ANNUAL REVIEWS

Annual Review of Statistics and Its Application

A Brief Tour of Deep Learning from a Statistical Perspective

Eric Nalisnick,[1] Padhraic Smyth,[2] and Dustin Tran[3]

[1]Informatics Institute, University of Amsterdam, Amsterdam, Netherlands; email: e.t.nalisnick@uva.nl

[2]Departments of Computer Science and Statistics, University of California, Irvine, California, USA; email: smyth@ics.uci.edu

[3]Google Brain, Mountain View, California, USA; email: trandustin@google.com

Annu. Rev. Stat. Appl. 2023.10:219-246. Downloaded from www.annualreviews.org
Access provided by University of California - Irvine on 09/10/23. See copyright for approved use.

ANNUAL REVIEWS CONNECT

www.annualreviews.org

- Download figures
- Navigate cited references
- Keyword search
- Explore related articles
- Share via email or social media

Annu. Rev. Stat. Appl. 2023. 10:219–46

The *Annual Review of Statistics and Its Application* is online at statistics.annualreviews.org

https://doi.org/10.1146/annurev-statistics-032921-013738

Copyright © 2023 by the author(s). This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited. See credit lines of images or other third-party material in this article for license information.

Keywords

deep learning, neural networks, pattern recognition, optimization

Abstract

We expose the statistical foundations of deep learning with the goal of facilitating conversation between the deep learning and statistics communities. We highlight core themes at the intersection; summarize key neural models, such as feedforward neural networks, sequential neural networks, and neural latent variable models; and link these ideas to their roots in probability and statistics. We also highlight research directions in deep learning where there are opportunities for statistical contributions.

OPEN ACCESS CC BY

Annu. Rev. Stat. Appl. 2023.10:219-246. Downloaded from www.annualreviews.org
Access provided by University of California - Irvine on 09/10/23. See copyright for approved use.

1. INTRODUCTION

In recent years, the umbrella of techniques known as deep learning (DL) (Hinton & Salakhutdinov 2006, LeCun et al. 2015, Schmidhuber 2015, Goodfellow et al. 2016) has made significant progress for prediction problems across computer vision (Krizhevsky et al. 2012), speech recognition (Dahl et al. 2012), and natural language processing (NLP) (Manning 2015). These success stories are often attributed to factors such as highly expressive models with millions of parameters, massive labeled data sets, scalable optimization algorithms, software for automatic differentiation, and hardware innovations. Yet many of DL's foundations are closely related to concepts well-known in statistics: log-likelihood functions, hierarchical modeling, latent variables, and regularization. Despite this conceptual overlap, the two worlds of statistics and DL remain relatively disconnected. One reason may be that the role of statistical thinking in DL is not widely acknowledged or advertised. Engineering feats and technological improvements are often championed by industry stakeholders, which may leave statisticians believing that they lack the expertise to push forward the research frontier. Moreover, the DL literature inherits terminology from its roots in cognitive science (e.g., "neuron," "activation function") and has developed a vernacular of its own (e.g., "attention"). This lack of a shared language may deter a statistician who is curious enough to read a DL paper. In this article, we aim to provide a bridge between the two communities by exposing the statistical foundations of DL, with two primary goals:

1. Explain DL's concepts, methods, and research trends in a manner accessible to a wide statistically minded audience
2. Identify opportunities in DL where statistical researchers can contribute new theories, models, and methods

There have been multiple other articles making these connections in the past, including papers in the 1990s and early 2000s focusing on non-deep neural networks (NNs) (White 1989, MacKay 1992, Cheng & Titterington 1994, Neal 1994, Ripley 1996, Stern 1996, Lee 2004), as well as more recent reviews that specifically make connections with DL (Mohamed 2015, Efron & Hastie 2016, Polson & Sokolov 2017, Yuan et al. 2020, Bartlett et al. 2021, Fan et al. 2021). While there is inevitably a degree of overlap across all of these reviews, including this article, our review complements the existing literature by providing an introduction that balances coverage and detail (a brief tour). Due to the breadth of work in DL, it is impractical to attempt an exhaustive review—for instance, we do not cover deep reinforcement learning. Readers interested in more details on DL may wish to delve into textbooks such as those by Goodfellow et al. (2016) and Murphy (2022).

DL and statistics differ not only in terminology and methodology but also (importantly) in terms of perspective. DL places a strong emphasis on data-driven predictive accuracy for validating models, whereas in statistics, there is significant emphasis on model interpretability and uncertainty quantification. This distinction is not new: Breiman (2001) famously argued as much, Welling (2015) modernized the argument for the DL era, and Efron (2020) provided a recent perspective. Questions that are natural in a statistical context, such as asymptotic consistency or posterior concentration, are much less relevant (or arguably not relevant at all) in DL, where models routinely have thousands, if not millions, of parameters. In essence, deep learners tend to focus on $\hat{y}$ rather than $\hat{\theta}$.

At least part of the reason for DL's focus on prediction goes back to its roots in pattern recognition and an emphasis on representation learning, where high-dimensional inputs often need to be transformed into representations (features) that are useful for prediction. For example, in both image classification and speech recognition, researchers for decades used a two-stage approach to

build classifiers: First, manually define useful functions (filters, templates) to extract features from signals, and then, learn classification models using the predefined features. A major contribution of DL has been to replace this two-stage approach with a single model trained end-to-end, from signals to outputs, where signals (pixels, audio) are transformed to intermediate representations, in a layer-by-layer manner. The most visible successes of DL have been for problems involving prediction from perceptual low-level signals (images, speech, text) for which feature extraction is essential.

In addition, the idea of building models by composing simple building blocks is a fundamental concept in both DL and statistics, but the two fields approach the concept of compositionality in different ways. In statistics, there is a rich tradition of using random variables as building blocks, allowing for likelihood functions to represent complex data-generating mechanisms, sharing statistical strength across groups and hierarchies, representing dynamics over time, or capturing random effects and interactions. In contrast, in DL, while the input-output mapping of a deep model may have probabilistic semantics, the internal building blocks are typically deterministic functions composed in a layered fashion and combined with operations such as convolution. This determinism yields both strengths and weaknesses. For example, it allows the modeler more flexibility by removing the need for distributional assumptions, but it makes uncertainty quantification more challenging. A notable exception in this context is deep latent variable models, as discussed in Section 4, which use internal representations that combine both stochastic variables and deterministic transformations.

The two fields also different significantly in terms of scale: scale of model complexity, scale of data sets, and scale of computation. The desire to learn internal representations has led DL researchers to work with complex NN architectures that contain many learnable weights. This complexity, in turn, has created a need for ever-larger data sets to build such models. More data support the learning of more complex (and potentially better predictive) internal representations, with the result that state-of-the-art models in image, speech, and language modeling are routinely trained on millions to billions of data points (Bommasani et al. 2022). In contrast, for many typical statistical analysis problems, data sets at this scale may be completely unavailable (particularly in application areas such as medicine). Furthermore, in order to handle the very large scales of models and data, DL also requires significant engineering advances: automatic differentiation to support high-level model specification, stochastic gradient methods for efficient optimization, and graphics processing units (GPUs) for efficient linear algebra computations. These have all played key roles in making DL practical.

2. VISUAL PATTERN RECOGNITION WITH FEEDFORWARD ARCHITECTURES

The early development of artificial NNs was heavily influenced by ideas from cognitive neuroscience and human visual perception (McCulloch & Pitts 1943). NNs made a more pragmatic turn toward applications in the late 1980s and early 1990s, with hand-written digit recognition being a challenging benchmark that also attracted interest from the United States Postal Service (LeCun et al. 1989). Progress slowed throughout the 2000s, but the empirical success of DL in the 2012 ImageNet benchmark competition (Krizhevsky et al. 2012), among other empirical successes in the early 2010s, focused attention on the field. Since then, deep neural networks (DNNs) have become a crucial subcomponent of a variety of systems used in problems ranging from language modeling (Devlin et al. 2019) to autonomous driving (Grigorescu et al. 2020) to playing go (Silver et al. 2017) to protein folding prediction (Jumper et al. 2021), bolstering DL's role as the primary methodology of interest within the fields of machine learning and artificial intelligence over the past decade.

Annu. Rev. Stat. Appl. 2023.10:219-246. Downloaded from www.annualreviews.org
Access provided by University of California - Irvine on 09/10/23. See copyright for approved use.

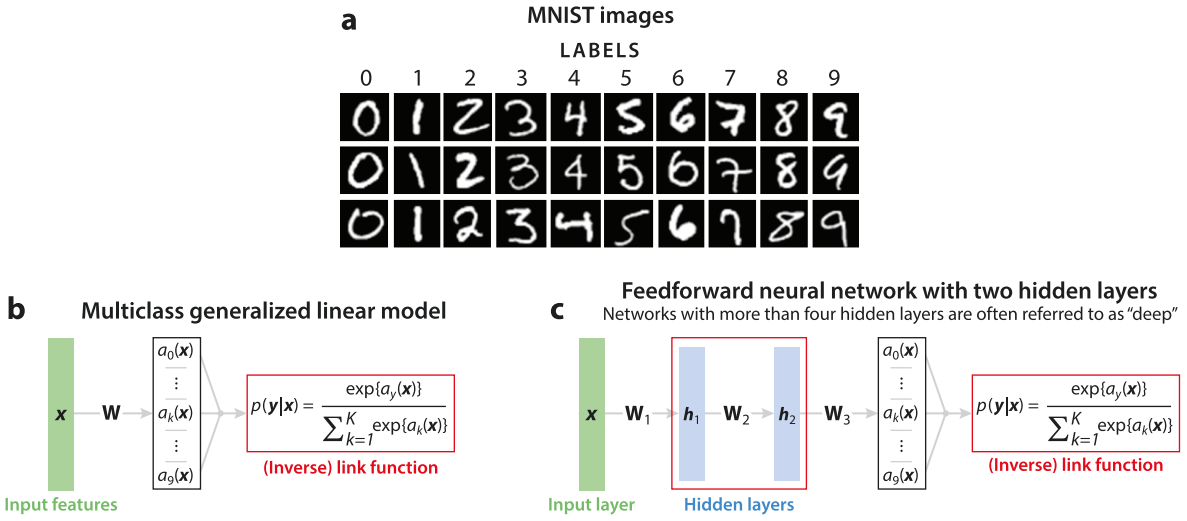

Figure 1

Modified National Institute of Standards and Technology (MNIST) classification: each image $\boldsymbol{x}$ is a 28 × 28 array, each element of which is a pixel intensity value from {0, . . . , 255}, and each image is associated with a label in {0, . . . , 9}. Abbreviations: a, activation/logit; $\boldsymbol{h}$, hidden layer; K, number of classes; k, class index; $\mathbf{W}$, weight matrix (parameters); $\boldsymbol{x}$, array of pixel values; $\boldsymbol{y}$, class label.

Hence, we begin our discussion by focusing on visual pattern recognition and, in particular, the task of classifying images into one of K categories or classes. We assume the simplest setting, in which each image contains one (and only one) of the K candidate objects. As a running example, we use the well-known Modified National Institute of Standards and Technology (MNIST) image classification data set (LeCun et al. 1998). This data set is often used for pedagogical purposes, as it is small enough that training and evaluating models are easy enough to carry out on a typical laptop. Each MNIST image $\boldsymbol{x}_n$ has a resolution of 28 × 28 pixels, which can be represented as a two-dimensional matrix with each element being a pixel value having an intensity $x \in [0, 1]$. **Figure 1*a*** shows samples from the data set for each digit class. The corresponding label is a value $y_n \in \{0, \ldots, 9\}$. The standard data set has a total of $N = 70{,}000$ image–label pairs, typically split into 50,000 training images and 10,000 test images, with 10,000 images used for hyperparameter tuning and validation.

2.1. Feedforward Neural Networks

Consider how we might go about building a model to predict labels y given pixels $\boldsymbol{x}$ for our image classification problem. In terms of notation, we can represent the image pixels as a $d \times 1$ vector $\boldsymbol{x}$, where $d = 784 = 28 \times 28$ pixel values for our MNIST images. This vectorization ignores the spatial relationship of pixels but nonetheless can produce accurate classifiers, as we see below—and we introduce some spatial information to the modeling setup when we discuss convolutional models later in this section. Ideally we would like to build a conditional model $p(y|\boldsymbol{x})$ for our problem, i.e., a mapping $\mathcal{X}^d \mapsto \Delta_K$, where $\mathcal{X}^d$ is the d-dimensional pixel space, Δ_K is the K-dimensional simplex, and K is the number of class labels.

We could consider a simple statistical model for $p(y|\boldsymbol{x})$, e.g., in the form of categorical generalized linear model (GLM), or multinomial logistic regression:

$$y|\boldsymbol{x} \sim \text{Categorical}\,(\boldsymbol{\pi}), \quad \pi_k = g_k^{-1}(\mathbf{W}^{\mathrm{T}}\boldsymbol{x}) = \frac{\exp\left\{\mathbf{w}_k^{\mathrm{T}}\boldsymbol{x}\right\}}{\sum_{j=1}^{K}\exp\left\{\mathbf{w}_j^{\mathrm{T}}\boldsymbol{x}\right\}}, \qquad 1.$$

Annu. Rev. Stat. Appl. 2023.10:219-246. Downloaded from www.annualreviews.org
Access provided by University of California - Irvine on 09/10/23. See copyright for approved use.

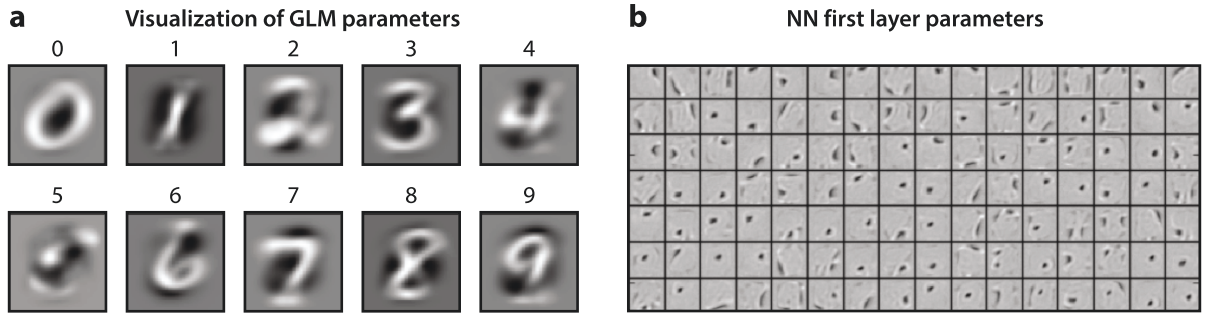

Figure 2

Parameter visualization with more positive values of weights shown in white. The NN weight visualization in panel *b* consists of a $7 \times 16 = 112$ array of weight images, with each of the individual images corresponding to the weight matrix for each of the 112 hidden units in the first layer of the NN. Abbreviations: GLM, generalized linear model; NN, neural network.

