, and optimization methods. In addition, they also demonstrate that the same double descent phenomenon can happen for training time (“epoch-wise double descent”) and dataset size (“sample-wise non-monotonicity”). One observes the epoch-wise double descent phenomenon for sufficiently large models. Holding model size fixed and training for longer also exhibits the double descent behavior.

However, for medium-sized models, as a function of training time, the test error follows the classic U-shaped curve. For small models, the test error decreases as the training time increases. Oddly, the phenomenon may lead to a regime where more data hurt. The performance first improves, then gets worse, and then improves again with increasing data size. In summary, double descent occurs not just as a function of model size, but also as a function of training time and dataset size, which means more data may surprisingly lead to worse test performance.

Based on the universal double descent phenomenon with respect to model size, training time or sample size, Nakkiran et al. [57] propose a new measure of
complexity, the effective model complexity (EMC). EMC is related to other classical complexity measures such as Rademacher complexity and VC dimension. However, neither Rademacher complexity nor VC dimension can be applied to training time double descent effects since they depend on the model and the distribution of the data, but not on the training procedure. The effect of double descent phenomenon can be avoided through optimal regularization, e.g., Nakkiran et al. [58].

The double descent is a universal phenomenon in terms of model size, training time and sample size. But complete understanding of the mechanisms and theory behind double descent in DL is an area of active research.

### 2.6 Kullback–Leibler Divergence Reversal

Suppose (as with GANs) one wants to generate complex data that resemble reality, such as fake images. One has a sample of images \(X_1, \ldots, X_n\) that is i.i.d. \(G\), but the analyst has incorrectly specified the model family as \(F_{\theta}\). The analyst seeks to estimate \(\theta\) by \(\hat{\theta}\), in order to draw samples from \(F_{\hat{\theta}}\) that seem realistic.

To a statistician, the natural approach is maximum likelihood estimation, which gives

\[
\hat{\theta} = \arg\max_{\theta} n^{-1} \sum \ln f_{\theta}(x_i) \quad (1)
\]

\[
\approx \arg\max_{\theta} \mathbb{E}_{G}[\ln f_{\theta}(x)]. \quad (2)
\]

where \(f_{\theta}(x)\) is the density of \(F_{\theta}(x)\). We note, for future use, that the maximizer does not change if one adds the entropy term for \(G\), so

\[
\hat{\theta} \approx \arg\max_{\theta} \mathbb{E}_{G}[\ln f_{\theta}(x)] - \mathbb{E}_{G} \ln g(x) \quad (3)
\]

\[
= \mathbb{E}_{G}[\ln \frac{f_{\theta}(x)}{g(x)}] \quad (4)
\]

\[
= \text{KL}(G||F_{\theta}) \quad (5)
\]

where the last term is the Kullback–Leibler divergence from \(F_{\theta}\) to \(G\).

In contrast, in DL, Zhao et al. [93] reversed the roles of \(G\) and \(F_{\theta}\) and sought

\[
\tilde{\theta} = \arg\max_{\theta} \mathbb{E}_{F_{\theta}}[\ln g(x)].
\]

Obviously, this cannot be calculated, since the analyst does not know \(G\), but it turns out there is still a way forward.

Suppose that the support of \(F_{\theta}\) is a proper subset of the support of \(G\) (in a very-high-dimensional image space). Then, \(G\) gives positive probability to pixel arrays that cannot be generated from \(F_{\hat{\theta}}\) and thus draws from \(F_{\hat{\theta}}\) will likely not resemble draws from \(G\). But the role reversal done in DL ensures that the \(\tilde{\theta}\) maximizer assigns positive probability to images or text generated by \(G\). They will seem real, although they may not explore the full range of realizations from \(G\).
To encourage wider support, [93] imposed an entropy regularization, obtaining

$$\tilde{\theta} = \arg\max_{\theta} \mathbb{E}_{F_\theta} [\ln g(x)] - \mathbb{E}_{F_\theta} [\ln f_\theta(x)]$$

(6)

$$= \arg\max_{\theta} \mathbb{E}_{F_\theta} \ln \frac{g(x)}{f_\theta(x)}$$

(7)

$$= \arg\min_{\theta} \text{KL} (F_\theta || G)$$

(8)

which is the Kullback–Leibler divergence from $G$ to $F_\theta$. The entropy term favors $F_\theta$ that have large support.

Since $F_\theta$ is unknown, the divergence cannot be calculated. But let $v(x) = \ln \frac{g(x)}{f_\theta(x)}$ and suppose one is able to easily sample from a distribution $H(z)$, which has an analytic form. Then construct the model $x = w_\theta(z)$ where $w_\theta(z)$ is a deep network and $z$ is a latent variable (the “reparameterization trick”). Now the problem is to find

$$\tilde{\theta} = \arg\max_{\theta} \mathbb{E}_{H} [g(w_\theta(z))]$$

which brings $\theta$ inside the expectation.

Suppose one knew $g(x)$ and $f_\theta(x)$ and wanted to find the optimal classifier to determine from which distribution an observation came. The optimal classifier is just the log likelihood ratio $v(x)$. Since one has samples from $g(x)$ (the data) and can sample $f_\theta(x)$ through the latent variable $z$, then one can construct a second deep network that does binary classification to determine whether an observation was drawn from $f_\theta(x)$ or from $q(x)$. This trains the new deep network to learn $v(x)$. By training both neural networks (the GAN framework), they converge to an equilibrium point at which the deep network corresponding to $f_\theta(x)$ produces outputs (e.g., images) that look like observations produced by $g(x)$.

### 2.7 Toward Uncertainty Quantification

DL models are often viewed as deterministic functions, which limits use in many applications. First, model uncertainty cannot be assessed; statisticians know this can lead to poor prediction (Gal et al. [21]). Second, most network architectures are designed through trial and error, or are based upon high-level abstractions. Thus, the process of finding an optimal network architecture for the task at hand, given the training data, can be cumbersome. Third, deep models have many parameters and thus require huge amounts of training data. Such data are not always available, and so deep models are often hampered by overfitting.

