components (equation 17.4)? State the possible range of values that each parameter could take.

Problem 17.2 A function is concave if its second derivative is less than or equal to zero everywhere. Show that this is true for the function $\mathrm{g}[x] = \log[x]$.

Problem 17.3 For convex functions, Jensen's inequality works the other way around.

$$g\big[\mathbb{E}[y]\big] \leq \mathbb{E}\big[g[y]\big]. \tag{17.31}$$

A function is convex if its second derivative is greater than or equal to zero everywhere. Show that the function $g[x] = x^{2n}$ is convex for arbitrary $n \in [1, 2, 3, \ldots]$. Use this result with Jensen's inequality to show that the square of the mean $\mathbb{E}[x]$ of a distribution $Pr(x)$ must be less than or equal to its second moment $\mathbb{E}[x^2]$.

Problem 17.4* Show that the ELBO, as expressed in equation 17.18, can alternatively be derived from the KL divergence between the variational distribution $q(\mathbf{z}|\mathbf{x})$ and the true posterior distribution $Pr(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})$:

$$\mathrm{D}_{KL}\Big[q(\mathbf{z}|\mathbf{x})\Big|\Big|Pr(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})\Big] = \int q(\mathbf{z}|\mathbf{x}) \log\left[\frac{q(\mathbf{z}|\mathbf{x})}{Pr(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})}\right] d\mathbf{z}. \tag{17.32}$$

Start by using Bayes' rule (equation 17.19).

Problem 17.5 The reparameterization trick computes the derivative of an expectation of a function $\mathrm{f}[x]$:

$$\frac{\partial}{\partial \boldsymbol{\phi}} \mathbb{E}_{Pr(x|\boldsymbol{\phi})}\big[\mathrm{f}[x]\big], \tag{17.33}$$

with respect to the parameters $\boldsymbol{\phi}$ of the distribution $Pr(x|\boldsymbol{\phi})$ that the expectation is over. Show that this derivative can also be computed as:

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{\phi}} \mathbb{E}_{Pr(x|\boldsymbol{\phi})}\big[\mathrm{f}[x]\big] &= \mathbb{E}_{Pr(x|\boldsymbol{\phi})}\left[\mathrm{f}[x]\frac{\partial}{\partial \boldsymbol{\phi}}\log\big[Pr(x|\boldsymbol{\phi})\big]\right] \\ &\approx \frac{1}{I}\sum_{i=1}^{I} \mathrm{f}[x_i]\frac{\partial}{\partial \boldsymbol{\phi}}\log\big[Pr(x_i|\boldsymbol{\phi})\big].\end{aligned} \tag{17.34}$$

This method is known as the *REINFORCE algorithm* or *score function estimator*.

Problem 17.6 Why is it better to use spherical linear interpolation rather than regular linear interpolation when moving between points in the latent space? Hint: consider figure 8.13.

Problem 17.7* Derive the EM algorithm for the 1D mixture of Gaussians algorithm with N components. To do this, you need to (i) find an expression for the posterior distribution $Pr(z|x)$ over the latent variable $z \in \{1, 2, \ldots, N\}$ for a data point x and (ii) find an expression that updates the evidence lower bound given the posterior distributions for all of the data points. You will need to use Lagrange multipliers to ensure that the weights $\lambda_1, \ldots, \lambda_N$ of the Gaussians sum to one.

Chapter 18

Diffusion models

Chapter 15 described generative adversarial models, which produce plausible-looking samples but do not define a probability distribution over the data. Chapter 16 discussed normalizing flows. These do define such a probability distribution but must place architectural constraints on the network; each layer must be invertible, and the determinant of its Jacobian must be easy to calculate. Chapter 17 introduced variational autoencoders, which also have a solid probabilistic foundation but where the computation of the likelihood is intractable and must be approximated by a lower bound.

This chapter introduces diffusion models. Like normalizing flows, these are probabilistic models that define a nonlinear mapping from latent variables to the observed data where both quantities have the same dimension. Like variational autoencoders, they approximate the data likelihood using a lower bound based on an encoder that maps *to* the latent variable. However, in diffusion models, this encoder is predetermined; the goal is to learn a decoder that is the inverse of this process and can be used to produce samples. Diffusion models are easy to train and can produce very high-quality samples that exceed the realism of those produced by GANs. The reader should be familiar with variational autoencoders (chapter 17) before reading this chapter.

18.1 Overview

A diffusion model consists of an *encoder* and a *decoder*. The encoder takes a data sample $\mathbf{x}$ and maps it through a series of intermediate latent variables $\mathbf{z}_1 \ldots \mathbf{z}_T$. The decoder reverses this process; it starts with $\mathbf{z}_T$ and maps back through $\mathbf{z}_{T-1}, \ldots, \mathbf{z}_1$ until it finally (re-)creates a data point $\mathbf{x}$. In both encoder and decoder, the mappings are stochastic rather than deterministic.

The encoder is prespecified; it gradually blends the input with samples of white noise (figure 18.1). With enough steps, the conditional distribution $q(\mathbf{z}_T|\mathbf{x})$ and marginal distribution $q(\mathbf{z}_T)$ of the final latent variable both become the standard normal distribution. Since this process is prespecified, all the learned parameters are in the decoder.

In the decoder, a series of networks are trained to map backward between each

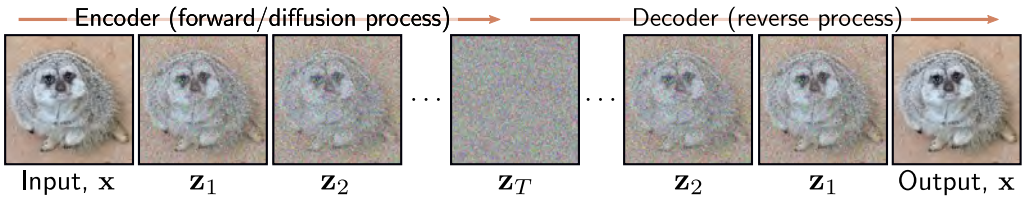


Figure 18.1 Diffusion models. The encoder (forward, or diffusion process) maps the input $\mathbf{x}$ through a series of latent variables $\mathbf{z}_1 \ldots \mathbf{z}_T$. This process is pre-specified and gradually mixes the data with noise until only noise remains. The decoder (reverse process) is learned and passes the data back through the latent variables, removing noise at each stage. After training, new examples are generated by sampling noise vectors $\mathbf{z}_T$ and passing them through the decoder.

adjacent pair of latent variables $\mathbf{z}_t$ and $\mathbf{z}_{t-1}$. The loss function encourages each network to invert the corresponding encoder step. The result is that noise is gradually removed from the representation until a realistic-looking data example remains. To generate a new data example $\mathbf{x}$, we draw a sample from $q(\mathbf{z}_T)$ and pass it through the decoder.

In section 18.2, we consider the encoder in detail. Its properties are non-obvious but are critical for the learning algorithm. In section 18.3, we discuss the decoder. Section 18.4 derives the training algorithm, and section 18.5 reformulates it to be more practical. Section 18.6 discusses implementation details, including how to make the generation conditional on text prompts.

18.2 Encoder (forward process)

The *diffusion* or *forward* process[1] (figure 18.2) maps a data example $\mathbf{x}$ through a series of intermediate variables $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_T$ with the same size as $\mathbf{x}$ according to:

$$\begin{aligned} \mathbf{z}_1 &= \sqrt{1-\beta_1}\cdot\mathbf{x} + \sqrt{\beta_1}\cdot\boldsymbol{\epsilon}_1 \\ \mathbf{z}_t &= \sqrt{1-\beta_t}\cdot\mathbf{z}_{t-1} + \sqrt{\beta_t}\cdot\boldsymbol{\epsilon}_t \qquad \forall\, t \in 2,\ldots,T, \end{aligned} \tag{18.1}$$

where $\boldsymbol{\epsilon}_t$ is noise drawn from a standard normal distribution. The first term attenuates the data plus any noise added so far, and the second adds more noise. The hyperparameters $\beta_t \in [0,1]$ determine how quickly the noise is blended and are collectively known as the *noise schedule*. The forward process can equivalently be written as:

[1]Note, this is the opposite nomenclature to normalizing flows, where the inverse mapping moves from the data to the latent variable, and the forward mapping moves back again.

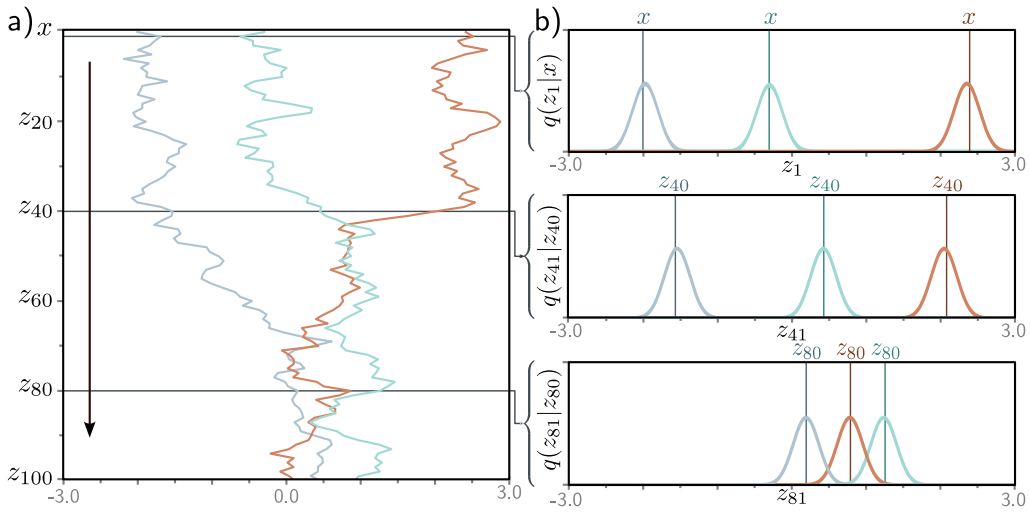


Figure 18.2 Forward process. a) We consider one-dimensional data x with $T = 100$ latent variables $z_1, \ldots, z_{100}$ and $\beta = 0.03$ at all steps. Three values of x (gray, cyan, and orange) are initialized (top row). These are propagated through $z_1, \ldots, z_{100}$. At each step, the variable is updated by attenuating its value by $\sqrt{1-\beta}$ and adding noise with mean zero and variance β (equation 18.1). Accordingly, the three examples noisily propagate through the variables with a tendency to move toward zero. b) The conditional probabilities $Pr(z_1|x)$ and $Pr(z_t|z_{t-1})$ are normal distributions with a mean that is slightly closer to zero than the current point and a fixed variance β_t (equation 18.2).

$$\begin{aligned} q(\mathbf{z}_1|\mathbf{x}) &= \text{Norm}_{\mathbf{z}_1}\left[\sqrt{1-\beta_1}\mathbf{x}, \beta_1\mathbf{I}\right] \\ q(\mathbf{z}_t|\mathbf{z}_{t-1}) &= \text{Norm}_{\mathbf{z}_t}\left[\sqrt{1-\beta_t}\mathbf{z}_{t-1}, \beta_t\mathbf{I}\right] \qquad \forall\, t \in \{2, \ldots, T\}. \end{aligned} \tag{18.2}$$

This is a *Markov chain* because the probability of $\mathbf{z}_t$ is determined entirely by the value of the immediately preceding variable $\mathbf{z}_{t-1}$. With sufficient steps T, all traces of the original data are removed, and $q(\mathbf{z}_T|\mathbf{x}) = q(\mathbf{z}_T)$ becomes a standard normal distribution.[2]

Problem 18.1

The joint distribution of all of the latent variables $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_T$ given input $\mathbf{x}$ is:

$$q(\mathbf{z}_{1\ldots T}|\mathbf{x}) = q(\mathbf{z}_1|\mathbf{x}) \prod_{t=2}^{T} q(\mathbf{z}_t|\mathbf{z}_{t-1}). \tag{18.3}$$

[2]We use $q(\mathbf{z}_t|\mathbf{z}_{t-1})$ rather than $Pr(\mathbf{z}_t|\mathbf{z}_{t-1})$ to match the notation in the description of the VAE encoder in the previous chapter.

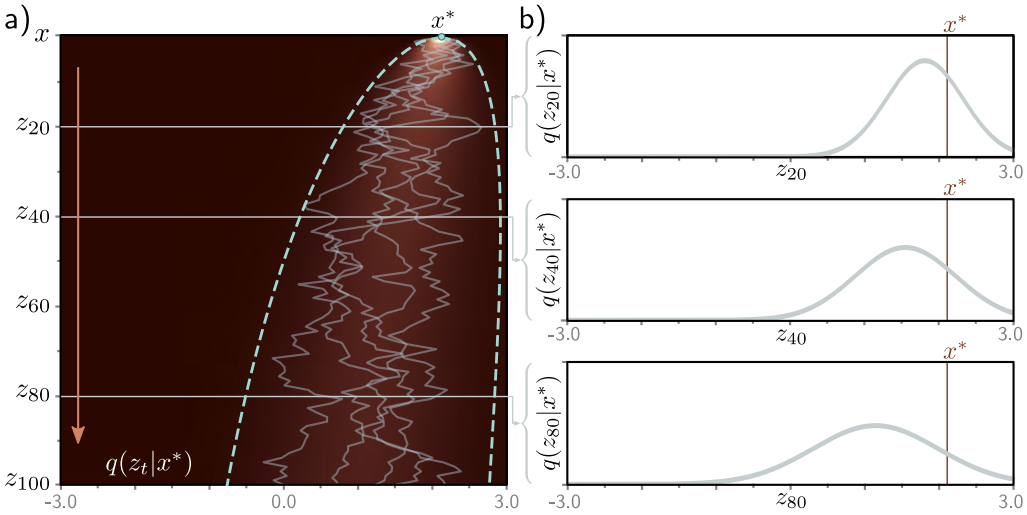


Figure 18.3 Diffusion kernel. a) The point $x^* = 2.0$ is propagated through the latent variables using equation 18.1 (five paths shown in gray). The diffusion kernel $q(z_t|x^*)$ is the probability distribution over variable z_t given that we started from x^*. It can be computed in closed-form and is a normal distribution whose mean moves toward zero and whose variance increases as t increases. Heatmap shows $q(z_t|x^*)$ for each variable. Cyan lines show ± 2 standard deviations from the mean. b) The diffusion kernel $q(z_t|x^*)$ is shown explicitly for $t = 20, 40, 80$. In practice, the diffusion kernel allows us to sample a latent variable z_t corresponding to a given x^* without computing the intermediate variables $z_1, \ldots, z_{t-1}$. When t becomes very large, the diffusion kernel becomes a standard normal.

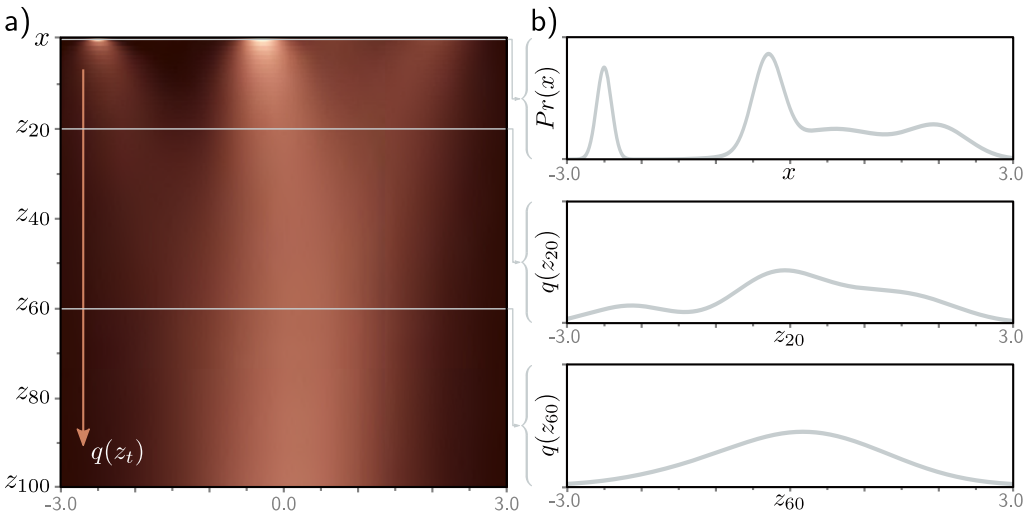


Figure 18.4 Marginal distributions. a) Given an initial density $Pr(x)$ (top row), the diffusion process gradually blurs the distribution as it passes through the latent variables z_t and moves it toward a standard normal distribution. Each subsequent horizontal line of heatmap represents a marginal distribution $q(z_t)$. b) The top graph shows the initial distribution $Pr(x)$. The other two graphs show the marginal distributions $q(z_{20})$ and $q(z_{60})$, respectively.

18.2.1 Diffusion kernel $q(\mathbf{z}_t|\mathbf{x})$

To train the decoder to invert this process, we use multiple samples $\mathbf{z}_t$ at time t for the same example $\mathbf{x}$. However, generating these sequentially using equation 18.1 is time-consuming when t is large. Fortunately, there is a closed-form expression for $q(\mathbf{z}_t|\mathbf{x})$, which allows us to directly draw samples $\mathbf{z}_t$ given initial data point $\mathbf{x}$ without computing the intermediate variables $\mathbf{z}_1 \ldots \mathbf{z}_{t-1}$. This is known as the *diffusion kernel* (figure 18.3).

To derive an expression for $q(\mathbf{z}_t|\mathbf{x})$, consider the first two steps of the forward process:

$$\begin{aligned}\mathbf{z}_1 &= \sqrt{1-\beta_1}\cdot\mathbf{x} + \sqrt{\beta_1}\cdot\boldsymbol{\epsilon}_1 \\ \mathbf{z}_2 &= \sqrt{1-\beta_2}\cdot\mathbf{z}_1 + \sqrt{\beta_2}\cdot\boldsymbol{\epsilon}_2.\end{aligned} \tag{18.4}$$

Substituting the first equation into the second, we get:

$$\begin{aligned}\mathbf{z}_2 &= \sqrt{1-\beta_2}\left(\sqrt{1-\beta_1}\cdot\mathbf{x} + \sqrt{\beta_1}\cdot\boldsymbol{\epsilon}_1\right) + \sqrt{\beta_2}\cdot\boldsymbol{\epsilon}_2 \\ &= \sqrt{1-\beta_2}\left(\sqrt{1-\beta_1}\cdot\mathbf{x} + \sqrt{1-(1-\beta_1)}\cdot\boldsymbol{\epsilon}_1\right) + \sqrt{\beta_2}\cdot\boldsymbol{\epsilon}_2 \\ &= \sqrt{(1-\beta_2)(1-\beta_1)}\cdot\mathbf{x} + \sqrt{1-\beta_2-(1-\beta_2)(1-\beta_1)}\cdot\boldsymbol{\epsilon}_1 + \sqrt{\beta_2}\cdot\boldsymbol{\epsilon}_2.\end{aligned} \tag{18.5}$$

The last two terms are independent samples from mean-zero normal distributions with variances $1-\beta_2-(1-\beta_2)(1-\beta_1)$ and β_2, respectively. The mean of this sum is zero, and its variance is the sum of the component variances (see problem 18.2), so:

Problem 18.2

$$\mathbf{z}_2 = \sqrt{(1-\beta_2)(1-\beta_1)}\cdot\mathbf{x} + \sqrt{1-(1-\beta_2)(1-\beta_1)}\cdot\boldsymbol{\epsilon}, \tag{18.6}$$

where $\boldsymbol{\epsilon}$ is also a sample from a standard normal distribution.

If we continue this process by substituting this equation into the expression for $\mathbf{z}_3$ and so on, we can show that:

Problem 18.3

$$\mathbf{z}_t = \sqrt{\alpha_t}\cdot\mathbf{x} + \sqrt{1-\alpha_t}\cdot\boldsymbol{\epsilon}, \tag{18.7}$$

where $\alpha_t = \prod_{s=1}^{t} 1-\beta_s$. We can equivalently write this in probabilistic form:

$$q(\mathbf{z}_t|\mathbf{x}) = \text{Norm}_{\mathbf{z}_t}\left[\sqrt{\alpha_t}\cdot\mathbf{x}, (1-\alpha_t)\mathbf{I}\right]. \tag{18.8}$$

For any starting data point $\mathbf{x}$, variable $\mathbf{z}_t$ is normally distributed with a known mean and variance. Consequently, if we don't care about the history of the evolution through the intermediate variables $\mathbf{z}_1 \ldots \mathbf{z}_{t-1}$, it is easy to generate samples from $q(\mathbf{z}_t|\mathbf{x})$.

18.2.2 Marginal distributions $q(\mathbf{z}_t)$

The marginal distribution $q(\mathbf{z}_t)$ is the probability of observing a value of $\mathbf{z}_t$ given the distribution of possible starting points $\mathbf{x}$ and the possible diffusion paths for each starting

point (figure 18.4). It can be computed by considering the joint distribution $q(\mathbf{x}, \mathbf{z}_{1\ldots t})$ and marginalizing over all the variables except $\mathbf{z}_t$:

Appendix C.1.2 Marginalization

$$\begin{aligned} q(\mathbf{z}_t) &= \int\!\!\int q(\mathbf{z}_{1\ldots t}, \mathbf{x}) d\mathbf{z}_{1\ldots t-1} d\mathbf{x} \\ &= \int\!\!\int q(\mathbf{z}_{1\ldots t}|\mathbf{x}) Pr(\mathbf{x}) d\mathbf{z}_{1\ldots t-1} d\mathbf{x}, \end{aligned} \tag{18.9}$$

where $q(\mathbf{z}_{1\ldots t}|\mathbf{x})$ was defined in equation 18.3.

However, since we now have an expression for the diffusion kernel $q(\mathbf{z}_t|\mathbf{x})$ that "skips" the intervening variables, we can equivalently write:

$$q(\mathbf{z}_t) = \int q(\mathbf{z}_t|\mathbf{x}) Pr(\mathbf{x}) d\mathbf{x}. \tag{18.10}$$

Hence, if we repeatedly sample from the data distribution $Pr(\mathbf{x})$ and superimpose the diffusion kernel $q(\mathbf{z}_t|\mathbf{x})$ on each sample, the result is the marginal distribution $q(\mathbf{z}_t)$ (figure 18.4). However, the marginal distribution cannot be written in closed form because we don't know the original data distribution $Pr(\mathbf{x})$.

Notebook 18.1 Diffusion encoder

18.2.3 Conditional distribution $q(\mathbf{z}_{t-1}|\mathbf{z}_t)$

We defined the conditional probability $q(\mathbf{z}_t|\mathbf{z}_{t-1})$ as the mixing process (equation 18.2). To reverse this process, we apply Bayes' rule:

Appendix C.1.4 Bayes' rule

$$q(\mathbf{z}_{t-1}|\mathbf{z}_t) = \frac{q(\mathbf{z}_t|\mathbf{z}_{t-1}) q(\mathbf{z}_{t-1})}{q(\mathbf{z}_t)}. \tag{18.11}$$

This is intractable since we cannot compute the marginal distribution $q(\mathbf{z}_{t-1})$.

For this simple 1D example, it's possible to evaluate $q(\mathbf{z}_{t-1}|\mathbf{z}_t)$ numerically (figure 18.5). In general, their form is complex, but in many cases, they are well-approximated by a normal distribution. This is important because when we build the decoder, we will approximate the reverse process using a normal distribution.

18.2.4 Conditional diffusion distribution $q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})$

There is one final distribution related to the encoder to consider. We noted above that we could not find the conditional distribution $q(\mathbf{z}_{t-1}|\mathbf{z}_t)$ because we do not know the marginal distribution $q(\mathbf{z}_{t-1})$. However, if we know the starting variable $\mathbf{x}$, then we *do* know the distribution $q(\mathbf{z}_{t-1}|\mathbf{x})$ at the time before. This is just the diffusion kernel (figure 18.3), and it is normally distributed.

Hence, it is possible to compute the conditional diffusion distribution $q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})$ in closed form (figure 18.6). This distribution is used to train the decoder. It is the distribution over $\mathbf{z}_{t-1}$ when we know the current latent variable $\mathbf{z}_t$ and the training

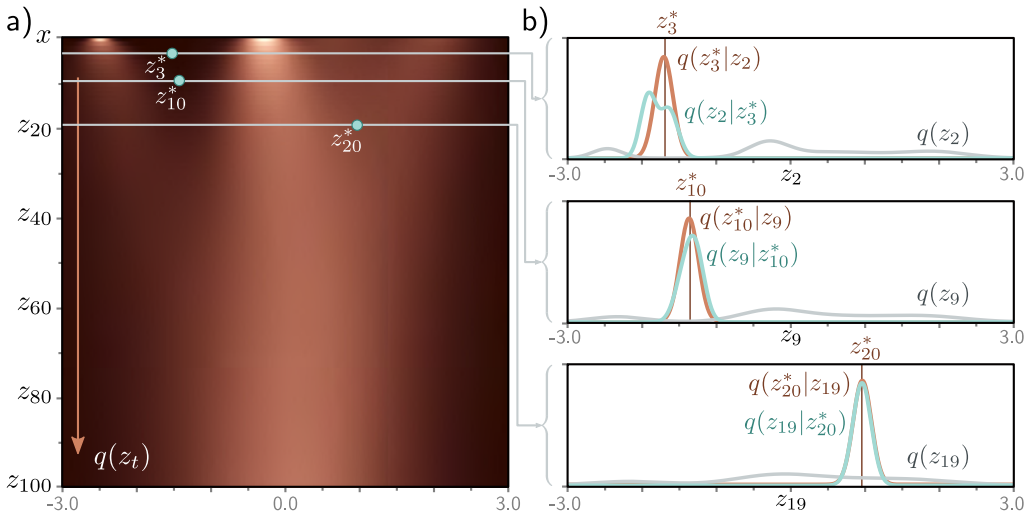


Figure 18.5 Conditional distribution $q(z_{t-1}|z_t)$. a) The marginal densities $q(z_t)$ with three points z_t^* highlighted. b) The probability $q(z_{t-1}|z_t^*)$ (cyan curves) is computed via Bayes' rule and is proportional to $q(z_t^*|z_{t-1})q(z_{t-1})$. In general, it is not normally distributed (top graph), although often the normal is a good approximation (bottom two graphs). The first likelihood term $q(z_t^*|z_{t-1})$ is normal in z_{t-1} (equation 18.2) with a mean that is slightly further from zero than z_t^* (brown curves). The second term is the marginal density $q(z_{t-1})$ (gray curves).