Annu. Rev. Stat. Appl. 2023.10:219-246. Downloaded from www.annualreviews.org
Access provided by University of California - Irvine on 09/10/23. See copyright for approved use.

where $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_K) = p(\mathrm{y}|\boldsymbol{x}) = \mathbf{g}^{-1}(\mathbf{W}^{\mathrm{T}}\boldsymbol{x})$ is a vector of conditional class probabilities and $\mathbf{g}^{-1}$ is the inverse link function, as illustrated in **Figure 1*b*** with $a_k(\boldsymbol{x}) = \mathbf{w}_k^{\mathrm{T}}\boldsymbol{x}$.

The parameters of the GLM are represented by the weight matrix $\mathbf{W}$, consisting of K weight vectors $\mathbf{w}_k$, $k = 1, \ldots, K$, each weight vector being of the same dimension d as the inputs. We can visualize the weights for any weight vector $\mathbf{w}_k$ by spatially by mapping each weight to the pixel it corresponds to in image coordinates. **Figure 2*a*** shows the resulting spatial visualization of the weights $\mathbf{w}_k$ for each of the digit classes, following maximum likelihood estimation of a categorical GLM on the MNIST training data. We see that the GLM has in effect learned templates for each digit. Whichever template has the largest activation for a given input—as quantified by $\mathbf{w}_k^{\mathrm{T}}\boldsymbol{x}$—determines the class with the highest conditional probability. This trained GLM can achieve roughly 92% classification accuracy on the standard test set—well above the 10% accuracy of random predictions but still far from the near-perfect test accuracy (99.7%) of typical DNNs for this problem.

With this GLM as a point of reference, now consider tackling the same classification problem with a feedforward NN (a.k.a. a multilayer perceptron). Formulating the NN as a statistical model, we can build on the idea of a GLM as follows. Instead of $a_k(\boldsymbol{x}) = \mathbf{w}_k^{\mathrm{T}}\boldsymbol{x}$, we use $a_k(\boldsymbol{x}) = \mathbf{w}_k^{\mathrm{T}}\boldsymbol{b}(\boldsymbol{x})$, as in **Figure 1*c***, where $\boldsymbol{b}(\boldsymbol{x})$ can be viewed as a new nonlinear vector representation of the original inputs $\boldsymbol{x}$, a representation that is learned by the NN. The NN model has L layers, where the final layer still looks like a GLM, parameterizing the conditional expectation of the label: $\mathbb{E}[\mathrm{y}|\boldsymbol{x}] = \boldsymbol{\pi} = \mathbf{g}^{-1}(\mathbf{W}_L^{\mathrm{T}}\boldsymbol{b}(\boldsymbol{x}))$, where $\mathbf{W}_L$ is the weight matrix for the final layer.

In the NN model, each internal vector $\boldsymbol{b}_l(\boldsymbol{x})$ at the lth layer, $l = 1, \ldots, L-1$, is termed a hidden layer, with each element in each vector termed a hidden unit or neuron. Each layer is computed recursively as $\boldsymbol{b}_l(\boldsymbol{x}) = \sigma(\mathbf{W}_l^{\mathrm{T}}\boldsymbol{b}_{l-1}(\boldsymbol{x}))$, where $\sigma(\cdot)$ is a referred to as an activation function.[1] The logistic function $\sigma(z) = 1/(1 + \exp\{-z\})$ has historically been a popular choice for activation functions; we discuss other options later. The initial layer can be defined to simply be the original inputs: $\boldsymbol{b}_0(\boldsymbol{x}) = \boldsymbol{x}$, e.g., the pixel values for our MNIST problem represented in vector form. Dropping the function notation for simplicity [i.e., letting $\boldsymbol{b}_l(\boldsymbol{x}) = \boldsymbol{b}_l$], the complete feedforward

[1]While NN terminology (e.g., neuron, activation) still reflects the field's early roots in neuroscience, this point of view does not necessarily help with understanding NNs as statistical models, nor are many modern developments in DL driven by biological inspirations.

NN can be defined as

$$\mathbb{E}[\mathrm{y}|\boldsymbol{x}] = g^{-1}(\mathbf{W}_L^{\mathrm{T}}\boldsymbol{h}_{L-1}), \quad \boldsymbol{h}_l = \sigma(\mathbf{W}_l^{\mathrm{T}}\boldsymbol{h}_{l-1}), \quad \boldsymbol{h}_0 = \boldsymbol{x}, \tag{2.}$$

where $\boldsymbol{h}_l \in \mathbb{R}^{d_l}$ such that d_l is the representation's dimensionality at layer l. The different matrices of parameters, denoted $\boldsymbol{W}_l \in \mathbb{R}^{d_{l-1} \times d_l}$, define the successive transformations $1, \ldots, L$. There is also usually an additive bias term at each layer, $\mathbf{W}_l^{\mathrm{T}}\boldsymbol{h}_{l-1} + \boldsymbol{b}_l$—we can assume that it is subsumed into $\mathbf{W}_l$ by appending a dimension containing a constant value of 1 to $\boldsymbol{h}_{l-1}$.

NN models of this form are described as feedforward since information propagates from the input $\boldsymbol{x}$ to the output $\mathbb{E}[\mathrm{y}|\boldsymbol{x}]$ through parallel computations of each hidden unit and sequential computation of each hidden layer, as shown in **Figure 1*c***, with the analogous figure for the GLM shown in **Figure 1*b*** for comparison. The primary difference between the two models is in the intermediate features $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$ that constitute the hidden layers. The arrows between all layers emphasize how the $\mathbf{W}$ parameters propagate information forward. The modeler is left to choose the size of the architecture: the number of hidden layers L and the dimensionality (or width) d_l of each hidden representation $\boldsymbol{h}_l$. Principled procedures for setting these values are desirable but hard to achieve. Thus, in practice, these hyperparameters are often tuned simply by brute force search: estimating parameters on the training set for different fixed combinations of hyperparameter values (e.g., on a grid) and selecting the combination that optimizes performance (e.g., classification accuracy) on a held-out validation data set.

We can interpret the hidden layers $\boldsymbol{h}$ as adaptive, nonlinear basis functions. These allow the model itself to transform the original feature space into representations that are better suited for the classification task. This concept of internal representation learning (Bengio et al. 2013a) is arguably the single most important characteristic responsible for the success of NNs. **Figure 2*b*** shows the features learned by the first hidden layer when a four-hidden-layer NN is trained on MNIST. This visualization is analogous to the GLM's in **Figure 2*a***. The NN learns features that are local edge detectors, unlike the GLM's global templates. This allows the model to gradually build a feature hierarchy, layer by layer. The second hidden layer will compose these features, and so on with each layer. This behavior makes NNs most effective on low-level raw signals, as the hidden layers can gradually aggregate information into higher levels of abstraction, e.g., in the context of classification, learning discriminative features that are useful for prediction at the output layer.

2.2. Maximum Likelihood and Stochastic Optimization

Having defined the feedforward NN, we now turn to model fitting. DL models are usually trained with maximum likelihood estimation, often with an independent and identically distributed assumption for classification problems. The log-likelihood can thus be written as

$$\ell(\mathbf{W}_1, \ldots, \mathbf{W}_L) = \sum_{n=1}^{N} \log\ p(\mathrm{y}_n|\mathbf{x}_n; \mathbf{W}_1, \ldots, \mathbf{W}_L), \tag{3.}$$

where $p(\mathrm{y}_n|\mathbf{x}_n; \boldsymbol{W}_1, \ldots, \boldsymbol{W}_L) = \pi_{\mathrm{y}_n}$, the probability of the observed class. Usually some form of regularization is also incorporated into training. Perhaps the most common technique is known as weight decay in the DL literature. This is, in effect, equivalent to a ridge penalty in statistical terminology: $\ell(\boldsymbol{W}_1, \ldots, \boldsymbol{W}_L) + \lambda \sum_{l=1}^{L} ||\boldsymbol{W}_l||_2^2$, where $||\cdot||_2^2$ denotes the squared Frobenius norm and λ is a parameter that controls the strength of the regularization. Other common regularization strategies include data augmentation and stochastic perturbations such as dropout (Srivastava et al. 2014), which randomly sets hidden units to zero during training.

Annu. Rev. Stat. Appl. 2023.10:219-246. Downloaded from www.annualreviews.org
Access provided by University of California - Irvine on 09/10/23. See copyright for approved use.

Despite this regularization, a statistician may still worry about overfitting with NNs, given that they are so overparameterized. Having a held-out validation data set (or using multi-fold cross validation, in the small data setting) is the most effective strategy to prevent overfitting. For example, a useful strategy is early stopping: When fitting the network with an iterative optimization procedure, we keep training the NN until the validation set's accuracy begins to decrease—a sign that overfitting has begun. Yet, NNs can avoid overfitting even when a large reserve of held-out data is not available. This is because overparameterization is not as dangerous to generalization as previously suggested by the classical theory of bias versus variance. We discuss this more in Section 5.1, but good generalization can be observed even in the case of overparameterized linear models (Hastie et al. 2022).

Returning to the log-likelihood, maximizing $\ell(\mathbf{W}_1, \ldots, \mathbf{W}_L)$ is a nonconvex optimization problem with no unique solution for the weight parameters due to invariances and nonidentifiability. In spite of these challenges, relatively simple gradient-based optimization methods are by far the most widely used and empirically successful method for training NNs. Gradient ascent is a first-order iterative method for maximization (or, equivalently, gradient descent if performed on the negated objective), updating an initial set of parameters (randomly initialized) and taking a step in the direction of steepest increase of the objective function. Given a log-likelihood ℓ, the updates from iteration t to $t + 1$ for a single parameter w are performed via

$$w_{t+1} = w_t + \alpha \frac{\partial}{\partial w_t} \ell(\mathbf{W}_1, \ldots, \mathbf{W}_L) = w_t + \alpha \sum_{n=1}^{N} \frac{\partial}{\partial w_t} \log\, p(\mathrm{y}_n | \mathbf{x}_n; \mathbf{W}_1, \ldots, \mathbf{W}_L), \qquad 4.$$

where α is a scalar learning rate (a.k.a. step size).

Computing the full gradient above requires summing gradients over each of the N data points, which may be prohibitively expensive for training sets with millions of high-dimensional data points. However, a noisy estimate of the gradient can be found by evaluating the likelihood on only a subset (potentially very small) of data. Define a random mini-batch $\mathcal{B}$ of data to be a subset of B data points created by drawing from the full set of observations (e.g., without replacement). We can then perform stochastic gradient descent (SGD) (Robbins & Monro 1951, Bottou 2010) by using the mini-batched likelihood $\tilde{l}$ in place of the full gradient using all N data points:

$$w_{t+1} = w_t + \alpha \frac{N}{B} \frac{\partial}{\partial w_t} \tilde{\ell}(\mathbf{W}_1, \ldots, \mathbf{W}_L) = w_t + \alpha \frac{N}{B} \sum_{i \in \mathcal{B}} \frac{\partial}{\partial w_t} \log\, p(\mathrm{y}_i | \mathbf{x}_i; \mathbf{W}_1, \ldots, \mathbf{W}_L). \qquad 5.$$

The method is termed stochastic because the gradient estimate is now a random variable. We rescale the derivative by N/B so that the likelihood is on the same scale as it would be for the full data set, which can also be viewed as an adjustment to the learning rate α. The key idea behind SGD is that when B is much smaller than N, one can make multiple noisy (but computationally cheaper) parameter updates, moving in the direction of a noisy gradient at each step, and potentially converge much faster (in wall-clock time) compared with steps using the full gradient.

Figure 3 shows the optimization progress for an NN trained with gradient updates computed using 100, 10, and 1 data point(s). While the three variations start at the same value of the log-likelihood (y-axis), the curves for 1 and 10 data points are able to make progress much faster as a function of the total number of data points seen by the optimization algorithm (x-axis). Even though they are using noisy gradient estimates, there is still enough signal in the estimate that the computational benefits outweigh the noise in the estimates. In this case, eventually all methods converge to roughly the same log-likelihood value (past the right-hand side of the frame of the plot), although in other cases, the varying noise levels may introduce different inductive biases.

While it may seem hopelessly naive to apply a crude first-order method to training deep networks, SGD has been found empirically to be a reliable optimization strategy. In fact, the success

Annu. Rev. Stat. Appl. 2023.10:219-246. Downloaded from www.annualreviews.org
Access provided by University of California - Irvine on 09/10/23. See copyright for approved use.

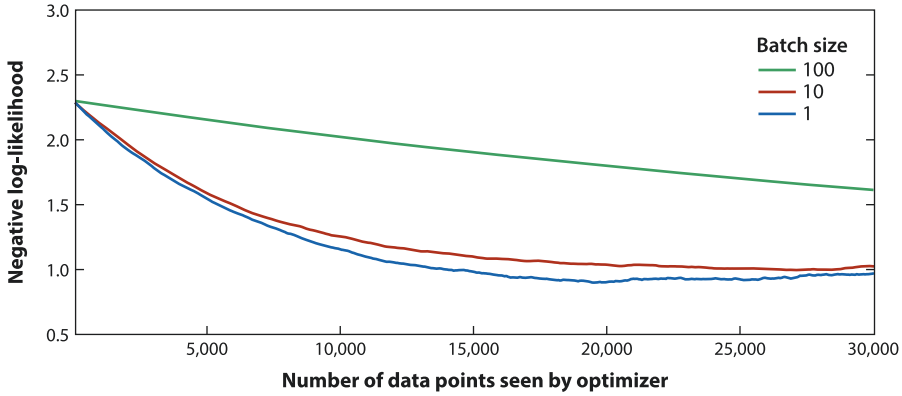

Figure 3

Empirical illustration of the benefits of using stochastic gradients under different sized batches of data points to compute the gradient estimates with batch size $B = 1, 10, 100$.

of DL is a testament to the perhaps more surprising success of SGD. Before 2012, NNs were conjectured to be limited in their usefulness because they would be restricted by the limitations of SGD (Cheng & Titterington 1994). A more complete understanding of why stochastic gradient works as well as it does is still an area of active research, but initial evidence suggests the noise introduced in the gradient estimates can actually be beneficial—for example, for escaping saddle points, which comprise the majority of critical points in the NN's optimization surface (Pascanu et al. 2014).