Potentially, these issues could be addressed using Bayesian statistics for uncertainty quantification (UQ). UQ builds a Gaussian process emulator for a complex simulator, which is one way to view DL networks. That approximation enables probabilistic expression of uncertainty (Gramacy [28]). It also enables fast but indirect study of architectures and training regimens. Random effects modeling makes the Bayesian
emulator resistant to overfit. Uncertainty quantification for deep learning has recently gained momentum (Zhu et al. [95], Tripathy and Bilionis [84]). The current status of this research is summarized below.

- Bayesian modeling and variational inference. The true posterior $p(w|X, Y)$ cannot usually be evaluated analytically. Instead, a variational distribution can be evaluated to approximate the posterior distribution. Variational inference is a standard technique to replace the Bayesian modeling marginalization with optimization. Compared to the optimization approaches used in DL, this method optimizes over distributions instead of point estimates. This approach preserves many of the advantages of Bayesian modeling (such as the balance between complex models and models that explain the data well) and results in probabilistic models that capture model uncertainty.

- Bayesian neural networks. These offer a probabilistic interpretation of DL models by inferring distributions over the models’ weights. They offer robustness to overfitting, uncertainty estimates, and can learn from small datasets.

- Deep Gaussian Process (DGP) Methods. The idea behind using DGP methods for non-stationary functions comes from the DL’s approximation of complexity through composition—one can create very flexible functions from compositions of simpler ones. So DGP methods with functional compositions of GPs have been developed to construct probabilistic prediction models and handle forecasting uncertainties (Lee et al. [45]).

These UQ methods enhance DL by enabling probabilistic statements of uncertainty about DL classification decisions and predictions.

### 2.8 Adversarial Learning

It is important to recognize that DL can fail badly. In particular, DL networks can be blindsided by small, imperceptible perturbations in the inputs, also known as data poisoning (Naveiro et al. [59], Vorobeychik [87]). The attackers need not even know the details of the machine learning model used by defenders (Moosavi-Dezfooli et al. [56]).

One important case is autonomous vehicles, where DL networks decide, e.g., whether the latest sequence of images implies that the car should apply its brakes. There are now many famous cases in which changing a few pixels in training data can create holes in DL network performance (Modas [55], Su et al. [81]). Symmetrically, a small perturbation of an image, such as applying a post-it note to a stop sign, can fool the DL network into classifying it has a billboard and not recognizing that the vehicle should brake (Eykholt et al. [20]).
3 Detailed Syllabus and Example Homework

The content of a DL course must depend upon the background of the students, the goal of the department, and the needs of the students. The proposed syllabus assumes the target audience are PhD graduate students in statistics, mathematics, and/or computer science. It assumes that proficiency in practice is wanted by both the department and the students, with a secondary goal of fostering some insight into the theory.

These assumptions describe the situation encountered by one of the authors in developing such a course. The class was not part of any department’s core program, but its popularity signalled the appetite for such material. Students were able to train deep networks on their laptops, but it is important to create assignments that prescribe limits on the amount of training that is done, partly to ensure comparability during grading and partly to prevent overuse of computing resources.

In a DL class, there are challenges and choices in terms of both coverage and evaluation. Our sense is that for most cases (with the possible exception of short courses) regular homework assignments that build skills in training deep networks are highly valued by the students.

3.1 Syllabus

The following syllabus and its rationale grew out of two graduate courses in DL; one was taught in the fall of 2019, and the other was taught in the summer of 2020. Our syllabus has evolved based on those experiences.

1. The first lecture should motivate the material. It should begin by listing some of the successes of DL (e.g., Tesla’s Autopilot, Google Translate, Alpha Go, its defeat of the Dota2 champions) and also cool applications such as style transfer, image reconstruction, image synthesis, object segmentation, colorization, super-resolution, and caricature creation.

   Next, one could show the improvement over time in the ImageNet classification competition and relate that to the depth of the DL network. Emphasize that DL requires lots of training data.

2. Define the perceptron and describe how it works. Define the input, output and hidden layers—a visual image is obviously helpful and ubiquitously available on the Internet. Mention the affine transformation and the activation function, but not in detail—that will get covered in the next lectures. If the audience are mostly statisticians, point out that in the DL world, the constant in the affine transformation is usually called a bias, although to us it would be an intercept. Emphasize that a deep network creates complex outcomes by function composition rather than linear modeling, as statisticians have been trained to do.

   Whether or not the class is going to do exercises (and we strongly recommend that it should), it is important for students to know some of the standard benchmark datasets. We used the MNIST digit and fashion data, the Cifar-10 data, and
the Google Street View House Numbers data (these are listed in order of increasing difficulty). Later in the course, for text applications instead of images, we used the IMDB Movie Review data. Both MNIST datasets, the Cifar10 data and the IMDB data are available at https://keras.io/api/datasets/, and the Streetview data are at http://ufldl.stanford.edu/housenumbers/. One should not assign the students datasets that are too difficult—these require lots of data, lots of time, and lots of electricity to train, which interferes with the educational objectives.

3. Compare and contrast some of the standard loss functions and activation functions. But the main content for this lecture should be the backpropagation algorithm. Define the gradient and show how the chain rule operates on compositions of function to allow estimation of the coefficients in the affine transformations in the activation functions. Introduce stochastic gradient descent.

If the class is mathematically advanced, one can describe the Universal Approximation Theorem. It reassures many statisticians that DL has theoretical underpinnings.

4. Generalize the previous lecture to tensors and discuss TPU s. This is probably a good time to talk about the enormous number of parameters in DL and introduce ideas of sparsity, regularization, the bias–variance trade-off and the Curse of Dimensionality. Discuss the surprising success of DL, given these considerations, and mention double descent.

5. Introduce initialization, weight decay, minibatches, learning rates and epochs. Explain saturation and momentum.

The following topics vary in their practicality, but if the course has a theoretical component, they should be addressed: AdaGrad, RMSProp, Nesterov momentum, second-order gradient methods, conjugate gradient descent, block coordinate gradient descent, batch renormalization and Polyak averaging (approximately in this order). This will probably take two lectures.

6. A statistical audience will want to know about the Robbins–Monro algorithm ([68]) and how it relates to DL. Even if the audience is non-statistical, a sketch of the algorithm is valuable. However, if pressed for time or focused on applications much more than theory, this topic is dispensable.