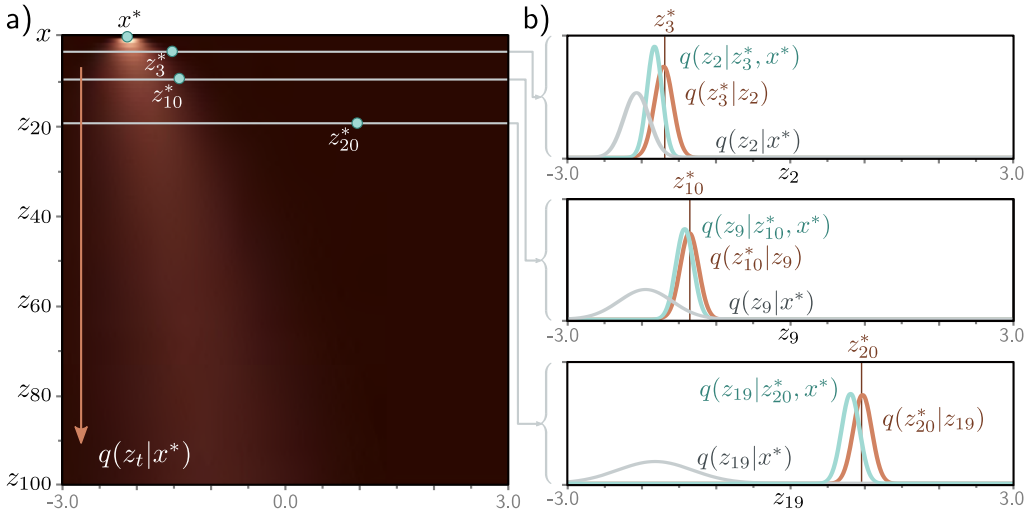


Figure 18.6 Conditional distribution $q(z_{t-1}|z_t, x)$. a) Diffusion kernel for $x^* = -2.1$ with three points z_t^* highlighted. b) The probability $q(z_{t-1}|z_t^*, x^*)$ is computed via Bayes' rule and is proportional to $q(z_t^*|z_{t-1})q(z_{t-1}|x^*)$. This *is* normally distributed and can be computed in closed form. The first likelihood term $q(z_t^*|z_{t-1})$ is normal in z_{t-1} (equation 18.2) with a mean that is slightly further from zero than z_t^* (brown curves). The second term is the diffusion kernel $q(z_{t-1}|x^*)$ (gray curves).

data example $\mathbf{x}$ (which, of course, we do when training). To compute an expression for $q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})$ we start with Bayes' rule:

$$\begin{aligned} q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x}) &= \frac{q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x})q(\mathbf{z}_{t-1}|\mathbf{x})}{q(\mathbf{z}_t|\mathbf{x})} \\ &\propto q(\mathbf{z}_t|\mathbf{z}_{t-1})q(\mathbf{z}_{t-1}|\mathbf{x}) \\ &= \text{Norm}_{\mathbf{z}_t}\left[\sqrt{1-\beta_t}\cdot\mathbf{z}_{t-1}, \beta_t\mathbf{I}\right]\text{Norm}_{\mathbf{z}_{t-1}}\left[\sqrt{\alpha_{t-1}}\cdot\mathbf{x}, (1-\alpha_{t-1})\mathbf{I}\right] \\ &\propto \text{Norm}_{\mathbf{z}_{t-1}}\left[\frac{1}{\sqrt{1-\beta_t}}\mathbf{z}_t, \frac{\beta_t}{1-\beta_t}\mathbf{I}\right]\text{Norm}_{\mathbf{z}_{t-1}}\left[\sqrt{\alpha_{t-1}}\cdot\mathbf{x}, (1-\alpha_{t-1})\mathbf{I}\right] \end{aligned} \tag{18.12}$$

where between the first two lines, we have used the fact that $q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}) = q(\mathbf{z}_t|\mathbf{z}_{t-1})$ because the diffusion process is Markov, and all information about $\mathbf{z}_t$ is captured by $\mathbf{z}_{t-1}$. Between lines three and four, we use the Gaussian change of variables identity:

Appendix C.3.4 Gaussian change of variables

$$\text{Norm}_{\mathbf{v}}\left[\mathbf{A}\mathbf{w}, \mathbf{B}\right] \propto \text{Norm}_{\mathbf{w}}\left[\left(\mathbf{A}^T\mathbf{B}^{-1}\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{B}^{-1}\mathbf{v}, \left(\mathbf{A}^T\mathbf{B}^{-1}\mathbf{A}\right)^{-1}\right], \tag{18.13}$$

to rewrite the first distribution in terms of $\mathbf{z}_{t-1}$. We then use a second Gaussian identity:

Problems 18.4–18.5

$$\begin{aligned} \text{Norm}_{\mathbf{w}}[\mathbf{a}, \mathbf{A}]\cdot\text{Norm}_{\mathbf{w}}[\mathbf{b}, \mathbf{B}] &\propto \\ &\text{Norm}_{\mathbf{w}}\left[\left(\mathbf{A}^{-1}+\mathbf{B}^{-1}\right)^{-1}(\mathbf{A}^{-1}\mathbf{a}+\mathbf{B}^{-1}\mathbf{b}), \left(\mathbf{A}^{-1}+\mathbf{B}^{-1}\right)^{-1}\right], \end{aligned} \tag{18.14}$$

to combine the two normal distributions in $\mathbf{z}_{t-1}$, which gives:

Problem 18.6

$$q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x}) = \text{Norm}_{\mathbf{z}_{t-1}}\left[\frac{(1-\alpha_{t-1})}{1-\alpha_t}\sqrt{1-\beta_t}\mathbf{z}_t + \frac{\sqrt{\alpha_{t-1}}\beta_t}{1-\alpha_t}\mathbf{x}, \frac{\beta_t(1-\alpha_{t-1})}{1-\alpha_t}\mathbf{I}\right]. \tag{18.15}$$

Note that the constants of proportionality in equations 18.12, 18.13, and 18.14 must cancel out since the final result is already a correctly normalized probability distribution.

18.3 Decoder model (reverse process)

When we learn a diffusion model, we learn the *reverse process*. In other words, we learn a series of probabilistic mappings back from latent variable $\mathbf{z}_T$ to $\mathbf{z}_{T-1}$, from $\mathbf{z}_{T-1}$ to $\mathbf{z}_{T-2}$, and so on, until we reach the data $\mathbf{x}$. The true reverse distributions $q(\mathbf{z}_{t-1}|\mathbf{z}_t)$ of the diffusion process are complex multi-modal distributions (figure 18.5) that depend on the data distribution $Pr(\mathbf{x})$. We approximate these as normal distributions:

$$\begin{aligned} Pr(\mathbf{z}_T) &= \text{Norm}_{\mathbf{z}_T}[\mathbf{0}, \mathbf{I}] \\ Pr(\mathbf{z}_{t-1}|\mathbf{z}_t, \boldsymbol{\phi}_t) &= \text{Norm}_{\mathbf{z}_{t-1}}\left[\mathbf{f}_t[\mathbf{z}_t, \boldsymbol{\phi}_t], \sigma_t^2\mathbf{I}\right] \\ Pr(\mathbf{x}|\mathbf{z}_1, \boldsymbol{\phi}_1) &= \text{Norm}_{\mathbf{x}}\left[\mathbf{f}_1[\mathbf{z}_1, \boldsymbol{\phi}_1], \sigma_1^2\mathbf{I}\right], \end{aligned} \tag{18.16}$$

where $\mathbf{f}_t[\mathbf{z}_t, \boldsymbol{\phi}_t]$ is a neural network that computes the mean of the normal distribution in the estimated mapping from $\mathbf{z}_t$ to the preceding latent variable $\mathbf{z}_{t-1}$. The terms $\{\sigma_t^2\}$ are predetermined. If the hyperparameters β_t in the diffusion process are close to zero (and the number of time steps T is large), then this normal approximation will be reasonable.

We generate new examples from $Pr(\mathbf{x})$ using ancestral sampling. We start by drawing $\mathbf{z}_T$ from $Pr(\mathbf{z}_T)$. Then we sample $\mathbf{z}_{T-1}$ from $Pr(\mathbf{z}_{T-1}|\mathbf{z}_T, \boldsymbol{\phi}_T)$, sample $\mathbf{z}_{T-2}$ from $Pr(\mathbf{z}_{T-2}|\mathbf{z}_{T-1}, \boldsymbol{\phi}_{T-1})$ and so on until we finally generate $\mathbf{x}$ from $Pr(\mathbf{x}|\mathbf{z}_1, \boldsymbol{\phi}_1)$.

18.4 Training

The joint distribution of the observed variable $\mathbf{x}$ and the latent variables $\{\mathbf{z}_t\}$ is:

$$Pr(\mathbf{x}, \mathbf{z}_{1\ldots T}|\boldsymbol{\phi}_{1\ldots T}) = Pr(\mathbf{x}|\mathbf{z}_1, \boldsymbol{\phi}_1)\prod_{t=2}^{T} Pr(\mathbf{z}_{t-1}|\mathbf{z}_t, \boldsymbol{\phi}_t) \cdot Pr(\mathbf{z}_T). \tag{18.17}$$

Appendix C.1.2 Marginalization

The likelihood of the observed data $Pr(\mathbf{x}|\boldsymbol{\phi}_{1\ldots T})$ is found by marginalizing over the latent variables:

$$Pr(\mathbf{x}|\boldsymbol{\phi}_{1\ldots T}) = \int Pr(\mathbf{x}, \mathbf{z}_{1\ldots T}|\boldsymbol{\phi}_{1\ldots T}) d\mathbf{z}_{1\ldots T}. \tag{18.18}$$

To train the model, we maximize the log-likelihood of the training data $\{\mathbf{x}_i\}$ with respect to the parameters $\boldsymbol{\phi}$:

$$\hat{\boldsymbol{\phi}}_{1\ldots T} = \underset{\boldsymbol{\phi}_{1\ldots T}}{\operatorname{argmax}} \left[\sum_{i=1}^{I} \log\Big[Pr(\mathbf{x}_i|\boldsymbol{\phi}_{1\ldots T})\Big]\right]. \tag{18.19}$$

We can't maximize this directly because the marginalization in equation 18.18 is intractable. Hence, we use Jensen's inequality to define a lower bound on the likelihood and optimize the parameters $\boldsymbol{\phi}_{1\ldots T}$ with respect to this bound exactly as we did for the VAE (see section 17.3.1).

18.4.1 Evidence lower bound (ELBO)

To derive the lower bound, we multiply and divide the log-likelihood by the encoder distribution $q(\mathbf{z}_{1\ldots T}|\mathbf{x})$ and apply Jensen's inequality (see section 17.3.2):

$$\begin{aligned}
\log\left[Pr(\mathbf{x}|\boldsymbol{\phi}_{1\ldots T})\right] &= \log\left[\int Pr(\mathbf{x}, \mathbf{z}_{1\ldots T}|\boldsymbol{\phi}_{1\ldots T}) d\mathbf{z}_{1\ldots T}\right] \\
&= \log\left[\int q(\mathbf{z}_{1\ldots T}|\mathbf{x}) \frac{Pr(\mathbf{x}, \mathbf{z}_{1\ldots T}|\boldsymbol{\phi}_{1\ldots T})}{q(\mathbf{z}_{1\ldots T}|\mathbf{x})} d\mathbf{z}_{1\ldots T}\right] \\
&\geq \int q(\mathbf{z}_{1\ldots T}|\mathbf{x}) \log\left[\frac{Pr(\mathbf{x}, \mathbf{z}_{1\ldots T}|\boldsymbol{\phi}_{1\ldots T})}{q(\mathbf{z}_{1\ldots T}|\mathbf{x})}\right] d\mathbf{z}_{1\ldots T}.
\end{aligned} \tag{18.20}$$

This gives us the evidence lower bound (ELBO):

$$\text{ELBO}\big[\boldsymbol{\phi}_{1\ldots T}\big] = \int q(\mathbf{z}_{1\ldots T}|\mathbf{x}) \log\left[\frac{Pr(\mathbf{x}, \mathbf{z}_{1\ldots T}|\boldsymbol{\phi}_{1\ldots T})}{q(\mathbf{z}_{1\ldots T}|\mathbf{x})}\right] d\mathbf{z}_{1\ldots T}. \tag{18.21}$$

In the VAE, the encoder $q(\mathbf{z}|\mathbf{x})$ approximates the posterior distribution over the latent variables to make the bound tight, and the decoder maximizes this bound (figure 17.10). In diffusion models, the decoder must do all the work since the encoder has no parameters. It makes the bound tighter by both (i) changing its parameters so that the static encoder does approximate the posterior $Pr(\mathbf{z}_{1\ldots T}|\mathbf{x}, \boldsymbol{\phi}_{1\ldots T})$ and (ii) optimizing its own parameters with respect to that bound (see figure 17.6).

18.4.2 Simplifying the ELBO

We now manipulate the log term from the ELBO into the final form that we will optimize. We first substitute in the definitions for the numerator and denominator from equations 18.17 and 18.3, respectively:

$$\begin{aligned}\log\left[\frac{Pr(\mathbf{x}, \mathbf{z}_{1\ldots T}|\boldsymbol{\phi}_{1\ldots T})}{q(\mathbf{z}_{1\ldots T}|\mathbf{x})}\right] &= \log\left[\frac{Pr(\mathbf{x}|\mathbf{z}_1, \boldsymbol{\phi}_1)\prod_{t=2}^{T} Pr(\mathbf{z}_{t-1}|\mathbf{z}_t, \boldsymbol{\phi}_t)\cdot Pr(\mathbf{z}_T)}{q(\mathbf{z}_1|\mathbf{x})\prod_{t=2}^{T} q(\mathbf{z}_t|\mathbf{z}_{t-1})}\right] \\ &= \log\left[\frac{Pr(\mathbf{x}|\mathbf{z}_1, \boldsymbol{\phi}_1)}{q(\mathbf{z}_1|\mathbf{x})}\right] + \log\left[\frac{\prod_{t=2}^{T} Pr(\mathbf{z}_{t-1}|\mathbf{z}_t, \boldsymbol{\phi}_t)}{\prod_{t=2}^{T} q(\mathbf{z}_t|\mathbf{z}_{t-1})}\right] + \log\Big[Pr(\mathbf{z}_T)\Big].\end{aligned} \tag{18.22}$$

Then we expand the denominator of the second term:

$$q(\mathbf{z}_t|\mathbf{z}_{t-1}) = q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}) = \frac{q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})q(\mathbf{z}_t|\mathbf{x})}{q(\mathbf{z}_{t-1}|\mathbf{x})}, \tag{18.23}$$

where the first equality follows because all of the information about variable $\mathbf{z}_t$ is encompassed in $\mathbf{z}_{t-1}$, so the extra conditioning on the data $\mathbf{x}$ is irrelevant. The second equality is a straightforward application of Bayes' rule.

Appendix C.1.4 Bayes' rule

Substituting in this result gives:

$$\begin{aligned}&\log\left[\frac{Pr(\mathbf{x}, \mathbf{z}_{1\ldots T}|\boldsymbol{\phi}_{1\ldots T})}{q(\mathbf{z}_{1\ldots T}|\mathbf{x})}\right] \\ &\quad= \log\left[\frac{Pr(\mathbf{x}|\mathbf{z}_1, \boldsymbol{\phi}_1)}{q(\mathbf{z}_1|\mathbf{x})}\right] + \log\left[\frac{\prod_{t=2}^{T} Pr(\mathbf{z}_{t-1}|\mathbf{z}_t, \boldsymbol{\phi}_t)\cdot q(\mathbf{z}_{t-1}|\mathbf{x})}{\prod_{t=2}^{T} q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})\cdot q(\mathbf{z}_t|\mathbf{x})}\right] + \log\Big[Pr(\mathbf{z}_T)\Big] \\ &\quad= \log\left[Pr(\mathbf{x}|\mathbf{z}_1, \boldsymbol{\phi}_1)\right] + \log\left[\frac{\prod_{t=2}^{T} Pr(\mathbf{z}_{t-1}|\mathbf{z}_t, \boldsymbol{\phi}_t)}{\prod_{t=2}^{T} q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})}\right] + \log\left[\frac{Pr(\mathbf{z}_T)}{q(\mathbf{z}_T|\mathbf{x})}\right] \\ &\quad\approx \log\left[Pr(\mathbf{x}|\mathbf{z}_1, \boldsymbol{\phi}_1)\right] + \sum_{t=2}^{T} \log\left[\frac{Pr(\mathbf{z}_{t-1}|\mathbf{z}_t, \boldsymbol{\phi}_t)}{q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})}\right],\end{aligned} \tag{18.24}$$

where all but two of the terms in the product of the ratios $q(\mathbf{z}_{t-1}|\mathbf{x})/q(\mathbf{z}_t|\mathbf{x})$ cancel out between lines two and three leaving only $q(\mathbf{z}_1|\mathbf{x})$ and $q(\mathbf{z}_T|\mathbf{x})$. The last term in the third line is approximately $\log[1] = 0$ since the result of the forward process $q(\mathbf{z}_T|\mathbf{x})$ is a standard normal distribution, and so is equal to the prior $Pr(\mathbf{z}_T)$.

The simplified ELBO is hence:

$$\begin{aligned}
&\text{ELBO}\left[\boldsymbol{\phi}_{1\ldots T}\right] \qquad (18.25)\\
&= \int q(\mathbf{z}_{1\ldots T}|\mathbf{x}) \log\left[\frac{Pr(\mathbf{x}, \mathbf{z}_{1\ldots T}|\boldsymbol{\phi}_{1\ldots T})}{q(\mathbf{z}_{1\ldots T}|\mathbf{x})}\right] d\mathbf{z}_{1\ldots T}\\
&\approx \int q(\mathbf{z}_{1\ldots T}|\mathbf{x}) \left(\log\left[Pr(\mathbf{x}|\mathbf{z}_1, \boldsymbol{\phi}_1)\right] + \sum_{t=2}^{T} \log\left[\frac{Pr(\mathbf{z}_{t-1}|\mathbf{z}_t, \boldsymbol{\phi}_t)}{q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})}\right]\right) d\mathbf{z}_{1\ldots T}\\
&= \mathbb{E}_{q(\mathbf{z}_1|\mathbf{x})}\Big[\log\left[Pr(\mathbf{x}|\mathbf{z}_1, \boldsymbol{\phi}_1)\right]\Big] - \sum_{t=2}^{T} \mathbb{E}_{q(\mathbf{z}_t|\mathbf{x})}\Big[\mathrm{D}_{KL}\Big[q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})\Big|\Big|Pr(\mathbf{z}_{t-1}|\mathbf{z}_t, \boldsymbol{\phi}_t)\Big]\Big],
\end{aligned}$$

Problem 18.7
Appendix C.5.1 KL divergence

where we have marginalized over the irrelevant variables in $q(\mathbf{z}_{1\ldots T}|\mathbf{x})$ between lines two and three and used the definition of KL divergence (see problem 18.7).

18.4.3 Analyzing the ELBO

The first probability term in the ELBO was defined in equation 18.16:

$$Pr(\mathbf{x}|\mathbf{z}_1, \boldsymbol{\phi}_1) = \text{Norm}_{\mathbf{x}}\Big[\mathbf{f}_1[\mathbf{z}_1, \boldsymbol{\phi}_1], \sigma_1^2\mathbf{I}\Big], \qquad (18.26)$$

and is equivalent to the reconstruction term in the VAE. The ELBO will be larger if the model prediction matches the observed data. As for the VAE, we will approximate the expectation over the log of this quantity using a Monte Carlo estimate (see equations 17.22–17.23), in which we estimate the expectation with a sample from $q(\mathbf{z}_1|\mathbf{x})$.

The KL divergence terms in the ELBO measure the distance between $Pr(\mathbf{z}_{t-1}|\mathbf{z}_t, \boldsymbol{\phi}_t)$ and $q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})$, which were defined in equations 18.16 and 18.15, respectively:

$$\begin{aligned}
Pr(\mathbf{z}_{t-1}|\mathbf{z}_t, \boldsymbol{\phi}_t) &= \text{Norm}_{\mathbf{z}_{t-1}}\Big[\mathbf{f}_t[\mathbf{z}_t, \boldsymbol{\phi}_t], \sigma_t^2\mathbf{I}\Big] \qquad (18.27)\\
q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x}) &= \text{Norm}_{\mathbf{z}_{t-1}}\left[\frac{(1-\alpha_{t-1})}{1-\alpha_t}\sqrt{1-\beta_t}\mathbf{z}_t + \frac{\sqrt{\alpha_{t-1}}\beta_t}{1-\alpha_t}\mathbf{x}, \frac{\beta_t(1-\alpha_{t-1})}{1-\alpha_t}\mathbf{I}\right].
\end{aligned}$$

Appendix C.5.4 KL divergence between normal distributions
Problem 18.8

The KL divergence between two normal distributions has a closed-form expression. Moreover, many of the terms in this expression do not depend on $\boldsymbol{\phi}$ (see problem 18.8), and the expression simplifies to the squared difference between the means plus a constant C:

$$\begin{aligned}
D_{KL}\Big[q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})\Big|\Big|Pr(\mathbf{z}_{t-1}|\mathbf{z}_t, \boldsymbol{\phi}_t)\Big] &= \qquad (18.28)\\
\frac{1}{2\sigma_t^2}\left\|\frac{(1-\alpha_{t-1})}{1-\alpha_t}\sqrt{1-\beta_t}\mathbf{z}_t + \frac{\sqrt{\alpha_{t-1}}\beta_t}{1-\alpha_t}\mathbf{x} - \mathbf{f}_t[\mathbf{z}_t, \boldsymbol{\phi}_t]\right\|^2 + C.
\end{aligned}$$

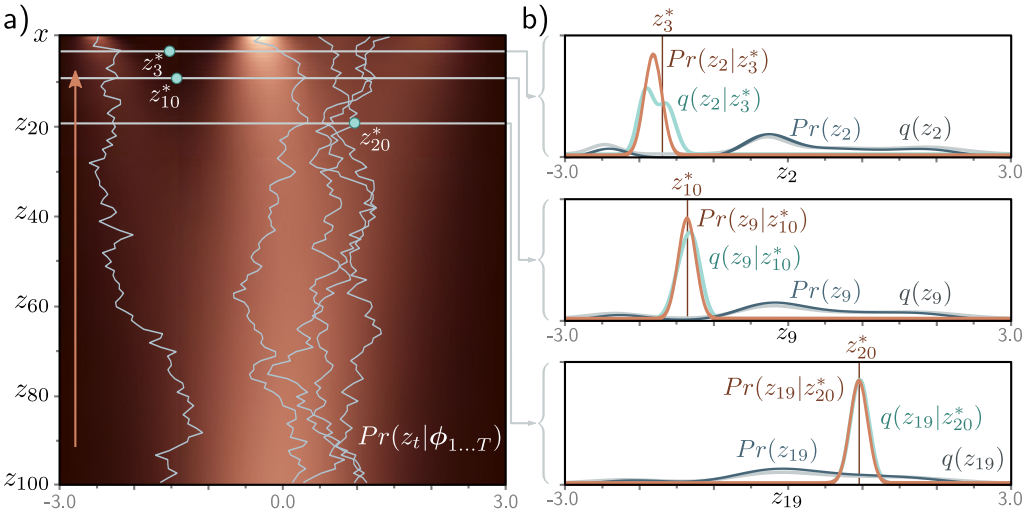


Figure 18.7 Fitted Model. a) Individual samples can be generated by sampling from the standard normal distribution $Pr(z_T)$ (bottom row) and then sampling z_{T-1} from $Pr(z_{T-1}|z_T) = \text{Norm}_{\mathbf{z}_{T-1}}[\text{f}_T[z_T, \boldsymbol{\phi}_T], \sigma_T^2\mathbf{I}]$ and so on until we reach x (five paths shown). The estimated marginal densities (heatmap) are the aggregation of these samples and are similar to the true marginal densities (figure 18.4). b) The estimated distribution $Pr(z_{t-1}|z_t)$ (brown curve) is a reasonable approximation to the true posterior of the diffusion model $q(z_{t-1}|z_t)$ (cyan curve) from figure 18.5. The marginal distributions $Pr(z_t)$ and $q(z_t)$ of the estimated and true models (dark blue and gray curves, respectively) are also similar.

18.4.4 Diffusion loss function

To fit the model, we maximize the ELBO with respect to the parameters $\boldsymbol{\phi}_{1...T}$. We recast this as a minimization by multiplying with minus one and approximating the expectations with samples to give the loss function:

$$\begin{aligned} L[\boldsymbol{\phi}_{1...T}] \;=\; \sum_{i=1}^{I}\Bigg(& \overbrace{-\log\Big[\text{Norm}_{\mathbf{x}_i}\left[\mathbf{f}_1[\mathbf{z}_{i1}, \boldsymbol{\phi}_1], \sigma_1^2\mathbf{I}\right]\Big]}^{\text{reconstruction term}} \\ & + \sum_{t=2}^{T} \frac{1}{2\sigma_t^2}\Bigg\| \underbrace{\frac{1-\alpha_{t-1}}{1-\alpha_t}\sqrt{1-\beta_t}\mathbf{z}_{it} + \frac{\sqrt{\alpha_{t-1}}\beta_t}{1-\alpha_t}\mathbf{x}_i}_{\text{target, mean of } q(\mathbf{z}_{t-1}|\mathbf{z}_t,\mathbf{x})} - \underbrace{\mathbf{f}_t[\mathbf{z}_{it}, \boldsymbol{\phi}_t]}_{\text{predicted } \mathbf{z}_{t-1}} \Bigg\|^2\Bigg), \end{aligned} \tag{18.29}$$

where $\mathbf{x}_i$ is the i^{th} data point, and $\mathbf{z}_{it}$ is the associated latent variable at diffusion step t.

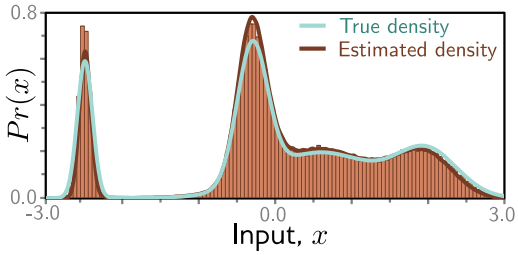


Figure 18.8 Fitted model results. Cyan and brown curves are original and estimated densities and correspond to the top rows of figures 18.4 and 18.7, respectively. Vertical bars are binned samples from the model, generated by sampling from $Pr(\mathbf{z}_T)$ and propagating back through the variables $\mathbf{z}_{T-1}, \mathbf{z}_{T-2}, \ldots$ as shown for the five paths in figure 18.7.

18.4.5 Training procedure

This loss function can be used to train a network for each diffusion time step. It minimizes the difference between the estimate $\mathbf{f}_t[\mathbf{z}_t, \boldsymbol{\phi}_t]$ of the hidden variable at the previous time step and the most likely value that it took given the ground truth de-noised data $\mathbf{x}$.

Figures 18.7 and 18.8 show the fitted reverse process for the simple 1D example. This model was trained by (i) taking a large dataset of examples $\mathbf{x}$ from the original density, (ii) using the diffusion kernel to predict many corresponding values for the latent variable $\mathbf{z}_t$ at each time t, and then (iii) training the models $\mathbf{f}_t[\mathbf{z}_t, \boldsymbol{\phi}_t]$ to minimize the loss function in equation 18.29. These models were nonparametric (i.e., lookup tables relating 1D input to 1D output), but more typically, they would be deep neural networks.

Notebook 18.2 1D diffusion model

18.5 Reparameterization of loss function

Although the loss function in equation 18.29 can be used, diffusion models have been found to work better with a different parameterization; the loss function is modified so that the model aims to predict the noise that was mixed with the original data example to create the current variable. Section 18.5.1 discusses reparameterizing the target (first two terms in second line of equation 18.29), and section 18.5.2 discusses reparameterizing the network (last term in second line of equation 18.29).