It is natural to ask why DL relies on first-order rather than second-order information, i.e., the Hessian. Second-order methods were, in fact, of interest in the early days of NN research (Parker 1987, Becker & LeCun 1989) and are, of course, widely used in statistics in the form of Fisher scoring. However, the very large number of parameters in modern NNs makes it impractical to compute and store all second derivatives. Moreover, the conditioning matrix can often be singular. For these reasons, first-order stochastic gradient methods, and adaptive variations in particular, have become the default practical choice in training DNNs (Duchi et al. 2011, Kingma & Ba 2014). There are a number of different varieties of such methods, but most operate by storing the empirical moments of the gradients (usually first and second) and using these to condition the next step.

Despite the success of SGD, optimizing NNs does not come without obstacles. To further illuminate the mechanics of optimization, assume w is a parameter residing at some intermediate layer of the NN. Expanding the likelihood's derivative with respect to w via the chain rule, we have

$$\frac{\partial \ell}{\partial w} = \frac{\partial \ell}{\partial \boldsymbol{b}_L} \frac{\partial \boldsymbol{b}_L}{\boldsymbol{b}_{L-1}} \cdots \frac{\partial \boldsymbol{b}_{l+1}}{\boldsymbol{b}_l} \frac{\boldsymbol{b}_l}{\partial w}. \quad 6.$$

The derivative is found by multiplicatively passing information backwards from the log-likelihood ℓ through the hidden representations $\boldsymbol{b}_l$ to the to-be-updated parameter w. Because of this intuition—that information is propagated backward through the NN—gradient-based optimization of NNs has been termed backpropagation of errors, or backpropagation, or backprop (Parker 1985, Le Cun 1986, Rumelhart et al. 1986). As NNs become increasingly deep [e.g., He et al. (2016) train NNs with 1,000+ layers], it is crucial that the intermediate derivates $\partial \boldsymbol{b}_l / \partial \boldsymbol{b}_{l-1}$ remain well-conditioned. For instance, if just one term approaches zero, then all parameters at shallower layers in the NN will receive a gradient of zero due to the multiplicative construction of backpropagation. This specific problem is known as having vanishing gradients, and it can result

Annu. Rev. Stat. Appl. 2023.10:219-246. Downloaded from www.annualreviews.org
Access provided by University of California - Irvine on 09/10/23. See copyright for approved use.

in slow convergence, in the best case. There is also the converse problem for large values, known as exploding gradients.

Returning to our discussion of the activation function $\sigma(\cdot)$, the logistic function used to be a popular choice but has fallen out of favor in recent years. To see why, note that the logistic's derivative is $\sigma' = \sigma(1-\sigma)$, and therefore the gradient signal starts to vanish when either $\sigma \approx 0$ or $\sigma \approx 1$, an effect known as saturation. Rectified activations such as rectified linear units (ReLUs) (Maas et al. 2013) do not have a bounded range in one or both directions, preventing the type of saturation that leads to vanishing gradients. Yet changing the activation function alone is often not enough to mitigate optimization pathologies. Normalization of the hidden units or their preactivation has become common as well (Ba et al. 2016, Salimans & Kingma 2016, Klambauer et al. 2017). The most popular instantiation of this regularization is known as batch normalization (Ioffe & Szegedy 2015) (known as batch norm for short). Roughly speaking, this method applies the standard z-transform $(a - \hat{\mu})/\hat{\sigma}$ to the preactivation values a at each internal layer, where $\hat{\mu}$ and $\hat{\sigma}$ are the empirical mean and standard deviation of the current training batch at a particular layer.

2.3. Uncertainty Quantification

Given the large number of parameters in NNs, it is worth considering how model uncertainty can be quantified and controlled. Up to this point in our discussion we have focused on frameworks, such as stochastic gradient methods, that seek point estimates of parameters—i.e., that optimize an objective function. An obvious alternative is to turn to Bayesian methodologies, placing a prior on the parameters, obtaining the posterior, and computing predictions with the posterior predictive distribution

$$p(\mathbf{y}^*|\boldsymbol{x}^*, \mathcal{D}) = \int_{\mathbf{W}_1,\ldots,\mathbf{W}_L} p(\mathbf{y}^*|\boldsymbol{x}^*, \mathbf{W}_1, \ldots, \mathbf{W}_L)\, p(\mathbf{W}_1, \ldots, \mathbf{W}_L|\mathcal{D}) \mathrm{d}\mathbf{W}_1, \ldots, \mathbf{W}_L,$$

where $\boldsymbol{x}^*$ is a new observation and $\mathcal{D}$ is the training set. This is a very attractive approach for addressing the all but unavoidable model uncertainty from having such an underdetermined model. However, the Bayesian approach has two obstacles to its effective implementation for DL (Izmailov et al. 2021). The first obstacle is that of setting a meaningful prior on the weights. Given that the weights lack identifiability and even a semantic interpretation, it is difficult to set a prior beyond one that simply encourages regularization via sparsity or shrinkage. The second major obstacle is that even if a good prior is found, posterior inference is challenging for NNs of any practical size. Variational methods can scale to rather large NNs but have intrinsic bias due to the variational family usually being misspecified. Scaling Markov chain Monte Carlo (MCMC) methods to large deep networks is currently an active focus of research in Bayesian DL (Izmailov et al. 2021).

Frequentist-based inference methods can also be applied. The bootstrap may first come to mind, but simply training an ensemble of networks, each with a different initialization, has been shown to be more effective than the bootstrap at uncertainty quantification (Lakshminarayanan et al. 2017). Post-hoc calibration techniques (Guo et al. 2017) are also popular for correcting for misspecification. A third promising method is that of conformal prediction (Shafer & Vovk 2008, Angelopoulos et al. 2020), which provides tools for constructing distribution-free guarantees on the (marginal) coverage of the true label. **Figure 4** demonstrates some of these inference procedures via a one-dimensional regression task (better suited for visualization than classification in this context). **Figure 4** compares a point-estimated NN (**Figure 4*a***) against a Bayesian NN (**Figure 4*b***) whose posterior is obtained via MCMC. The predictive variance is shown, and as expected, the MCMC solution collapses its uncertainty where the data are observed and inflates it elsewhere. **Figure 4*c*** and **Figure 4*d*** show common strategies to approximate model uncertainty.

Annu. Rev. Stat. Appl. 2023.10:219-246. Downloaded from www.annualreviews.org
Access provided by University of California - Irvine on 09/10/23. See copyright for approved use.

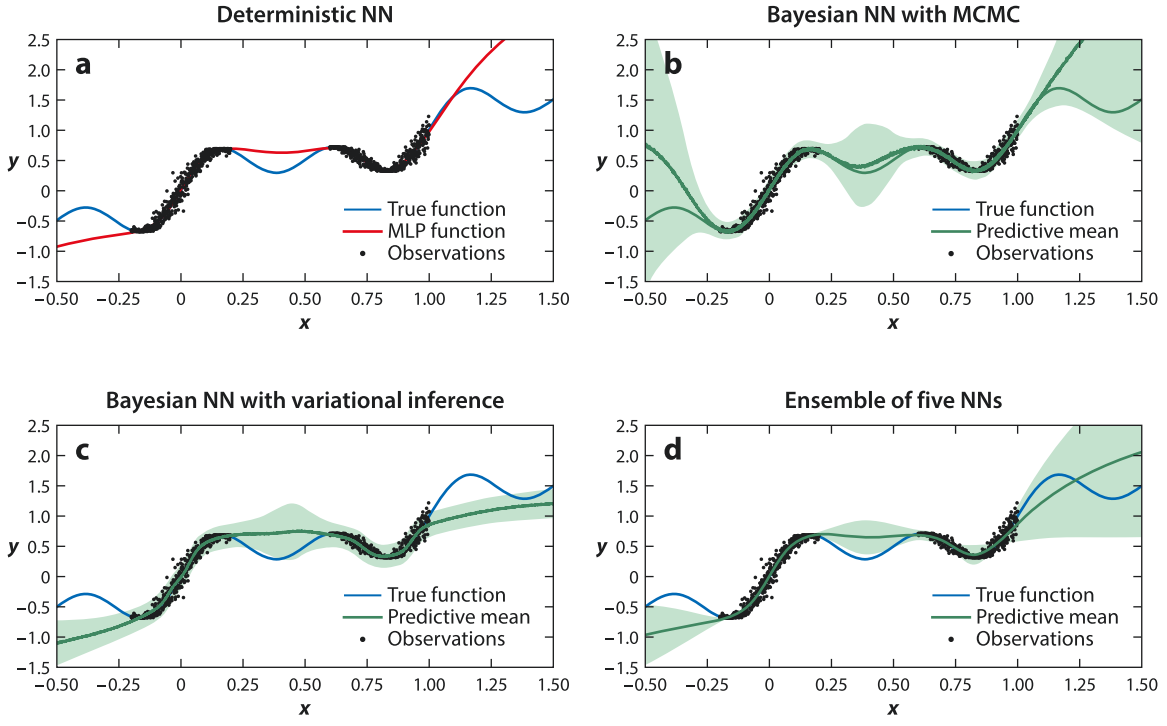

Figure 4

Real-valued regression: To demonstrate model uncertainty, we plot the predictive distribution of (*a*) a point-estimated NN, (*b*) a Bayesian NN (MCMC posterior), (*c*) a variational Bayesian NN (Gaussian-approximated posterior), and (*d*) an ensemble of point-estimated NNs. Abbreviations: MCMC, Markov chain Monte Carlo; MLP, multilayer perceptron; NN, neural network.

While imperfect, variational inference and ensembling are two of the few methods that can scale to large NNs.

2.4. Convolutional and Other Layer Types

For the sake of simplicity, we introduced only fully connected weight transformations to compute hidden activations at each layer. However, unsurprisingly, other architectures have also been developed. A particularly popular example is the convolutional layer: For inputs in the form of images, two-dimensional weight matrices known as filters are spatially convolved across the input, ensuring translation invariance in the input signal. The different hidden units each have their own associated convolutional filter—in effect, their own feature detector. Convolutional layers are commonly used for object detection, especially when it is assumed that an object can appear anywhere in the input image. For MNIST, the digits are all at the center of each image, but if they could appear elsewhere in the image, then using a convolutional NN would be essential. Despite MNIST not needing translation invariance for good performance, using a convolutional NN results in about a 0.3% test error rate compared with about 1% for a nonconvolutional feedforward NN, and about 7.6% for a logistic GLM.[2]

[2]These results are from **https://en.wikipedia.org/wiki/MNIST_database**.

Annu. Rev. Stat. Appl. 2023.10:219-246. Downloaded from www.annualreviews.org
Access provided by University of California - Irvine on 09/10/23. See copyright for approved use.

Of course, other data types necessitate alternative layer designs. For instance, we may wish to apply an NN to a task from astronomy: classifying galaxies into types such as elliptical, spiral, and irregular. As objects in space have no natural orientation, it is common to use rotationally equivariant NNs for this task (Cohen et al. 2018). In another example, modeling relational data can benefit from the use of a graph NN (Wu et al. 2020); this approach has been used in applications ranging from quantum chemistry (Gilmer et al. 2017) to computer program synthesis (Allamanis et al. 2017) to protein folding (Jumper et al. 2021).

3. SEQUENTIAL MODELS

We next turn our attention to DL models for sequential data, extending beyond the feedforward models in the last section. We primarily focus on modeling categorical sequences of the form $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t, \ldots, \boldsymbol{y}_T$, where t can represent relative position or time. Each $\boldsymbol{y}_t = (y_{t1}, \ldots, y_{tK})$ is a K-dimensional indicator vector. From a prediction perspective, we are interested in autoregressive factorizations of the form $p(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T) = \prod_{t=1}^{T} p(\boldsymbol{y}_t \mid \boldsymbol{y}_{<t})$, where $p(\boldsymbol{y}_t \mid \boldsymbol{y}_{<t})$ is a distribution over the K categories at position t conditioned on the history of the sequence $\boldsymbol{y}_{<t} = \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}$ prior to t. Although our focus below is primarily on categorical sequences, as we discuss later, the general ideas of sequential deep models are also applicable to other sequential and time-series modeling problems.

A very common application of categorical sequential modeling in machine learning is in NLP, where the categories are characters or words. DL models in this context are known as language models and have revolutionized the field of NLP in recent years (Brown et al. 2020, McClelland et al. 2020). Common applications include predicting the next character or word $\boldsymbol{y}_{t+1}$ in a sequence conditioned on $\boldsymbol{y}_{<t+1}$; generation of new text $\boldsymbol{y}'_{t+1}, \boldsymbol{y}'_{t+2}, \ldots$ conditioned on preceding context; classification of an entire passage of text; or translating a sentence from one language into another. While the DL approaches for each of these tasks differ in the details, there are many common characteristics.

3.1. Example: Modeling Text at the Character Level

To illustrate some of the basic concepts in sequential DL models, we begin by focusing on the relatively simple problem of learning an NN model that can predict the next character in English text and that can generate new text conditioned on a partial sequence. For this problem, the K categories correspond to lower- and uppercase letters a–z/A–Z, digits 0–9, punctuation marks, and a variety of other symbols, with typically between $K = 50$ and 100 categories depending on what symbols we include in the vocabulary for a particular model.[3] As a running example in the discussion below, we use as our text source a concatenation of a number of publicly available arXiv LaTeX files authored by well-known researchers in statistics, with $K = 96$ unique characters and over 150,000 characters in total length.

A simple traditional approach to modeling such data would be to use an mth-order Markov model, requiring $O(K^m)$ parameters, referred to as n-gram models in NLP, with $n = m + 1$. Variations on these types of n-gram models have historically been widely used for modeling text (Halevy et al. 2009) but are obviously limited in terms of their ability to capture high-order dependencies. An alternative option would be to use a state-space model, perhaps with a real-valued low-dimensional state variable $\mathbf{z}_t$ with linear-Gaussian dynamics as a function of t, coupled to

[3] Realistic language models typically use words (or parts of words) as categories, with $K \sim 10^4$–10^6 depending on the particular application—for simplicity of exposition, we use the simpler problem of character-level modeling here.

Annu. Rev. Stat. Appl. 2023.10:219-246. Downloaded from www.annualreviews.org
Access provided by University of California - Irvine on 09/10/23. See copyright for approved use.

a transformation at each position t from the state-space to categorical observations. However, the parametric assumption of Gaussian dynamics would likely lack the flexibility to effectively represent the types of dependencies that occur in natural language sequences.