7. At some point, we recommend stepping back from the mathematics and comparing the designed experiments that are embedded into the homework assignments (see Sect. 3.2). At this point, the class has probably completed three or four assignments on the MNIST fashion data and/or the Cifar10 data. The experiments can compare the effect of activation functions, learning rates, breadth versus depth, and minibatch size on performance.

8. We recommend revisiting regularization techniques. In DL, these include weight decay, L1 penalties, quantization, quantization through k-means clustering, and quantization through Huffman coding. Discuss data augmentation, noise robustness, label smoothing, the softmax function, early stopping, bagging, stacking, and dropout. This will take two to three lectures.

If one needs to fill out the third lecture, one can introduce the computation graph. But this topic may also arise later.

9. Now do a deep dive on convolutional neural networks (CNNs). Introduce the convolution operation, talk about sparse interactions, parameter sharing, zero
(and smarter) padding, and pooling. Discuss convolution with stride and tiled convolution.

It is probably good to remind students of the successful applications of CNNs mentioned in the first lecture. This is a good motivator for a lecture on reinforcement learning.

Showing some simple computation graphs is helpful, and will feed forward into the next lesson. Covering this material will take about 2.5 hours worth of lecture time.

10. Next come recurrent neural networks (RNNs). Discuss the standard architectures and introduce backpropagation through time. Computation graphs and other visuals are essential. Describe teacher forcing, skip connections, and leaky units. Then proceed to long short-term memory, gated recurrent units, the constant error carousel, augmented RNNs (including neural Turing machines) and echo state networks.

RNNs are prone to vanishing and exploding gradients, so although that also can arise in CNNs, we recommend covering it here. Mention gradient clipping for exploding gradients. In our experience, this is about 1.5 hours of material.

11. RNNs are dominant when dealing with text or language, so the homework on this unit should employ the IMDB database. When covering text examples, it is helpful to describe Word2Vec, Seq2Seg, and Google’s BERT. The concept of attention arises naturally.

Recursive neural networks (RvNNs) are not as prominent as other DL techniques, but a short description is appropriate here.

12. Variational autoencoding is the next topic. This can be motivated to a statistical audience by pointing out that a shallow network with a single hidden layer and linear activation functions is equivalent to principal components analysis. Describe the denoising autoencoder. There is hardly any new theory in this unit, but there are a number of compelling applications.

For the homework in this segment, we recommend that the class assignments compare information loss across a range of architectures for simulated data in which the signal-to-noise ratios are known. Insofar as we are aware, little is known about how to pick the right autoencoding architecture to match an application with specific characteristics.

13. The last main unit concerns generative adversarial networks (GANs). After introducing the key ideas and the game-theoretic underpinning, one can proceed to conditional GANs, ancestral sampling, and diffusion inversion. Emphasize that the GANs are learning a low-dimensional manifold on which the images (or other material, such as text) concentrate. Wasserstein GANs (WGANs) should be mentioned and perhaps covered in some detail. WGANs show more stability during training than regular GANs and use a loss function that better reflects image quality (Gulranjani [30]).

For a statistical audience, the work in Zhao et al. [93] nicely connects ideas in maximum likelihood estimation to GANs. Since it is relatively new, Sect. 2.6 reviews it.
This syllabus lays out one path through DL material, emphasizing connections to statistical thinking. If time permits, the instructor may want to include more material adjacent to DL, such as empirical risk bounds, or other statistical approaches to classification and text data.

### 3.2 Homework

If all students are given the same assignment, say to train a deep network to classify the Google Streetview data, and if all are using the same programming environment, then one must be very prescriptive to ensure that the results are comparable. The Tesla automated driver DL system took 70,000 GPU hours to train (see Autopilot AI, [https://www.tesla.com/autopilotAI](https://www.tesla.com/autopilotAI)), and we do not want students to be marked on the basis of how long they let their programs run. So one must specify the number of training epochs, the architecture of their networks, and many other details.

The MNIST digit data is quite easy to classify well, so it is a good starting point for homework. But later assignments should be more difficult in order to illustrate realistic DL performance.

We found it interesting to set up the homework as a designed experiment. Specifically, for a large class, we recommend that triplets of students be given the same assignment, and the assignments should vary in ways that study some of the DL strategies. This enables an ANOVA with three replicates for each combination of factor levels.

For example, if the students are not all using the same programming environment, then the environment might be one of the design factors, so students can compare, say, PyTorch to TensorFlow to Keras. Similar designed experiments can study the impact of different architectures, learning rates, and dropout protocols. Such experiments can help students understand the impact of different decisions that are typically made when training a DL network. (Note: Typically the results of such assignments will be error rates, which are binomial random variables; therefore, we recommend using the arcsine square-root transformation before fitting the ANOVA.)

In our class, the first project was to fit a fully connected feedforward network to the MNIST digit data. There were to be 3 hidden layers, with each layer having 512 nodes. Students whose ID number ended in a 0 or 1 (or 2, 3, or 4, 5 or 6, 7 or 8, 9) used the ReLU activation function (or sigmoidal, tanh, linear, or leaky ReLU, respectively), and students whose terminal ID digit was even used minibatches of size 100 for two epochs and those that were odd used minibatch sizes 200 for one epoch (so the total training effort is equivalent). The loss function was Kullback–Leibler divergence. Students reported results classification accuracy and training time.

As a slightly more advanced example, one might direct students to fit a fully connected feedforward network with four hidden layers, where each hidden layer has 1,024 nodes, to the Google Streetview data. Students whose ID number ends in a 0 or 1 (or 2, 3, or 4, 5 or 6, 7 or 8, 9) should use a learning rate of 0.8 (or 0.85, 0.9, 0.95, or 0.99, respectively), and students whose terminal ID digit is even use the cross-entropy loss and those that are odd use 0-1 loss. All minibatches should be of...
size 200, and the network should be trained for ten epochs. Students should report classification accuracy and training time.

In general, even more prescriptive detail will be needed to ensure reasonable comparability among the graded homework.

4 Review of DL Resources

Numerous DL resources and courses have been developed and are available online. The question is where to start. We summarize the most popular DL resources to provide a learning path for students who are new to DL and also to those who want to explore it further.