18.5.1 Reparameterization of target

The original diffusion update was given by:

$$\mathbf{z}_t = \sqrt{\alpha_t} \cdot \mathbf{x} + \sqrt{1-\alpha_t} \cdot \boldsymbol{\epsilon}. \tag{18.30}$$

It follows that the data term $\mathbf{x}$ in equation 18.28 can be expressed as the diffused image minus the noise that was added to it:

$$\mathbf{x} = \frac{1}{\sqrt{\alpha_t}} \cdot \mathbf{z}_t - \frac{\sqrt{1-\alpha_t}}{\sqrt{\alpha_t}} \cdot \boldsymbol{\epsilon}. \tag{18.31}$$

Substituting this into the target terms from equation 18.29 gives:

$$\begin{aligned}\frac{(1-\alpha_{t-1})}{1-\alpha_t}\sqrt{1-\beta_t}\mathbf{z}_t + \frac{\sqrt{\alpha_{t-1}}\beta_t}{1-\alpha_t}\mathbf{x} \qquad (18.32)\\ = \frac{(1-\alpha_{t-1})}{1-\alpha_t}\sqrt{1-\beta_t}\mathbf{z}_t + \frac{\sqrt{\alpha_{t-1}}\beta_t}{1-\alpha_t}\left(\frac{1}{\sqrt{\alpha_t}}\mathbf{z}_t - \frac{\sqrt{1-\alpha_t}}{\sqrt{\alpha_t}}\boldsymbol{\epsilon}\right)\\ = \frac{(1-\alpha_{t-1})}{1-\alpha_t}\sqrt{1-\beta_t}\mathbf{z}_t + \frac{\beta_t}{1-\alpha_t}\left(\frac{1}{\sqrt{1-\beta_t}}\mathbf{z}_t - \frac{\sqrt{1-\alpha_t}}{\sqrt{1-\beta_t}}\boldsymbol{\epsilon}\right),\end{aligned}$$

where we have used the fact that $\sqrt{\alpha_t}/\sqrt{\alpha_{t-1}} = \sqrt{1-\beta_t}$ between the second and third lines. Simplifying further, we get: Problem 18.9

$$\begin{aligned}\frac{(1-\alpha_{t-1})}{1-\alpha_t}\sqrt{1-\beta_t}\mathbf{z}_t + \frac{\sqrt{\alpha_{t-1}}\beta_t}{1-\alpha_t}\mathbf{x} \qquad (18.33)\\ = \left(\frac{(1-\alpha_{t-1})\sqrt{1-\beta_t}}{1-\alpha_t} + \frac{\beta_t}{(1-\alpha_t)\sqrt{1-\beta_t}}\right)\mathbf{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}\sqrt{1-\beta_t}}\boldsymbol{\epsilon}\\ = \left(\frac{(1-\alpha_{t-1})(1-\beta_t)}{(1-\alpha_t)\sqrt{1-\beta_t}} + \frac{\beta_t}{(1-\alpha_t)\sqrt{1-\beta_t}}\right)\mathbf{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}\sqrt{1-\beta_t}}\boldsymbol{\epsilon}\\ = \frac{(1-\alpha_{t-1})(1-\beta_t)+\beta_t}{(1-\alpha_t)\sqrt{1-\beta_t}}\mathbf{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}\sqrt{1-\beta_t}}\boldsymbol{\epsilon}\\ = \frac{1-\alpha_t}{(1-\alpha_t)\sqrt{1-\beta_t}}\mathbf{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}\sqrt{1-\beta_t}}\boldsymbol{\epsilon}\\ = \frac{1}{\sqrt{1-\beta_t}}\mathbf{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}\sqrt{1-\beta_t}}\boldsymbol{\epsilon},\end{aligned}$$

where we have multiplied the numerator and denominator of the first term by $\sqrt{1-\beta_t}$ between lines two and three, multiplied out the terms, and simplified the numerator in the first term between lines three and four.

Substituting this back into the loss function (equation 18.29), we have: Problem 18.10

$$\begin{aligned}L[\boldsymbol{\phi}_{1\ldots T}] &= \sum_{i=1}^{I} -\log\Big[\text{Norm}_{\mathbf{x}_i}\left[\mathbf{f}_1[\mathbf{z}_{i1},\boldsymbol{\phi}_1],\sigma_1^2\mathbf{I}\right]\Big] \qquad (18.34)\\ &\quad + \sum_{t=2}^{T}\frac{1}{2\sigma_t^2}\left\|\left(\frac{1}{\sqrt{1-\beta_t}}\mathbf{z}_{it} - \frac{\beta_t}{\sqrt{1-\alpha_t}\sqrt{1-\beta_t}}\boldsymbol{\epsilon}_{it}\right) - \mathbf{f}_t[\mathbf{z}_{it},\boldsymbol{\phi}_t]\right\|^2.\end{aligned}$$

18.5.2 Reparameterization of network

Now we replace the model $\hat{\mathbf{z}}_{t-1} = \mathbf{f}_t[\mathbf{z}_t,\boldsymbol{\phi}_t]$ with a new model $\hat{\boldsymbol{\epsilon}} = \mathbf{g}_t[\mathbf{z}_t,\boldsymbol{\phi}_t]$, which predicts the noise $\boldsymbol{\epsilon}$ that was mixed with $\mathbf{x}$ to create $\mathbf{z}_t$:

$$\mathbf{f}_t[\mathbf{z}_t,\boldsymbol{\phi}_t] = \frac{1}{\sqrt{1-\beta_t}}\mathbf{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}\sqrt{1-\beta_t}}\mathbf{g}_t[\mathbf{z}_t,\boldsymbol{\phi}_t]. \qquad (18.35)$$

Substituting the new model into equation 18.34 produces the criterion:

$$L[\boldsymbol{\phi}_{1\ldots T}] = \sum_{i=1}^{I} -\log\Big[\text{Norm}_{\mathbf{x}_i}\left[\mathbf{f}_1[\mathbf{z}_{i1},\boldsymbol{\phi}_1],\sigma_1^2\mathbf{I}\right]\Big] + \sum_{t=2}^{T}\frac{\beta_t^2}{(1-\alpha_t)(1-\beta_t)2\sigma_t^2}\Big\|\mathbf{g}_t[\mathbf{z}_{it},\boldsymbol{\phi}_t]-\boldsymbol{\epsilon}_{it}\Big\|^2. \tag{18.36}$$

The log normal can be written as a least squares loss plus a constant C_i (section 5.3.1):

$$L[\boldsymbol{\phi}_{1\ldots T}] = \sum_{i=1}^{I}\frac{1}{2\sigma_1^2}\Big\|\mathbf{x}_i-\mathbf{f}_1[\mathbf{z}_{i1},\boldsymbol{\phi}_1]\Big\|^2 + \sum_{t=2}^{T}\frac{\beta_t^2}{(1-\alpha_t)(1-\beta_t)2\sigma_t^2}\Big\|\mathbf{g}_t[\mathbf{z}_{it},\boldsymbol{\phi}_t]-\boldsymbol{\epsilon}_{it}\Big\|^2 + C_i.$$

Problem 18.11

Substituting in the definitions of $\mathbf{x}$ and $\mathbf{f}_1[\mathbf{z}_1,\boldsymbol{\phi}_1]$ from equations 18.31 and 18.35, respectively, the first term simplifies to:

$$\frac{1}{2\sigma_1^2}\Big\|\mathbf{x}_i-\mathbf{f}_1[\mathbf{z}_{i1},\boldsymbol{\phi}_1]\Big\|^2 = \frac{1}{2\sigma_1^2}\Big\|\frac{\beta_1}{\sqrt{1-\alpha_1}\sqrt{1-\beta_1}}\mathbf{g}_1[\mathbf{z}_{i1},\boldsymbol{\phi}_1] - \frac{\beta_1}{\sqrt{1-\alpha_1}\sqrt{1-\beta_1}}\boldsymbol{\epsilon}_{i1}\Big\|^2. \tag{18.37}$$

Adding this back to the final loss function yields:

$$L[\boldsymbol{\phi}_{1\ldots T}] = \sum_{i=1}^{I}\sum_{t=1}^{T}\frac{\beta_t^2}{(1-\alpha_t)(1-\beta_t)2\sigma_t^2}\Big\|\mathbf{g}_t[\mathbf{z}_{it},\boldsymbol{\phi}_t]-\boldsymbol{\epsilon}_{it}\Big\|^2, \tag{18.38}$$

where we have disregarded the additive constants C_i.

In practice, the scaling factors (which might be different at each time step) are ignored, giving an even simpler formulation:

$$\begin{aligned} L[\boldsymbol{\phi}_{1\ldots T}] &= \sum_{i=1}^{I}\sum_{t=1}^{T}\Big\|\mathbf{g}_t[\mathbf{z}_{it},\boldsymbol{\phi}_t]-\boldsymbol{\epsilon}_{it}\Big\|^2 \\ &= \sum_{i=1}^{I}\sum_{t=1}^{T}\Big\|\mathbf{g}_t\Big[\sqrt{\alpha_t}\cdot\mathbf{x}_i+\sqrt{1-\alpha_t}\cdot\boldsymbol{\epsilon}_{it},\boldsymbol{\phi}_t\Big]-\boldsymbol{\epsilon}_{it}\Big\|^2, \end{aligned} \tag{18.39}$$

where we have rewritten $\mathbf{z}_t$ using the diffusion kernel (equation 18.30) in the second line.

18.6 Implementation

This leads to straightforward algorithms for both training the model (algorithm 18.1) and sampling (algorithm 18.2). The training algorithm has the advantages that it is (i) simple to implement and (ii) naturally augments the dataset; we can reuse every original data point $\mathbf{x}_i$ as many times as we want at each time step with different noise instantiations $\boldsymbol{\epsilon}$. The sampling algorithm has the disadvantage that it requires serial processing of many neural networks $\mathbf{g}_t[\mathbf{z}_t,\boldsymbol{\phi}_t]$ and is hence time-consuming.

Notebook 18.3 Reparameterized model

Algorithm 18.1: Diffusion model training

Input: Training data $\mathbf{x}$
Output: Model parameters $\boldsymbol{\phi}_t$
repeat
  for $i \in \mathcal{B}$ **do** // For every training example index in batch
    $t \sim \text{Uniform}[1, \ldots T]$ // Sample random timestep
    $\boldsymbol{\epsilon} \sim \text{Norm}[\mathbf{0}, \mathbf{I}]$ // Sample noise
    $\ell_i = \left\| \mathbf{g}_t\left[\sqrt{\alpha_t}\mathbf{x}_i + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}, \boldsymbol{\phi}_t\right] - \boldsymbol{\epsilon} \right\|^2$ // Compute individual loss
  Accumulate losses for batch and take gradient step
until converged

Algorithm 18.2: Sampling

Input: Model, $\mathbf{g}_t[\bullet, \boldsymbol{\phi}_t]$
Output: Sample, $\mathbf{x}$
$\mathbf{z}_T \sim \text{Norm}_{\mathbf{z}}[\mathbf{0}, \mathbf{I}]$ // Sample last latent variable
for $t = T \ldots 2$ **do**
  $\hat{\mathbf{z}}_{t-1} = \frac{1}{\sqrt{1-\beta_t}}\mathbf{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}\sqrt{1-\beta_t}}\mathbf{g}_t[\mathbf{z}_t, \boldsymbol{\phi}_t]$ // Predict previous latent variable
  $\boldsymbol{\epsilon} \sim \text{Norm}_{\boldsymbol{\epsilon}}[\mathbf{0}, \mathbf{I}]$ // Draw new noise vector
  $\mathbf{z}_{t-1} = \hat{\mathbf{z}}_{t-1} + \sigma_t\boldsymbol{\epsilon}$ // Add noise to previous latent variable
$\mathbf{x} = \frac{1}{\sqrt{1-\beta_1}}\mathbf{z}_1 - \frac{\beta_1}{\sqrt{1-\alpha_1}\sqrt{1-\beta_1}}\mathbf{g}_1[\mathbf{z}_1, \boldsymbol{\phi}_1]$ // Generate sample from $\mathbf{z}_1$ without noise

18.6.1 Application to images

Diffusion models have been very successful in modeling image data. Here, we need to construct models that can take a noisy image and predict the noise that was added at each step. The obvious architectural choice for this image-to-image mapping is the U-Net (figure 11.10). However, there may be a very large number of diffusion steps, and training and storing multiple U-Nets is inefficient. The solution is to train a single U-Net that also takes a predetermined vector representing the time step as input (figure 18.9). In practice, this is resized to match the number of channels at each stage of the U-Net and used to offset and/or scale the representation at each spatial position.

A large number of time steps are needed as the conditional probabilities $q(\mathbf{z}_{t-1}|\mathbf{z}_t)$ become closer to normal when the hyperparameters β_t are close to zero, matching the form of the decoder distributions $Pr(\mathbf{z}_{t-1}|\mathbf{z}_t, \boldsymbol{\phi}_t)$. However, this makes sampling slow. We might have to run the U-Net model through $T = 1000$ steps to generate good images.

18.6.2 Improving generation speed

The loss function (equation 18.39) requires the diffusion kernel to have the form $q(\mathbf{z}_t|\mathbf{x}) = \text{Norm}[\sqrt{\alpha_t}\mathbf{x}, \sqrt{1-\alpha_t} \cdot \mathbf{I}]$. The same loss function will be valid for *any* forward process

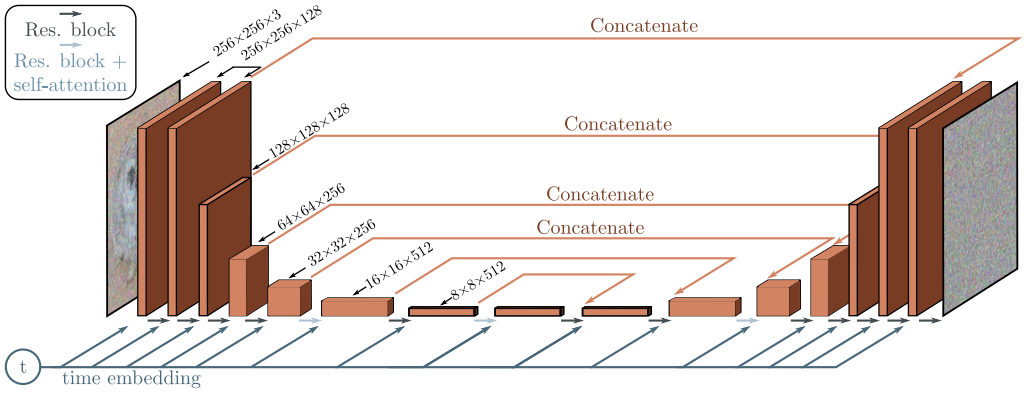


Figure 18.9 U-Net as used in diffusion models for images. The network aims to predict the noise that was added to the image. It consists of an encoder which reduces the scale and increases the number of channels and a decoder which increases the scale and reduces the number of channels. The encoder representations are concatenated to their partner in the decoder. Connections between adjacent representations consist of residual blocks, and periodic global self-attention in which every spatial position interacts with every other spatial position. A single network is used for all time steps, by passing a sinusoidal time embedding (figure 12.5) through a shallow neural network and adding the result to the channels at every spatial position at every stage of the U-Net.

with this relation, and there is a family of such compatible processes. These are all optimized by the same loss function but have different rules for the forward process and different corresponding rules for how to use the estimated noise $\mathbf{g}[\mathbf{z}_t, \boldsymbol{\phi}_t]$ to predict $\mathbf{z}_{t-1}$ from $\mathbf{z}_t$ in the reverse process (figure 18.10).

Among this family are *denoising diffusion implicit models*, which are no longer stochastic after the first step from $\mathbf{x}$ to $\mathbf{z}_1$, and *accelerated sampling* models, where the forward process is defined only on a sub-sequence of time steps. This allows a reverse process that skips time steps and hence makes sampling much more efficient; good samples can be created with 50 time steps when the forward process is no longer stochastic. This is much faster than before but still slower than most other generative models.

Notebook 18.4 Families of diffusion models

18.6.3 Conditional generation

If the data has associated labels c, these can be exploited to control the generation. Sometimes this can improve generation results in GANs, and we might expect this to be the case in diffusion models as well; it's easier to denoise an image if you have some information about what that image contains. One approach to conditional synthesis in diffusion models is *classifier guidance*. This modifies the denoising update from $\mathbf{z}_t$ to $\mathbf{z}_{t-1}$ to take into account class information c. In practice, this means adding an extra

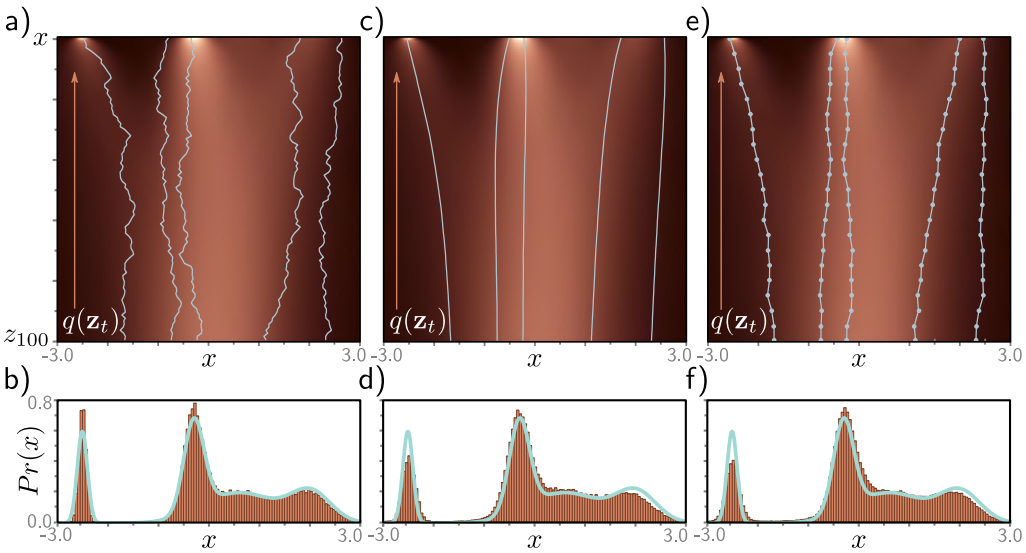


Figure 18.10 Different diffusion processes that are compatible with the same model. a) Five sampled trajectories of the reparameterized model superimposed on the ground truth marginal distributions. Top row represents $Pr(\mathbf{x})$ and subsequent rows represent $q(\mathbf{x}_t)$. b) Histogram of samples generated from reparameterized model plotted alongside ground truth density curve $Pr(\mathbf{x})$. The same trained model is compatible with a family of diffusion models (and corresponding updates in the opposite direction), including the denoising diffusion implicit (DDIM) model, which is deterministic and does not add noise at each step. c) Five trajectories from DDIM model. d) Histogram of samples from DDIM model. The same model is also compatible with accelerated diffusion models that skip inference steps for increased sampling speed. e) Five trajectories from accelerated model. f) Histogram of samples from accelerated model.

term into the final update step in algorithm 18.2 to yield:

$$\mathbf{z}_{t-1} = \hat{\mathbf{z}}_{t-1} + \sigma_t^2 \frac{\partial \log\big[\Pr(c|\mathbf{z}_t)\big]}{\partial \mathbf{z}_t} + \sigma_t \boldsymbol{\epsilon}. \tag{18.40}$$

The new term depends on the gradient of a classifier $Pr(c|\mathbf{z}_t)$ that is based on the latent variable $\mathbf{z}_t$. This maps features from the downsampling half of the U-Net to the class c. Like the U-Net, it is usually shared across all time steps and takes time as an input. The update from $\mathbf{z}_t$ to $\mathbf{z}_{t-1}$ now makes the class c more likely.

Classifier-free guidance avoids learning a separate classifier $Pr(c|\mathbf{z}_t)$ but instead incorporates class information into the main model $\mathbf{g}_t[\mathbf{z}_t, \boldsymbol{\phi}_t, c]$. In practice, this usually takes the form of adding an embedding based on c to the layers of the U-Net in a similar way to how the time step is added (see figure 18.9). This model is jointly trained on conditional and unconditional objectives by randomly dropping the class information

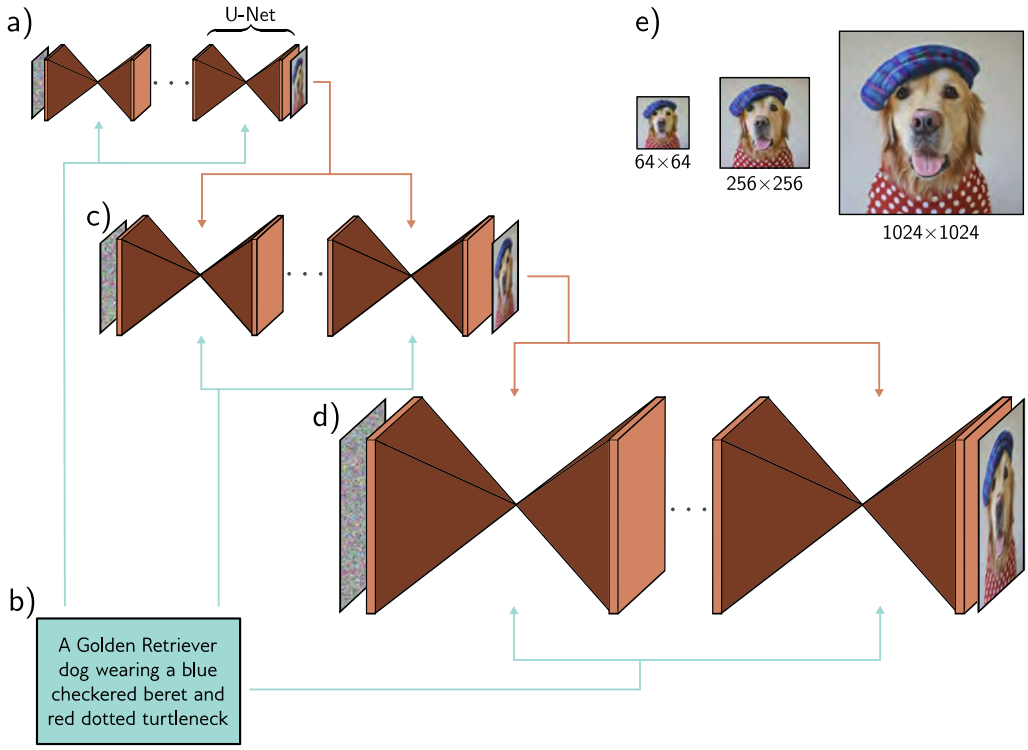


Figure 18.11 Cascaded conditional generation based on a text prompt. a) A diffusion model consisting of a series of U-Nets is used to generate a 64×64 image. b) This generation is conditioned on a sentence embedding computed by a language model. c) A higher resolution 256×256 image is generated and conditioned on the smaller image *and* the text encoding. d) This is repeated to create a 1024×1024 image. e) Final image sequence. Adapted from Saharia et al. (2022b).

Problem 18.12

during training. Hence, it can both generate unconditional or conditional data examples at test time or any weighted combination of the two. This brings a surprising advantage; if the conditioning information is over-weighted, the model tends to produce very high quality but slightly stereotypical examples. This is somewhat analogous to the use of truncation in GANs (figure 15.10).

18.6.4 Improving generation quality

As for other generative models, the highest quality results result from applying a combination of tricks and extensions to the basic model. First, it's been noted that it also helps to estimate the variances σ_t^2 of the reverse process as well as the mean (i.e., the widths

of the brown normal distributions in figure 18.7). This particularly improves the results when sampling with fewer steps. Second, it's possible to modify the noise schedule in the forward process so that β_t varies at each step, and this can also improve results.

Third, to generate high-resolution images, a cascade of diffusion models is used. The first creates a low-resolution image (possibly guided by class information). The subsequent diffusion models generate progressively higher-resolution images. They condition on the lower-resolution image by resizing this and appending it to the layers of the constituent U-Net, as well as any other class information (figure 18.11).

Combining all of these techniques allows the generation of very high-quality images. Figure 18.12 shows examples of images generated from a model conditioned on the ImageNet class. It is particularly impressive that the same model can learn to generate such diverse classes. Figure 18.13 shows images generated from a model that is trained to condition on text captions encoded by a language model like BERT, which are inserted into the model in the same way as the time step (figures 18.9 and 18.11). This results in very realistic images that agree with the caption. Since the diffusion model is stochastic by nature, it's possible to generate multiple images that are conditioned on the same caption.

18.7 Summary

Diffusion models map the data examples through a series of latent variables by repeatedly blending the current representation with random noise. After sufficient steps, the representation becomes indistinguishable from white noise. Since these steps are small, the reverse denoising process at each step can be approximated with a normal distribution and predicted by a deep learning model. The loss function is based on the evidence lower bound (ELBO) and ultimately results in a simple least-squares formulation.

For image generation, each denoising step is implemented using a U-Net, so sampling is slow compared to other generative models. To improve generation speed, it's possible to change the diffusion model to a deterministic formulation, and here sampling with fewer steps works well. Several methods have been proposed to condition generation on class information, images, and text information. Combining these methods produces impressive text-to-image synthesis results.

Notes

Denoising diffusion models were introduced by Sohl-Dickstein et al. (2015), and early related work based on score-matching was carried out by Song & Ermon (2019). Ho et al. (2020) produced image samples that were competitive with GANs and kick-started a wave of interest in this area. Most of the exposition in this chapter, including the original formulation and the reparameterization, is derived from this paper. Dhariwal & Nichol (2021) improved the quality of these results and showed for the first time that images from diffusion models were quantitatively superior to GAN models in terms of Fréchet Inception Distance. At the time

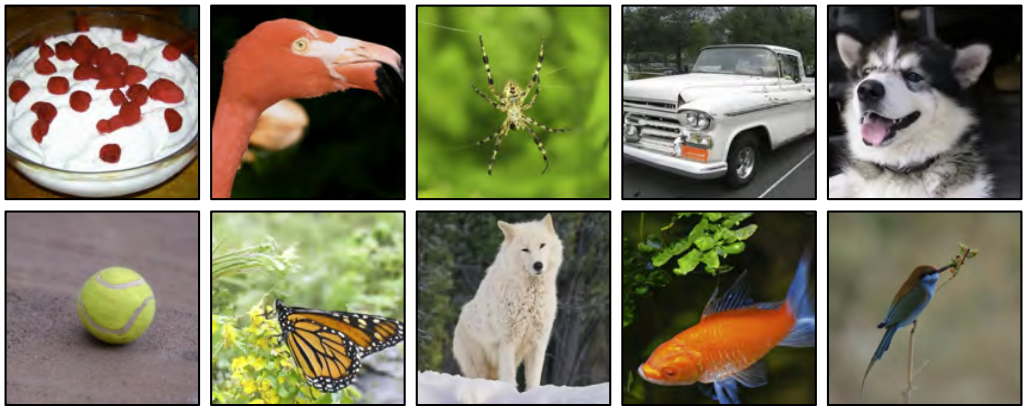


Figure 18.12 Conditional generation using classifier guidance. Image samples conditioned on different ImageNet classes. The same model produces high quality samples of highly varied image classes. Adapted from Dhariwal & Nichol (2021).