A key innovation in DL in this context was the development of the recurrent neural network (RNN) [also referred to as an Elman RNN (Elman 1990)] as an improvement over models in the observation space such as n-grams. An RNN builds on the concept of a state-space model where the state and observation equations for a standard RNN are typically defined as

$$\mathbf{z}_t = \sigma(\mathbf{H}\mathbf{z}_{t-1} + \mathbf{W}\boldsymbol{x}_t), \quad 7.$$

$$p(\boldsymbol{y}_t|\boldsymbol{y}_{<t}) = g^{-1}(\mathbf{A}\mathbf{z}_t), \quad t = 2, 3, \ldots, \quad 8.$$

where $\boldsymbol{x}_t$ is an input to the model at position t, with $\boldsymbol{x}_t = \boldsymbol{y}_{t-1}$ (e.g., the previous character or word) in an autoregressive modeling context, and $\boldsymbol{z}_t \in \mathcal{R}^h$ is a hidden state vector of dimension $h \times 1$. Both the inputs $\boldsymbol{x}_t$ and outputs $\boldsymbol{y}_{t-1}$ are indicator vectors of dimension $K \times 1$; e.g., for the character data set above with $K = 96$, each vector has value 1 for the vector component corresponding to a particular character and value 0 for all other vector components. A standard convention is to define the initial hidden state vector $\mathbf{z}_1$ as all zeros, so that the first RNN computation unit for the sequence has input $\boldsymbol{x}_2 = \boldsymbol{y}_1$, hidden state vector $\mathbf{z}_2 = \sigma(\mathbf{W}\boldsymbol{x}_2)$, and output $p(\boldsymbol{y}_2|\boldsymbol{y}_1) = g^{-1}(\mathbf{A}\mathbf{z}_2)$.

The parameters of this RNN model are weight matrices **W, A**, and **H** of dimension $h \times K$, $K \times h$, and $h \times h$, respectively. Analogous to the hidden units in feedforward networks, $\sigma(.)$ is a nonlinear recurrent activation function (e.g., logistic or ReLu), which gives the model nonlinear dynamics, and g^{-1} is an output link function that maps the linear transformation of the hidden (deterministic) state $\mathbf{z}_t$ to the output domain (typically a multinomial logit, as with feedforward models with categorical outputs). More generally, for real-valued observations, g^{-1} can map to the mean of $p(\boldsymbol{y}_t|\boldsymbol{y}_{<t})$ for some parametric form p, with an additional additive noise term $\mathbf{v}_t$, analogous to a standard state-space modeling approach.

Figure 5*a* provides a visual representation of the RNN state and observation equations at position t. The RNN bears some similarity to the feedforward model from Section 2 with the important difference that the hidden variable $\boldsymbol{z}_t$ is now both a function of the input $\boldsymbol{x}_t$ as well

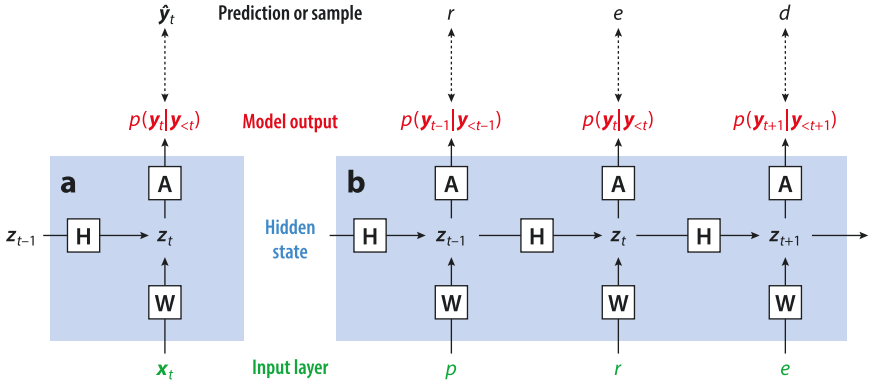

Figure 5

(*a*) A visual representation of a recurrent neural network (RNN) computational unit, implementing the state and observation equations for position t, with recurrent dependence of $\boldsymbol{z}_t$ on $\boldsymbol{z}_{t-1}$, resulting in a categorical prediction or sample $\hat{\boldsymbol{y}}_t$, and with RNN model parameters (weight matrices) $\theta = \mathbf{W}, \mathbf{H}, \mathbf{A}$. (*b*) An example of multiple RNN units being chained together to create an RNN model that can predict the next character in a sequence of characters, specifically $p, r, e, d, \ldots$.

Annu. Rev. Stat. Appl. 2023.10:219-246. Downloaded from www.annualreviews.org
Access provided by University of California - Irvine on 09/10/23. See copyright for approved use.

as a function of the hidden state variable z_{t-1} from the previous position, providing context for the current prediction in a recurrent manner based on the past history of the sequence. Deep versions of the simple RNN unit in **Figure 5*a*** can be created by vertically stacking additional hidden recurrent layers between each input and output.

Figure 5*b* shows an example of an RNN applied to our character modeling problem. Specifically, for the partial sequence `pred`, we see that at each position, the model combines the hidden state and observed character from the previous position to produce the current hidden state, from which the model output is generated. The observed data (in this case, a subsequence from the word `predict`) are shown at the top, indicating the true target output for the model. The parameters (weight matrices) $\theta = \mathbf{W}, \mathbf{A}, \mathbf{H}$ are shared across the different positions t in the model. The number of individual weights for this model scales as $O(Kh + h^2)$, avoiding the $O(K^n)$ explosion of parameters for observation-level models such as n-grams, which becomes impractical even for relatively small n when the number of categories K is large [e.g., $K \approx O(10^5)$ for word-level language models].

Once we know the parameters of an RNN model, we can use it in a generative autoregressive fashion to simulate sequences, sampling an output y'_t at each timestep t from the current conditional distribution, then using this as input at position $t + 1$ to combine with $\boldsymbol{z}_t$ to generate the next hidden state vector $\boldsymbol{z}_{t+1}$, sampling a y'_{t+1} from the new conditional output distribution at $t + 1$, and so on. The dynamics $p_t(\boldsymbol{y}_{t+1}|\boldsymbol{y}_t, \boldsymbol{z}_t)$ at the observation level are not homogeneous as a function of t but are functions of the history (as summarized by $\boldsymbol{z}_t$), in contrast to (say) a fixed-order Markov model.

3.2. Estimating the Parameters of a Recurrent Neural Network

The unknown parameters $\mathbf{W}, \mathbf{A}$, and $\mathbf{H}$ of an RNN model in **Figure 5*b*** are learned in a manner similar to that for categorical outputs in DL feedforward models, i.e., by maximizing a categorical conditional log-likelihood:

$$\ell(\mathbf{W}, \mathbf{A}, \mathbf{H}; \boldsymbol{y}) = \sum_{t=1}^{T} \log p(\boldsymbol{y}_t | \boldsymbol{y}_{<t}, \mathbf{W}, \mathbf{A}, \mathbf{H}). \quad 9.$$

This sum is typically over multiple sequences (e.g., multiple sentences), where each sequence is treated as conditionally independent of the others—here, for simplicity, we write the log-likelihood as one sequence of length T. Regularization terms are typically added to the log-likelihood as with training of feedforward models. As with feedforward models, it is also common practice in DL to use first-order gradient methods to train sequential deep models given the typical numbers of parameters in RNN models. SGD using mini-batches is, again, also widely used, given that in many NLP applications, models are trained on vast amounts of text data—e.g., all of Wikipedia or large crawls of public Web pages, resulting in billions of words being used during model training.

From **Figure 5*b*** we see that, in principle, the relevant gradients (per parameter) for the log-likelihood can be computed by propagating backwards (backpropagation in time) the relevant information from later predictions to earlier parts of the model (e.g., see Jurafsky & Martin 2022, chapter 9). In practice, long text sequences are divided into multiple shorter segments to make this approach practical. However, as with feedforward models, significant numerical issues (such as unstable gradients) can arise in gradient-based training of RNN models. This has led to the development of modified RNN computation units that exert more direct control of how information is passed along the hidden-unit chain. For example, Hochreiter & Schmidhuber (1997b) proposed the long short-term memory (LSTM) unit by introducing more complex

Annu. Rev. Stat. Appl. 2023.10:219-246. Downloaded from www.annualreviews.org
Access provided by University of California - Irvine on 09/10/23. See copyright for approved use.

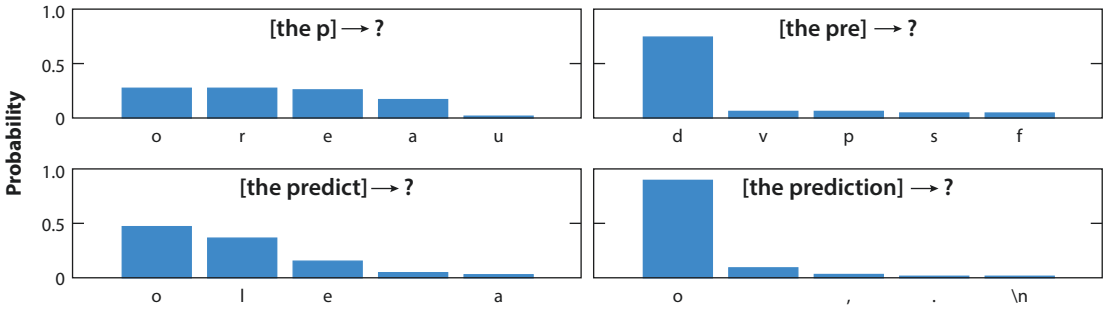

Figure 6

Predicted character-level probabilities for an RNN model conditioned on different prefixes (from `[the p]` to `[the prediction]`). The five highest conditional probabilities and the associated characters are shown for each prefix. The characters include letters as well as punctuation such as space, comma, and period.

Annu. Rev. Stat. Appl. 2023.10:219-246. Downloaded from www.annualreviews.org
Access provided by University of California - Irvine on 09/10/23. See copyright for approved use.

RNN computation units that can control or gate how much information is passed forwards and backwards (compared with the standard unit illustrated in **Figure 5*a***). In addition to improving optimization aspects of RNNs, LSTMs can also improve the effectiveness of how the hidden states represent past sequence information. Most modern usages of RNNs in DL use the LSTM unit or similar ideas for gating information (Cho et al. 2014).

To illustrate these ideas, we fit an RNN to the LaTeX text described earlier, with $K = 96$ unique characters and a text sequence of length 152,499 characters, with $h = 128$ as the dimensionality of the hidden unit layer, optimizing the conditional log-likelihood using SGD. **Figure 6** shows illustrative examples of conditional distributions produced by the model, conditioned on different subsequences from the history `[the prediction]`. As the history of the sequence progresses, the RNN captures the predictive uncertainty, from high uncertainty at the start of the word `prediction`, to low uncertainty toward the end.

We can also simulate sequences of characters from this model in a generative fashion, e.g.,

```
And matrix $\WLS$ is a given to describe a space of the diffusion of the
set of the accuracy of $\mu$ is will be the restrict the bagged
confidence studying the response ...
```

We see that while the trained RNN has captured many local aspects of character dependence (including some LaTeX syntax), the longer text lacks both syntactic and semantic coherence, and readers need not worry that RNNs will soon be writing statistics papers. However, with more training data and using more advanced models that extend beyond the relatively simple RNN, modern deep language models are now able to generate surprisingly coherent text (Brown et al. 2020).

3.3. Generalizing the Recurrent Neural Network Concept

The basic RNN model described above can be extended and generalized in a variety of different ways. One such variant is to have input and output sequences $\boldsymbol{x}$ and $\boldsymbol{y}$ that are in one-to-one correspondence but from different vocabularies, for example, in NLP, where the input is a sequence of words and the output sequence corresponds to the parts of speech to be predicted (nouns, verbs, adjectives, etc.) for each word. Another common NLP task is to build a model to assign a categorical class label y to an entire sequence $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$, where the training data consist of (sequence, label) pairs, e.g., assigning a positive, neutral, or negative label to a review [the problem of sentiment analysis (Wang et al. 2018)]. A more challenging NLP task involves mapping

from one sequence to another, where the two sequences can be of different lengths [also known as sequence transduction (Graves 2012)]. This type of sequence mapping problem is at the heart of problems such as machine translation (mapping a sentence in one language to a sentence in another) or automated chat-bots (generating a response sentence given a human-generated sentence in a conversation), where the training data consist of pairs of such sequences. A well-known DL approach for this type of problem is to use two coupled RNNs [the seq-to-seq approach (Sutskever et al. 2014)], with one RNN (the encoder) producing a hidden representation $\boldsymbol{z}$ of the first sequence at the end of an RNN chain, and the second RNN (the decoder) taking as input this encoded representation $\boldsymbol{z}$ and generating the second (output) sequence. Although the modeling details differ across the various models above, training of these models proceeds in a largely similar manner to that for the standard autoregressive RNN described earlier: minimizing the negative log-likelihood (or some regularized variant) using stochastic gradient methods while paying close attention to computational and numerical issues related to sequence length and vanishing gradients.

Models that perform sequential processing, like RNNs, can struggle with remembering the relevant information (e.g., across multiple sentences). Consider modeling the text "Rose lives in the Netherlands.... She enjoys speaking [X]," where [X] is the word to be predicted. "Netherlands" is a strong clue in predicting the next word (i.e., that she speaks Dutch), but it could be hard for a model to retrieve that information, depending on how much content is in the "..." part of the sequence. The concept of attention (Bahdanau et al. 2015) aims to break this dependence by allowing the NN to directly access the information at previous timesteps. Yet, attention alone does not necessarily break the sequential nature of RNN computation. To admit parallelized computation, Vaswani et al. (2017) introduced the Transformer model. The core idea is to use masking—indicator variables that allow some inputs to be included in the computation and others not—to preserve the autoregressive structure. The usual architecture design decisions as described for object recognition are applied, as one must select, for example, the sequencing and choice of layers, architecture depth, and layer width.

While RNNs and attention-based models have largely been developed for categorical sequences such as text, the basic concepts underlying these models are applicable to a much broader variety of prediction problems involving sequences and time. For example, RNNs have been adapted to develop models for problems that are familiar to statisticians, such as time-series forecasting (Wang et al. 2019b, Hewamalage et al. 2021, Lim & Zohren 2021), continuous-time point processes (Mei & Eisner 2017, Chen et al. 2020), and survival analysis (Ranganath et al. 2016, Wang et al. 2019a). There is also a growing body of work on models and inference methods that bridge the gap between RNNs and more traditional statistical models, such as stochastic RNNs (Krishnan et al. 2017), deep state-space models (Rangapuram et al. 2018), and Bayesian RNNs (McDermott & Wikle 2019), as well as latent space approaches for continuous-time and irregularly sampled time series using ordinary differential equation models parameterized via NNs (Chen et al. 2018). The development of DL models in these areas has yet to see the types of dramatic improvements in prediction performance that accompanied the development of DL, in part because many of the typical applications (in medicine, economics, and climate) do not have access to the massive volumes of data used in building DL models from text data, for example.

4. LATENT VARIABLE MODELS AND IMAGE GENERATION

Our focus up to this point has been on supervised learning. But since the early days of NN research, there has always been significant interest in unsupervised learning, motivated broadly by ideas from artificial intelligence and cognitive science. For example, can NNs mimic the ability

Annu. Rev. Stat. Appl. 2023.10:219-246. Downloaded from www.annualreviews.org
Access provided by University of California - Irvine on 09/10/23. See copyright for approved use.