4.1 Online Courses

There are many DL courses and tutorials available online. The most popular ones include:

- The Stanford University Online Open Courses.
  - CS230: Deep Learning (https://cs230.stanford.edu/)
  - CS231n: Convolutional Neural Networks for Visual Recognition (http://cs231n.stanford.edu/)
  - CS224d: Deep Learning for Natural Language Processing (http://web.stanford.edu/class/cs224n/).

  This series of courses was compiled by deep learning experts, such as Fei-Fei Li, Andrej Karpathy, Samy Bengio, Tom Dean, Andrew Ng, and Richard Socher. The courses offer video explanations of topics including logistic regression and regularization of the cost function, vector implementation, polynomial regression, all from a DL perspective. These courses are designed for DL beginners.

- A Deep learning specialization course series by Andrew Ng on Coursera (https://www.deeplearning.ai/deep-learning-specialization/). This is a suite of five courses on DL, starting with starter course and covering all major types of DL networks. This series provides an excellent in-depth introduction for beginners to DL models. There are exercises on how to use these tools.

- The Deep Learning Nanodegree by Udacity. It is a popular training course which is designed in collaboration with leading AI experts (https://www.udacity.com/course/deep-learning-nanodegree--nd101). This course is offered by a team of instructors who are AI engineers and industry experts with a notable emphasis on hands-on projects and learning by doing. Each student will need to complete a portfolio of five real-world DL projects during the course to earn a nanodegree certification. The course also provides interactive mentor feedback to assist student learning. The course is not free, but it is designed for students who want to improve practical problem-solving skills and learn hands-on experience that prepares them to work in AI tech companies.
• DataCamp: Interactive Learning Course (https://learn.datacamp.com/) is an emerging platform that provides interactive training courses on data science and programming. All the courses are designed to let students learn by doing. They apply each lesson immediately and get instant feedback. The DL courses target beginners and teach how to implement different neural network models quickly in an interactive learning environment with real-time feedback.

• Simplilearn’s Deep Learning Course (with Keras & TensorFlow) Certification Training (https://www.simplilearn.com/deep-learning-course-with-tensorflow-training) is a popular platform to teach DL. It is led by experienced trainers who use many real-life examples. An important part of this teaching platform is that it provides well-organized prerequisite courses.

4.2 Books

There are many DL books. Some of the best include:

• Deep Learning by Goodfellow, Bengio and Courville [27]. It is aimed at people who have no knowledge of deep learning, but are mathematically sophisticated. It introduces DL methods and concepts, as well as advanced DL topics, including research directions and practical applications.

• Deep Learning with Python by Chollet [14]. This book is written by the inventor of Keras inventor and introduces DL using Python and the Keras library. It balances theory and coding with intuitive explanations and practical examples.

• Hands-On Machine Learning with Scikit-Learn and TensorFlow by Geron [24]. This is a popular book that first employs Scikit-Learn to introduce fundamental machine learning tasks and employs Keras and TensorFlow to learn deep neural networks. It uses examples to build understanding of the concepts and tools and has practice exercises in each chapter.

• Deep Learning Tutorial was developed by the LISA lab at University of Montreal. It is a concise tutorial. The book http://deeplearning.net/tutorial/deeplearning.pdf emphasizes use of the Theano library for building DL models in Python. See https://github.com/lisa-lab/DeepLearningTutorials.

• Neural Network Design (2nd Edition) by Hagan et al. [32]: This book provides a clear and detailed survey of basic neural network architectures and learning rules. The authors emphasize the mathematical analysis of networks, methods for training networks, and application of networks to practical engineering problems in pattern recognition, signal processing, and control systems.

4.3 Popular Online Resources

In addition to courses and books, there are also resources available online to learn about DL, real-world applications, and recent advances. Some of the most active online resources are:
• Github: There are many educational resources and open-source projects on Github. One can access many machine learning and deep learning projects and example implementations in Python and other programming languages. See https://github.com/topics/deep-learning/.

• Quora: A great resource for AI and machine learning. Many of the top researchers answer questions on the site https://www.quora.com/, and it provides an active platform to share knowledge of DL-related topics. Under the Quora Deep Learning feed https://www.quora.com/topic/Deep-Learning/, one can check a curated list of questions discussed by the community.

• Reddit: There is an active DL community on Reddit. One can follow subreddits on machine learning https://www.reddit.com/r/MachineLearning/ and Deep Learning https://www.reddit.com/r/deeplearning/ to keep up with the latest news and research.

• Blogs: There are a number of active blogs that post consistently on AI-related topics with original materials. The top viewed ones include DeepMind Blog https://deepmind.com/blog, OpenAI Blog https://openai.com/blog/, Machine Learning Mastery https://machinelearningmastery.com/, distill.pub (a modern medium for presenting research), Data Science Central https://www.dataciencecentral.com/, Towards Data Science https://towardsdatascience.com/.

4.4 Developing Programming Skills

Python will remain a dominant programming language in DL, and Jupyter notebooks simplify many tasks. But students should expect to learn many new languages and environments and tools over the course of their careers. Statisticians are most comfortable with R and SAS, but these are not much used in DL. And the traditional statistical curriculum does not usually teach DL programming, for many reasons.

As a result, most students will need to teach themselves the programming environments they need to master. Fortunately, there are online options:

• CheckiO is a gamified website containing programming tasks that can be solved in Python 3.

• Codecademy provides lessons on Python.

• Code The Blocks combines Python programming with a 3D environment where one places blocks to construct structures. It also comes with Python tutorials that teach one to create progressively elaborate 3D structures.

• Computer Science Circles has 30 lessons, 100 exercises, and a message system where one can ask for help. Teachers can use it with their students. It is available in English, Dutch, French, German, and Lithuanian.

• DataCamp Python Tutorial is unlike most other Python tutorials. It lasts four hours and focuses on Python for Data Science. It has 57 interactive exercises and 11 videos.

• Finxter tests and trains with 300 handpicked Python puzzles.

• How to Think Like a Computer Scientist: Interactive Edition is a re-imagination of the Downey et al. [18] Elkbook with visualizations and audio explanations.
For students with little programming experience, several books can be useful. These books can be purchased online, but also have free electronic versions.