Figure 18.13 Conditional generation using text prompts. Synthesized images from a cascaded generation framework, conditioned on a text prompt encoded by a large language model. The stochastic model can produce many different images compatible with the prompt. The model can count objects and incorporate text into images. Adapted from Saharia et al. (2022b).

of writing, the state-of-the-art results for conditional image synthesis have been achieved by Karras et al. (2022). Surveys of denoising diffusion models can be found in Croitoru et al. (2022), Cao et al. (2022), Luo (2022), and Yang et al. (2022).

Applications for images: Applications of diffusion models include text-to-image generation (Nichol et al., 2022; Ramesh et al., 2022; Saharia et al., 2022b), image-to-image tasks such as colorization, inpainting, uncropping and restoration (Saharia et al., 2022a), super-resolution (Saharia et al., 2022c), image editing (Hertz et al., 2022; Meng et al., 2021), removing adversarial perturbations (Nie et al., 2022), semantic segmentation (Baranchuk et al., 2022), and medical imaging (Song et al., 2021b; Chung & Ye, 2022; Chung et al., 2022; Peng et al., 2022; Xie & Li, 2022; Luo et al., 2022) where the diffusion model is sometimes used as a prior.

Different data types: Diffusion models have also been applied to video data (Ho et al., 2022b; Harvey et al., 2022; Yang et al., 2022; Höppe et al., 2022; Voleti et al., 2022) for generation, past and future frame prediction, and interpolation. They have been used for 3D shape generation (Zhou et al., 2021; Luo & Hu, 2021), and recently a technique has been introduced to generate 3D models using only a 2D text-to-image diffusion model (Poole et al., 2023). Austin et al. (2021) and Hoogeboom et al. (2021) investigated diffusion models for discrete data. Kong et al. (2021) and Chen et al. (2021d) applied diffusion models to audio data.

Alternatives to denoising: The diffusion models in this chapter mix noise with the data and build a model to gradually denoise the result. However, degrading the image using noise is not necessary. Rissanen et al. (2022) devised a method that progressively blurred the image and Bansal et al. (2022) showed that the same ideas work with a large family of degradations that do not have to be stochastic. These include masking, morphing, blurring, and pixelating.

Comparison to other generative models: Diffusion models synthesize higher quality images than other generative models and are simple to train. They can be thought of as a special case of a hierarchical VAE (Vahdat & Kautz, 2020; Sønderby et al., 2016b) where the encoder is fixed, and the latent space is the same size as the data. They are probabilistic, but in their basic form, they can only compute a lower bound on the likelihood of a data point. However, Kingma et al. (2021) show that this lower bound improves on the exact log-likelihoods for test data from normalizing flows and autoregressive models. The likelihood for diffusion models can be computed by converting to an ordinary differential equation (Song et al., 2021c) or by training a continuous normalizing flow model with a diffusion-based criterion (Lipman et al., 2022). The main disadvantages of diffusion models are that they are slow and that the latent space has no semantic interpretation.

Improving quality: Many techniques have been proposed to improve image quality. These include the reparameterization of the network described in section 18.5 and the equal weighting of the subsequent terms (Ho et al., 2020). Choi et al. (2022) subsequently investigated different weightings of terms in the loss function.

Kingma et al. (2021) improved the test log-likelihood of the model by learning the denoising weights β_t. Conversely, Nichol & Dhariwal (2021) improved performance by learning separate variances σ^2 of the denoising estimate at each time step in addition to the mean. Bao et al. (2022) show how to learn the variances *after* training the model.

Ho et al. (2022a) developed the cascaded method for producing very high-resolution images (figure 18.11). To prevent artifacts in lower-resolution images from being propagated to higher resolutions, they introduced *noise conditioning augmentation*; here, the lower-resolution conditioning image is degraded by adding noise at each training step. This reduces the reliance on the exact details of the lower-resolution image during training. It is also done during inference, where the best noise level is chosen by sweeping over different values.

Improving speed: One of the major drawbacks of diffusion models is that they take a long time to train and sample from. *Stable diffusion* (Rombach et al., 2022) projects the original data to a smaller latent space using a conventional autoencoder and then runs the diffusion process in this smaller space. This has the advantages of reducing the dimensionality of the training data for the diffusion process and allowing other data types (text, graphs, etc.) to be described by diffusion models. Vahdat et al. (2021) applied a similar approach.

Song et al. (2021a) showed that an entire family of diffusion processes is compatible with the training objective. Most of these processes are non-Markovian (i.e., the diffusion step does not only depend on the results of the previous step). One of these models is the denoising diffusion implicit model (DDIM), in which the updates are not stochastic (figure 18.10b). This model is amenable to taking larger steps (figure 18.10b) without inducing large errors. It effectively converts the model into an ordinary differential equation (ODE) in which the trajectories have low curvature and allows efficient numerical methods for solving ODEs to be applied.

Song et al. (2021c) propose converting the underlying stochastic differential equations into a *probability flow ODE* which has the same marginal distributions as the original process. Vahdat et al. (2021), Xiao et al. (2022b), and Karras et al. (2022) all exploit techniques for solving ODEs to speed up synthesis. Karras et al. (2022) identified the best-performing time discretization for sampling and evaluated different sampler schedules. The result of these and other improvements has been a significant drop in steps required during synthesis.

Sampling is slow because many small diffusion steps are required to ensure that the posterior distribution $q(\mathbf{z}_{t-1}|\mathbf{z}_t)$ is close to Gaussian (figure 18.5), so the Gaussian distribution in the decoder is appropriate. If we use a model that describes a more complex distribution at each denoising step, then we can use fewer diffusion steps in the first place. To this end, Xiao et al. (2022b) have investigated using conditional GAN models, and Gao et al. (2021) investigated using conditional energy-based models. Although these models cannot describe the original data distribution, they suffice to predict the (much simpler) reverse diffusion step.

Salimans & Ho (2022) distilled adjacent steps of the denoising process into a single step to speed up synthesis. Dockhorn et al. (2022) introduced momentum into the diffusion process. This makes the trajectories smoother and so more amenable to coarse sampling.

Conditional generation: Dhariwal & Nichol (2021) introduced classifier guidance, in which a classifier learns to identify the category of object being synthesized at each step, and this is used to bias the denoising update toward that class. This works well, but training a separate classifier is expensive. *Classifier-free guidance* (Ho & Salimans, 2022) concurrently trains conditional and unconditional denoising models by dropping the class information some proportion of the time in a process akin to dropout. This technique allows control of the relative contributions of the conditional and unconditional components. Over-weighting the conditional component causes the model to produce more typical and realistic samples.

The standard technique for conditioning on images is to append the (resized) image to the different layers of the U-Net. For example, this was used in the cascaded generation process for super-resolution (Ho et al., 2022a). Choi et al. (2021) provide a method for conditioning on images in an unconditional diffusion model by matching the latent variables with those of a conditioning image. The standard technique for conditioning on text is to linearly transform the text embedding to the same size as the U-Net layer and then add it to the representation in the same way that the time embedding is introduced (figure 18.9).

Existing diffusion models can also be fine-tuned to be conditioned on edge maps, joint positions, segmentation, depth maps, etc., using a neural network structure called a *control network* (Zhang & Agrawala, 2023).

Text-to-image: Before diffusion models, state-of-the-art text-to-image systems were based on transformers (e.g., Ramesh et al., 2021). GLIDE (Nichol et al., 2022) and Dall·E 2 (Ramesh et al., 2022) are both conditioned on embeddings from the CLIP model (Radford et al., 2021),

which generates joint embeddings for text and image data. Imagen (Saharia et al., 2022b) showed that text embeddings from a large language model could produce even better results (see figure 18.13). The same authors introduced a benchmark (DrawBench) which is designed to evaluate the ability of a model to render colors, numbers of objects, spatial relations, and other characteristics. Feng et al. (2022) developed a Chinese text-to-image model.

Connections to other models: This chapter described diffusion models as hierarchical variational autoencoders because this approach connects most closely with the other parts of this book. However, diffusion models also have close connections with stochastic differential equations (consider the paths in figure 18.5) and with score matching (Song & Ermon, 2019, 2020). Song et al. (2021c) presented a framework based on stochastic differential equations that encompasses both the denoising and score matching interpretations. Diffusion models also have close connections to normalizing flows (Zhang & Chen, 2021). Yang et al. (2022) present an overview of the relationship between diffusion models and other generative approaches.

Problems

Problem 18.1 Show that if $\text{Var}[\mathbf{x}_{t-1}] = \mathbf{I}$ and we use the update:

$$\mathbf{x}_t = \sqrt{1-\beta_t}\cdot\mathbf{x}_{t-1} + \sqrt{\beta_t}\cdot\boldsymbol{\epsilon}_t, \tag{18.41}$$

then $\text{Var}[\mathbf{x}_t] = \mathbf{I}$, so the variance stays the same.

Problem 18.2 Consider the variable:

$$z = a\cdot\epsilon_1 + b\cdot\epsilon_2, \tag{18.42}$$

where both ϵ_1 and ϵ_2 are drawn from independent standard normal distributions with mean zero and unit variance. Show that:

$$\begin{aligned}\mathbb{E}[z] &= 0 \\ \text{Var}[z] &= a^2 + b^2,\end{aligned} \tag{18.43}$$

so we could equivalently compute $z = \sqrt{a^2+b^2}\cdot\epsilon$ where ϵ is also drawn from a standard normal distribution.

Problem 18.3 Continue the process in equation 18.5 to show that:

$$\mathbf{z}_3 = \sqrt{(1-\beta_3)(1-\beta_2)(1-\beta_1)}\cdot\mathbf{x} + \sqrt{1-(1-\beta_3)(1-\beta_2)(1-\beta_1)}\cdot\boldsymbol{\epsilon}', \tag{18.44}$$

where $\boldsymbol{\epsilon}'$ is a draw from a standard normal distribution.

Problem 18.4* Prove the relation:

$$\text{Norm}_{\mathbf{v}}\left[\mathbf{A}\mathbf{w}, \mathbf{B}\right] \propto \text{Norm}_{\mathbf{w}}\Big[(\mathbf{A}^T\mathbf{B}^{-1}\mathbf{A})^{-1}\mathbf{A}^T\mathbf{B}^{-1}\mathbf{v}, (\mathbf{A}^T\mathbf{B}^{-1}\mathbf{A})^{-1}\Big]. \tag{18.45}$$

Problem 18.5* Prove the relation:

$$\text{Norm}_{\mathbf{x}}[\mathbf{a}, \mathbf{A}]\text{Norm}_{\mathbf{x}}[\mathbf{b}, \mathbf{B}] \;\propto\; \text{Norm}_{\mathbf{x}}\Big[(\mathbf{A}^{-1}+\mathbf{B}^{-1})^{-1}(\mathbf{A}^{-1}\mathbf{a}+\mathbf{B}^{-1}\mathbf{b}), (\mathbf{A}^{-1}+\mathbf{B}^{-1})^{-1}\Big]. \tag{18.46}$$

Problem 18.6* Derive equation 18.15.

Problem 18.7* Derive the third line of equation 18.25 from the second line.

Problem 18.8* The KL-divergence between two normal distributions in D dimensions with means $\mathbf{a}$ and $\mathbf{b}$ and covariance matrices $\mathbf{A}$ and $\mathbf{B}$ is given by:

$$D_{KL}\Big[\text{Norm}_{\mathbf{w}}[\mathbf{a}, \mathbf{A}]||\text{Norm}_{\mathbf{w}}[\mathbf{b}, \mathbf{B}]\Big] \;=\; \frac{1}{2}\left(\text{tr}\left[\mathbf{B}^{-1}\mathbf{A}\right] - d + (\mathbf{a}-\mathbf{b})^T\mathbf{B}^{-1}(\mathbf{a}-\mathbf{b}) + \log\left[\frac{|\mathbf{B}|}{|\mathbf{A}|}\right]\right). \tag{18.47}$$

Substitute the definitions from equation 18.27 into this expression and show that the only term that depends on the parameters $\boldsymbol{\phi}$ is the first term from equation 18.28.

Problem 18.9* If $\alpha_t = \prod_{s=1}^{t} 1-\beta_s$, then show that:

$$\sqrt{\frac{\alpha_t}{\alpha_{t-1}}} = \sqrt{1-\beta_t}. \tag{18.48}$$

Problem 18.10* If $\alpha_t = \prod_{s=1}^{t} 1-\beta_s$, then show that:

$$\frac{(1-\alpha_{t-1})(1-\beta_t)+\beta_t}{(1-\alpha_t)\sqrt{1-\beta_t}} = \frac{1}{\sqrt{1-\beta_t}}. \tag{18.49}$$

Problem 18.11* Prove equation 18.37.

Problem 18.12 Classifier-free guidance allows us to create more stereotyped "canonical" images of a given class. When we described transformer decoders, generative adversarial networks, and the GLOW algorithm, we also discussed methods to reduce the amount of variation and produce more stereotyped outputs. What were these? Do you think it's inevitable that we should limit the output of generative models in this way?

Chapter 19

Reinforcement learning

Reinforcement learning (RL) is a sequential decision-making framework in which agents learn to perform actions in an environment with the goal of maximizing received rewards. For example, an RL algorithm might control the moves (actions) of a character (the agent) in a video game (the environment), aiming to maximize the score (the reward). In robotics, an RL algorithm might control the movements (actions) of a robot (the agent) in the world (the environment) to perform a task (earning a reward). In finance, an RL algorithm might control a virtual trader (the agent) who buys or sells assets (the actions) on a trading platform (the environment) to maximize profit (the reward).

Consider learning to play chess. Here, there is a reward of $+1$, -1, or 0 at the end of the game if the agent wins, loses, or draws and 0 at every other time step. This illustrates the challenges of RL. First, the reward is sparse; here, we must play an entire game to receive feedback. Second, the reward is temporally offset from the action that caused it; a decisive advantage might be gained thirty moves before victory. We must associate the reward with this critical action. This is termed the *temporal credit assignment problem*. Third, the environment is stochastic; the opponent doesn't always make the same move in the same situation, so it's hard to know if an action was truly good or just lucky. Finally, the agent must balance exploring the environment (e.g., trying new opening moves) with exploiting what it already knows (e.g., sticking to a previously successful opening). This is termed the *exploration-exploitation trade-off*.

Reinforcement learning is an overarching framework that does not necessarily require deep learning. However, in practice, state-of-the-art systems often use deep networks. They encode the environment (the video game display, robot sensors, financial time series, or chessboard) and map this directly or indirectly to the next action (figure 1.13).

19.1 Markov decision processes, returns, and policies

Reinforcement learning maps observations of an environment to actions, aiming to maximize a numerical quantity that is connected to the rewards received. In the most common case, we learn a *policy* that maximizes the expected *return* in a *Markov decision process*. This section explains these terms.

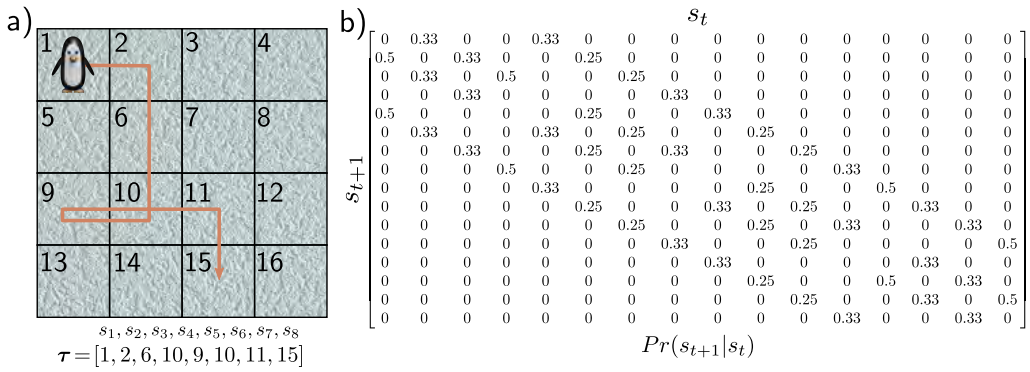


Figure 19.1 Markov process. A Markov process consists of a set of states and transition probabilities $Pr(s_{t+1}|s_t)$ that define the probability of moving to state s_{t+1} given the current state is s_t. a) The penguin can visit 16 different positions (states) on the ice. b) The ice is slippery, so at each time, it has an equal probability of moving to any adjacent state. For example, in position 6, it has a 25% chance of moving to states 2, 5, 7, and 10. A trajectory $\tau = [s_1, s_2, s_3, \ldots]$ from this process consists of a sequence of states.

19.1.1 Markov process

A *Markov process* assumes that the world is always in one of a set of possible states. The word *Markov* implies that the probability of being in a state depends only on the previous state and not on the states before. The changes between states are captured by the *transition probabilities* $Pr(s_{t+1}|s_t)$ of moving to the next state s_{t+1} given the current state s_t, where t indexes the time step. Hence, a Markov process is an evolving system that produces a sequence $s_1, s_2, s_3 \ldots$ of states (figure 19.1).

19.1.2 Markov reward process

A *Markov reward process* extends the Markov process to include a distribution $Pr(r_{t+1}|s_t)$ over the possible rewards r_{t+1} received at the next time step, given that we are in state s_t. This produces a sequence $s_1, r_2, s_2, r_3, s_3, r_4 \ldots$ of states and the associated rewards (figure 19.2). The Markov reward process also includes a *discount factor* $\gamma \in (0, 1]$ that is used to compute the *return* G_t at time t:

Problem 19.1

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}. \tag{19.1}$$

The return is the sum of the cumulative discounted future rewards; it measures the future benefit of being on this trajectory. A discount factor of less than one makes rewards that are closer in time more valuable than rewards that are further away.

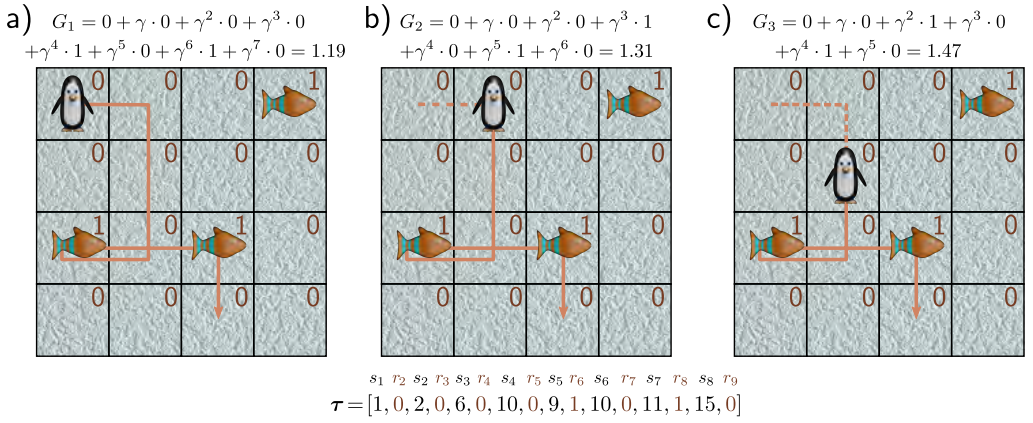


Figure 19.2 Markov reward process. This associates a distribution $Pr(r_{t+1}|s_t)$ of rewards r_{t+1} with each state s_t. a) Here, the rewards are deterministic; the penguin will receive a reward of $+1$ if it lands on a fish and 0 otherwise. The trajectory τ now consists of a sequence $s_1, r_2, s_2, r_3, s_3, r_4 \ldots$ of alternating states and rewards, terminating after eight steps. The return G_t of the sequence is the sum of discounted future rewards, here with discount factor $\gamma = 0.9$. b-c) As the penguin proceeds along the trajectory and gets closer to reaching the rewards, the return increases.

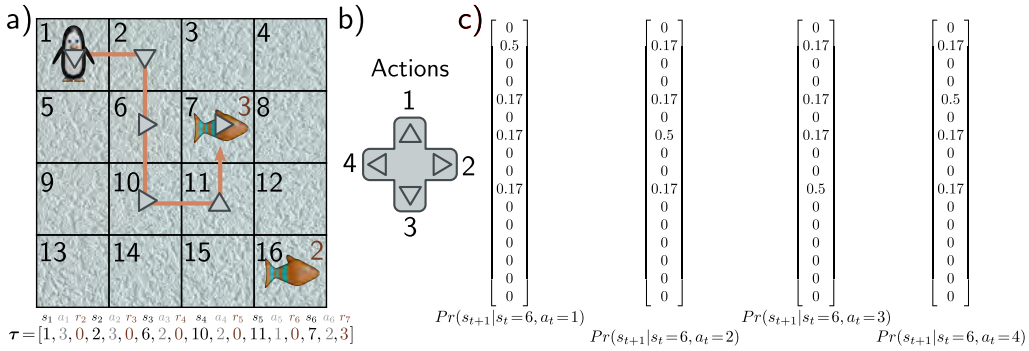


Figure 19.3 Markov decision process. a) The agent (penguin) can perform one of a set of actions in each state. The action influences both the probability of moving to the successor state and the probability of receiving rewards. b) Here, the four actions correspond to moving up, right, down, and left. c) For any state (here, state 6), the action changes the probability of moving to the next state. The penguin moves in the intended direction with 50% probability, but the ice is slippery, so it may slide to one of the other adjacent positions with equal probability. Accordingly, in panel (a), the action taken (gray arrows) doesn't always line up with the trajectory (orange line). Here, the action does not affect the reward, so $Pr(r_{t+1}|s_t, a_t) = Pr(r_{t+1}|s_t)$. The trajectory $\boldsymbol{\tau}$ from an MDP consists of a sequence $s_1, a_1, r_2, s_2, a_2, r_3, s_3, a_3, r_4 \ldots$ of alternating states s_t, actions a_t, and rewards, r_{t+1}. Note that here the penguin receives the reward when it *leaves* a state with a fish (i.e., the reward is received for passing through the fish square, regardless of whether the penguin arrived there intentionally or not).

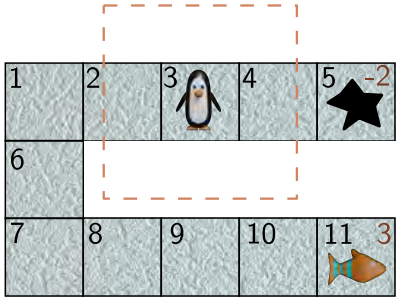


Figure 19.4 Partially observable Markov decision process (POMDP). In a POMDP, the agent does not have access to the entire state. Here, the penguin is in state three and can only see the region in the dashed box. This is indistinguishable from what it would see in state nine. In the first case, moving right leads to the hole in the ice (with -2 reward) and, in the latter, to the fish (with +3 reward).

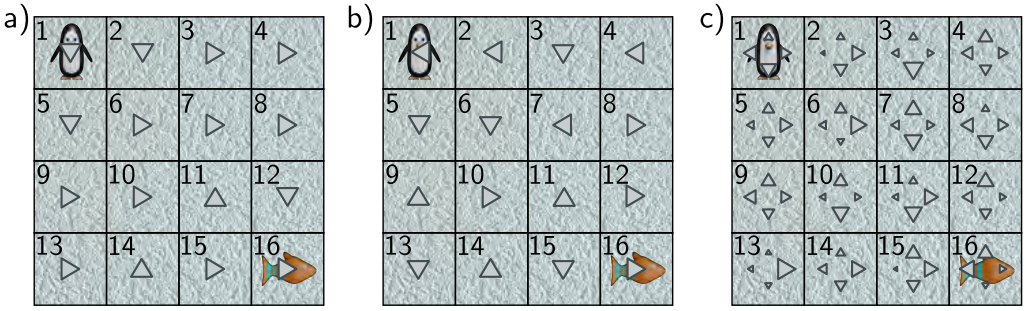


Figure 19.5 Policies. a) A deterministic policy always chooses the same action in each state (indicated by arrow). Some policies are better than others. This policy is not optimal but still generally steers the penguin from top-left to bottom-right where the reward lies. b) This policy is more random. c) A stochastic policy has a probability distribution over actions for each state (probability indicated by size of arrows). This has the advantage that the agent explores the states more thoroughly and can be necessary for optimal performance in partially observable Markov decision processes.

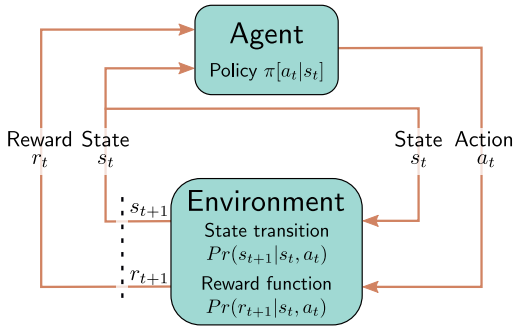


Figure 19.6 Reinforcement learning loop. The agent takes an action a_t at time t based on the state s_t, according to the policy $\pi[a_t|s_t]$. This triggers the generation of a new state s_{t+1} (via the state transition function) and a reward r_{t+1} (via the reward function). Both are passed back to the agent, which then chooses a new action.

19.1.3 Markov decision process

A *Markov decision process* or *MDP* adds a set of possible *actions* at each time step. The action a_t changes the transition probabilities, which are now written as $Pr(s_{t+1}|s_t, a_t)$. The rewards can also depend on the action and are now written as $Pr(r_{t+1}|s_t, a_t)$. An MDP produces a sequence $s_1, a_1, r_2, s_2, a_2, r_3, s_3, a_3, r_4 \ldots$ of states, actions, and rewards (figure 19.3). The entity that performs the actions is known as the *agent*.

19.1.4 Partially observable Markov decision process

In a *partially observable Markov decision process* or *POMDP*, the state is not directly visible (figure 19.4). Instead, the agent receives an observation o_t drawn from $Pr(o_t|s_t)$. Hence, a POMDP generates a sequence $s_1, o_1, a_1, r_2, s_2, o_2, a_2, r_3, o_3, a_3, s_3, r_4, \ldots$ of states, observations, actions, and rewards. In general, each observation will be more compatible with some states than others but insufficient to identify the state uniquely.

19.1.5 Policy

The rules that determine the agent's action for each state are known as the *policy* (figure 19.5). This may be stochastic (the policy defines a distribution over actions for each state) or deterministic (the agent always takes the same action in a given state). A stochastic policy $\pi[a|s]$ returns a probability distribution over each possible action a for state s, from which a new action is sampled. A deterministic policy $\pi[a|s]$ returns one for the action a that is chosen for state s and zero otherwise. A *stationary* policy depends only on the current state. A *non-stationary* policy also depends on the time step.

The environment and the agent form a loop (figure 19.6). The agent receives the state s_t and reward r_t from the last time step. Based on this, it can modify the policy $\pi[a_t|s_t]$ if desired and choose the next action a_t. The environment then assigns the next state according to $Pr(s_{t+1}|s_t, a_t)$ and the reward according to $Pr(r_{t+1}|s_t, a_t)$.

Notebook 19.1 Markov decision processes

19.2 Expected return

The previous section introduced the Markov decision process and the idea of an agent carrying out actions according to a policy. We want to choose a policy that maximizes the expected return. In this section, we make this idea mathematically precise. To do that, we assign a *value* to each state s_t and state-action pair $\{s_t, a_t\}$.