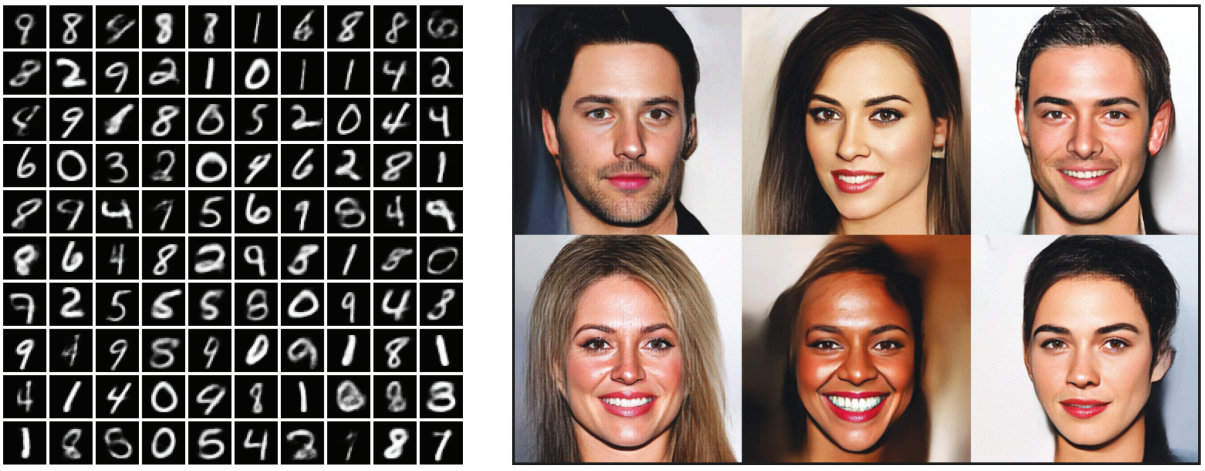

Figure 7

Samples from neural generative models. (*a*) The variational autoencoder samples were produced by Doersch (2021), and (*b*) the Glow samples by Kingma & Dhariwal (2018); both are used with permission of the authors.

of humans to learn structure from perceptual signals (e.g., audio, visual) from the world around them? As a concrete example, consider the images of digits shown in **Figure 7*a***. Do these look like images from the MNIST data set (see **Figure 1*a*** for reference)? Despite their visual similarity to MNIST, they are not from the data set but rather samples generated from an NN fit to MNIST. Next, consider **Figure 7*b***. These images are not of real people. Rather, the images were also generated by an NN, this one trained on a data set of celebrity images known as CelebA. These are cases of what is known as generative modeling in DL: The primary goal is to generate novel samples that plausibly could have been part of the training set. The task is similar to (nonparametric) density estimation, where we hope to capture the true distribution $\mathbb{P}(\mathbf{x})$ as faithfully as possible. As we will see, for some of the models in this class, we do have access to a density estimator, and for others we do not. Yet the quality of the samples drawn from the model is usually given more emphasis, as density and sample quality do not always correlate (Theis et al. 2016).

Models based on unsupervised learning have applications ranging from dimensionality reduction to data synthesis, although much of the excitement in this area stems from the desire to build intelligent systems. The intuition is that if our models can perfectly capture the training distribution, then they must understand the data. Models that only discriminate (e.g., classifiers) are then performing an easier cognitive task—just like it is easier to recognize quality art than to produce it. While the field of statistics also tries to build models to represent the data with as high fidelity as possible, a major difference is that these neural generative models are built with complete data agnosticism. Relatively few, if any, bespoke modeling decisions are made, but rather, the NN-based model is designed to be as powerful and rich as computational limits allow.

4.1. Dimensionality Reduction with Autoencoders

To introduce this class of models, consider the task of dimensionality reduction: We wish to learn a new representation of the data that discards noise and other unimportant information. Principal component analysis (PCA), manifold learning, and clustering are all well-known and well-studied methods for such a task. As discussed in Section 2, DNNs, too, are performing dimensionality reduction by nature of learning their hidden layers. Yet in that case, the dimensionality reduction

Annu. Rev. Stat. Appl. 2023.10:219-246. Downloaded from www.annualreviews.org
Access provided by University of California - Irvine on 09/10/23. See copyright for approved use.

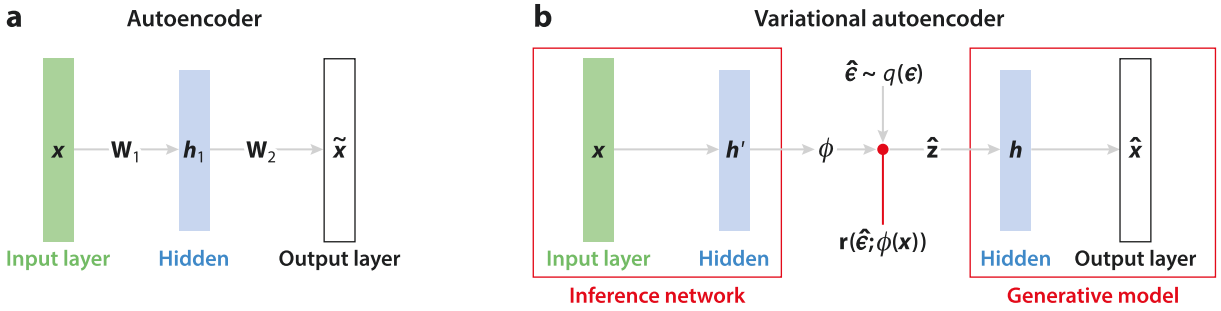

Figure 8

Autoencoder diagrams comparing (*a*) a fully deterministic architecture with (*b*) its stochastic counterpart. In the latter, there is a stochastic latent variable, denoted with $\mathbf{z}$. During optimization of the model, this latent variable is sampled using a reparameterization—denoted $\mathbf{r}(\hat{\boldsymbol{\epsilon}}; \boldsymbol{\phi}(\boldsymbol{x}))$, where $\boldsymbol{\epsilon}$ is the reparameterized version of $\mathbf{z}$ and $\boldsymbol{\phi}(\boldsymbol{x})$ are the parameters of the variational approximation—that allows for end-to-end differentiation. Abbreviations: $\boldsymbol{x}$, data; $\mathbf{W}$, weights for computing hidden units; $\boldsymbol{h}$, hidden units; $\tilde{\boldsymbol{x}}$, reconstruction of data; $\boldsymbol{\phi}(\boldsymbol{x})$, parameters of variational approximation (as a function of $\boldsymbol{x}$); $\hat{\mathbf{z}}$, sample of latent variable; $\hat{\boldsymbol{\epsilon}}$, samples from reparameterized distribution; $\hat{\boldsymbol{x}}$, sample from model.

is done with respect to the supervision signal (e.g., the class), so that the information that informs the prediction is preserved rather than a general summary of the data.

The autoencoder (AE) (a.k.a. a diablo network or auto-associator) (Bourlard & Kamp 1988, Baldi & Hornik 1989, Cottrell 1989, Hinton & Salakhutdinov 2006) is the simplest NN architecture designed for unsupervised learning and dimensionality reduction. The AE's goal is to reconstruct the data from a lossy representation of those same data. Specifically, the model takes an observation $\boldsymbol{x}$ as input, computes at least one hidden layer $\boldsymbol{h}$, and then tries to predict the observation $\boldsymbol{x}$ back from $\boldsymbol{h}$. An AE with multiple hidden layers can be defined as

$$\tilde{\boldsymbol{x}} = \mathbf{g}^{-1}(\mathbf{W}_L^{\mathrm{T}} \boldsymbol{h}_{L-1}), \quad \boldsymbol{h}_l = \sigma(\mathbf{W}_l^{\mathrm{T}} \boldsymbol{h}_{l-1}), \quad \boldsymbol{h}_0 = \boldsymbol{x}, \qquad 10.$$

where $\tilde{\boldsymbol{x}}$ is the predicted reconstruction of the input $\boldsymbol{x}$. $\mathbf{g}^{-1}$ is again a link function that maps to the domain of the data. $\mathbf{W}$, $\boldsymbol{h}$, and σ are defined as before for feedforward NNs. The AE is fit by minimizing an appropriate reconstruction loss between $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$, e.g., $||\boldsymbol{x} - \tilde{\boldsymbol{x}}||$, with respect to the parameters $\mathbf{W}_1, \ldots, \mathbf{W}_L$. A depiction of a simple one-hidden-layer AE can be seen in **Figure 8*a***.

Despite their lack of a probabilistic interpretation, AEs can be grounded by noticing that they are equivalent to PCA under special conditions (Baldi & Hornik 1989). When (*a*) the reconstruction error is the squared loss, (*b*) σ is the identity function, and (*c*) there is one hidden layer such that $\mathbf{W}_1 = \mathbf{W}_2^{\mathrm{T}}$—meaning the weight matrices are tied—then an AE performs PCA. In this restricted case, the dimensionality of $\boldsymbol{h}$'s role as an information bottleneck is clear: The number of hidden units corresponds to the number of eigenvectors used in the corresponding PCA.

4.2. Probabilistic Autoencoders for Generative Modeling

If the AE were given a probabilistic interpretation, then it could both perform dimensionality reduction and generate samples. The latter would be useful for synthesizing data as well as telling the user the degree of information loss. One simple variant that gives the AE a probabilistic formulation is the denoising autoencoder (DAE) (Vincent et al. 2008, 2010). Instead of passing $\boldsymbol{x}$ into the first layer, a DAE takes as input a corruption of $\boldsymbol{x}$: $\boldsymbol{x}' \sim \mathrm{P}(\mathbf{x}'|\boldsymbol{x})$, where $\mathrm{P}(\mathbf{x}'|\boldsymbol{x})$ is the noise model. Gaussian noise is one example: $\boldsymbol{x}' \sim \mathrm{N}(\boldsymbol{x}, \Sigma)$. Bengio et al. (2013b) showed that the DAE can then be interpreted as a transition operator generating an ergodic Markov chain whose asymptotic

Annu. Rev. Stat. Appl. 2023.10:219-246. Downloaded from www.annualreviews.org
Access provided by University of California - Irvine on 09/10/23. See copyright for approved use.

distribution is the data-generating distribution $\mathbb{P}(\mathbf{x})$. Vincent (2011) also provides a probabilistic interpretation via score matching.

A more direct probabilistic interpretation can be had by thinking of AE-like architectures as latent variable models. The earliest work in this direction is the density network (MacKay & Gibbs 1999), which one can view as a form of nonlinear factor analysis with the NN serving as the nonlinearity (McDonald 1962, Yalcin & Amemiya 2001). MacKay & Gibbs (1999) define a latent variable $\mathbf{z}$ and assume the data are generated by an NN-parameterized conditional distribution:

$$\mathbf{x} \sim p\left(\mathbf{x}|\mathbf{z};\mathbf{W}\right), \quad \mathbf{z} \sim p\left(\mathbf{z}\right), \quad \mathbb{E}[\mathbf{x}|\mathbf{z}] = \mathbf{g}^{-1}(\mathbf{W}_L^{\mathrm{T}}\boldsymbol{h}_{L-1}), \quad \text{and } \boldsymbol{h}_0 = \mathbf{z}, \tag{11.}$$

where $p(\mathbf{z})$ denotes a prior on the latent variable. An NN with L layers of parameters $\mathbf{W} = \{\mathbf{W}_1, \ldots, \mathbf{W}_L\}$ takes as input $\mathbf{z}$ and outputs the mean of the conditional distribution. MacKay & Gibbs (1999) used importance sampling to estimate the marginal likelihood, $p(\mathbf{x};\ \mathbf{W}) = \int_{\mathbf{z}} p(\mathbf{x}|\mathbf{z};\mathbf{W})p(\mathbf{z})\mathrm{d}\mathbf{z}$, and fit the NN weights using this objective.

Yet MacKay & Gibbs's (1999) approach does not scale to large NNs. This class of models fell out of favor until Kingma & Welling (2014) and Rezende et al. (2014) noticed that NNs could also be used to perform inference for the latent variables and the whole architecture could be trained with end-to-end differentiation. This insight leads to a unified model known as the variational autoencoder (VAE). The core idea is to define an inference network to form a posterior approximation:

$$q\left(\mathbf{z};\boldsymbol{\phi}(\boldsymbol{x})\right) \approx p(\mathbf{z}|\mathbf{x}), \quad \boldsymbol{\phi}(\boldsymbol{x}) = \mathbf{g}^{-1}(\mathbf{U}_L^{\mathrm{T}}\boldsymbol{h}'_{L-1}), \quad \text{and } \boldsymbol{h}'_0 = \boldsymbol{x} \tag{12.}$$

where $\boldsymbol{\phi}(\boldsymbol{x})$ are the parameters of the posterior approximation (as a function of a given $\boldsymbol{x}$) and $\mathbf{U}_1, \ldots, \mathbf{U}_L$ are the parameters of the inference NN. Both networks can be trained simultaneously using a reparameterized stochastic evidence lower bound:

$$\begin{aligned} p(\mathbf{x};\mathbf{W}) &\geq \mathbb{E}_{q(\mathbf{z};\boldsymbol{\phi}(\boldsymbol{x}))}\left[\log p\left(\mathbf{x}|\mathbf{z};\mathbf{W}\right)\right] - \mathrm{KLD}\left[q(\mathbf{z};\boldsymbol{\phi}(\boldsymbol{x}))||p(\mathbf{z})\right] \\ &= \mathbb{E}_{q(\boldsymbol{\epsilon})}\left[\log p\left(\mathbf{x}|\mathbf{r}(\boldsymbol{\epsilon};\boldsymbol{\phi}(\boldsymbol{x}));\mathbf{W}\right)\right] - \mathrm{KLD}\left[q(\mathbf{z};\boldsymbol{\phi}(\boldsymbol{x}))||p(\mathbf{z})\right] \\ &\approx \frac{1}{S}\sum_{s=1}^{S} \log p\left(\mathbf{x}|\mathbf{r}(\hat{\boldsymbol{\epsilon}}_s;\boldsymbol{\phi}(\boldsymbol{x}));\mathbf{W}\right) - \mathrm{KLD}\left[q(\mathbf{z};\boldsymbol{\phi}(\boldsymbol{x}))||p(\mathbf{z})\right], \end{aligned} \tag{13.}$$

where s indexes the samples in the Monte Carlo expectation and $\mathrm{KLD}[q(\mathbf{z};\boldsymbol{\phi})||p(\mathbf{z})]$ denotes the Kullback–Leibler divergence between the approximate posterior and the prior. Most crucially, $\mathbf{r}(\boldsymbol{\epsilon};\boldsymbol{\phi}(\boldsymbol{x}))$ represents a reparameterization that allows us to draw samples from $q(\mathbf{z};\boldsymbol{\phi}(\boldsymbol{x}))$ via a fixed distribution $q(\boldsymbol{\epsilon})$. One example of such a function is the location-scale form for Normals: $\hat{\mathbf{z}} = \mathbf{r}(\hat{\boldsymbol{\epsilon}};\boldsymbol{\mu}_\phi(\boldsymbol{x}),\boldsymbol{\sigma}_\phi(\boldsymbol{x})) = \boldsymbol{\mu}_\phi(\boldsymbol{x}) + \boldsymbol{\sigma}_\phi(\boldsymbol{x}) \odot \hat{\boldsymbol{\epsilon}}$, where $\hat{\boldsymbol{\epsilon}} \sim \mathrm{N}(\mathbf{0}, \mathbb{1})$. Another example would be inverse transform sampling using $q(\mathbf{z})$'s CDF. Representing the stochastic variable $\mathbf{z}$ in this way allows for end-to-end differentiation, as we now have access to the partials with respect to the inference network's parameters: $\partial\hat{\mathbf{z}}/\partial\mathbf{U}_l = (\partial\hat{\mathbf{z}}/\partial\boldsymbol{\phi})(\partial\boldsymbol{\phi}/\partial\boldsymbol{h}'_L)\ldots(\partial\boldsymbol{h}'_l/\partial\mathbf{U}_l)$. **Figure 8*b*** shows a diagram of the VAE, with the inference and generative networks composed via $\mathbf{r}(\hat{\boldsymbol{\epsilon}};\boldsymbol{\phi}(\boldsymbol{x}))$. When the inference and generative processes are thought of as a unified computational pipeline, the resulting structure resembles a traditional AE, which is how the VAE got its name. The VAE was one of the first modern generative models that showed a compelling ability to generate high-fidelity samples, as is demonstrated in **Figure 7*a***. The VAE can also perform density estimation, but only approximately via Monte Carlo integration.