- **Automate the Boring Stuff with Python: Practical Programming for Total Beginners** by Sweigart [82] is written for office workers, students, administrators, and anyone who uses a computer to learn how to code small, practical programs.
- **How To Think Like a Computer Scientist** by Downey et al. [18] is a classic open-source Python book. It was updated to Python 3 by Peter Wentworth.

5 Real-World Application Areas and Examples

We describe two interdisciplinary application areas (medicine and astronomy) that use DL but which have deep statistical roots. We encourage DL courses to motivate students with real-world examples and projects. We further urge traditional statistical consulting classes to be re-engineered to include DL consulting experiences, of the kind described in the following examples.

In medicine, previous landmark studies have addressed a variety of classification tasks to detect patterns from image data sets using CNNs (Ardia et al. [4], Coudray et al. [15], Esteva, et al. [19], Bejnordi et al. [8], Gulshan er al. [31]) or an ensemble of CNNs (McKinney et al. [54]). The studies apply DL to the various medical image data sets for diagnostic or screening purposes. Students who know traditional classification methods in statistics can compare the performances on these tasks.

Recent work uses DL to predict patients’ prognoses, including survival, recurrence, hospitalization, and treatment complications. Studies have built DL-based predictive models using RNNs (Rajkomar et al. [66], Tomasev et al. [85]) or feed-forward networks with fully connected layers (Lee et al. [44]). Long short-term memory architecture has been used to automate natural language processing to extract clinically relevant information from electronic health records (Liang et al. [48]). Probabilistic prediction and quantification of uncertainties are crucial topics to address, but the DL community spends less attention on this aspect than statisticians do.

Patients’ electronic health records are usually longitudinal data with time-to-event endpoints. To statisticians, understanding the nature of the data generation mechanisms and statistical modeling is key. Missing data problems arise frequently in these records. Strategies to deal with missing data need to be tailored to the specific problem. But in DL, missing data are handled in rudimentary ways.

In astronomy, DL has become popular with exoplanet detection, and detection of gravitational waves using LIGO-Virgo data. DL will be widely used by the Rubin Observatory Legacy Survey of Space and Time (LSST).

NASA’s Kepler mission collected over 150,000 light curves on brightness of stars in our galaxy with the aim of detecting planets outside our solar system. However, Earth-sized planets orbiting Sun-like stars are on the very edge of the mission’s detection sensitivity. DL is being used to classify potential planet signals (Shallue and Vanderburg [77]) and thus determine the frequency of Earth-sized planets.
CNNs are trained to predict whether a given signal is an exoplanet or a false positive caused by astrophysical or instrumental phenomena.

A new era dawned in 2015 with the detection of gravitational waves (GWs). The first detection of GW150914 by the Laser Interferometer Gravitational-Wave Observatory (LIGO) was from the merger of two black holes, each about 30 times the mass of our sun. But such events are rare, and DL is data hungry, so the majority of the DL work in this area is based on simulated data. George and Huerta [23] take advantage of scientific simulations to produce the data for needed for training. They focus on labeled data obtained from physics simulations (with domain knowledge) to train CNNs to detect signals embedded in noise and also estimate multiple parameters of the source. Li et al. [47] use DL to detect GW events based on simulated signals contaminated with white Gaussian noise. It is a classification problem, and the method is particularly valuable when the potential GW signal shapes are too complex to be characterized. Statistical knowledge is essential in creating faithful simulations.

The forthcoming LSST should detect \( \sim 300 \text{M} \) quasars out of the \( \sim 40 \text{B} \) objects in the all-sky survey, but experts expect to confidently identify only about 10% of them. This is in contrast to the Sloan Digital Sky Survey (SDSS) dataset where about 98% can be identified. While the true situation is somewhere between 10% and 98%, the cause of the problem is that the training set for quasars is based on bright objects, while LSST will see much fainter objects. If fainter quasars were to reside in the same location as brighter quasars in the 4-dimensional SDSS color space, then the bright training set would work. But this is not true: fainter quasars are typically more distant with higher redshift, causing different emission lines to enter/exit different bands; quasars probably have intrinsically different properties both at earlier cosmic times (called ‘cosmic evolution’) and at lower luminosities (where host galaxy contamination becomes important).

This problem arises in many applications, where the training sets will be based on certain samples which, if used with conventional classification procedures, will recover only biased subsamples of the members of the class. This kind of bias is well understood by statisticians and should be taught in DL courses.

### 6 Final Remarks

It is imperative that statisticians be involved in DL. We can learn a lot, but also contribute much. However, involvement requires us to learn and teach this material, which poses practical, pedagogic and technical challenges.

This paper proposes partial solutions to many of those challenges. We hope it will help teachers who are trying to extend their curricula, and students who are eager to learn this new field.

Teaching DL requires us to find new ways to think and to deliver educational content. We believe that the statistics profession is up for the challenge.

**Acknowledgements** We thank deep learning working group at the Statistical and Applied Mathematical Sciences Institute (SAMSI), USA, for helpful discussions. This work was supported by National...
Science Foundation Grant DMS-1638521. Hyunsoon Cho’s work was partially supported by the National Research Foundation of Korea Grant (No. 2020R1A2C1A01011584) funded by the Korea Ministry of Science and ICT. Hailin Sang’s work was partially supported by the Simons Foundation Grant (No. 586789). Shouyi Wang’s work was partially supported by the National Science Foundation Grant ECCS-1938895.