19.2.1 State and action values

The return G_t depends on the state s_t and the policy $\pi[a|s]$. From this state, the agent will pass through a sequence of states, taking actions and receiving rewards. This sequence differs every time the agent starts in the same place since, in general, the policy

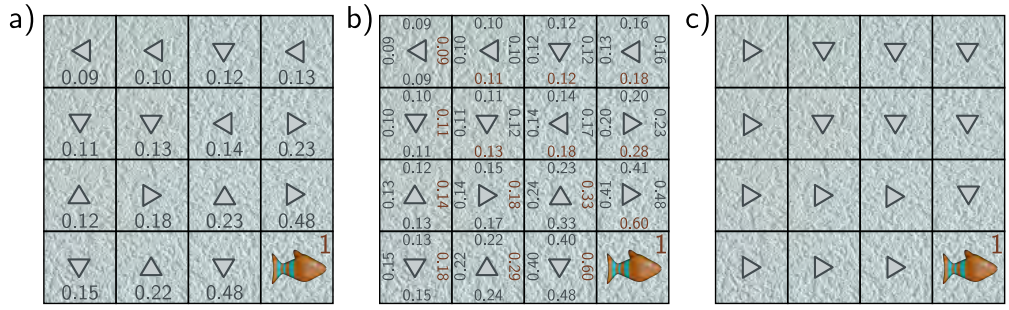


Figure 19.7 State and action values. a) The value $v[s_t|\pi]$ of a state s_t (number at each position) is the expected return for this state for a given policy π (gray arrows). It is the average sum of discounted rewards received over many trajectories started from this state. Here, states closer to the fish are more valuable. b) The value $q[s_t, a_t, \pi]$ of an action a_t in state s_t (four numbers at each position/state corresponding to four actions) is the expected return given that this particular action is taken in this state. In this case, it gets larger as we get closer to the fish and is larger for actions that head in the direction of the fish. c) If we know the action values at a state, then the policy can be modified so that it chooses the maximum of these values (red numbers in panel b).

$\pi[a_t|s_t]$, the state transitions $Pr(s_{t+1}|s_t, a_t)$, and the rewards issued $Pr(r_{t+1}|s_t, a_t)$ are all stochastic.

Appendix C.2 Expectation

We can characterize how "good" a state is under a given policy π by considering the expected return $v[s_t|\pi]$. This is the return that would be received on average from sequences that start from this state and is termed the *state value* or *state-value function* (figure 19.7a):

$$v[s_t|\pi] = \mathbb{E}\Big[G_t|s_t, \pi\Big]. \tag{19.2}$$

Informally, the state value tells us the *long-term* reward we can expect on average if we start in this state and follow the specified policy thereafter. It is highest for states where it's probable that subsequent transitions will bring large rewards soon (assuming the discount factor γ is less than one).

Similarly, the *action value* or *state-action value function* $q[s_t, a_t|\pi]$ is the expected return from executing action a_t in state s_t (figure 19.7b):

$$q[s_t, a_t|\pi] = \mathbb{E}\Big[G_t|s_t, a_t, \pi\Big]. \tag{19.3}$$

The action value tells us the long-term reward we can expect on average if we start in this state, take this action, and follow the specified policy thereafter. Through this quantity, reinforcement learning algorithms connect future rewards to current actions (i.e., resolve the temporal credit assignment problem).

19.2.2 Optimal policy

We want a policy that maximizes the expected return. For MDPs (but not POMDPs), there is always a deterministic, stationary policy that maximizes the value of every state. If we know this optimal policy, then we get the optimal state-value function $v^*[s_t]$:

$$v^*[s_t] = \max_{\pi}\Big[\mathbb{E}\Big[G_t|s_t,\pi\Big]\Big]. \tag{19.4}$$

Similarly, the optimal state-action value function is obtained under the optimal policy:

$$q^*[s_t,a_t] = \max_{\pi}\Big[\mathbb{E}\Big[G_t|s_t,a_t,\pi\Big]\Big]. \tag{19.5}$$

Turning this on its head, if we knew the optimal action-values $q^*[s_t,a_t]$, then we can derive the optimal policy by choosing the action a_t with the highest value (figure 19.7c):[1]

$$\pi[a_t|s_t] \leftarrow \underset{a_t}{\text{argmax}}\Big[q^*[s_t,a_t]\Big]. \tag{19.6}$$

Indeed, some reinforcement learning algorithms are based on alternately estimating the action values and the policy (see section 19.3).

19.2.3 Bellman equations

We may not know the state values $v[s_t]$ or action values $q[s_t,a_t]$ for any policy.[2] However, we know that they must be consistent with one another, and it's easy to write relations between these quantities. The state value $v[s_t]$ can be found by taking a weighted sum of the action values $q[s_t,a_t]$, where the weights depend on the probability under the policy $\pi[a_t|s_t]$ of taking that action (figure 19.8):

$$v[s_t] = \sum_{a_t}\pi[a_t|s_t]q[s_t,a_t]. \tag{19.7}$$

Similarly, the value of an action is the immediate reward $r_{t+1} = r[s_t,a_t]$ generated by taking the action, plus the value $v[s_{t+1}]$ of being in the subsequent state s_{t+1} discounted by γ (figure 19.9).[3] Since the assignment of s_{t+1} is not deterministic, we weight the values $v[s_{t+1}]$ according to the transition probabilities $Pr(s_{t+1}|s_t,a_t)$:

$$q[s_t,a_t] = r[s_t,a_t] + \gamma\cdot\sum_{s_{t+1}}Pr(s_{t+1}|s_t,a_t)v[s_{t+1}]. \tag{19.8}$$

Substituting equation 19.8 into equation 19.7 provides a relation between the state value at time t and $t+1$:

[1] The notation $\pi[a_t|s_t] \leftarrow a$ in equations 19.6, 19.12, and 19.13 means set $\pi[a_t|s]$ to one for action a and $\pi[a_t|s]$ to zero for other actions.

[2] For simplicity, we will just write $v[s_t]$ and $q[s_t,a_t]$ instead of $v[s_t|\pi]$ and $q[s_t,a_t|\pi]$ from now on.

[3] We also assume from now on that the rewards are deterministic and can be written as $r[s_t,a_t]$.

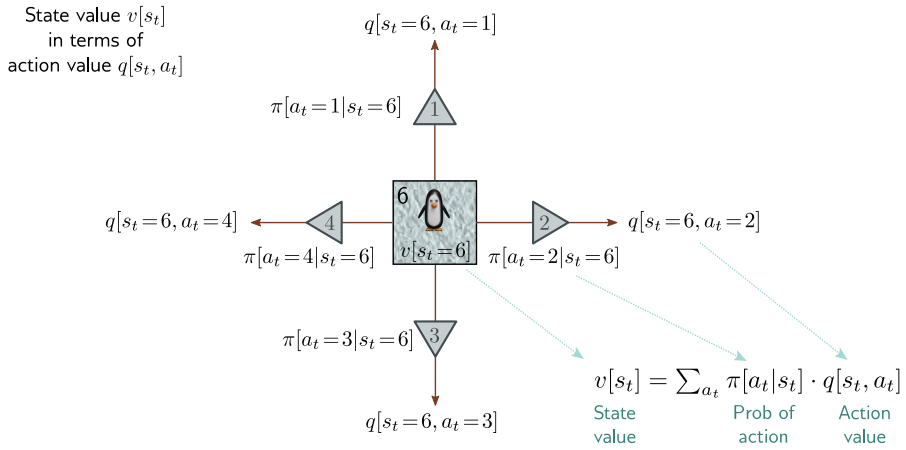


Figure 19.8 Relationship between state values and action values. The value of state six $v[s_t=6]$ is a weighted sum of the action values $q[s_t=6, a_t]$ at state six, where the weights are the policy probabilities $\pi[a_t|s_t=6]$ of taking that action.

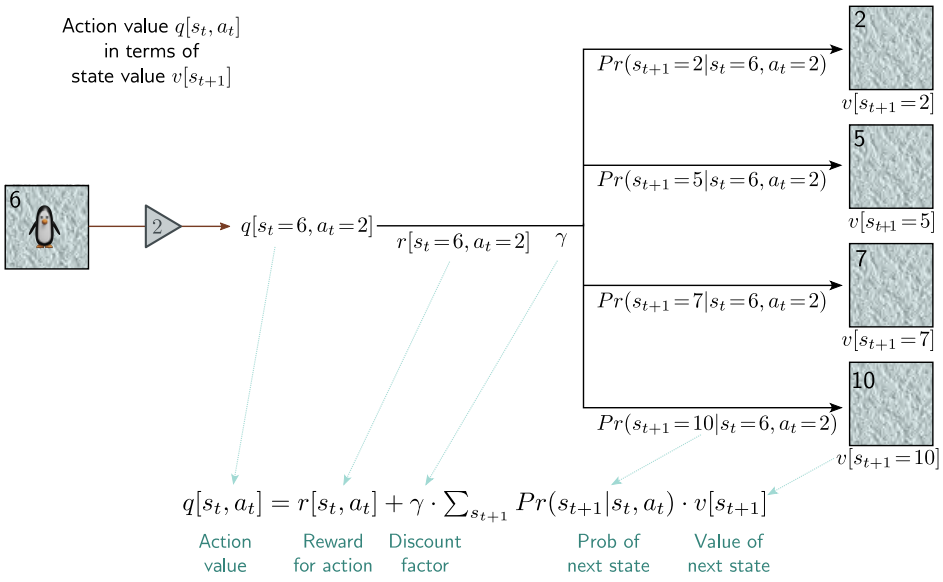


Figure 19.9 Relationship between action values and state values. The value $q[s_t=6, a_t=2]$ of taking action two in state six is the reward $r[s_t=6, a_t=2]$ from taking that action plus a weighted sum of the discounted values $v[s_{t+1}]$ of being in successor states, where the weights are the transition probabilities $Pr(s_{t+1}|s_t=6, a_t=2)$. The Bellman equations chain this relation with that of figure 19.8 to link the current and next (i) state values and (ii) action values.

$$v[s_t] = \sum_{a_t} \pi[a_t|s_t] \left(r[s_t, a_t] + \gamma \cdot \sum_{s_{t+1}} Pr(s_{t+1}|s_t, a_t) v[s_{t+1}] \right). \tag{19.9}$$

Similarly, substituting equation 19.7 into equation 19.8 provides a relation between the action value at time t and $t+1$:

$$q[s_t, a_t] = r[s_t, a_t] + \gamma \cdot \sum_{s_{t+1}} Pr(s_{t+1}|s_t, a_t) \left(\sum_{a_{t+1}} \pi[a_{t+1}|s_{t+1}] q[s_{t+1}, a_{t+1}] \right). \tag{19.10}$$

The latter two relations are the *Bellman equations* and are the backbone of many RL methods. In short, they say that the state (action) values have to be self-consistent. Consequently, when we update an estimate of one state (action) value, this will have a ripple effect that causes modifications to all the others.

19.3 Tabular reinforcement learning

Tabular RL algorithms (i.e., those that don't rely on function approximation) are divided into *model-based* and *model-free* methods. *Model-based methods* use the MDP structure explicitly and find the best policy from the transition matrix $Pr(s_{t+1}|s_t, a_t)$ and reward structure $r[s, a]$. If these are known, this is a straightforward optimization problem that can be tackled using *dynamic programming*. If they are unknown, they must first be estimated from observed MDP trajectories.[4]

Conversely, *model-free* methods eschew a model of the MDP and fall into two classes:

1. *Value estimation* approaches estimate the optimal state-action value function and then assign the policy according to the action in each state with the greatest value.
2. *Policy estimation* approaches directly estimate the optimal policy using a gradient descent technique without the intermediate steps of estimating the model or values.

Within each family, *Monte Carlo* methods simulate many trajectories through the MDP for a given policy to gather information from which this policy can be improved. Sometimes it is not feasible or practical to simulate many trajectories before updating the policy. *Temporal difference (TD)* methods update the policy *while* the agent traverses the MDP.

We now briefly describe dynamic programming methods, Monte Carlo value estimation methods, and TD value estimation methods. Section 19.4 describes how deep networks have been used in TD value estimation methods. We return to policy estimation in section 19.5.

[4] In RL, a *trajectory* is an observed sequence of states, rewards, and actions. A *rollout* is a *simulated* trajectory. An *episode* is a trajectory that starts in an initial state and ends in a terminal state (e.g., a full game of chess starting from the standard opening position and ending in a win, lose, or draw.)

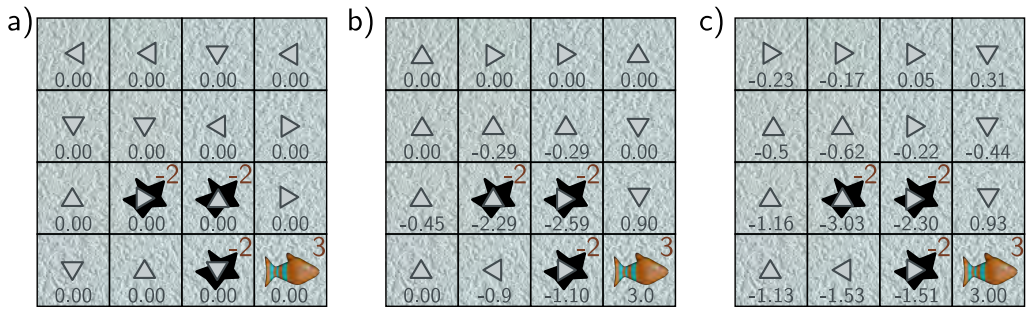


Figure 19.10 Dynamic programming. a) The state values are initialized to zero, and the policy (arrows) is chosen randomly. b) The state values are updated to be consistent with their neighbors (equation 19.11, shown after two iterations). The policy is updated to move the agent to states with the highest value (equation 19.12). c) After several iterations, the algorithm converges to the optimal policy, in which the penguin tries to avoid the holes and reach the fish.

19.3.1 Dynamic programming

Dynamic programming algorithms assume we have *perfect* knowledge of the transition and reward structure. In this respect, they are distinguished from most RL algorithms which observe the agent interacting with the environment to gather information about these quantities indirectly.

The state values $v[s]$ are initialized arbitrarily (usually to zero). The deterministic policy $\pi[a|s]$ is also initialized (e.g., by choosing a random action for each state). The algorithm then alternates between iteratively computing the state values for the current policy (*policy evaluation*) and improving that policy (*policy improvement*).

Policy evaluation: We sweep through the states s_t, updating their values:

$$v[s_t] \leftarrow \sum_{a_t} \pi[a_t|s_t] \left(r[s_t, a_t] + \gamma \cdot \sum_{s_{t+1}} Pr(s_{t+1}|s_t, a_t) v[s_{t+1}] \right), \tag{19.11}$$

where s_{t+1} is the successor state and $Pr(s_{t+1}|s_t, a_t)$ is the state transition probability. Each update makes $v[s_t]$ consistent with the value at the successor state s_{t+1} using the Bellman equation for state values (equation 19.9). This is termed *bootstrapping*.

Policy improvement: To update the policy, we greedily choose the action that maximizes the value for each state:

$$\pi[a_t|s_t] \leftarrow \underset{a_t}{\text{argmax}} \left[r[s_t, a_t] + \gamma \cdot \sum_{s_{t+1}} Pr(s_{t+1}|s_t, a_t) v[s_{t+1}] \right]. \tag{19.12}$$

This is guaranteed to improve the policy according to the *policy improvement theorem*.

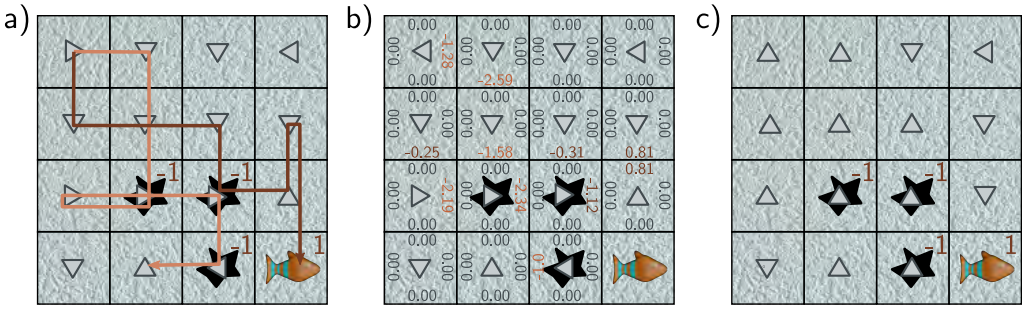


Figure 19.11 Monte Carlo methods. a) The policy (arrows) is initialized randomly. The MDP is repeatedly simulated, and the trajectories of these episodes are stored (orange and brown paths represent two trajectories). b) The action values are empirically estimated based on the observed returns averaged over these trajectories. In this case, the action values were all initially zero and have been updated where an action was observed. c) The policy can then be updated according to the action which received the best (or least bad) reward.

These two steps are iterated until the policy converges (figure 19.10).

Problems 19.2–19.3

There are many variations of this approach. In *policy iteration*, the policy evaluation step is iterated until convergence before policy improvement. The values can be updated either in place or synchronously in each sweep. In *value iteration*, the policy evaluation procedure sweeps through the values just once before policy improvement. *Asynchronous* dynamic programming algorithms don't have to systematically sweep through all the values at each step but can update a subset of the states in place in an arbitrary order.

Notebook 19.2 Dynamic programming

19.3.2 Monte Carlo methods

Unlike dynamic programming algorithms, Monte Carlo methods don't assume knowledge of the MDP's transition probabilities and reward structure. Instead, they gain experience by repeatedly sampling trajectories from the MDP and observing the rewards. They alternate between computing the action values (based on this experience) and updating the policy (based on the action values).

To estimate the action values $q[s, a]$, a series of *episodes* are run. Each starts with a given state and action and thereafter follows the current policy, producing a series of actions, states, and returns (figure 19.11a). The action value for a given state-action pair under the current policy is estimated as the average of the empirical returns that follow after each time this pair is observed (figure 19.11b). Then the policy is updated by choosing the action with the maximum value at every state (figure 19.11c):

$$\pi[a|s] \leftarrow \underset{a}{\text{argmax}}\Big[q[s, a]\Big]. \tag{19.13}$$

This is an *on-policy* method; the current best policy is used to guide the agent through the environment. This policy is based on the observed action values in every state, but of course, it's not possible to estimate the value of actions that haven't been used, and there is nothing to encourage the algorithm to explore these. One solution is to use *exploring starts*. Here, episodes with all possible state-action pairs are initiated, so every combination is observed at least once. However, this is impractical if the number of states is large or the starting point cannot be controlled. A different approach is to use an *epsilon greedy* policy, in which a random action is taken with probability ϵ, and the optimal action is allotted the remaining probability. The choice of ϵ trades off exploitation and exploration. Here, an on-policy method will seek the best policy from this epsilon-greedy family, which will *not* generally be the best overall policy.

Problem 19.4

Conversely, in *off-policy* methods, the optimal policy π (the *target policy*) is learned based on episodes generated by a different *behavior policy* π'. Typically, the target policy is deterministic, and the behavior policy is stochastic (e.g., an epsilon-greedy policy). Hence, the behavior policy can explore the environment, but the learned target policy remains efficient. Some off-policy methods explicitly use importance sampling (section 17.8.1) to estimate the action value under policy π using samples from π'. Others, such as Q-learning (described in the next section), estimate the values based on the greedy action, even though this is not necessarily what was chosen.

Notebook 19.3 Monte Carlo methods

19.3.3 Temporal difference methods

Dynamic programming methods use a bootstrapping process to update the values to make them self-consistent under the current policy. Monte Carlo methods sampled the MDP to acquire information. Temporal difference (TD) methods combine both bootstrapping and sampling. However, unlike Monte Carlo methods, they update the values and policy *while* the agent traverses the states of the MDP instead of afterward.

SARSA (State-Action-Reward-State-Action) is an on-policy algorithm with update:

$$q[s_t, a_t] \leftarrow q[s_t, a_t] + \alpha\Big(r[s_t, a_t] + \gamma \cdot q[s_{t+1}, a_{t+1}] - q[s_t, a_t]\Big), \tag{19.14}$$

where $\alpha \in \mathbb{R}^+$ is the learning rate. The bracketed term is called the *TD error* and measures the consistency between the estimated action value $q[s_t, a_t]$ and the estimate $r[s_t, a_t] + \gamma \cdot q[s_{t+1}, a_{t+1}]$ after taking a single step.

By contrast, *Q-Learning* is an off-policy algorithm with update (figure 19.12):

$$q[s_t, a_t] \leftarrow q[s_t, a_t] + \alpha\Big(r[s_t, a_t] + \gamma \cdot \max_a\big[q[s_{t+1}, a]\big] - q[s_t, a_t]\Big), \tag{19.15}$$

where now the choice of action at each step is derived from a different behavior policy π'.

Notebook 19.4 Temporal difference methods

In both cases, the policy is updated by taking the maximum of the action values at each state (equation 19.13). It can be shown that these updates are contraction mappings (see equation 16.20); the action values will eventually converge, assuming that every state-action pair is visited an infinite number of times.

Problem 19.5

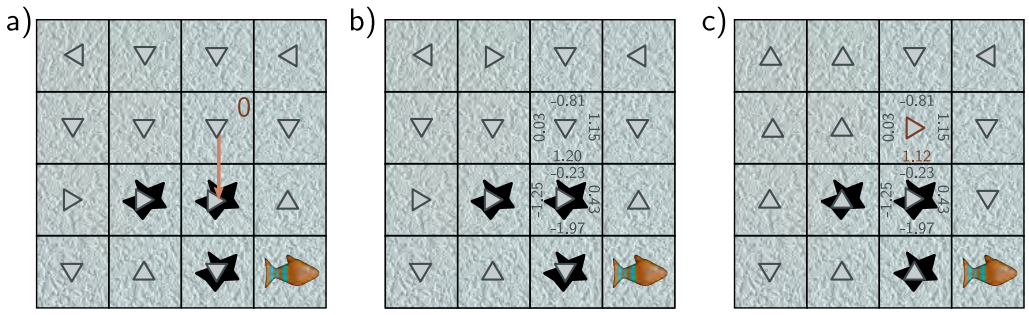


Figure 19.12 Q-learning. a) The agent starts in state s_t and takes action $a_t = 2$ according to the policy. It does not slip on the ice and moves downward, receiving reward $r[s_t, a_t] = 0$ for leaving the original state. b) The maximum action value at the new state is found (here 0.43). c) The action value for action 2 in the original state is updated to 1.12 based on the current estimate of the maximum action value at the subsequent state, the reward, discount factor $\gamma = 0.9$, and learning rate $\alpha = 0.1$. This changes the highest action value at the original state, so the policy changes.

19.4 Fitted Q-learning

The tabular Monte Carlo and TD algorithms described above repeatedly traverse the entire MDP and update the action values. However, this is only practical if the state-action space is small. Unfortunately, this is rarely the case; even for the constrained environment of a chessboard, there are more than 10^{40} possible legal states.

In *fitted Q-learning*, the discrete representation $q[s_t, a_t]$ of the action values is replaced by a machine learning model $q[\mathbf{s}_t, a_t, \boldsymbol{\phi}]$, where now the state is represented by a vector $\mathbf{s}_t$ rather than just an index. We then define a least squares loss based on the consistency of adjacent action values (similarly to in Q-learning, see equation 19.15):

$$L[\boldsymbol{\phi}] = \left(r[\mathbf{s}_t, a_t] + \gamma \cdot \max_a\Big[q[\mathbf{s}_{t+1}, a, \boldsymbol{\phi}]\Big] - q[\mathbf{s}_t, a_t, \boldsymbol{\phi}]\right)^2, \tag{19.16}$$

which in turn leads to the update:

$$\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} + \alpha\left(r[\mathbf{s}_t, a_t] + \gamma \cdot \max_a\Big[q[\mathbf{s}_{t+1}, a, \boldsymbol{\phi}]\Big] - q[\mathbf{s}_t, a_t, \boldsymbol{\phi}]\right)\frac{\partial q[\mathbf{s}_t, a_t, \boldsymbol{\phi}]}{\partial \boldsymbol{\phi}}. \tag{19.17}$$

Fitted Q-learning differs from Q-Learning in that convergence is no longer guaranteed. A change to the parameters potentially modifies both the target $r[\mathbf{s}_t, a_t] + \gamma \cdot \max_{a_{t+1}}[q[\mathbf{s}_{t+1}, a_{t+1}, \boldsymbol{\phi}]]$ (the maximum value may change) and the prediction $q[\mathbf{s}_t, a_t, \boldsymbol{\phi}]$. This can be shown both theoretically and empirically to damage convergence.

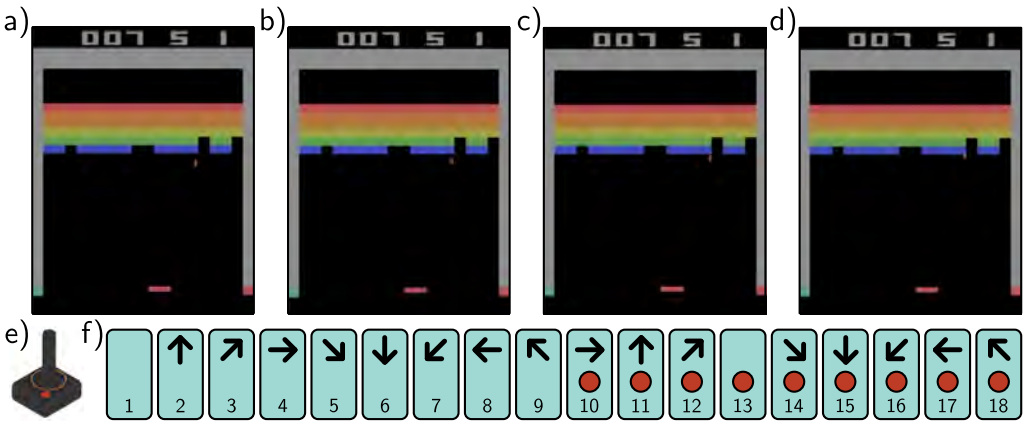


Figure 19.13 Atari Benchmark. The Atari benchmark consists of 49 Atari 2600 games, including Breakout (pictured), Pong, and various shoot-em-up, platform, and other types of games. a-d) Even for games with a single screen, the state is not fully observable from a single frame because the velocity of the objects is unknown. Consequently, it is usual to use several adjacent frames (here, four) to represent the state. e) The action simulates the user input via a joystick. f) There are eighteen actions corresponding to eight directions of movement or no movement, and for each of these nine cases, the button being pressed or not.

19.4.1 Deep Q-networks for playing ATARI games

Deep networks are ideally suited to making predictions from a high-dimensional state space, so they are a natural choice for the model in fitted Q-learning. In principle, they could take both state and action as input and predict the values, but in practice, the network takes only the state and simultaneously predicts the values for each action.