4.3. Other Types of Neural Generative Models

A variety of other deep generative models have been developed, and we briefly outline them here. One of the most popular is the generative adversarial network (GAN) (Goodfellow et al. 2014).

Annu. Rev. Stat. Appl. 2023.10:219-246. Downloaded from www.annualreviews.org
Access provided by University of California - Irvine on 09/10/23. See copyright for approved use.

GANs reformulate the task of density modeling into an adversarial game in which a generator NN tries to simulate data so that a discriminator NN cannot tell the difference between the generated and observed samples. The assumption is that if the discriminator cannot tell the two apart, then the generator must be a good model of the data. The concept is similar in spirit to approximate Bayesian computation (ABC) (Rubin 1984), which compares simulated data to the observations via some statistic or metric and retains the parameters that generated the simulation—treating them as a posterior sample—if the statistic is within some threshold. In GANs, the discriminator serves as the metric comparing the fake and real data. The major difference between ABC and GANs is that GANs are trained by differentiating through the adversarial process, treating it as an optimization objective. Mohamed & Lakshminarayanan (2017) discuss GANs from a generalized framework, showing various proper scoring rules resulting in valid discriminators. The GAN framework can also be used for approximate inference for model parameters (Mescheder et al. 2017, Tran et al. 2017), although using GANs for inference is made difficult by their inability to provide a density estimate.

Another type of neural generative model is the normalizing flow (NF) (Tabak & Turner 2013, Rezende & Mohamed 2015, Papamakarios et al. 2021). These models use NNs to reparameterize a simple distribution into one with richer complexity. Specifically, the data density $p(\boldsymbol{x})$ is modeled as $p(\mathbf{x}; \boldsymbol{\psi}) = p_z(T_{\boldsymbol{\psi}}^{-1}(\mathbf{x}))\,|\partial\, T_{\boldsymbol{\psi}}^{-1}/\partial \mathbf{x}|$, where $p_z(\mathbf{z})$ is the simple base distribution that is being reparameterized via the NN function $T_{\boldsymbol{\psi}}$, and where $\boldsymbol{\psi}$ are the parameters (weights) of the NN. After performing maximum likelihood estimation for $\boldsymbol{\psi}$, samples can be drawn via $\hat{\boldsymbol{z}} \sim p(\mathbf{z}), \hat{\boldsymbol{x}} = T_{\boldsymbol{\psi}}(\hat{\boldsymbol{z}})$. The NNs are carefully designed so that the volume element $|\partial\, T_{\boldsymbol{\psi}}^{-1} \partial \mathbf{x}|$ is easy to compute (i.e., does not require computing an arbitrary Jacobian determinant). For instance, autoregressive flows allow for a triangular Jacobian matrix whose determinant is just the product of the diagonal terms (Kingma et al. 2016, Papamakarios et al. 2017, Huang et al. 2018). The images shown in **Figure 7*b*** were generated by a particular NF model known as a Glow (Kingma & Dhariwal 2018). NFs have the added benefit that their density function can usually be evaluated quickly, as is necessary for model fitting.

Two final types of popular deep generative models are diffusion models (DFs) and energy-based models (EBMs). For DFs, like with NFs, the underlying idea is to transform a simple distribution $\mathbf{x}_0 \sim p_0(\mathbf{x}_0)$ into a richer distribution that can model real-world data sources. Whereas NFs use one deterministic transformation, DFs instead use a series of conditional distributions: $p(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{\mathrm{T}} p(\mathbf{x}_t|\mathbf{x}_{t-1})$. EBMs are the most general class as they define a Gibbs density: $p(\mathbf{x}) = \exp\{-\mathrm{E}(\mathbf{x})\}/\int_{\boldsymbol{x}} \exp\{-\mathrm{E}(\mathbf{x})\}\mathrm{d}\mathbf{x}$, where $\mathrm{E}(\mathbf{x}) \in \mathbb{R}$ is known as an energy function and the denominator is the normalizing constant. While the energy function can be defined in many ways, recent work using classifiers to define the free energy has been shown to be quite effective (Grathwohl et al. 2019).

5. SELECTED TOPICS FROM THE RESEARCH FRONTIER

Our review above touches on some of the more well-established aspects of DL. In this final section of the article, we discuss additional topics in DL that involve a number of open research problems that may be of particular interest to statisticians.

5.1. Theories of Deep Learning

Attempts to theoretically characterize DNNs are primarily concerned with their expressive power, the characteristics of their optimization landscape, and their ability to generalize to unseen data. Regarding expressivity, Cybenko (1989) showed that sigmoidal architectures are universal approximators. These approximation results, however, may require the NN to have an exponential

Annu. Rev. Stat. Appl. 2023.10:219-246. Downloaded from www.annualreviews.org
Access provided by University of California - Irvine on 09/10/23. See copyright for approved use.

number of hidden units, and there have been recent attempts to characterize the depth (Yarotsky 2017) and width (Lu et al. 2017) required to achieve a given approximation level. Moreover, there are parallel attempts to understand the differences in the classes of functions able to be represented by deep versus shallow networks. Results such as those of Baldi & Vershynin (2019) and Eldan & Shamir (2016) suggest that deep networks represent fewer, but more sophisticated, functions than shallow counterparts.

While DNNs have been shown to be universal approximators for some time, these results do not guarantee anything about the class of functions that can be reached by SGD. Thus, there has been much interest in studying the optimization landscape of these models. For many years, it was thought that NN optimization would be hopelessly plagued by local minima (Cheng & Titterington 1994). However, this concern has been alleviated, to a degree, with more recent conjectures that it is not local minima but saddle points that comprise many of the loss surface's critical points (Dauphin et al. 2014, Kawaguchi 2016). The intuition is that it is unlikely that the optimization surface will be going the same direction in every dimension, as is necessary to build a local minimum. In consequence, much attention has been given to escaping saddle points efficiently (Jin et al. 2017). In addition to classifying critical points, the qualities of the minima are also of interest. In particular, whether minima are wide and flat versus narrow and sharp has been of keen interest (Hochreiter & Schmidhuber 1997a, Keskar et al. 2017). The intuition is that wide minima are likely to generalize to never-before-seen data since there is a neighborhood of parameters that represent roughly equivalent solutions.

Lastly, understanding the mystery of generalization performance of DNNs is another very active research topic: While DNNs as a model may be expressive and can capture expressive functions through optimization, how do they avoid overfitting? Methods for determining model complexity by counting the number of parameters (e.g., information criterions) are notably ineffective for determining whether NNs are likely to be overfit to a training set. In fact, the classic bias-variance trade-off has been shown to break down for NNs. Recent results have shown the existence of a double descent curve. Consider plotting generalization (test) error for a deep network as a function of model complexity (e.g., number of parameters or weights). As model complexity increases (on the x-axis), the generalization error (on the y-axis) exhibits the expected bias-variance U-shaped trade-off as underfitting gives way to overfitting. However, once the complexity is at the point where the model has fully interpolated the data, the generalization error can again decrease (hence, double descent) and decrease to the point that the best model (in terms of generalization error) has far more parameters than data points. This type of phenomenon has been observed in the past for overparameterized models (Duin 2000), and the general topic of double-descent is now a very active area of research in DL (Belkin et al. 2020, Nakkiran et al. 2021, Viering & Loog 2021), although theoretical progress (not surprisingly) has primarily occurred in terms of understanding this phenomenon with simpler nonneural models (Hastie et al. 2022, Bartlett et al. 2020, Mei & Montanari 2022).

5.2. Interpretability, Causality, Fairness, and Trustworthiness

DNNs are often criticized for being black boxes. The complexity of a typical DNN makes it difficult to extract an understanding of how it makes predictions, when or why it may perform poorly, and what assumptions are baked into the model (Lipton 2018). Recent work in interpretability (Doshi-Velez & Kim 2017, Guidotti et al. 2018) can be broadly understood under three main directions: developing methods to better understand existing architectures, designing models that by construction are more interpretable, and designing methods to investigate the data that influence the fit of a model. As an example of the first, one can examine the gradient of an NN with respect

Annu. Rev. Stat. Appl. 2023.10:219-246. Downloaded from www.annualreviews.org
Access provided by University of California - Irvine on 09/10/23. See copyright for approved use.

to its input features in order to understand their importance for prediction (Simonyan et al. 2014). As an example of the second, the knowledge encoded by an NN can be approximated by a decision tree with the hope of getting the predictive power of the former and the interpretability of the latter (Letham et al. 2015). Thirdly, Aamodt & Plaza (1994) and Kim et al. (2016) use statistical tools in model criticism in order to find patterns in the data not explained by prototypical examples. This delivers insights into parts of the input space that do not provide good explanations.

Related to explainability is the notion of causal inference (Pearl 2009). As causal inference relies on flexible function approximation, DL provides a toolbox of methods that are attractive to plug into existing semiparametric inferential frameworks. For instance, within the potential outcomes framework, Shi et al. (2019) propose an NN for estimating treatment effects. In the structural framework, Xia et al. (2021) propose an NN-based structural causal model. Looking forward, Schölkopf et al. (2021) highlight future directions, including that of using the representation learning capabilities of NNs to identify high-level causal variables from low-level observations.

There is also significant interest in the fairness of DL models, where the goal is to ensure nondiscrimination, due process, and understandability in decision-making (Zemel et al. 2013, Mehrabi et al. 2021). Policymakers, regulators, and advocates have expressed fears about the potentially discriminatory impact of machine learning, with many calling for further technical research into the dangers of inadvertently encoding bias into automated decisions. Recent work poses the problem under causal inference (Kusner et al. 2017) where evaluating the fairness of a model can be formalized as reasoning about counterfactuals such as how a classifier may change predictions if the demographic group or gender of the predicted individual were different. Of particular relevance to DL is bias, in terms of predictive disparities in a model due to underrepresentation of certain demographic groups. Given that DL models for images and text are often trained on millions or billions of examples, this bias can be implicit in data sets and hard to detect and remove, which has led to recent interest in debiasing methodologies for DL (Savani et al. 2020). Also of relevance are the notions of differential privacy (Dwork 2011) and differential fairness (Foulds et al. 2020), which aim to bound the effect of including different data points or features, respectively, on the model fit.

Since DNNs overwhelmingly are used to parameterize conditional distributions, there is perhaps even more concern about ensuring these models receive only proper inputs—that is, inputs that are drawn from the same distribution as the original training set. NN verification (a.k.a. validation) has received attention since the early 1990s (Bishop 1994), and most approaches to this problem take the perspective of satisfiability (Zakrzewski 2001), showing that the DNN's error is bounded. Another approach is to derive theoretical guarantees on the robustness (usually taking the form of class prediction invariance) within regions of input feature space (Wong & Kolter 2018, Zhang et al. 2019). This line of work is especially relevant for defending against adversarial examples (Goodfellow et al. 2015), small (imperceptible) input perturbations that are designed to result in an incorrect prediction. Another popular trend is to expose the model to samples that are unlike the training set and optimize so that the model's predictive distribution is highly entropic for these samples (Malinin & Gales 2018, Hafner et al. 2019, Hendrycks et al. 2019).

5.3. Hierarchical Modeling and Meta-Learning

As in statistics (for example, in Bayesian hierarchical modeling), the development of hierarchical modeling frameworks, which allow for the sharing of knowledge and statistical strength across data sets and subtasks, is another active research area in DL. Given that NNs are simply nonlinear functions, they can be incorporated into hierarchical Bayesian modeling by using them to parameterize a random variable at one level as a function of a higher-level random variable. We

Annu. Rev. Stat. Appl. 2023.10:219-246. Downloaded from www.annualreviews.org
Access provided by University of California - Irvine on 09/10/23. See copyright for approved use.

covered perhaps the simplest instantiation of this in Section 4.2 with the VAE. Johnson et al. (2016) extended the idea further so that general graph structures can be used to define the latent random variables. The DL concepts of meta-learning (Finn 2018) and learning to learn (Heskes 2000, Andrychowicz et al. 2016) have a less rigorous tie to hierarchical modeling in statistics but still share similarities. To give an example of one variant, episodic meta-learning (Lake et al. 2015, Santoro et al. 2016, Finn et al. 2017, Ravi & Larochelle 2017) aims to define and estimate models that can generalize to several tasks, including tasks with very little data or tasks unlike the tasks on which it was trained (but that still share some conceptual overlap). Meta-learning approaches often use task-specific models, and these specialized models have some form of tied parameterization to allow for information sharing across tasks. One way to generate these task-specific models is to use a hypernetwork (Ha et al. 2017), an NN that outputs the parameters of another NN.