References

1. Abbas Q, Ibrahim MEA, Jaffar MA (2019) A comprehensive review of recent advances on deep vision systems. Artif Intell Rev 52:39–76
2. Advani MS, Saxe AM (2020) High-dimensional dynamics of generalization error in neural networks. Neural Netw. https://doi.org/10.1016/j.neunet.2020.08.022
3. Ahmed Z, Mohammed K, Zeeshan S, Dong X (2020) Artificial intelligence with multi-functional machine learning platform development for better healthcare and precision medicine. Database, 1–5 2020. https://doi.org/10.1093/database/baa010
4. Ardila D, Kiraly AP, Bharadwaj S, Choi B, Reicher JJ, Peng L, Tse D, Etemadi M, Ye W, Corrado G, Naidich DP, Shetty S (2019) End-to-end lung cancer screening with three-dimensional deep learning on low-dose chest computed tomography. Nat Med 25(6):954–961
5. Bartlett PL, Boucheron S, Lugosi G (2002) Model selection and error estimation. Mach Learn 48:85–113
6. Bartlett P, Foster DJ, Telgarsky MJ (2017) Spectrally-normalized margin bounds for neural networks. Adv Neural Inf Process Syst 6241–6250
7. Bartlett PL, Mendelson S (2002) Rademacher and gaussian complexities: risk bounds and structural results. J Mach Learn Res 3(3):463–482
8. Bejnordi EB, Veta M, van Diest PJ, van Ginneken B, Karssemeijer N, Litjens G, van der Laak JAWM, and the Camelyon Consortium (2017) Diagnostic assessment of deep learning algorithms for detection of lymph node metastases in women with breast cancer. JAMA 318(22):2199–2210
9. Belkin M, Hsu D, Ma S, Mandal S (2019) Reconciling modern machine learning and the bias-variance trade-off. PNAS 116(32):15849–15854
10. Betancourt M, Jordan MI, Wilson AC On symplectic optimization. arXiv preprint, pp 1–20. arXiv:1802.03653.pdf
11. Bircanoglu C, Atay M, Beser F, Genc O, Kiztrak MA (2018) RecycleNet: intelligent waste sorting using deep neural networks. IEEE international symposium on innovations in intelligent systems and applications, pp 1–7. https://ieeexplore.ieee.org/document/8466276
12. Chen C, Li O, Tao D, Barnett A, Rudin C, Su JK (2019) This looks like that: deep learning for interpretable image recognition. Adv Neural Inf Process Syst 8930–8941
13. Chen LC, Papandreou G, Kokkinos I, Murphy K, Yuille AL (2018) Deeplab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. IEEE Trans Pattern Anal Mach Intell 40:834–848
14. Chollet F (2018) Deep learning with python, vol. 361. New York, Manning
15. Coudray N, Ocampo PS, Sakellarooulos T, Narula N, Snuderl M, Fenyo D, Moreira AL, Razavian N, Tsirigos A (2018) Classification and mutation prediction from non-small cell lung cancer histopathology images using deep learning. Nat Med 24(10):1559–1567
16. Cybenko G (1989) Approximations by superpositions of sigmoidal functions. Math Control Signals Syst 2(4):303–314
17. Dai Y, Wang G (2018) A deep inference learning framework for healthcare. Pattern recognition letters, pp 1–9. https://doi.org/10.1016/j.patrec.2018.02.009
18. Downey A, Wentworth P, Elkner J, Meyers C (2016) How to think like a computer scientist: learning with python 3. Green Tea Press
19. Esteva A, Kuprel B, Novoa RA, Ko J, Swetter SM, Blau HM, Thrun S (2017) Dermatologist-level classification of skin cancer with deep neural networks. Nature 542(7639):115–118
20. Eykholt K, Evtimov I, Fernandes E, Li B, Rahmati A, Xiao C, Prakash A, Kohno T, Song D (2018) Robust physical-world attacks on deep learning visual classification. Proceedings of the IEEE conference on computer vision and pattern recognition 1625–1634
21. Gal Y, Islam R, Ghahramani Z (2017) Deep Bayesian active learning with image data. arXiv preprint, pp 1–10. arXiv:1703.02910.pdf
22. Geiger M, Spigler S, d’Ascoli S, Sagun L, Baiti-Jesi M, Biroli G, Wyart M (2019) Jamming transition as a paradigm to understand the loss landscape of deep neural networks. Phys Rev E 100(1)
23. George D, Huerta EA (2018) Deep neural networks to enable real-time multimessenger astrophysics. Phys Rev D 97:044039 (23pp). https://doi.org/10.1103/PhysRevD.97.044039
24. Geron A (2017) Hands-on machine learning with scikit-learn and tensorflow. O'Reilly Media
25. Gheisari M, Wang G, Bhuiyan MZA (2017) A survey on deep learning in Big Data. Proceedings of the IEEE international conference on embedded and ubiquitous computing, pp 1–8. https://ieeexplore.ieee.org/document/8005992
26. Golowich N, Rakhlin A, Shamir O (2018) Size-independent sample complexity of neural networks. PMLR 75:297–299
27. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press, Cambridge
28. Gramacy RB (2020) Surrogates: Gaussian process modeling, design, and optimization for the applied sciences. CRC Press, Boca Raton
29. Grigorescu S, Trasnea B, Cocias T, Macesanu G (2020) A survey of deep learning techniques for autonomous driving. J Field Robot 37:362–386
30. Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V, Courville A (2017) Improved Training of Wasserstein GANs. arXiv preprint. arXiv:1704.00028
31. Gulshan V, Peng L, Coram M, Stumpe MC, Wu D, Narayanaswamy A, Venugopalan S, Widner K, Madams T, Cuadros J, Kim R, Raman R, Nelson PC, Mega JL, Webster DR (2016) Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. JAMA 316(22):2402–2410
32. Hagan MT, Demuth HB, Beale MH, Jess OD (2014) Neural network design, 2nd edn. Martin Hagan
33. Hastie T, Tibshirani R, Friedman J (2009) The elements of statistical learning: data mining, inference, and prediction. Springer Science & Business Media
34. Heller M (2018) TensorFlow review: the best deep learning library gets better. InfoWorld. https://www.infoworld.com/article/3250826/tensorflow-review-the-best-deep-learning-library-gets-better.html
35. Hornik K, Stinchcombe M, White H (1989) Multilayer feedforward networks are universal approximators. Neural Netw 2(5):359–366
36. Huang H, Cheng Y, Bapna A, Firat O, Chen MX, Chen D, Lee H, Ngiam J, Le QV, Wu Y, Chen Z (2019) Gpipe: efficient training of giant neural networks using pipeline parallelism. NIPS
37. Huang H, Yu PS, Wang C (2018) An introduction to image synthesis with generative adversarial nets. arXiv preprint, pp 1–17. arXiv:1803.04469.pdf
38. Ignatov A (2018) Real-time human activity recognition from accelerometer data using convolutional neural networks. Appl Soft Comput 62:915–922
39. Kaushal M, Khehra B, Sharma A (2018) Soft computing based object detection and tracking approaches: state-of-the-art survey. Appl Soft Comput 70:423–464
40. Kolchinskii V (2001) Rademacher penalties and structural risk minimization. IEEE Trans Inf Theory 47(5):1902–1914
41. Kolchinskii V, Panchenko D (2000) Rademacher processes and bounding the risk of function learning. High Dimensional Probab II:443–459
42. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. Adv Neural Inf Process Syst 1097–1105
43. Lardinois F (2019) Google launches TensorFlow Enterprise with long-term support and managed services. Tech Crunch. https://techcrunch.com/2019/10/30/google-launches-tensorflow-enterprise-with-long-term-support-and-managed-services
44. Lee CK, Hofer I, Gabel E, Baldi P, Cannesson M (2018) Development and validation of a deep neural network model for prediction of postoperative in-hospital mortality. Anesthesiology 129(4):649–662
45. Lee J, Bahri Y, Novak R, Schoenholz S, Pennington J, Sohl-Dickstein J (2017) Deep neural networks as Gaussian processes, arXiv preprint. arXiv:1711.00165
46. Leshno M, Lin VY, Pinkus A, Schocken S (1993) Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. Neural Netw 6(6):861–867
47. Li X, Yu W, Fan X, Babu GJ (2020) Some optimizations on detecting gravitational wave using convolutional neural network. Frontiers of physics. 15, Article number: 54501. https://doi.org/10.1007/s11467-020-0966-4
48. Liang H, Tsui BY, Ni H, Valentim CCS, Baxter SL, Liu G, Cai W, Kermay DS, Sun X, Chen J, He L, Zhu J, Tian P, Shao H, Zheng L, Hou R, Hewett S, Li G, Liang P, Zang X, Zhang Z, Pan L, Cai H, Ling R, Li S, Cui Y, Tang S, Ye H, Huang X, He W, Liang W, Zhang Q, Jiang J, Yu W, Gao J, Ou W, Deng Y, Hou Q, Wang B, Yao C, Liang Y, Zhang S, Duan Y, Zhang R, Gibson S, Zhang CL, Li O, Zhang ED, Karin G, Nguyen N, Wu X, Wen C, Xu J, Xu W, Wang B, Wang W, Li J, Pizzato B, Bao C, Xiang D, He W, He S, Zhou Y, Haw W, Goldbaum M, Tremoulet A, Hsu CN, Carter H, Zhu L, Zhang K, Xia H (2019) Evaluation and accurate diagnoses of pediatric diseases using artificial intelligence. Nat Med 25(3):433–438