The *Deep Q-Network* was a breakthrough reinforcement learning architecture that exploited deep networks to learn to play ATARI 2600 games. The observed data comprises 220×160 images with 128 possible colors at each pixel (figure 19.13). This was reshaped to size 84×84, and only the brightness value was retained. Unfortunately, the full state is not observable from a single frame. For example, the velocity of game objects is unknown. To help resolve this problem, the network ingests the last four frames at each time step to form $\mathbf{s}_t$. It maps these frames through three convolutional layers followed by a fully connected layer to predict the value of every action (figure 19.14).

Several modifications were made to the standard training procedure. First, the rewards (which were driven by the score in the game) were clipped to -1 for a negative change and $+1$ for a positive change. This compensates for the wide variation in scores between different games and allows the same learning rate to be used. Second, the system exploited *experience replay*. Rather than update the network based on the tuple $<\mathbf{s}_t, a_t, r_{t+1}, \mathbf{s}_{t+1}>$ at the current step or with a batch of the last I tuples, all recent

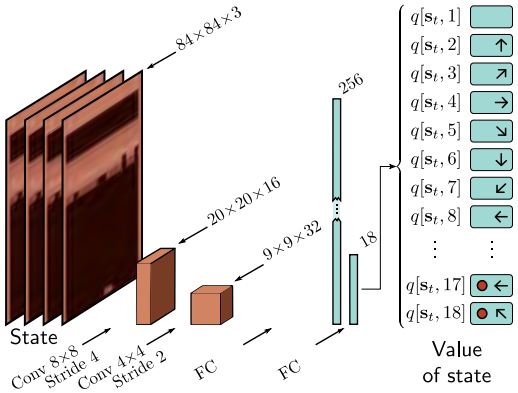


Figure 19.14 Deep Q-network architecture. The input $\mathbf{s}_t$ consists of four adjacent frames of the ATARI game. Each is resized to 84×84 and converted to grayscale. These frames are represented as four channels and processed by an 8×8 convolution with stride four, followed by a 4×4 convolution with stride 2, followed by two fully connected layers. The final output predicts the action value $q[\mathbf{s}_t, a_t]$ for each of the 18 actions in this state.

tuples were stored in a buffer. This buffer was sampled randomly to generate a batch at each step. This approach reuses data samples many times and reduces correlations between the samples in the batch that arise due to the similarity of adjacent frames.

Finally, the issue of convergence in fitted Q-Networks was tackled by fixing the target parameters to values $\boldsymbol{\phi}^-$ and only updating them periodically. This gives the update:

$$\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} + \alpha\Big(r[\mathbf{s}_t, a_t] + \gamma \cdot \max_a\Big[q[\mathbf{s}_{t+1}, a, \boldsymbol{\phi}^-]\Big] - q[\mathbf{s}_t, a_t, \boldsymbol{\phi}]\Big)\frac{\partial q[\mathbf{s}_t, a_t, \boldsymbol{\phi}]}{\partial \boldsymbol{\phi}}. \tag{19.18}$$

Now the network no longer chases a moving target and is less prone to oscillation.

Using these and other heuristics and with an ϵ-greedy policy, Deep Q-Networks performed at a level comparable to a professional game tester across a set of 49 games using the same network (trained separately for each game). It should be noted that the training process was data-intensive. It took around 38 full days of experience to learn each game. In some games, the algorithm exceeded human performance. On other games like "Montezuma's Revenge," it barely made any progress. This game features sparse rewards and multiple screens with quite different appearances.

19.4.2 Double Q-learning and double deep Q-networks

One potential flaw of Q-Learning is that the maximization over the actions in the update:

$$q[s_t, a_t] \leftarrow q[s_t, a_t] + \alpha\Big(r[s_t, a_t] + \gamma \cdot \max_a\big[q[s_{t+1}, a]\big] - q[s_t, a_t]\Big) \tag{19.19}$$

leads to a systematic bias in the estimated state values $q[s_t, a_t]$. Consider two actions that provide the same average reward, but one is stochastic and the other deterministic. The stochastic reward will exceed the average roughly half of the time and be chosen by the maximum operation, causing the corresponding action value $q[s_t, a_t]$ to be overestimated. A similar argument can be made about random inaccuracies in the output of the network $q[\mathbf{s}_t, a_t, \boldsymbol{\phi}]$ or random initializations of the q-function.

The underlying problem is that the same network both selects the target (by the maximization operation) and updates the value. Double Q-Learning tackles this problem by training two models $q_1[s_t, a_t, \pi_1]$ and $q_2[s_t, a_t, \pi_2]$ simultaneously:

$$\begin{aligned} q_1[s_t, a_t] &\leftarrow q_1[s_t, a_t] + \alpha\Big(r[s_t, a_t] + \gamma \cdot q_2\left[s_{t+1}, \underset{a}{\operatorname{argmax}}\Big[q_1[s_{t+1}, a]\Big]\right] - q_1[s_t, a_t]\Big) \\ q_2[s_t, a_t] &\leftarrow q_2[s_t, a_t] + \alpha\Big(r[s_t, a_t] + \gamma \cdot q_1\left[s_{t+1}, \underset{a}{\operatorname{argmax}}\Big[q_2[s_{t+1}, a]\Big]\right] - q_2[s_t, a_t]\Big). \end{aligned} \tag{19.20}$$

Now the choice of the target and the target itself are decoupled, which helps prevent these biases. In practice, new tuples $< s, a, r, s' >$ are randomly assigned to update one model or another. This is known as *double Q-learning*. *Double deep Q-networks* or *double DQNs* use deep networks $q[\mathbf{s}_t, a_t, \boldsymbol{\phi}_1]$ and $q[\mathbf{s}_t, a_t, \boldsymbol{\phi}_2]$ to estimate the action values, and the update becomes:

$$\begin{aligned} \boldsymbol{\phi}_1 &\leftarrow \boldsymbol{\phi}_1 + \alpha\left(r[\mathbf{s}_t, a_t] + \gamma \cdot q\left[\mathbf{s}_{t+1}, \underset{a}{\operatorname{argmax}}\Big[q[\mathbf{s}_{t+1}, a, \boldsymbol{\phi}_1]\Big], \boldsymbol{\phi}_2\right] - q[\mathbf{s}_t, a_t, \boldsymbol{\phi}_1]\right)\frac{\partial q[\mathbf{s}_t, a_t, \boldsymbol{\phi}_1]}{\partial \boldsymbol{\phi}_1} \\ \boldsymbol{\phi}_2 &\leftarrow \boldsymbol{\phi}_2 + \alpha\left(r[\mathbf{s}_t, a_t] + \gamma \cdot q\left[\mathbf{s}_{t+1}, \underset{a}{\operatorname{argmax}}\Big[q[\mathbf{s}_{t+1}, a, \boldsymbol{\phi}_2]\Big], \boldsymbol{\phi}_1\right] - q[\mathbf{s}_t, a_t, \boldsymbol{\phi}_2]\right)\frac{\partial q[\mathbf{s}_t, a_t, \boldsymbol{\phi}_2]}{\partial \boldsymbol{\phi}_2}. \end{aligned} \tag{19.21}$$

19.5 Policy gradient methods

Q-learning estimates the action values first and then uses these to update the policy. Conversely, *policy-based methods* directly learn a stochastic policy $\pi[a_t|\mathbf{s}_t, \boldsymbol{\theta}]$. This is a function with trainable parameters $\boldsymbol{\theta}$ that maps a state $\mathbf{s}_t$ to a distribution $Pr(a_t|\mathbf{s}_t)$ over actions a_t from which we can sample. In MDPs, there is always an optimal deterministic policy. However, there are three reasons to use a stochastic policy:

1. A stochastic policy naturally helps with exploration of the space; we are not obliged to take the best action at each time step.
2. The loss changes smoothly as we modify a stochastic policy. This means we can use gradient descent methods even though the rewards are discrete. This is similar to using maximum likelihood in (discrete) classification problems. The loss changes smoothly as the model parameters change to make the true class more likely.
3. The MDP assumption is often incorrect; we usually don't have complete knowledge of the state. For example, consider an agent navigating in an environment where it can only observe nearby locations (e.g., figure 19.4). If two locations look identical, but the nearby reward structure is different, a stochastic policy allows the possibility of taking different actions until this ambiguity is resolved.

19.5.1 Derivation of gradient update

Consider a trajectory $\boldsymbol{\tau} = [\mathbf{s}_1, a_1, \mathbf{s}_2, a_2, \ldots, \mathbf{s}_T, a_T]$ through an MDP. The probability of this trajectory $Pr(\boldsymbol{\tau}|\boldsymbol{\theta})$ depends on both the state evolution function $Pr(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t)$ and the current stochastic policy $\pi[a_t|\mathbf{s}_t, \boldsymbol{\theta}]$:

$$Pr(\boldsymbol{\tau}|\boldsymbol{\theta}) = Pr(\mathbf{s}_1)\prod_{t=1}^{T}\pi[a_t|\mathbf{s}_t,\boldsymbol{\theta}]Pr(\mathbf{s}_{t+1}|\mathbf{s}_t,a_t). \tag{19.22}$$

Policy gradient algorithms aim to maximize the expected return $r[\boldsymbol{\tau}]$ over many such trajectories:

$$\boldsymbol{\theta} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\Big[\mathbb{E}_{\boldsymbol{\tau}}\Big[r[\boldsymbol{\tau}]\Big]\Big] = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\Big[\int Pr(\boldsymbol{\tau}|\boldsymbol{\theta})r[\boldsymbol{\tau}]d\boldsymbol{\tau}\Big], \tag{19.23}$$

where the return is the sum of all the rewards received along the trajectory.

To maximize this quantity, we use the gradient ascent update:

$$\begin{aligned}\boldsymbol{\theta} &\leftarrow \boldsymbol{\theta} + \alpha\cdot\frac{\partial}{\partial\boldsymbol{\theta}}\int Pr(\boldsymbol{\tau}|\boldsymbol{\theta})r[\boldsymbol{\tau}]d\boldsymbol{\tau}\\ &= \boldsymbol{\theta} + \alpha\cdot\int\frac{\partial Pr(\boldsymbol{\tau}|\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}r[\boldsymbol{\tau}]d\boldsymbol{\tau}.\end{aligned} \tag{19.24}$$

where α is the learning rate.

We want to approximate this integral with a sum over empirically observed trajectories. These are drawn from the distribution $Pr(\boldsymbol{\tau}|\boldsymbol{\theta})$, so to make progress, we multiply and divide the integrand by this distribution:

$$\begin{aligned}\boldsymbol{\theta} &\leftarrow \boldsymbol{\theta} + \alpha\cdot\int\frac{\partial Pr(\boldsymbol{\tau}|\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}r[\boldsymbol{\tau}]d\boldsymbol{\tau}\\ &= \boldsymbol{\theta} + \alpha\cdot\int Pr(\boldsymbol{\tau}|\boldsymbol{\theta})\frac{1}{Pr(\boldsymbol{\tau}|\boldsymbol{\theta})}\frac{\partial Pr(\boldsymbol{\tau}|\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}r[\boldsymbol{\tau}]d\boldsymbol{\tau}\\ &\approx \boldsymbol{\theta} + \alpha\cdot\frac{1}{I}\sum_{i=1}^{I}\frac{1}{Pr(\boldsymbol{\tau}_i|\boldsymbol{\theta})}\frac{\partial Pr(\boldsymbol{\tau}_i|\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}r[\boldsymbol{\tau}_i].\end{aligned} \tag{19.25}$$

This equation has a simple interpretation (figure 19.15); the update changes the parameters $\boldsymbol{\theta}$ to increase the likelihood $Pr(\boldsymbol{\tau}_i|\boldsymbol{\theta})$ of an observed trajectory $\boldsymbol{\tau}_i$ in proportion to the reward $r[\boldsymbol{\tau}_i]$ from that trajectory. However, it also normalizes by the probability of observing that trajectory in the first place to compensate for the fact that some trajectories are observed more often than others. If a trajectory is already common and yields high rewards, then we don't need to change much. The biggest updates will come from trajectories that are uncommon but create large rewards.

We can simplify this expression using the *likelihood ratio identity*:

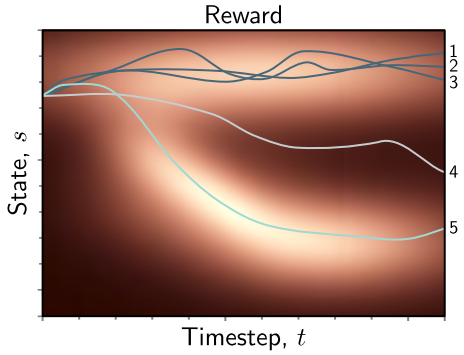


Figure 19.15 Policy gradients. Five episodes for the same policy (brighter indicates higher reward). Trajectories 1, 2, and 3 generate consistently high rewards, but similar trajectories already frequently occur with this policy, so there is no need to change. Conversely, trajectory 4 receives low rewards, so the policy should be modified to avoid producing similar trajectories. Trajectory 5 receives high rewards *and* is unusual. This will cause the largest change to the policy under equation 19.25.

$$\frac{\partial \log[\mathrm{f}[z]]}{\partial z} = \frac{1}{f[z]}\frac{\partial \mathrm{f}[z]}{\partial z}, \tag{19.26}$$

which yields the update:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \cdot \frac{1}{I}\sum_{i=1}^{I} \frac{\partial \log\big[Pr(\boldsymbol{\tau}_i|\boldsymbol{\theta})\big]}{\partial \boldsymbol{\theta}} r[\boldsymbol{\tau}_i]. \tag{19.27}$$

The log probability $\log[Pr(\boldsymbol{\tau}|\boldsymbol{\theta})]$ of a trajectory is given by:

$$\begin{aligned}\log[Pr(\boldsymbol{\tau}|\boldsymbol{\theta})] &= \log\Big[Pr(\mathbf{s}_1)\prod_{t=1}^{T}\pi[a_t|\mathbf{s}_t,\boldsymbol{\theta}]Pr(\mathbf{s}_{t+1}|\mathbf{s}_t,a_t)\Big] \\ &= \log\big[Pr(\mathbf{s}_1)\big] + \sum_{t=1}^{T}\log\big[\pi[a_t|\mathbf{s}_t,\boldsymbol{\theta}]\big] + \sum_{t=1}^{T}\log\big[Pr(\mathbf{s}_{t+1}|\mathbf{s}_t,a_t)\big],\end{aligned} \tag{19.28}$$

and noting that only the center term depends on $\boldsymbol{\theta}$, we can rewrite the update from equation 19.27 as:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \cdot \frac{1}{I}\sum_{i=1}^{I}\sum_{t=1}^{T} \frac{\partial \log\big[\pi[a_{it}|\mathbf{s}_{it},\boldsymbol{\theta}]\big]}{\partial \boldsymbol{\theta}} r[\boldsymbol{\tau}_i], \tag{19.29}$$

where $\mathbf{s}_{it}$ is the state at time t in episode i, and a_{it} is the action taken at time t in episode i. Note that since the terms relating to the state evolution $Pr(\mathbf{s}_{t+1}|\mathbf{s}_t,a_t)$ disappear, this parameter update does not assume a Markov time evolution process.

We can further simplify this by noting that:

$$r[\boldsymbol{\tau}_i] = \sum_{t=1}^{T} r_{it} = \sum_{k=1}^{t-1} r_{ik} + \sum_{k=t}^{T} r_{ik}, \tag{19.30}$$

where r_{it} is the reward at time t in the i^{th} episode. The first term (the rewards before time t) does not affect the update from time t, so we can write:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \cdot \frac{1}{I}\sum_{i=1}^{I}\sum_{t=1}^{T}\frac{\partial \log\left[\pi[a_{it}|\mathbf{s}_{it},\boldsymbol{\theta}]\right]}{\partial\boldsymbol{\theta}}\sum_{k=t}^{T} r_{ik}. \tag{19.31}$$

19.5.2 REINFORCE algorithm

REINFORCE is an early policy gradient algorithm that exploits this result and incorporates discounting. It is a Monte Carlo method that generates episodes $\boldsymbol{\tau}_i = [\mathbf{s}_{i1}, a_{i1}, r_{i2}, \mathbf{s}_{i2}, a_{i2}, r_{i3}, \ldots, r_{iT}]$ based on the current policy $\pi[a|\mathbf{s},\boldsymbol{\theta}]$. For discrete actions, this policy could be determined by a neural network $\pi[\mathbf{s}|\boldsymbol{\theta}]$, which takes the current state $\mathbf{s}$ and returns one output for each possible action. These outputs are passed through a softmax function to create a distribution over actions, which is sampled at each time step.

For each episode i, we loop through each step t and calculate the empirical discounted return for the partial trajectory $\boldsymbol{\tau}_{it}$ that starts at time t:

$$r[\boldsymbol{\tau}_{it}] = \sum_{k=t+1}^{T}\gamma^{k-t-1} r_{ik}, \tag{19.32}$$

and then we update the parameters for each time step t in each trajectory:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\cdot\gamma^{t}\frac{\partial \log\left[\pi_{a_{it}}[\mathbf{s}_{it},\boldsymbol{\theta}]\right]}{\partial\boldsymbol{\theta}} r[\boldsymbol{\tau}_{it}] \qquad \forall\, i,t, \tag{19.33}$$

where $\pi_{a_t}[\mathbf{s}_t,\boldsymbol{\theta}]$ is the probability of a_t produced by the neural network given the current state $\mathbf{s}_t$ and parameters $\boldsymbol{\theta}$, and α is the learning rate. The extra term γ^t ensures that the rewards are discounted relative to the start of the sequence because we maximize the log probability of returns in the whole sequence (equation 19.23).

19.5.3 Baselines

Policy gradient methods have the drawback that they exhibit high variance; many episodes may be needed to get stable updates of the derivatives. One way to reduce this variance is to subtract the trajectory returns $r[\boldsymbol{\tau}]$ from a baseline b:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \cdot \frac{1}{I}\sum_{i=1}^{I}\sum_{t=1}^{T}\frac{\partial \log\left[\pi_{a_{it}}[\mathbf{s}_{it},\boldsymbol{\theta}]\right]}{\partial\boldsymbol{\theta}}\left(r[\boldsymbol{\tau}_{it}] - b\right). \tag{19.34}$$

As long as the baseline b doesn't depend on the actions: Problem 19.6

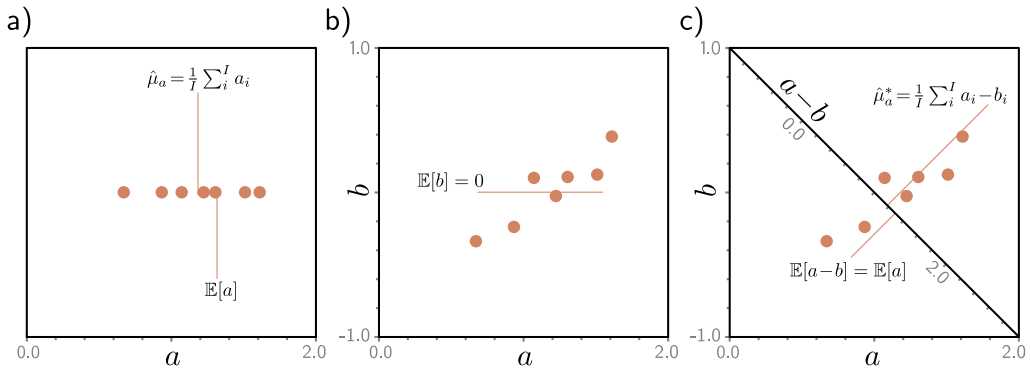


Figure 19.16 Decreasing variance of estimates using control variates. a) Consider trying to estimate $\mathbb{E}[a]$ from a small number of samples. The estimate (the mean of the samples) will vary based on the number of samples and the variance of those samples. b) Now consider observing another variable b that co-varies with a and has $\mathbb{E}[b] = 0$ and the same variance as a. c) The variance of the samples of $a - b$ is much less than that of a, but the expected value $\mathbb{E}[a - b] = \mathbb{E}[a]$, so we get an estimator with lower variance.

$$\mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t=1}^{T} \frac{\partial \log\left[\pi_{a_{it}}[\mathbf{s}_{it}, \boldsymbol{\theta}]\right]}{\partial \boldsymbol{\theta}} \cdot b\right] = 0, \tag{19.35}$$

Notebook 19.5 Control variates

and the expected value will not change. However, if the baseline co-varies with irrelevant factors that add uncertainty, then subtracting it reduces the variance (figure 19.16). This is a special case of the method of *control variates* (see problem 19.7).

Problem 19.7

This raises the question of how we should choose b. We can find the value of b that minimizes the variance by writing an expression for the variance, taking the derivative with respect to b, setting the result to zero, and solving to yield:

Problem 19.8

$$b = \sum_i \frac{\sum_{t=1}^{T}\left(\partial \log\left[\pi_{a_{it}}[\mathbf{s}_{it}, \boldsymbol{\theta}]\right]/\partial \boldsymbol{\theta}\right)^2 r[\boldsymbol{\tau}_{it}]}{\sum_{t=1}^{T}\left(\partial \log\left[\pi_{a_{it}}[\mathbf{s}_{it}, \boldsymbol{\theta}]\right]/\partial \boldsymbol{\theta}\right)^2}. \tag{19.36}$$

In practice, this is often approximated as:

$$b = \frac{1}{I}\sum_i r[\boldsymbol{\tau}_i]. \tag{19.37}$$

Subtracting this baseline factors out variance that might occur when the returns $r[\boldsymbol{\tau}_i]$ from all trajectories are greater than is typical but only because they happen to pass through states with higher than average returns *whatever* actions are taken.

19.5.4 State-dependent baselines

A better option is to use a baseline $b[\mathbf{s}_{it}]$ that depends on the current state $\mathbf{s}_{it}$.

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \cdot \frac{1}{I}\sum_{i=1}^{I}\sum_{t=1}^{T} \frac{\partial \log\left[\pi_{a_{it}}[\mathbf{s}_{it}, \boldsymbol{\theta}]\right]}{\partial \boldsymbol{\theta}} \left(r[\boldsymbol{\tau}_{it}] - b[\mathbf{s}_{it}]\right). \tag{19.38}$$

Here, we are compensating for variance introduced by some states having greater overall returns than others, whichever actions we take.

A sensible choice is the expected future reward based on the current state, which is just the state value $v[\mathbf{s}]$. In this case, the difference between the empirically observed rewards and the baseline is known as the *advantage estimate*. Since we are in a Monte Carlo context, this can be parameterized by a neural network $b[\mathbf{s}] = v[\mathbf{s}, \boldsymbol{\phi}]$ with parameters $\boldsymbol{\phi}$, which we can fit to the observed returns using least squares loss:

$$L[\boldsymbol{\phi}] = \sum_{i=1}^{I}\sum_{t=1}^{T}\left(v[\mathbf{s}_{it}, \boldsymbol{\phi}] - \sum_{j=y}^{T} r_{ij}\right)^2. \tag{19.39}$$

19.6 Actor-critic methods

Actor-critic algorithms are temporal difference (TD) policy gradient algorithms. They can update the parameters of the policy network at each step. This contrasts with the Monte Carlo REINFORCE algorithm, which *must* wait for one or more episodes to complete before updating the parameters.

In the TD approach, we do not have access to the future rewards $r[\boldsymbol{\tau}_t] = \sum_{k=t}^{T} r_k$ along this trajectory. Actor-critic algorithms approximate the sum over all the future rewards with the observed current reward plus the discounted value of the next state:

$$r[\boldsymbol{\tau}_{it}] \approx r_{it} + \gamma \cdot v[\mathbf{s}_{i,t+1}, \boldsymbol{\phi}]. \tag{19.40}$$

Here the value $v[\mathbf{s}_{i,t+1}, \boldsymbol{\phi}]$ is estimated by a second neural network with parameters $\boldsymbol{\phi}$.

Substituting this into equation 19.38 gives the update:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \cdot \frac{1}{I}\sum_{i=1}^{I}\sum_{t=1}^{T} \frac{\partial \log\left[Pr(a_{it}|\mathbf{s}_{it}, \boldsymbol{\theta})\right]}{\partial \boldsymbol{\theta}} \Big(r_{it} + \gamma \cdot v[\mathbf{s}_{i,t+1}, \boldsymbol{\phi}] - v[\mathbf{s}_{i,t}, \boldsymbol{\phi}]\Big). \tag{19.41}$$

Concurrently, we update the parameters $\boldsymbol{\phi}$ by bootstrapping using the loss function:

$$L[\boldsymbol{\phi}] = \sum_{i=1}^{I}\sum_{t=1}^{T}\left(r_{it} + \gamma \cdot v[\mathbf{s}_{i,t+1}, \boldsymbol{\phi}] - v[\mathbf{s}_{i,t}, \boldsymbol{\phi}]\right)^2. \tag{19.42}$$

The policy network $\pi[\mathbf{s}_t, \boldsymbol{\theta}]$ that predicts $Pr(a|\mathbf{s}_t)$ is termed the *actor*. The value network $v[\mathbf{s}_t, \boldsymbol{\phi}]$ is termed the *critic*. Often the same network represents both actor and

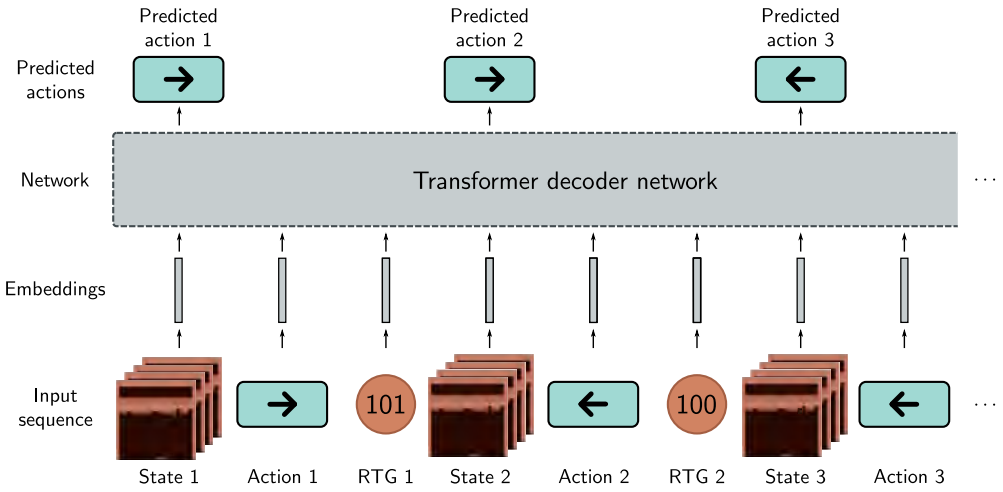


Figure 19.17 Decision transformer. The decision transformer treats offline reinforcement learning as a sequence prediction task. The input is a sequence of states, actions, and returns-to-go (remaining rewards in the episode), each of which is mapped to a fixed-size embedding. At each time step, the network predicts the next action. During testing, the returns-to-go are unknown; in practice, an initial estimate is made from which subsequent observed rewards are subtracted.

the critic, with two sets of outputs that predict the policy and the values, respectively. Note that although actor-critic methods can update the policy parameters at each step, this is rarely done in practice. The agent typically collects a batch of experience over many time steps before the policy is updated.