6. CONCLUSION

During our brief tour of DL, we have presented the foundations of feedforward, sequential, and unsupervised architectures. While the particular details are sure to change going forward, DL will continue to thrive when prediction is the primary task and a hierarchy of representations is needed to extract signal from data. Despite its success, innovations in DL are still needed in order for it to keep pace with requirements such as interpretability, uncertainty quantification, reliability, and safety, as dictated by modern applications. From autonomous driving to finance to health care, tried and true methods from statistics, such as model validation and criticism, are likely to be very useful in deploying DL models with confidence. Given that DL operates at new scales (in both model and data size) that are not yet common in statistics, the field of statistics has the opportunity to enrich itself by engaging with these new challenges. We hope our article facilitates such discussions, bringing about innovation at the intersection of statistics, data science, and DL.

DISCLOSURE STATEMENT

The authors are not aware of any affiliations, memberships, funding, or financial holdings that might be perceived as affecting the objectivity of this review.

ACKNOWLEDGMENTS

The authors thank David Blei, Yee Whye Teh, and Brian Vegetabile for discussions and feedback in the early development of this article. Alex Boyd assisted in providing code and examples for neural sequence models. The work of P.S. was supported by the US National Science Foundation under awards 1925741, 1900644, 1927245, 1633631, and 1839336, by the HPI Research Center in Machine Learning and Data Science at UC Irvine, and by a Qualcomm Faculty Award.

LITERATURE CITED

Aamodt A, Plaza E. 1994. Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Commun.* 7(1):39–59

Allamanis M, Brockschmidt M, Khademi M. 2017. Learning to represent programs with graphs. arXiv:1711.00740 [cs.LG]

Andrychowicz M, Denil M, Gomez S, Hoffman MW, Pfau D, et al. 2016. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems 29 (NIPS 2016)*, ed. D Lee, M Sugiyama, U Luxburg, I Guyon, R Garnett, pp. 3988–96. Red Hook, NY: Curran

Annu. Rev. Stat. Appl. 2023.10:219-246. Downloaded from www.annualreviews.org
Access provided by University of California - Irvine on 09/10/23. See copyright for approved use.

Annu. Rev. Stat. Appl. 2023.10:219-246. Downloaded from www.annualreviews.org
Access provided by University of California - Irvine on 09/10/23. See copyright for approved use.

Angelopoulos AN, Bates S, Jordan M, Malik J. 2020. *Uncertainty sets for image classifiers using conformal prediction*. Presented at International Conference on Learning Representations, virtual, April 26–May 1

Ba JL, Kiros JR, Hinton GE. 2016. Layer normalization. arXiv:1607.06450 [stat.ML]

Bahdanau D, Cho K, Bengio Y. 2015. *Neural machine translation by jointly learning to align and translate*. Presented at International Conference on Learning Representations, San Diego, CA, May 7–9

Baldi P, Hornik K. 1989. Neural networks and principal component analysis: learning from examples without local minima. *Neural Netw.* 2(1):53–58

Baldi P, Vershynin R. 2019. The capacity of feedforward neural networks. *Neural Netw.* 116:288–311

Bartlett PL, Long PM, Lugosi G, Tsigler A. 2020. Benign overfitting in linear regression. *PNAS* 117(48):30063–70

Bartlett PL, Montanari A, Rakhlin A. 2021. Deep learning: a statistical viewpoint. *Acta Numer.* 30:87–201

Becker S, LeCun Y. 1989. Improving the convergence of back-propagation learning with second order methods. In *Proceedings of the 1988 Connectionist Models Summer School*, ed. D Touretzky, G Hinton, T Sejnowski, pp. 29–37. San Francisco, CA: Morgan Kaufmann

Belkin M, Hsu D, Xu J. 2020. Two models of double descent for weak features. *SIAM J. Math. Data Sci.* 2(4):1167–80

Bengio Y, Courville A, Vincent P. 2013a. Representation learning: a review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intel.* 35(8):1798–828

Bengio Y, Yao L, Alain G, Vincent P. 2013b. Generalized denoising auto-encoders as generative models. In *Advances in Neural Information Processing Systems 26 (NIPS 2013)*, ed. C Burges, L Bottou, M Welling, Z Ghahramani, K Weinberger, pp. 899–907. Red Hook, NY: Curran

Bishop CM. 1994. Novelty detection and neural network validation. *IEEE Proc. Vis. Image Signal Proc.* 141(4):217–22

Bommasani R, Hudson DA, Adeli E, Altman R, Arora S, et al. 2022. On the opportunities and risks of foundation models. arXiv:2108.07258 [cs.LG]

Bottou L. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, ed. Y Lechevallier, G Saporta, pp. 177–86. Berlin: Springer

Bourlard H, Kamp Y. 1988. Auto-association by multilayer perceptrons and singular value decomposition. *Biol. Cybernet.* 59(4–5):291–94

Breiman L. 2001. Statistical modeling: the two cultures. *Stat. Sci.* 16(3):199–231

Brown T, Mann B, Ryder N, Subbiah M, Kaplan JD, et al. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, ed. I Guyon, UV Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, R Garnett, pp. 1877–901. Red Hook, NY: Curran

Chen RT, Amos B, Nickel M. 2020. *Neural spatio-temporal point processes*. Presented at International Conference on Learning Representations, virtual, April 26–May 1

Chen RT, Rubanova Y, Bettencourt J, Duvenaud DK. 2018. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*, ed. S Bengio, H Wallach, H Larochelle, K Grauman, N Cesa-Bianchi, R Garnett, pp. 6572–83. Red Hook, NY: Curran

Cheng B, Titterington M. 1994. Neural networks: a review from a statistical perspective. *Stat. Sci.* 9(1):2–30

Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, et al. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv:1406.1078 [cs.CL]

Cohen TS, Geiger M, Köhler J, Welling M. 2018. *Spherical CNNs*. Presented at International Conference on Learning Representations, Vancouver, Apr. 30–May 3

Cottrell GW. 1989. Image compression by back propagation: a demonstration of extensional programming. *Models Cogn.* 3:208–40

Cybenko G. 1989. Approximation by superpositions of a sigmoidal function. *Math. Control Signals Syst.* 2(4):303–14

Dahl GE, Yu D, Deng L, Acero A. 2012. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Trans. Audio Speech Lang. Proc.* 20(1):30–42

Dauphin YN, Pascanu R, Gulcehre C, Cho K, Ganguli S, Bengio Y. 2014. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, ed. Z Ghahramani, M Welling, C Cortes, N Lawrence, K Weinberger, pp. 2933–41. Red Hook, NY: Curran

Annu. Rev. Stat. Appl. 2023.10:219-246. Downloaded from www.annualreviews.org
Access provided by University of California - Irvine on 09/10/23. See copyright for approved use.

Devlin J, Chang MW, Lee K, Toutanova K. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4171–86. Stroudsburg, PA: Assoc. Comput. Linguist.

Doersch C. 2021. Tutorial on variational autoencoders. arXiv:1606.05908 [stat.ML]

Doshi-Velez F, Kim B. 2017. Towards a rigorous science of interpretable machine learning. arXiv:1702.08608 [stat.ML]

Duchi J, Hazan E, Singer Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.* 12(7):2121–59

Duin RP. 2000. Classifiers in almost empty spaces. In *Proceedings of the 15th International Conference on Pattern Recognition, ICPR-2000*, Vol. 2, pp. 1–7. New York: IEEE

Dwork C. 2011. Differential privacy. In *Encyclopedia of Cryptography and Security*, ed. HCA van Tilborg, S Jajodia, pp. 338–40. Berlin: Springer

Efron B. 2020. Prediction, estimation, and attribution. *Int. Stat. Rev.* 88:S28–59

Efron B, Hastie T. 2016. *Computer Age Statistical Inference*. Cambridge, UK: Cambridge Univ. Press

Eldan R, Shamir O. 2016. The power of depth for feedforward neural networks. *PMLR* 49:907–40

Elman JL. 1990. Finding structure in time. *Cogn. Sci.* 14(2):179–211

Fan J, Ma C, Zhong Y. 2021. A selective overview of deep learning. *Stat. Sci.* 36(2):264

Finn C. 2018. *Learning to learn with gradients*. PhD Thesis, University of California, Berkeley

Finn C, Abbeel P, Levine S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. *PMLR* 70:1126–35

Foulds JR, Islam R, Keya KN, Pan S. 2020. An intersectional definition of fairness. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pp. 1918–21. New York: IEEE

Gilmer J, Schoenholz SS, Riley PF, Vinyals O, Dahl GE. 2017. Neural message passing for quantum chemistry. *PMLR* 70:1263–72

Goodfellow I, Bengio Y, Courville A. 2016. *Deep Learning*. Cambridge, MA: MIT Press

Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, et al. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, ed. Z Ghahramani, M Welling, C Cortes, N Lawrence, K Weinberger, pp. 2672–80. Red Hook, NY: Curran

Goodfellow I, Shlens J, Szegedy C. 2015. *Explaining and harnessing adversarial examples*. Presented at International Conference on Learning Representations, San Diego, CA, May 7–9

Grathwohl W, Wang KC, Jacobsen JH, Duvenaud D, Norouzi M, Swersky K. 2019. *Your classifier is secretly an energy based model and you should treat it like one.* Presented at International Conference on Learning Representations, New Orleans, LA, May 6–9

Graves A. 2012. Sequence transduction with recurrent neural networks. arXiv:1211.3711 [cs.NE]

Grigorescu S, Trasnea B, Cocias T, Macesanu G. 2020. A survey of deep learning techniques for autonomous driving. *J. Field Robot.* 37(3):362–86

Guidotti R, Monreale A, Ruggieri S, Turini F, Giannotti F, Pedreschi D. 2018. A survey of methods for explaining black box models. *ACM Comput. Surv.* 51(5):93

Guo C, Pleiss G, Sun Y, Weinberger KQ. 2017. On calibration of modern neural networks. *PMLR* 70:1321–30

Ha D, Dai A, Le QV. 2017. *Hypernetworks*. Presented at International Conference on Learning Representations, Toulon, France, Apr. 24–26

Hafner D, Tran D, Lillicrap T, Irpan A, Davidson J. 2019. Noise contrastive priors for functional uncertainty. *PMLR* 115:894–904

Halevy A, Norvig P, Pereira F. 2009. The unreasonable effectiveness of data. *IEEE Intel. Syst.* 24(2):8–12

Hastie T, Montanari A, Rosset S, Tibshirani RJ. 2022. Surprises in high-dimensional ridgeless least squares interpolation. *Ann. Stat.* 50(2):949–86

He K, Zhang X, Ren S, Sun J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–78. New York: IEEE

Hendrycks D, Mazeika M, Dietterich T. 2019. *Deep anomaly detection with outlier exposure.* Presented at International Conference on Learning Representations, New Orleans, LA, May 6–9

Heskes T. 2000. Empirical Bayes for learning to learn. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, ed. P Langley, pp. 367–74. San Francisco, CA: Morgan Kaufmann

Hewamalage H, Bergmeir C, Bandara K. 2021. Recurrent neural networks for time series forecasting: current status and future directions. *Int. J. Forecast.* 37(1):388–427

Hinton GE, Salakhutdinov RR. 2006. Reducing the dimensionality of data with neural networks. *Science* 313(5786):504–7

Hochreiter S, Schmidhuber J. 1997a. Flat minima. *Neural Comput.* 9(1):1–42

Hochreiter S, Schmidhuber J. 1997b. Long short-term memory. *Neural Comput.* 9(8):1735–80

Huang CW, Krueger D, Lacoste A, Courville A. 2018. Neural autoregressive flows. *PMLR* 80:2078–87

Ioffe S, Szegedy C. 2015. Batch normalization: accelerating deep network training by reducing internal covariate shift. *PMLR* 37:448–56

Izmailov P, Vikram S, Hoffman MD, Wilson AGG. 2021. What are Bayesian neural network posteriors really like? *PMLR* 139:4629–40

Jin C, Ge R, Netrapalli P, Kakade SM, Jordan MI. 2017. How to escape saddle points efficiently. *PMLR* 70:1724–32

Johnson M, Duvenaud DK, Wiltschko A, Adams RP, Datta SR. 2016. Composing graphical models with neural networks for structured representations and fast inference. In *Advances in Neural Information Processing Systems 29 (NIPS 2016)*, ed. D Lee, M Sugiyama, U Luxburg, I Guyon, R Garnett, pp. 2954–62. Red Hook, NY: Curran

Jumper J, Evans R, Pritzel A, Green T, Figurnov M, et al. 2021. Highly accurate protein structure prediction with Alphafold. *Nature* 596(7873):583–89

Jurafsky D, Martin JH. 2022. *Speech and Language Processing*. Englewood Cliffs, NJ: Pearson. 3rd ed.

Kawaguchi K. 2016. Deep learning without poor local minima. In *Advances in Neural Information Processing Systems 29 (NIPS 2016)*, ed. D Lee, M Sugiyama, U Luxburg, I Guyon, R Garnett, pp. 586–94. Red Hook, NY: Curran

Keskar NS, Mudigere D, Nocedal J, Smelyanskiy M, Tang PTP. 2017. *On large-batch training for deep learning: generalization gap and sharp minima*. Presented at International Conference on Learning Representations, Toulon, France, Apr. 24–26

Kim B, Khanna R, Koyejo OO. 2016. Examples are not enough, learn to criticize! Criticism for interpretability. In *Advances in Neural Information Processing Systems 29 (NIPS 2016)*, ed. D Lee, M Sugiyama, U Luxburg, I Guyon, R Garnett, pp. 2288–96. Red Hook, NY: Curran

Kingma D, Ba J. 2014. *Adam: a method for stochastic optimization.* Presented at International Conference on Learning Representations, Banff, Canada, Apr. 14–16

Kingma D, Dhariwal P. 2018. Glow: generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*, ed. S Bengio, H Wallach, H Larochelle, K Grauman, N Cesa-Bianchi, R Garnett, pp. 10236–45. Red Hook, NY: Curran

Kingma D, Salimans T, Jozefowicz R, Chen X, Sutskever I, Welling M. 2016. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems 29 (NIPS 2016)*, ed. D Lee, M Sugiyama, U Luxburg, I Guyon, R Garnett, pp. 4743–51. Red Hook, NY: Curran

Kingma D, Welling M. 2014. *Auto-encoding variational Bayes.* Presented at International Conference on Learning Representations, Banff, Canada, Apr. 14–16

Klambauer G, Unterthiner T, Mayr A, Hochreiter S. 2017. Self-normalizing neural networks. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, ed. I Guyon, UV Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, R Garnett, pp. 972–81. Red Hook, NY: Curran

Krishnan R, Shalit U, Sontag D. 2017. Structured inference networks for nonlinear state space models. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pp. 2101–9. Menlo Park, CA: AAAI Press

Krizhevsky A, Sutskever I, Hinton GE. 2012. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25 (NIPS 2012)*, ed. F Pereira, C Burges, L Bottou, K Weinberger, pp. 1097–105. Red Hook, NY: Curran

Kusner MJ, Loftus J, Russell C, Silva R. 2017. Counterfactual fairness. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, ed. I Guyon, UV Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, R Garnett, pp. 4069–79. Red Hook, NY: Curran

Lake BM, Salakhutdinov R, Tenenbaum JB. 2015. Human-level concept learning through probabilistic program induction. *Science* 350(6266):1332–38

Annu. Rev. Stat. Appl. 2023.10:219-246. Downloaded from www.annualreviews.org
Access provided by University of California - Irvine on 09/10/23. See copyright for approved use.