49. Loh BCS, Then PHH (2017) Deep learning for cardiac computer aided diagnosis: benefits, issues and solutions. MHealth. 3:45. https://doi.org/10.21037/mhealth.2017.09.01

50. Luckow A, Cook M, Ashcraft N, Weill E, Djerekarov E, Vorster, B (2017) Deep learning in the automotive industry: applications and tools. Proceedings of the IEEE international conference on big data, pp 3759–768. https://ieeexplore.ieee.org/document/7841045

51. Luong MT, Manning CD (2016) Achieving open vocabulary neural machine translation with hybrid word-character models. arXiv preprint, pp 1–11. arXiv:1604.00788.pdf

52. Luong MT, Pham H, Manning CD (2015) Effective approaches to attention-based neural machine translation. arXiv preprint, pp 1–11. arXiv.org/pdf/1508.04025.pdf

53. Ma X, Yu H, Wang Y, Wang Y (2015) Large-scale transportation network congestion evolution prediction using deep learning theory. PLoS ONE 10. https://doi.org/10.1371/journal.pone.0119044

54. McKinney SM, Sieniek M, Godbole M, Godwin J, Antropova N, Ashrafian H, Back T, Chesnus M, Corrado GC, Darzi A, Etemadi M, Garcia-Vicente F, Gilbert FJ, Halling-Brown M, Hassabis D, Jansen S, Karthikesalingam A, Kelly CJ, King D, Ledsam JR, Melnick D, Mostofi H, Peng L, Reicher JJ, Romera-Paredes B, Sidebottom R, Suleyman M, Tse D, Young KC, De Fauw J, Shetty S (2020) International evaluation of an AI system for breast cancer screening. Nature 577(7788):89–94

55. Modas A, Moosavi-Dezfooli SM, Frossard P (2019) Sparsefool: a few pixels make a big difference. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 9087–9096

56. Moosavi-Dezfooli SM, Fawzi A, Fawzi O, Frossard P (2017) Universal adversarial perturbations. Proceedings of the ieee conference on computer vision and pattern recognition, pp 1765–1773

57. Nakirian P, Kaplun G, Bansal Y, Yang T, Barak B, Sutskever I (2020) Deep double descent: where bigger models and more data hurt. ICLR 2020 conference

58. Nakkiran P, Venkat P, Kakade S, Ma T (2020) Optimal regularization can mitigate double descent. arXiv:2003.01897

59. Naveiro R, Redondo A, Insua DR, Ruggeri F (2019) Adversarial classification: an adversarial risk analysis approach. Int J Approx Reason 113:133–148

60. Noda K, Yamaguchi Y, Nakadai K, Okuno HG, Ogata T (2015) Audio-visual speech recognition using deep learning. Appl Intell 42:722–737

61. Nweke HF, Teh YW, AI-garadi MA, Alo UR (2018) Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: state of the art and research challenges. Expert Syst Appl 105:233–261

62. Opper M (1995) Statistical mechanics of learning: generalization. The handbook of brain theory and neural networks 922–925

63. Opper M (2001) Learning to generalize. Front Life 3(part 2):763–775

64. Peng H, Obermeyer F (2019) Modeling censored time-to-event data using Pyro, an open source probabilistic programming language. Uber Engineering. https://eng.uber.com/modeling-censored-time-to-event-data-using-pyro

65. Perrotta F, Parry T, Neves LC (2017) Application of machine learning for fuel consumption modelling of trucks. IEEE international conference on big data, pp 1–6. https://ieeexplore.ieee.org/document/8258382

66. Rajkomar A, Oren E, Chen K, Dai AM, Hajaj N, Hardt M, Liu PJ, Liu X, Marcus J, Sun M, Sundberg P, Yee H, Zhang K, Zhang Y, Flores G, Duggan GE, Irvine J, Le Q, Litsch K, Mossin A, Tan- suwan J, Wang Wexler J, Wilson J, Ludwig D, Volchenboum SL, Chou K, Pearson M, Madabushi S, Shah NH, Butte AJ, Howell MD, CU, Corrado GS, Dean J (2018) Scalable and accurate deep learning with electronic health records. NPJ Digit Med 1(18)