19.7 Offline reinforcement learning

Interaction with the environment is at the core of reinforcement learning. However, there are some scenarios where it is not practical to send a naïve agent into an environment to explore the effect of different actions. This may be because erratic behavior in the environment is dangerous (e.g., driving autonomous vehicles) or because data collection is time-consuming or expensive (e.g., making financial trades).

However, it *is* possible to gather historical data from human agents in both cases. *Offline RL* or *batch RL* aims to learn how to take actions that maximize rewards on future episodes by observing past sequences $\mathbf{s}_1, a_1, r_2, \mathbf{s}_2, a_2, r_3, \ldots$, without ever interacting with the environment. It is distinct from *imitation learning*, a related technique that (i) does not have access to the rewards and (ii) attempts to replicate the performance of a historical agent rather than improve it.

Although there are offline RL methods based on Q-Learning and policy gradients,

this paradigm opens up new possibilities. In particular, we can treat this as a sequence learning problem, in which the goal is to predict the next action, given the history of states, rewards, and actions. The *decision transformer* exploits a transformer decoder framework (section 12.7) to make these predictions (figure 19.17).

However, the goal is to predict actions based on *future rewards*, and these are not captured in a standard $\mathbf{s}, a, r$ sequence. Hence, the decision transformer replaces the reward r_t with the *returns-to-go* $R_{t:T} = \sum_{t'=t}^{T} r_{t'}$ (i.e., the sum of the empirically observed future rewards). The remaining framework is very similar to a standard transformer decoder. The states, actions, and returns-to-go are converted to fixed-size embeddings via learned mappings. For Atari games, the state embedding might be converted via a convolutional network similar to that in figure 19.14. The embeddings for the actions and returns-to-go can be learned in the same way as word embeddings (figure 12.9). The transformer is trained with masked self-attention and position embeddings.

This formulation is natural during training but poses a quandary during inference because we don't know the returns-to-go. This can be resolved by using the desired total return at the first step and decrementing this as rewards are received. For example, in an Atari game, the desired total return would be the total score required to win.

Decision transformers can also be fine-tuned from online experience and hence learn over time. They have the advantage of dispensing with most of the reinforcement learning machinery and its associated instability and replacing this with standard supervised learning. Transformers can learn from enormous quantities of data and integrate information across large time contexts (making the temporal credit assignment problem more tractable). This represents an intriguing new direction for reinforcement learning.

19.8 Summary

Reinforcement learning is a sequential decision-making framework for Markov decision processes and similar systems. This chapter reviewed tabular approaches to RL, including dynamic programming (in which the environment model is known), Monte Carlo methods (in which multiple episodes are run and the action values and policy subsequently changed based on the rewards received), and temporal difference methods (in which these values are updated while the episode is ongoing).

Deep Q-Learning is a temporal difference method where deep neural networks are used to predict the action value for every state. It can train agents to perform Atari 2600 games at a level similar to humans. Policy gradient methods directly optimize the policy rather than assigning values to actions. They produce stochastic policies, which are important when the environment is partially observable. The updates are noisy, and many refinements have been introduced to reduce their variance.

Offline reinforcement learning is used when we cannot interact with the environment but must learn from historical data. The decision transformer leverages recent progress in deep learning to build a model of the state-action-reward sequence and predict the actions that will maximize the rewards.

Notes

Sutton & Barto (2018) cover tabular reinforcement learning methods in depth. Li (2017), Arulkumaran et al. (2017), François-Lavet et al. (2018), and Wang et al. (2022c) all provide overviews of deep reinforcement learning. Graesser & Keng (2019) is an excellent introductory resource that includes Python code.

Landmarks in deep reinforcement learning: Most landmark achievements of reinforcement learning have been in either video games or real-world games since these provide constrained environments with limited actions and fixed rules. Deep Q-Learning (Mnih et al., 2015) achieved human-level performance across a benchmark of ATARI games. AlphaGo (Silver et al., 2016) beat the world champion at Go. This game was previously considered very difficult for computers to play. Berner et al. (2019) built a system that beat the world champion team in the five vs. five-player game *Defense of the Ancients 2*, which requires cooperation across players. Ye et al. (2021) built a system that could beat humans on Atari games with limited data (in contrast to previous systems, which need much more experience than humans). More recently, the Cicero system demonstrated human-level performance in the game *Diplomacy* which requires natural language negotiations and coordination between players (FAIR, 2022).

RL has also been applied successfully to combinatorial optimization problems (see Mazyavkina et al., 2021). For example, Kool et al. (2019) learned a model that performed similarly to the best heuristics for the traveling salesman problem. Recently, AlphaTensor (Fawzi et al., 2022) treated matrix multiplication as a game and learned faster ways to multiply matrices using fewer multiplication operations. Since deep learning relies heavily on matrix multiplication, this is one of the first examples of self-improvement in AI.

Classical reinforcement learning methods: Very early contributions to the theory of MDPs were made by Thompson (1933) and Thompson (1935). The Bellman recursions were introduced by Bellman (1966). Howard (1960) introduced policy iteration. Sutton & Barto (2018) identify the work of Andreae (1969) as being the first to describe RL using the MDP formalism.

The modern era of reinforcement learning arguably originated in the Ph.D. theses of Sutton (1984) and Watkins (1989). Sutton (1988) introduced the term temporal difference learning. Watkins (1989) and Watkins & Dayan (1992) introduced Q-Learning and showed that it converges to a fixed point by Banach's theorem because the Bellman operator is a contraction mapping. Watkins (1989) made the first explicit connection between dynamic programming and reinforcement learning. SARSA was developed by Rummery & Niranjan (1994). Gordon (1995) introduced *fitted Q-learning* in which a machine learning model is used to predict the action value for each state-action pair. Riedmiller (2005) introduced *neural-fitted Q-learning*, which used a neural network to predict all the action values at once from a state. Early work on Monte Carlo methods was carried out by Singh & Sutton (1996), and the exploring starts algorithm was introduced by Sutton & Barto (1999). Note that this is an extremely cursory summary of more than fifty years of work. A much more thorough treatment can be found in Sutton & Barto (2018).

Deep Q-Networks: Deep Q-Learning was devised by Mnih et al. (2015) and is an intellectual descendent of neural-fitted Q-learning. It exploited the then-recent successes of convolutional networks to develop a fitted Q-Learning method that could achieve human-level performance on a benchmark of ATARI games. Deep Q-Learning suffers from the *deadly triad issue* (Sutton & Barto, 2018): training can be unstable in any scheme that incorporates (i) bootstrapping, (ii) off-policy learning, and (iii) function approximation. Much subsequent work has aimed to make training more stable. Mnih et al. (2015) introduced the experience replay buffer (Lin, 1992), which was subsequently improved by Schaul et al. (2016) to favor more important tuples and hence increase learning speed. This is termed *prioritized experience replay*.

The original Q-Learning paper concatenated four frames so the network could observe the velocities of objects and make the underlying process closer to fully observable. Hausknecht & Stone (2015) introduced *deep recurrent Q-learning*, which used a recurrent network architecture that only ingested a single frame at a time because it could "remember" the previous states. Van Hasselt (2010) identified the systematic overestimation of the state values due to the max operation and proposed double Q-Learning in which two models are trained simultaneously to remedy this. This was subsequently applied in the context of deep Q-learning (Van Hasselt et al., 2016), although its efficacy has since been questioned (Hessel et al., 2018). Wang et al. (2016) introduced *deep dueling networks* in which two heads of the same network predict (i) the state value and (ii) the *advantage* (relative value) of each action. The intuition here is that sometimes it is the state value that is important, and it doesn't matter much which action is taken, and decoupling these estimates improves stability.

Fortunato et al. (2018) introduced *noisy deep Q-Networks*, in which some weights in the Q-Network are multiplied by noise to add stochasticity to the predictions and encourage exploration. The network can learn to decrease the magnitudes of the noise over time as it converges to a sensible policy. Distributional DQN (Bellemare et al., 2017a; Dabney et al., 2018 following Morimura et al., 2010) aims to estimate more complete information about the distribution of returns than just the expectation. This potentially allows the network to mitigate against worst-case outcomes and can also improve performance, as predicting higher moments provides a richer training signal. *Rainbow* (Hessel et al., 2018) combined six improvements to the original deep Q-learning algorithm, including dueling networks, distributional DQN, and noisy DQN, to improve both the training speed and the final performance on the ATARI benchmark.

Policy gradients: Williams (1992) introduced the REINFORCE algorithm. The term "policy gradient method" dates to Sutton et al. (1999). Konda & Tsitsiklis (1999) introduced the actor-critic algorithm. Decreasing the variance by using different baselines is discussed in Greensmith et al. (2004) and Peters & Schaal (2008). It has since been argued that the value baseline primarily reduces the aggressiveness of the updates rather than their variance (Mei et al., 2022).

Policy gradients have been adapted to produce deterministic policies (Silver et al., 2014; Lillicrap et al., 2016; Fujimoto et al., 2018). The most direct approach is to maximize over the possible actions, but if the action space is continuous, this requires an optimization procedure at each step. The *deep deterministic policy gradient* algorithm (Lillicrap et al., 2016) moves the policy in the direction of the gradient of the action value (implying the use of an actor-critic method).

Modern policy gradients: We introduced policy gradients in terms of the parameter update. However, they can also be viewed as optimizing a surrogate loss based on importance sampling of the expected rewards, using trajectories from the current policy parameters. This view allows us to take multiple optimization steps validly. However, this can cause very large policy updates. Overstepping is a minor problem in supervised learning, as the trajectory can be corrected later. However, in RL, it affects future data collection and can be extremely destructive.

Several methods have been proposed to moderate these updates. *Natural policy gradients* (Kakade, 2001) are based on natural gradients (Amari, 1998), which modify the descent direction by the Fisher information matrix. This provides a better update which is less likely to get stuck in local plateaus. However, the Fisher matrix is impractical to compute in models with many parameters. In *trust-region policy optimization* or *TRPO* (Schulman et al., 2015), the surrogate objective is maximized subject to a constraint on the KL divergence between the old and new policies. Schulman et al. (2017) propose a simpler formulation in which this KL divergence appears as a regularization term. The regularization weight is adapted based on the distance between the KL divergence and a target indicating how much we want the policy to change. *Proximal policy optimization* or *PPO* (Schulman et al., 2017) is an even simpler approach in which the loss is clipped to ensure smaller updates.

Actor-critic: In the actor-critic algorithm (Konda & Tsitsiklis, 1999) described in section 19.6, the critic used a 1-step estimator. It's also possible to use k-step estimators (in which we

observe k discounted rewards and approximate subsequent rewards with an estimate of the state value). As k increases, the variance of the estimate increases, but the bias decreases. *Generalized advantage estimation* (Schulman et al., 2016) weights together estimates from many steps and parameterizes the weighting by a single term that trades off the bias and the variance. Mnih et al. (2016) introduced *asynchronous actor-critic* or *A3C* in which multiple agents are run independently in parallel environments and update the same parameters. Both the policy and value function are updated every T time steps using a mix of k-step returns. Wang et al. (2017) introduced several methods designed to make asynchronous actor-critic more efficient. *Soft actor-critic* (Haarnoja et al., 2018b) adds an entropy term to the cost function, which encourages exploration and reduces overfitting as the policy is encouraged to be less confident.

Offline RL: In offline reinforcement learning, the policy is learned by observing the behavior of other agents, including the rewards they receive, *without* the ability to change the policy. It is related to imitation learning, where the goal is to copy the behavior of another agent without access to rewards (see Hussein et al., 2017). One approach is to treat offline RL in the same way as off-policy reinforcement learning. However, in practice, the distributional shift between the observed and applied policy manifests in overly optimistic estimates of the action value and poor performance (see Fujimoto et al., 2019; Kumar et al., 2019a; Agarwal et al., 2020). Conservative Q-learning (Kumar et al., 2020b) learns conservative, lower-bound estimates of the value function by regularizing the Q-values. The decision transformer (Chen et al., 2021c) is a simple approach to offline learning that takes advantage of the well-studied self-attention architecture. It can subsequently be fine-tuned with online training (Zheng et al., 2022).

Reinforcement learning and chatbots: Chatbots can be trained using a technique known as *reinforcement learning with human feedback* or *RLHF* (Christiano et al., 2018; Stiennon et al., 2020). For example, *InstructGPT* (the forerunner of ChatGPT, Ouyang et al., 2022) starts with a standard transformer decoder model. This is then fine-tuned based on prompt-response pairs where the response was written by human annotators. During this training step, the model is optimized to predict the next word in the ground truth response.

Unfortunately, such training data are expensive to produce in sufficient quantities to support high-quality performance. To resolve this problem, human annotators then indicate which of several model responses they prefer. These (much cheaper) data are used to train a *reward model*. This is a second transformer network that ingests the prompt and model response and returns a scalar indicating how good the response is. Finally, the fine-tuned chatbot model is further trained to produce high rewards using the reward model as supervision. Here, standard gradient descent cannot be used as it's not possible to compute derivatives through the sampling procedure in the chatbot output. Hence, the model is trained with proximal policy optimization (a policy gradient method where the derivatives are tractable) to generate higher rewards.

Other areas of RL: Reinforcement learning is an enormous area, which easily justifies its own book, and this literature review is extremely superficial. Other notable areas of RL that we have not discussed include *model-based RL*, in which the state transition probabilities and reward functions are modeled (see Moerland et al., 2023). This allows forward planning and has the advantage that the same model can be reused for different reward structures. *Hybrid methods* such as AlphaGo (Silver et al., 2016) and MuZero (Schrittwieser et al., 2020) have separate models for the dynamics of the states, the policy, and the value of future positions.

This chapter has only discussed simple methods for exploration, like the epsilon-greedy approach, noisy Q-learning, and adding an entropy term to penalize overconfident policies. *Intrinsic motivation* refers to methods that add rewards for exploration and thus imbue the agent with "curiosity" (see Barto, 2013; Aubret et al., 2019). *Hierarchical reinforcement learning* (see Pateria et al., 2021) refers to methods that break down the final objective into sub-tasks. *Multi-agent reinforcement learning* (see Zhang et al., 2021a) considers the case where multiple agents coexist in a shared environment. This may be in either a competitive or cooperative context.

Problems

Problem 19.1 Figure 19.18 shows a single trajectory through the example MDP. Calculate the return for each step in the trajectory given that the discount factor γ is 0.9.

Problem 19.2* Prove the policy improvement theorem. Consider changing from policy π to policy π', where for state s_t the new policy π' chooses the action that maximizes the expected return:

$$\pi'[a_t|s_t] = \underset{a_t}{\text{argmax}}\Big[r[s_t, a_t] + \gamma \cdot \sum_{s_{t+1}} Pr(s_{t+1}|s_t, a_t) v[s_{t+1}|\pi]\Big]. \tag{19.43}$$

and for all other states, the policies are the same. Show that the value $v[s_t|\pi]$ for the original policy must be less than or equal to $v[s_t|\pi'] = q\big[s_t, \pi'[a|s_t]\big|\pi\big]$ for the new policy:

$$\begin{aligned} v[s_t|\pi] &\leq q\Big[s_t, \pi'[a_t|s_t]\Big|\pi\Big] \\ &= \mathbb{E}_{\pi'}\Big[r_{t+1} + \gamma \cdot v[s_{t+1}|\pi]\Big]. \end{aligned} \tag{19.44}$$

Hint: Start by writing the term $v[s_{t+1}|\pi]$ in terms of the new policy.

Problem 19.3 Show that when the state values and policy are initialized as in figure 19.10a, they become those in figure 19.10b after two iterations of (i) policy evaluation (in which all states are updated based on their current values and then replace the previous ones) and (ii) policy improvement. The state transition allots half the probability to the direction the policy indicates and divides the remaining probability equally between the other valid actions. The reward function returns -2 irrespective of the action when the penguin leaves a hole. The reward function returns +3 regardless of the action when the penguin leaves the fish tile and the episode ends, so the fish tile has a value of +3.

Problem 19.4 The *Boltzmann policy* strikes a balance between exploration and exploitation by basing the action probabilities $\pi[a|s]$ on the current state-action reward function $q[s, a]$:

$$\pi[a|s] = \frac{\exp\big[q[s, a]/\tau\big]}{\sum_{a'} \exp\big[q[s, a']/\tau\big]}. \tag{19.45}$$

Explain how the temperature parameter τ can be varied to prioritize exploration or exploitation.

Problem 19.5* When the learning rate α is one, the Q-Learning update is given by:

$$\mathrm{f}\big[q[s, a]\big] = r[s, a] + \gamma \cdot \max_a \big[q[s', a]\big]. \tag{19.46}$$

Show that this is a contraction mapping (equation 16.30) so that:

$$\Big|\Big|\mathrm{f}\big[q_1[s, a]\big] - \mathrm{f}\big[q_2[s, a]\big]\Big|\Big|_\infty < \Big|\Big|q_1[s, a] - q_2[s, a]\Big|\Big|_\infty \qquad \forall\, q_1, q_2. \tag{19.47}$$

where $||\bullet||_\infty$ represents the ℓ_∞ norm. It follows that a fixed point will exist by Banach's theorem and that the updates will eventually converge.

Appendix B.3.2 Vector norms

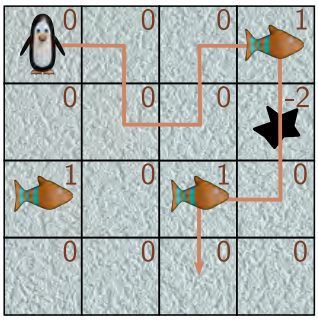


Figure 19.18 One trajectory through an MDP. The penguin receives a reward of +1 when it reaches the first fish tile, −2 when it falls in the hole, and +1 for reaching the second fish tile. The discount factor γ is 0.9.

Problem 19.6 Show that:

$$\mathbb{E}_{\boldsymbol{\tau}}\left[\frac{\partial}{\partial \boldsymbol{\theta}}\log\big[Pr(\boldsymbol{\tau}|\boldsymbol{\theta})\big]b\right] = 0, \tag{19.48}$$

and so adding a baseline update doesn't change the expected policy gradient update.

Problem 19.7* Suppose that we want to estimate a quantity $\mathbb{E}[a]$ from samples $a_1, a_2 \ldots a_I$. Consider that we also have paired samples $b_1, b_2 \ldots b_I$ that are samples that co-vary with a where $\mathbb{E}[b] = \mu_b$. We define a new variable:

$$a' = a - c(b - \mu_b). \tag{19.49}$$

Show that $\text{Var}[a'] \leq \text{Var}[a]$ when the constant c is chosen judiciously. Find an expression for the optimal value of c.

Problem 19.8 The estimate of the gradient in equation 19.34 can be written as:

$$\mathbb{E}_{\boldsymbol{\tau}}\Big[g[\boldsymbol{\theta}](r[\boldsymbol{\tau}_t] - b)\Big], \tag{19.50}$$

where

$$g[\theta] = \sum_{t=1}^{T}\frac{\partial \log\big[Pr(a_t|\mathbf{s}_t, \boldsymbol{\theta})\big]}{\partial \boldsymbol{\theta}}, \tag{19.51}$$

and

$$r[\boldsymbol{\tau}_t] = \sum_{k=t}^{T} r_k. \tag{19.52}$$

Show that the value of b that minimizes the variance of the gradient estimate is given by:

$$b = \frac{\mathbb{E}[g[\boldsymbol{\tau}]^2]r[\boldsymbol{\tau}]}{\mathbb{E}[g[\boldsymbol{\tau}]^2]}. \tag{19.53}$$

Chapter 20

Why does deep learning work?

This chapter differs from those that precede it. Instead of presenting established results, it poses questions about how and why deep learning works so well. These questions are rarely discussed in textbooks. However, it's important to realize that (despite the title of this book) understanding of deep learning is still limited.

We argue that it is surprising that deep networks are easy to train and also surprising that they generalize. Then we consider each of these topics in turn. We enumerate the factors that influence training success and discuss what is known about loss functions for deep networks. Then we consider the factors that influence generalization. We conclude with a discussion of whether networks need to be overparameterized and deep.

20.1 The case against deep learning

The MNIST-1D dataset (figure 8.1) has just forty input dimensions and ten output dimensions. With enough hidden units per layer, a two-layer fully connected network classifies 10000 MNIST-1D training data points perfectly and generalizes reasonably to unseen examples (figure 8.10a). Indeed, we now take it for granted that with sufficient hidden units, deep networks will classify almost any training set near-perfectly. We also take for granted that the fitted model will generalize to new data. However, it's not *at all* obvious either that the training process should succeed or that the resulting model should generalize. This section argues that both these phenomena are surprising.

20.1.1 Training

Performance of a two-layer fully connected network on 10000 MNIST-1D training examples is perfect once there are 43 hidden units per layer ($\sim$4000 parameters). However, finding the global minimum of an arbitrary non-convex function is NP-hard (Murty & Kabadi, 1987), and this is also true for certain neural network loss functions (Blum & Rivest, 1992). It's remarkable that the fitting algorithm doesn't get trapped in local minima or stuck near saddle points and that it can efficiently recruit spare model capacity

to fit unexplained training data wherever they lie.

Perhaps this success is less surprising when there are far more parameters than training data. However, it's debatable whether this is generally the case. AlexNet had ∼60 million parameters and was trained with ∼1 million data points. However, to complicate matters, each training example was augmented with 2048 transformations. GPT-3 had 175 billion parameters and was trained with 300 billion tokens. There is not a clear-cut case that either model was overparameterized, and yet they were successfully trained.

In short, it's surprising that we can fit deep networks reliably and efficiently. Either the data, the models, the training algorithms, or some combination of all three must have some special properties that make this possible.

20.1.2 Generalization

If the efficient fitting of neural networks is startling, their generalization to new data is *dumbfounding*. First, it's not obvious *a priori* that typical datasets are sufficient to characterize the input/output mapping. The curse of dimensionality implies that the training dataset is tiny compared to the *possible* inputs; if each of the 40 inputs of the MNIST-1D data were quantized into 10 possible values, there would be 10^{40} possible inputs, which is a factor of 10^{35} more than the number of training examples.

Problem 20.1

Second, deep networks describe *very* complicated functions. A fully connected network for MNIST-1D with two hidden layers of width 400 can create mappings with up to 10^{42} linear regions. That's roughly 10^{37} regions per training example, so very few of these regions contain data at any stage during training; regardless, those regions that *do* encounter data points constrain the remaining regions to behave reasonably.

Third, generalization gets *better* with more parameters (figure 8.10). The model in the previous paragraph has 177,201 parameters. Assuming it can fit one training example per parameter, it has 167,201 spare degrees of freedom. This surfeit gives the model latitude to do *almost anything* between the training data, and yet it behaves sensibly.

20.1.3 The unreasonable effectiveness of deep learning

To summarize, it's neither obvious that we should be able to fit deep networks nor that they should generalize. *A priori*, deep learning shouldn't work. And yet it does. This chapter investigates why. Sections 20.2–20.3 describe what we know about fitting deep networks and their loss functions. Sections 20.4–20.6 examine generalization.

20.2 Factors that influence fitting performance

Figure 6.4 showed that loss functions for nonlinear models can have both local minima and saddle points. However, we can reliably fit deep networks to complex training sets. For example, figure 8.10 shows perfect training performance on MNIST-1D, MNIST, and CIFAR-100. This section considers factors that might resolve this contradiction.

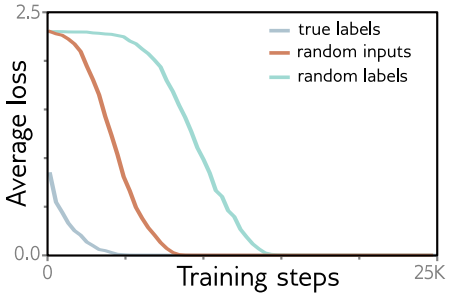


Figure 20.1 Fitting random data. Losses for AlexNet architecture trained on CIFAR-10 dataset with SGD. When the pixels are drawn from a Gaussian random distribution with the same mean and variance as the original data, the model can still be fit (albeit more slowly). When the labels are randomized, the model can still be fit (albeit even more slowly). Adapted from Zhang et al. (2017a).

20.2.1 Dataset

It's important to realize that we can't learn *any* function. Consider a completely random mapping from every possible 28×28 binary image to one of ten categories. Since there is no structure to this function, the only recourse is to memorize the 2^{784} assignments. However, it's easy to train a model on the MNIST dataset (figures 8.10 and 15.15), which contains 60,000 examples of 28×28 images labeled with one of ten categories. One explanation for this contradiction could be that it is easy to find global minima because the real-world functions that we approximate are relatively simple.[1]

This hypothesis was investigated by Zhang et al. (2017a), who trained AlexNet on the CIFAR-10 image classification dataset when (i) each image was replaced with Gaussian noise and (ii) the labels of the ten classes were randomly permuted (figure 20.1). These changes slowed down learning, but the network could still fit this finite dataset well. This suggests that the properties of the dataset aren't critical.

Notebook 20.1 Random data

Problem 20.2

20.2.2 Regularization

Another possible explanation for the ease with which models are trained is that some regularization methods like L2 regularization (weight decay) make the loss surface flatter and more convex. However, Zhang et al. (2017a) found that neither L2 regularization nor Dropout was required to fit random data. This does not eliminate implicit regularization due to the finite step size of the fitting algorithms (section 9.2). However, this effect increases with the learning rate (equation 9.9), and model-fitting does not get easier with larger learning rates.

20.2.3 Stochastic training algorithms

Chapter 6 argued that the SGD algorithm potentially allows the optimization trajectory to move between "valleys" during training. However, Keskar et al. (2017) show that several models (including fully connected and convolutional networks) can be fit to many

[1] In this chapter, we use the term "global minimum" loosely to mean any solution where all data are classified correctly. We have no way of knowing if there are solutions with a lower loss elsewhere.

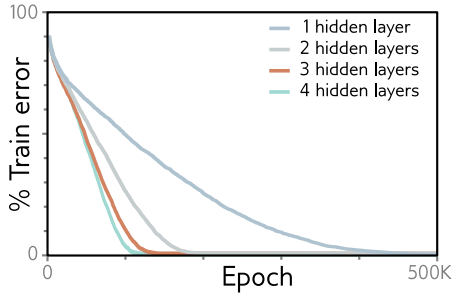


Figure 20.2 MNIST-1D training. Four fully connected networks were fit to 4000 MNIST-1D examples with random labels using full batch gradient descent, He initialization, no momentum or regularization, and learning rate 0.0025. Models with 1,2,3,4 layers had 298, 100, 75, and 63 hidden units per layer and 15208, 15210, 15235, and 15139 parameters, respectively. All models train successfully, but deeper models require fewer epochs.

datasets (including CIFAR-100 and MNIST) almost perfectly with very large batches of 5000-6000 images. This eliminates most of the randomness but training still succeeds.

Notebook 20.2 Full batch gradient descent

Problem 20.3

Figure 20.2 shows training results for four fully connected models fitted to 4000 MNIST-1D examples with randomized labels using full-batch (i.e., non-stochastic) gradient descent. There was no explicit regularization, and the learning rate was set to a small constant value of 0.0025 to minimize implicit regularization. Here, the true mapping from data to labels has no structure, the training is deterministic, and there is no regularization, and yet the training error *still* decreases to zero. This suggests that these loss functions may genuinely have no local minima.