Annu. Rev. Stat. Appl. 2023.10:219-246. Downloaded from www.annualreviews.org
Access provided by University of California - Irvine on 09/10/23. See copyright for approved use.

Lakshminarayanan B, Pritzel A, Blundell C. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, ed. I Guyon, UV Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, R Garnett, pp. 6405–16. Red Hook, NY: Curran

Le Cun Y. 1986. Learning process in an asymmetric threshold network. In *Disordered Systems and Biological Organization*, ed. E Bienenstock, FF Soulié, G Weisbuch, pp. 233–40. Berlin: Springer

LeCun Y, Bengio Y, Hinton G. 2015. Deep learning. *Nature* 521(7553):436–44

LeCun Y, Boser B, Denker J, Henderson D, Howard R, et al. 1989. Handwritten digit recognition with a back-propagation network. In *Advances in Neural Information Processing Systems (NIPS 1989)*, Vol. 2, ed. D Touretzky, pp. 396–404. San Francisco, CA: Morgan Kaufmann

LeCun Y, Bottou L, Bengio Y, Haffner P. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86(11):2278–324

Lee HK. 2004. *Bayesian Nonparametrics via Neural Networks*. Philadelphia, PA: ASA, SIAM

Letham B, Rudin C, McCormick TH, Madigan D. 2015. Interpretable classifiers using rules and Bayesian analysis: building a better stroke prediction model. *Ann. Appl. Stat.* 9(3):1350–71

Lim B, Zohren S. 2021. Time-series forecasting with deep learning: a survey. *Philos. Trans. R. Soc. A* 379(2194):20200209

Lipton ZC. 2018. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue* 16(3):31–57

Lu Z, Pu H, Wang F, Hu Z, Wang L. 2017. The expressive power of neural networks: a view from the width. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, ed. I Guyon, UV Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, R Garnett, pp. 6232–40. Red Hook, NY: Curran

Maas AL, Hannun AY, Ng AY. 2013. Rectifier nonlinearities improve neural network acoustic models. In *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*. N.p.: JMLR

MacKay DJ. 1992. *Bayesian methods for adaptive models*. PhD Thesis, Calif. Inst. Technol., Pasadena, CA

MacKay DJ, Gibbs MN. 1999. Density networks. In *Statistics and Neural Networks: Advances at the Interface*, ed. JW Kay, DM Titterington, pp. 129–46. Oxford, UK: Oxford Univ. Press

Malinin A, Gales M. 2018. Predictive uncertainty estimation via prior networks. In *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*, ed. S Bengio, H Wallach, H Larochelle, K Grauman, N Cesa-Bianchi, R Garnett, pp. 7047–58. Red Hook, NY: Curran

Manning CD. 2015. Computational linguistics and deep learning. *Comput. Linguist.* 41(4):701–7

McClelland JL, Hill F, Rudolph M, Baldridge J, Schütze H. 2020. Placing language in an integrated understanding system: next steps toward human-level performance in neural language models. *PNAS* 117(42):25966–74

McCulloch WS, Pitts W. 1943. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.* 5(4):115–33

McDermott PL, Wikle CK. 2019. Bayesian recurrent neural network models for forecasting and quantifying uncertainty in spatial-temporal data. *Entropy* 21(2):184

McDonald RP. 1962. A general approach to nonlinear factor analysis. *Psychometrika* 27(4):397–415

Mehrabi N, Morstatter F, Saxena N, Lerman K, Galstyan A. 2021. A survey on bias and fairness in machine learning. *ACM Comput. Surv.* 54(6):1–35

Mei H, Eisner JM. 2017. The neural Hawkes process: a neurally self-modulating multivariate point process. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, ed. I Guyon, UV Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, R Garnett, pp. 6757–67. Red Hook, NY: Curran

Mei S, Montanari A. 2022. The generalization error of random features regression: precise asymptotics and the double descent curve. *Commun. Pure Appl. Math.* 75(4):667–766

Mescheder LM, Nowozin S, Geiger A. 2017. Adversarial variational Bayes: unifying variational autoencoders and generative adversarial networks. *PMLR* 70:2391–400

Mohamed S. 2015. *A statistical view of deep learning*. Work. Pap. **http://blog.shakirm.com/wp-content/uploads/2015/07/SVDL.pdf**

Mohamed S, Lakshminarayanan B. 2017. Learning in implicit generative models. arXiv:1610.03483 [stat.ML]

Murphy KP. 2022. *Probabilistic Machine Learning: An Introduction*. Cambridge, MA: MIT Press

Nakkiran P, Kaplun G, Bansal Y, Yang T, Barak B, Sutskever I. 2021. Deep double descent: where bigger models and more data hurt. *J. Stat. Mech. Theory Exp.* 2021(12):124003

Neal RM. 1994. *Bayesian Learning for Neural Networks*. PhD Thesis, Univ. Toronto, Canada

Papamakarios G, Nalisnick E, Rezende DJ, Mohamed S, Lakshminarayanan B. 2021. Normalizing flows for probabilistic modeling and inference. *J. Mach. Learn. Res.* 22(1):1–64

Papamakarios G, Pavlakou T, Murray I. 2017. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, ed. I Guyon, UV Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, R Garnett, pp. 2335–44. Red Hook, NY: Curran

Parker DB. 1985. *Learning-logic.* Tech. Rep. TR-47, Cent. Comput. Res. Econ. Manag. Sci., MIT, Cambridge, MA

Parker DB. 1987. Optimal algorithms for adaptive networks: second order back propagation, second order direct propagation, and second order Hebbian learning. In *IEEE 1st International Conference on Neural Networks, San Diego*, Vol. 2, pp. 593–600. Piscataway, NJ: IEEE

Pascanu R, Dauphin YN, Ganguli S, Bengio Y. 2014. On the saddle point problem for non-convex optimization. arXiv:1405.4604 [cs.LG]

Pearl J. 2009. Causal inference in statistics: an overview. *Stat. Surv.* 3:96–146

Polson NG, Sokolov V. 2017. Deep learning: a Bayesian perspective. *Bayesian Anal.* 12(4):1275–304

Ranganath R, Perotte A, Elhadad N, Blei D. 2016. Deep survival analysis. *PMLR* 56:101–14

Rangapuram SS, Seeger MW, Gasthaus J, Stella L, Wang Y, Januschowski T. 2018. Deep state space models for time series forecasting. In *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*, ed. S Bengio, H Wallach, H Larochelle, K Grauman, N Cesa-Bianchi, R Garnett, pp. 7796–805. Red Hook, NY: Curran

Ravi S, Larochelle H. 2017. *Optimization as a model for few-shot learning.* Presented at International Conference on Learning Representations, Toulon, France, Apr. 24–26

Rezende D, Mohamed S. 2015. Variational inference with normalizing flows. *PMLR* 37:1530–38

Rezende DJ, Mohamed S, Wierstra D. 2014. Stochastic backpropagation and approximate inference in deep generative models. *PMLR* 32:1278–86

Ripley BD. 1996. *Pattern Recognition and Neural Networks*. Cambridge, UK: Cambridge Univ. Press

Robbins H, Monro S. 1951. A stochastic approximation method. *Ann. Math. Stat.* 22(3):400–7

Rubin DB. 1984. Bayesianly justifiable and relevant frequency calculations for the applied statistician. *Ann. Stat.* 12(4):1151–72

Rumelhart DE, Hinton GE, Williams RJ. 1986. Learning representations by back-propagating errors. *Nature* 323:533–36

Salimans T, Kingma D. 2016. Weight normalization: a simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems 29 (NIPS 2016)*, ed. D Lee, M Sugiyama, U Luxburg, I Guyon, R Garnett, pp. 901–9. Red Hook, NY: Curran

Santoro A, Bartunov S, Botvinick M, Wierstra D, Lillicrap T. 2016. Meta-learning with memory-augmented neural networks. *PMLR* 48:1842–50

Savani Y, White C, Govindarajulu NS. 2020. Intra-processing methods for debiasing neural networks. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, ed. H Larochelle, M Ranzato, R Hadsell, M Balcan, H Lin, pp. 2798–810. Red Hook, NY: Curran

Schmidhuber J. 2015. Deep learning in neural networks: an overview. *Neural Netw.* 61:85–117

Schölkopf B, Locatello F, Bauer S, Ke NR, Kalchbrenner N, et al. 2021. Toward causal representation learning. *Proc. IEEE* 109(5):612–34

Shafer G, Vovk V. 2008. A tutorial on conformal prediction. *J. Mach. Learn. Res.* 9(3):371–421

Shi C, Blei D, Veitch V. 2019. Adapting neural networks for the estimation of treatment effects. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, ed. H Wallach, H Larochelle, A Beygelzimer, F d'Alché Buc, E Fox, R Garnett, pp. 2507–17. Red Hook, NY: Curran

Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, et al. 2017. Mastering the game of Go without human knowledge. *Nature* 550(7676):354–59

Simonyan K, Vedaldi A, Zisserman A. 2014. Deep inside convolutional networks: visualising image classification models and saliency maps. arXiv:1312.6034 [cs.CV]

Annu. Rev. Stat. Appl. 2023.10:219-246. Downloaded from www.annualreviews.org
Access provided by University of California - Irvine on 09/10/23. See copyright for approved use.

Srivastava N, Hinton GE, Krizhevsky A, Sutskever I, Salakhutdinov R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* 15(1):1929–58

Stern HS. 1996. Neural networks in applied statistics. *Technometrics* 38(3):205–14

Sutskever I, Vinyals O, Le QV. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, ed. Z Ghahramani, M Welling, C Cortes, N Lawrence, K Weinberger, pp. 3104–12. Red Hook, NY: Curran

Tabak EG, Turner CV. 2013. A family of nonparametric density estimation algorithms. *Commun. Pure Appl. Math.* 66(2):145–64

Theis L, van den Oord A, Bethge M. 2016. *A note on the evaluation of generative models.* Presented at International Conference on Learning Representations, San Juan, Puerto Rico, May 2–4

Tran D, Ranganath R, Blei DM. 2017. Hierarchical implicit models and likelihood-free variational inference. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, ed. I Guyon, UV Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, R Garnett, pp. 5529–39. Red Hook, NY: Curran

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, et al. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, ed. I Guyon, UV Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, R Garnett, pp. 6000–10. Red Hook, NY: Curran

Viering T, Loog M. 2021. The shape of learning curves: a review. arXiv:2103.10948 [cs.LG]

Vincent P. 2011. A connection between score matching and denoising autoencoders. *Neural Comput.* 23(7):1661–74

Vincent P, Larochelle H, Bengio Y, Manzagol PA. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, ed. W Cohen, S Roweis, A McCallum, pp. 1096–103. New York: ACM

Vincent P, Larochelle H, Lajoie I, Bengio Y, Manzagol PA. 2010. Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.* 11(12):3371–408

Wang A, Singh A, Michael J, Hill F, Levy O, Bowman SR. 2018. GLUE: a multi-task benchmark and analysis platform for natural language understanding. arXiv:1804.07461 [cs.CL]

Wang P, Li Y, Reddy CK. 2019. Machine learning for survival analysis: a survey. *ACM Comput. Surv.* 51(6):1–36

Wang Y, Smola A, Maddix D, Gasthaus J, Foster D, Januschowski T. 2019. Deep factors for forecasting. *PMLR* 97:6607–17

Welling M. 2015. *Are ML and statistics complementary?* Presented at IMS-ISBA Meeting on Data Science in the Next 50 Years, Lenzerheide, Switz., Dec. 28. **https://staff.fnwi.uva.nl/m.welling/wp-content/uploads/papers/WhyMLneedsStatistics.pdf**

White H. 1989. Learning in artificial neural networks: a statistical perspective. *Neural Comput.* 1(4):425–64

Wong E, Kolter Z. 2018. Provable defenses against adversarial examples via the convex outer adversarial polytope. *PMLR* 80:5283–92

Wu Z, Pan S, Chen F, Long G, Zhang C, Philip SY. 2020. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* 32(1):4–24

Xia K, Lee KZ, Bengio Y, Bareinboim E. 2021. The causal-neural connection: expressiveness, learnability, and inference. In *Advances in Neural Information Processing Systems 34 (NeurIPS 2021)*, ed. M Ranzato, A Beygelzimer, Y Dauphin, P Liang, JW Vaughan, pp. 10823–36. Red Hook, NY: Curran

Yalcin I, Amemiya Y. 2001. Nonlinear factor analysis as a statistical method. *Stat. Sci.* 16(3):275–94

Yarotsky D. 2017. Error bounds for approximations with deep ReLU networks. *Neural Netw.* 94:103–14

Yuan Y, Deng Y, Zhang Y, Qu A. 2020. Deep learning from a statistical perspective. *Stat* 9(1):e294

Zakrzewski R. 2001. Verification of a trained neural network accuracy. In *International Joint Conference on Neural Networks*, Vol. 3, pp. 1657–62. New York: IEEE

Zemel R, Wu Y, Swersky K, Pitassi T, Dwork C. 2013. Learning fair representations. *PMLR* 28:325–33

Zhang H, Yu Y, Jiao J, Xing E, Ghaoui LE, Jordan MI. 2019. Theoretically principled trade-off between robustness and accuracy. *PMLR* 97:7472–82

Annu. Rev. Stat. Appl. 2023.10:219-246. Downloaded from www.annualreviews.org
Access provided by University of California - Irvine on 09/10/23. See copyright for approved use.

Annual Review of Statistics and Its Application

Volume 10, 2023

Contents

Annu. Rev. Stat. Appl. 2023.10:219-246. Downloaded from www.annualreviews.org
Access provided by University of California - Irvine on 09/10/23. See copyright for approved use.

Annu. Rev. Stat. Appl. 2023.10:219-246. Downloaded from www.annualreviews.org
Access provided by University of California - Irvine on 09/10/23. See copyright for approved use.

Errata

An online log of corrections to *Annual Review of Statistics and Its Application* articles may be found at http://www.annualreviews.org/errata/statistics

Annu. Rev. Stat. Appl. 2023.10:219-246. Downloaded from www.annualreviews.org
Access provided by University of California - Irvine on 09/10/23. See copyright for approved use.