67. Razzak MI, Naz S, Zaib A (2018) Deep learning for medical image processing: overview, challenges and the future. Classif BioApps 26:323–350

68. Robbins H, Monro S (1951) A stochastic approximation method. Ann Math Stat 22(3):400
69. Rudin F, Li GJ, Wang K (2017) An algorithm for power system fault analysis based on convolutional deep learning neural networks. Int J Res Educ Sci Methods 5:11–18
70. Sagar R (2019) How Tesla uses PyTorch. Analytics India Magazine. https://analyticsindiamag.com/tesla-pytorch-self-driving-computer-vision-karpathy-elon-musk-ai
71. Sakr GE, Mokbel M, Darwich A, Khneisser MN, Hadi A (2016) Comparing deep learning and support vector machines for autonomous waste sorting. IEEE international multidisciplinary conference on engineering technology, pp 1–6. https://ieeexplore.ieee.org/document/7777453
72. Samek W, Wiegand T, Müller K-R (2017) Explainable artificial intelligence: understanding, visualizing and interpreting deep learning models. arXiv preprint, pp 1–8. arXiv:1708.08296.pdf
73. See A, Luong MT, Manning CD (2016) Compression of neural machine translation models via pruning. arXiv preprint, pp 1–11. arXiv:1606.09274.pdf
74. Serizel RGD (2016) Deep-neural network approaches for speech recognition with heterogeneous groups of speakers including children. Nat Lang Eng 1:1–26
75. Sethy H, Patel A, Padmanabhan V (2015) Real time strategy games: a reinforcement learning approach. Procedia Comput Sci 54:257–264
76. Shalev-Shwartz S, Ben-David S (2014) Understanding machine learning: from theory to algorithms. Cambridge University Press, Cambridge
77. Shallue CJ, Vanderburg A (2018) Identifying exoplanets with deep learning: a five-planet resonant chain around Kepler-80 and an eighth planet around Kepler-90. Astron J 155: 94 (21pp)
78. Shang C, You F (2019) Data analytics and machine learning for smart process manufacturing: recent advances and perspectives in the Big Data era. Engineering 5:1010–1016
79. Soniya SP, Singh L (2015) A review on advances in deep learning. Proceedings of IEEE workshop on computational intelligence: theories, applications and future directions, pp 1–6. https://ieeexplore.ieee.org/document/7495514
80. Spigler S, Geiger M, d’Ascoli S, Sagun L, Biroli G, Wyatt M (2019) A jamming transition from under-to over-parametrization affects loss landscape and generalization. J Phys A Math Theor 52(47)
81. Su J, Vargas DV, Sakurai K (2019) One pixel attack for fooling deep neural networks. IEEE Trans Evol Comput 23:828–841
82. Sweeney A (2019) Automate the boring stuff with python: practical programming for total beginners. No Starch Press, San Francisco
83. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. CVPR
84. Tripathy RK, Bilioni S (2018) Deep UQ: learning deep neural network surrogate models for high dimensional uncertainty quantification. J Comput Phys 375:565–588
85. Tomasev N, Golor X, Rae JW, Zielinski M, Ashkan H, Saraiva A, Mottram A, Meyer C, Ravuri S, Protsyuk I, Connell A, Hughes CO, Karthikesalingam A, Cornebise J, Montgomery H, Rees G, Laing C, Baker CR, Peterson K, Reeves R, Hassabis D, King D, Suleyman M, Back T, Nielson C, Ledsam JR, Mohamed S (2019) A clinically applicable approach to continuous prediction of future acute kidney injury. Nature 572(7767):116–119
86. Vinyals O, Babuschkin I, Czarnecki WM, Mathieu M, Dudzik A, Chung J, Choi DH, Powell R, Ewalds T, Georgiev P, Oh J, Horgan D, Kroiss M, Danihelka I, Huang A, Sifre L, Cai T, Agapiou JP, Jaderberg M, Vezhnevets AS, Leblond R, Pohlen T, Dalibard V, Budden D, Sulsky Y, Molloy J, Paine TL, Gulcehre C, Wang Z, Pfaff T, Wu Y, Yogatama RRD, Wunsch D, McKinney K, Smith O, Schaul T, Lillicrap T, Kavukcuoglu K, Hassabis D, Apps C, Silver D (2019) Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature 575:350–354
87. Vorobeychik Y, Kantarcioglu M (2018) Adversarial machine learning. Synth Lect Artif Intell Mach Learn 12:1–169
88. Wang L, Sng D (2015) Deep learning algorithms with applications to video analytics for a smart city: a survey. arXiv preprint, pp 1–8. https://arxiv.org/pdf/1512.03131.pdf
89. Wu Z, Swietozanski P, Veaux C, Renals S (2015) A study of speaker adaptation for DNN-based speech synthesis. Proceedings of the interspeech conference 1–5
90. Xu Y, Wang X (2018) Understanding weight normalized deep neural networks with rectified linear units. Adv Neural Inf Process Syst 130–139
91. Zacharias J, Barz M, Sonntag D (2018) A survey on deep learning toolkits and libraries for intelligent user interfaces. arXiv preprint, pp 1–10. arXiv:1803.04818.pdf
92. Zhang Q, Yang LT, Chen Z, Li P (2018) A survey on deep learning for Big Data. Inf Fusion 42:146–157
93. Zhao M, Cong Y, Dai S, Carin L (2020) Bridging maximum likelihood and adversarial learning via $\alpha$-divergence. Association for the Advancement of Artificial Intelligence, pp 1–19. arXiv:2007.06178.pdf

94. Zhu G, Jiang B, Tong L, Xie Y, Zaharchuk G, Wintermark M (2019) Applications of deep learning to neuro-imaging techniques. Front Neurol 10:869. https://doi.org/10.3389/fneur.2019.00869

95. Zhu Y, Zabaras N, Koutsourelakis P-S, Perdikaris P (2019) Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. J Comput Phys 394:56–81

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.