20.2.4 Overparameterization

Overparameterization almost certainly *is* an important factor that contributes to ease of training. It implies that there is a large family of degenerate solutions, so there may always be a direction in which the parameters can be modified to decrease the loss. Sejnowski (2020) suggests that "... the degeneracy of solutions changes the nature of the problem from finding a needle in a haystack to a haystack of needles."

In practice, networks are frequently overparameterized by one or two orders of magnitude (figure 20.3). However, data augmentation makes it difficult to make precise statements. Augmentation may increase the data by several orders of magnitude, but these are manipulations of existing examples rather than independent new data points. Moreover, figure 8.10 shows that neural networks can sometimes fit the training data well when there are the same number or fewer parameters than data points. This is presumably due to redundancy in training examples from the same underlying function.

Several theoretical convergence results show that, *under certain circumstances*, SGD converges to a global minimum when the network is sufficiently overparameterized. For example, Du et al. (2019b) show that randomly initialized SGD converges to a global minimum for shallow fully connected ReLU networks with a least squares loss with enough hidden units. Similarly, Du et al. (2019a) consider deep, residual, and convolutional networks when the activation function is smooth and Lipschitz. Zou et al. (2020) analyzed the convergence of gradient descent on deep, fully connected networks using a hinge loss. Allen-Zhu et al. (2019) considered deep networks with ReLU functions.

If a neural network is sufficiently overparameterized so that it can memorize any

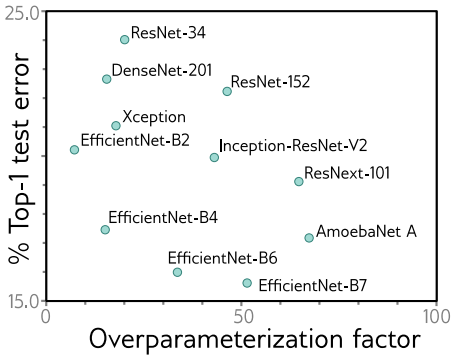


Figure 20.3 Overparameterization. ImageNet performance for convolutional nets as a function of overparameterization (in multiples of dataset size). Most models have 10–100 times more parameters than there were training examples. Models compared are ResNet (He et al., 2016a,b), DenseNet (Huang et al., 2017b), Xception (Chollet, 2017), EfficientNet (Tan & Le, 2019), Inception (Szegedy et al., 2017), ResNeXt (Xie et al., 2017), and AmoebaNet (Cubuk et al., 2019).

dataset of a fixed size, then all stationary points become global minima (Livni et al., 2014; Nguyen & Hein, 2017, 2018). Other results show that if the network is wide enough, local minima where the loss is higher than the global minimum are rare (see Choromanska et al., 2015; Pascanu et al., 2014; Pennington & Bahri, 2017). Kawaguchi et al. (2019) prove that as a network becomes deeper, wider, or both, the loss at local minima becomes closer to that at the global minimum for squared loss functions.

These theoretical results are intriguing but usually make unrealistic assumptions about the network structure. For example, Du et al. (2019a) show that residual networks converge to zero training loss when the width of the network D (i.e., the number of hidden units) is $\Omega[I^4K^2]$ where I is the amount of training data, and K is the depth of the network. Similarly, Nguyen & Hein (2017) assume that the network's width is larger than the dataset size, which is unrealistic in most practical scenarios. Overparameterization seems to be important, but theory cannot yet explain empirical fitting performance.

20.2.5 Activation functions

The activation function is also known to affect training difficulty. Networks where the activation only changes over a small part of the input range are harder to fit than ReLUs (which vary over half the input range) or Leaky ReLUs (which vary over the full range); For example, sigmoid and tanh nonlinearities (figure 3.13a) have shallow gradients in their tails; where the activation function is near-constant, the training gradient is near-zero, so there is no mechanism to improve the model.

20.2.6 Initialization

Another potential explanation is that Xavier/He initialization sets the parameters to values that are easy to optimize. Of course, for deeper networks, such initialization is necessary to avoid exploding and vanishing gradients, so in a trivial sense, initialization is critical to training success. However, for shallower networks, the initial variance of the weights is less important. Liu et al. (2023c) trained a 3-layer fully connected network with

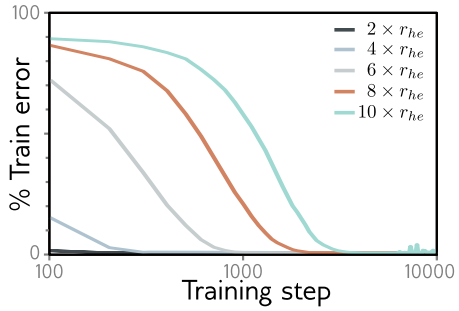


Figure 20.4 Initialization and fitting. A three-layer fully connected network with 200 hidden units per layer was trained on 1000 MNIST examples with AdamW using one-hot targets and mean-squared error loss. It takes longer to fit networks when larger multiples of He initialization are used, but this doesn't change the outcome. This may simply reflect the extra distance that the weights must move. Adapted from Liu et al. (2023c).

200 hidden units per layer on 1000 MNIST data points. They found that more iterations were required to fit the training data as the variance increased from that proposed by He (figure 20.4), but this did not ultimately impede fitting. Hence, initialization doesn't shed much light on why fitting neural networks is easy, although exploding/vanishing gradients do reveal initializations that make training difficult with finite precision arithmetic.

20.2.7 Network depth

Neural networks are harder to fit when the depth becomes very large due to exploding and vanishing gradients (figure 7.7) and shattered gradients (figure 11.3). However, these are (arguably) practical numerical issues. There is no definitive evidence that the underlying loss function is fundamentally more or less convex as the network depth increases. Figure 20.2 does show that for MNIST data with randomized labels and He initialization, deeper networks train in fewer iterations. However, this might be because either (i) the gradients in deeper networks are steeper or (ii) He initialization just starts wider, shallower networks further away from the optimal parameters.

Frankle & Carbin (2019) show that for small networks like VGG, you can get the same or better performance if you (i) train the network, (ii) prune the weights with the smallest magnitudes and (iii) retrain from the same initial weights. This does not work if the weights are randomly re-initialized. They concluded that the original overparameterized network contains small trainable sub-networks, which are sufficient to provide the performance. They term this the *lottery ticket hypothesis* and denote the sub-networks as *winning tickets*. This suggests that the effective number of sub-networks may have a key role to play in fitting. This (perhaps) varies with the network depth for a fixed parameter count, but a precise characterization of this idea is lacking.

Notebook 20.3
Lottery tickets

20.3 Properties of loss functions

The previous section discussed factors that contribute to the ease with which neural networks can be trained. The number of parameters (degree of overparameterization) and the choice of activation function are both important. Surprisingly, the choice of dataset,

the randomness of the fitting algorithm, and the use of regularization don't seem important. There is no definitive evidence that (for a fixed parameter count) the depth of the network matters (other than numerical problems due to exploding/vanishing/shattered gradients). This section tackles the same topic from a different angle by considering the empirical properties of loss functions. Most of this evidence comes from fully connected networks and CNNs; loss functions of transformer networks are less well understood.

20.3.1 Multiple global minima

We *expect* loss functions for deep networks to have a large family of equivalent global minima. In fully connected networks, the hidden units at each layer and their associated weights can be permuted without changing the output. In convolutional networks, permuting the channels and convolution kernels appropriately doesn't change the output. We can multiply the weight before any ReLU function and divide the weight after by a positive number without changing the output. Using BatchNorm induces another set of redundancies because the mean and variance of each hidden unit or channel are reset.

The above modifications all produce the same output for *every* input. However, the global minimum only depends on the output at the training data points. In overparameterized networks, there will also be families of solutions that behave identically at the data points but differently between them. All of these are also global minima.

20.3.2 Route to the minimum

Goodfellow et al. (2015b) considered a straight line between the initial parameters and the final values. They show that the loss function along this line usually decreases monotonically (except for a small bump near the start sometimes). This phenomenon is observed for several different types of networks and activation functions (figure 20.5a).

Of course, real optimization trajectories do not proceed in a straight line. However, Li et al. (2018b) find that they do lie in low-dimensional subspaces. They attribute this to the existence of large, nearly convex regions in the loss landscape that capture the trajectory early on and funnel it in a few important directions. Surprisingly, Li et al. (2018a) showed that networks still train well if optimization is *constrained* to lie in a random low-dimensional subspace (figure 20.6).

Li & Liang (2018) show that the relative change in the parameters during training decreases as network width increases; for larger widths, the parameters start at smaller values, change by a smaller proportion of those values, and converge in fewer steps.

20.3.3 Connections between minima

Goodfellow et al. (2015b) examined the loss function along a straight line between two minima that were found independently. They saw a pronounced increase in the loss between them (figure 20.5b); good minima are not generally linearly connected. However,

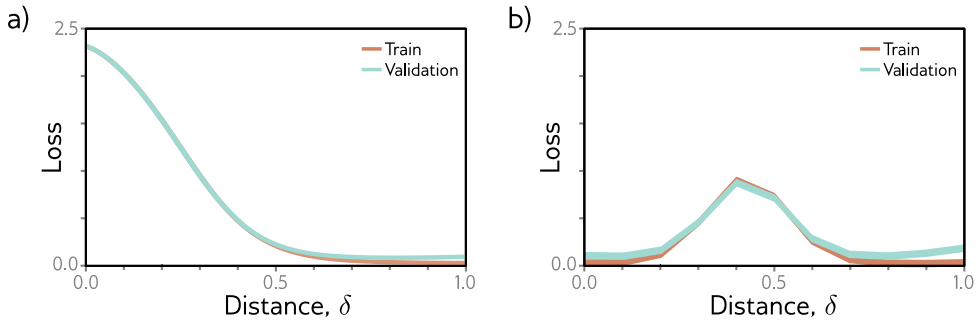
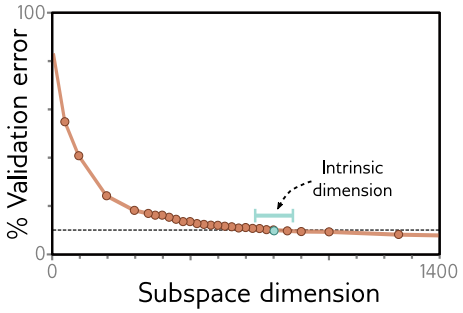


Figure 20.5 Linear slices through loss function. a) A two-layer fully connected ReLU network is trained on MNIST. The loss along a straight line starting at the initial parameters (δ=0) and finishing at the trained parameters (δ=1) descends monotonically. b) However, in this two-layer fully connected MaxOut network on MNIST, there is an increase in the loss along a straight line between one solution (δ=0) and another (δ=1). Adapted from Goodfellow et al. (2015b).

Figure 20.6 Subspace training. A fully connected network with two hidden layers, each with 200 units was trained on MNIST. Parameters were initialized using a standard method but then constrained to lie within a random subspace. Performance reaches 90% of the unconstrained level when this subspace is 750D (termed the *intrinsic dimension*), which is 0.4% of the original parameters. Adapted from Li et al. (2018a).



Frankle et al. (2020) showed that this increase vanishes if the networks are identically trained initially and later allowed to diverge by using different SGD noise and augmentation. This suggests that the solution is constrained early in training and that *some* families of minima are linearly connected.

Draxler et al. (2018) found minima with good (but different) performance on the CIFAR-10 dataset. They then showed that it is possible to construct paths from one to the other, where the loss function remains low along this path. They conclude that there is a single connected manifold of low loss (figure 20.7). This seems to be increasingly true as the width and depth of the network increase. Garipov et al. (2018) and Fort & Jastrzębski (2019) present other schemes for connecting minima.

20.3.4 Curvature of loss surface

Random Gaussian functions (in which points are jointly distributed with covariance given by a kernel function of their distance) have an interesting property: for points

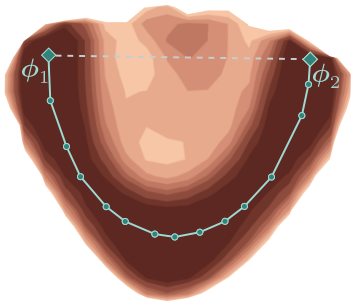


Figure 20.7 Connections between minima. A slice through the loss function of DenseNet on CIFAR-10. Parameters ϕ_1 and ϕ_2 are two independently discovered minima. Linear interpolation between these parameters reveals an energy barrier (dashed line). However, for sufficiently deep and wide networks, it is possible to find a curved path of low energy between two minima (cyan line). Adapted from Draxler et al. (2018).

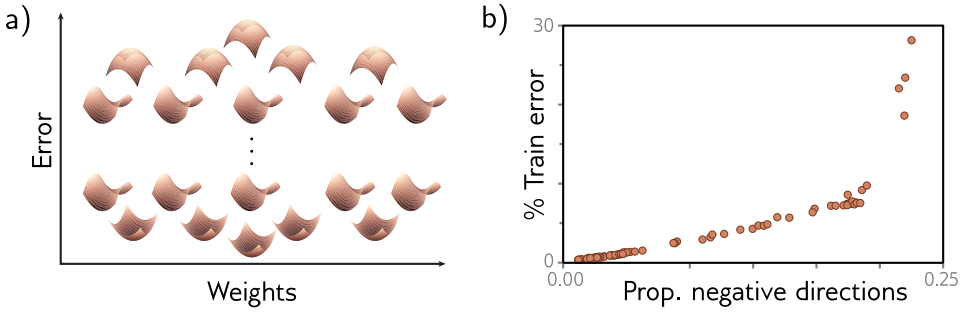


Figure 20.8 Critical points vs. loss. a) In random Gaussian functions, the number of directions in which the function curves down at points with zero gradient decreases with the height of the function, so minima all appear at lower function values. b) Dauphin et al. (2014) found critical points on a neural network loss surface (i.e., points with zero gradient). They showed that the proportion of negative eigenvalues (directions that point down) decreases with the loss. The implication is that all minima (points with zero gradient where no directions point down) have low losses. Adapted from Dauphin et al. (2014) and Bahri et al. (2020).

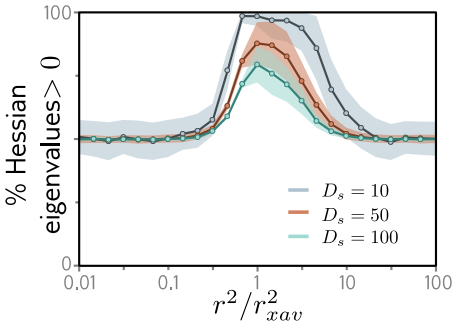


Figure 20.9 Goldilocks zone. The proportion of eigenvalues of the Hessian that are greater than zero (a measure of positive curvature/convexity) within a random subspace of dimension D_s in a two-layer fully connected network with ReLU functions applied to MNIST as a function of the squared radius r^2 of the parameters relative to Xavier initialization. There is a pronounced region of positive curvature known as the *Goldilocks zone*. Adapted from Fort & Scherlis (2019).

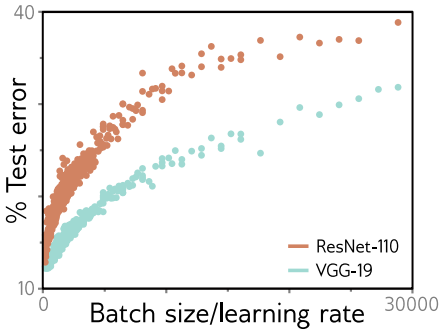


Figure 20.10 Batch size to learning rate ratio. Generalization of two models on the CIFAR-10 database depends on the ratio of batch size to the learning rate. As the batch size increases, generalization decreases. As the learning rate increases, generalization increases. Adapted from He et al. (2019).

where the gradient is zero, the fraction of directions where the function curves down becomes smaller when these points occur at lower loss values (see Bahri et al., 2020). Dauphin et al. (2014) searched for saddle points in a neural network loss function and similarly found a correlation between the loss and the number of negative eigenvalues (figure 20.8). Baldi & Hornik (1989) analyzed the error surface of a shallow network and found that there were *no local minima* but only saddle points. These results suggest that there are few or no bad local minima.

Fort & Scherlis (2019) measured the curvature at random points on a neural network loss surface; they showed that the curvature of the surface is unusually positive when the ℓ_2 norm of the weights lies within a certain range (figure 20.9), which they term the *Goldilocks zone*. He and Xavier initialization fall within this range.

20.4 Factors that determine generalization

The last two sections considered factors that determine whether the network trains successfully and what is known about neural network loss functions. This section considers factors that determine how well the network generalizes. This complements the discussion of regularization (chapter 9), which explicitly aims to encourage generalization.

20.4.1 Training algorithms

Since deep networks are usually overparameterized, the details of the training process determine which of the degenerate family of minima the algorithm converges to. Some of these details reliably improve generalization.

LeCun et al. (2012) show that SGD generalizes better than full-batch gradient descent. It has been argued that SGD generalizes better than Adam (e.g., Wilson et al., 2017; Keskar & Socher, 2017), but more recent studies suggest that there is little difference when the hyperparameter search is done carefully (Choi et al., 2019). Keskar et al. (2017) show that deep nets generalize better with smaller batch-size when no other form of regularization is used. It is also well-known that larger learning rates tend to generalize better (e.g., figure 9.5). Jastrzębski et al. (2018), Goyal et al. (2018), and He

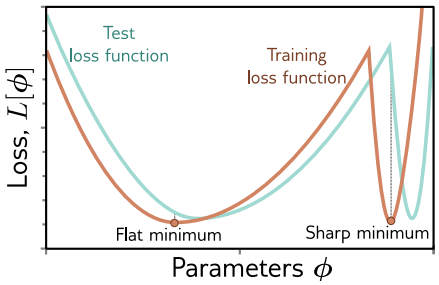


Figure 20.11 Flat vs. sharp minima. Flat minima are expected to generalize better. Small errors in estimating the parameters or in the alignment of the train and test loss functions are less problematic in flat regions. Adapted from Keskar et al. (2017).

et al. (2019) argue that the batch size/learning rate ratio is important. He et al. (2019) show a significant correlation between this ratio and the degree of generalization and prove a generalization bound for neural networks, which has a positive correlation with this ratio (figure 20.10).

These observations are aligned with the discovery that SGD implicitly adds regularization terms to the loss function (section 9.2), and their magnitude depends on the learning rate. The trajectory of the parameters is changed by this regularization, and they converge to a part of the loss function that generalizes well.

20.4.2 Flatness of minimum

There has been speculation dating at least to Hochreiter & Schmidhuber (1997a) that flat minima in the loss function generalize better than sharp minima (figure 20.11). Informally, if the minimum is flatter, then small errors in the estimated parameters are less important. This can also be motivated from various theoretical viewpoints. For example, minimum description length theory suggests models specified by fewer bits generalize better (Rissanen, 1983). For wide minima, the precision needed to store the weights is lower, so they should generalize better.

Flatness can be measured by (i) the size of the connected region around the minimum for which training loss is similar (Hochreiter & Schmidhuber, 1997a), (ii) the second-order curvature around the minimum (Chaudhari et al., 2019), or (iii) the maximum loss within a neighborhood of the minimum (Keskar et al., 2017). However, caution is required; estimated flatness can be affected by trivial reparameterizations of the network due to the non-negative homogeneity property of the ReLU function (Dinh et al., 2017).

Nonetheless, Keskar et al. (2017) varied the batch size and learning rate and showed that flatness correlates with generalization. Izmailov et al. (2018) average together weights from multiple points in a learning trajectory. This both results in flatter test and training surfaces at the minimum and improves generalization. Other regularization techniques can also be viewed through this lens. For example, averaging model outputs (ensembling) may also make the test loss surface flatter. Kleinberg et al. (2018) showed that large gradient variance during training helps avoid sharp regions. This may explain why reducing the batch size and adding noise helps generalization.

The above studies consider flatness for a single model and training set. However, sharpness is not a good criterion to predict generalization between datasets; when the

labels in the CIFAR dataset are randomized (making generalization impossible), there is no commensurate decrease in the flatness of the minimum (Neyshabur et al., 2017).

20.4.3 Architecture

The inductive bias of a network is determined by its architecture, and judicious choices of model can drastically improve generalization. Chapter 10 introduced convolutional networks, which are designed to process data on regular grids; they implicitly assume that the input statistics are the same across the input, so they share parameters across position. Similarly, transformers are suited for modeling data that is invariant to permutations, and graph neural networks are suited to data represented on irregular graphs. Matching the architecture to the properties of the data improves generalization over generic, fully connected architectures (see figure 10.8).

20.4.4 Norm of weights

Section 20.3.4 reviewed the finding of Fort & Scherlis (2019) that the curvature of the loss surface is unusually positive when the ℓ_2 norm of the weights lies within a certain range. The same authors provided evidence that generalization is also good when the ℓ_2 weight norm falls within this Goldilocks zone (figure 20.12). This is perhaps unsurprising. The norm of the weights is (indirectly) related to the Lipschitz constant of the model. If this norm is too small, then the model will not be able to change fast enough to capture the variation in the underlying function. If the norm is too large, then the model will be unnecessarily variable between training points and will not interpolate smoothly.

This finding was used by Liu et al. (2023c) to explain the phenomenon of *grokking* (Power et al., 2022), in which a sudden improvement in generalization can occur many epochs after the training error is already zero (figure 20.13). It is proposed that grokking occurs when the norm of the weights is initially too large; the training data fits well, but the variation of the model between the data points is large. Over time, implicit or explicit regularization decreases the norm of the weights until they reach the Goldilocks zone, and generalization suddenly improves.

20.4.5 Overparameterization

Figure 8.10 showed that generalization performance tends to improve with the degree of overparameterization. When combined with the bias/variance trade-off curve, this results in double descent. The putative explanation for this improvement is that the network has more latitude to become smoother *between* the training data points when the model is overparameterized.

It follows that the norm of the weights can also be used to explain double descent. The norm of the weights increases when the number of parameters is similar to the number of data points (as the model contorts itself to fit these points exactly), causing

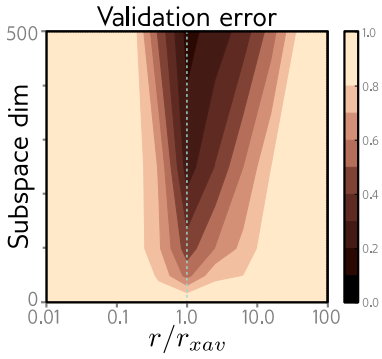


Figure 20.12 Generalization on hyperspheres. A fully connected network with two hidden layers, each with 200 units (198,450 parameters) was trained on the MNIST database. The parameters are initialized to a given ℓ_2 norm and then constrained to maintain this norm and to lie in a subspace (vertical direction). The network generalizes well in a small range around the radius r defined by Xavier initialization (cyan dotted line). Adapted from Fort & Scherlis (2019).

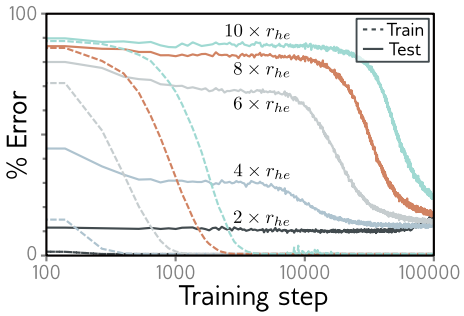


Figure 20.13 Grokking. When the parameters are initialized so that their ℓ_2 norm (radius) is considerably larger than is specified by He initialization, training takes longer (dashed lines), and generalization takes *much* longer (solid lines). The lag in generalization is attributed to the time taken for the norm of the weights to decrease back to the Goldilocks zone. Adapted from Liu et al. (2023c).

generalization to reduce. As the network becomes wider and the number of weights increases, the overall norm of these weights decreases; the weights are initialized with a variance that is inversely proportional to the width (i.e., with He or Glorot initialization), and the weights change very little from their original values.

20.4.6 Leaving the data manifold

Until this point, we have discussed how models generalize to new data that is drawn from the same distribution as the training data. This is a reasonable assumption for experimentation. However, systems deployed in the real world may encounter unexpected data due to noise, changes in the data statistics over time, or deliberate attacks. Of course, it is harder to make definite statements about this scenario, but D'Amour et al. (2020) show that the variability of identical models trained with different seeds on corrupted data can be enormous and unpredictable.

Goodfellow et al. (2015a) showed that deep learning models are susceptible to *adversarial attacks*. Consider perturbing an image that is correctly classified by the network as "dog" so that the probability of the correct class decreases as fast as possible until the class flips. If this image is now classified as an airplane, you might expect the perturbed image to look like a cross between a dog and an airplane. However, in practice, the perturbed image looks almost indistinguishable from the original dog image (figure 20.14).

Notebook 20.4
Adversarial attacks

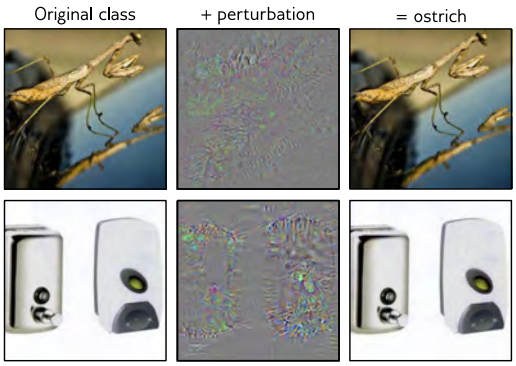


Figure 20.14 Adversarial examples. In each case, the left image is correctly classified by AlexNet. By considering the gradients of the network output with respect to the input, it's possible to find a small perturbation (center, magnified by 10 for visibility) that, when added to the original image (right), causes the network to misclassify it as an ostrich. This is despite the fact that the original and perturbed images are almost indistinguishable to humans. Adapted from Szegedy et al. (2014).

The conclusion is that there are positions that are close to but not on the data manifold that are misclassified. These are known as *adversarial examples*. Their existence is surprising; how can such a small change to the network input make such a drastic change to the output? The best current explanation is that adversarial examples aren't due to a lack of robustness to data from outside the training data manifold. Instead, they are exploiting a source of information that is in the training distribution but which has a small norm and is imperceptible to humans (Ilyas et al., 2019).

20.5 Do we need so many parameters?

Section 20.4 argued that models generalize better when over-parameterized. Indeed, there are almost no examples of state-of-the-art performance on complex datasets where the model has significantly fewer parameters than there were training data points.

However, section 20.2 reviewed evidence that training becomes easier as the number of parameters increases. Hence, it's not clear if some fundamental property of smaller models prevents them from performing as well or whether the training algorithms can't find good solutions for small models. *Pruning* and *distilling* are two methods for reducing the size of trained models. This section examines whether these methods can produce underparameterized models which retain the performance of overparameterized ones.

20.5.1 Pruning

Pruning trained models reduces their size and hence storage requirements (figure 20.15). The simplest approach is to remove individual weights. This can be done based on the second derivatives of the loss function (LeCun et al., 1990; Hassibi & Stork, 1993) or (more practically) based on the absolute value of the weight (Han et al., 2016, 2015). Other work prunes hidden units (Zhou et al., 2016a; Alvarez & Salzmann, 2016), channels in convolutional networks (Li et al., 2017a; Luo et al., 2017b; He et al., 2017; Liu et al., 2019a), or entire layers in residual nets (Huang & Wang, 2018). Often, the network is

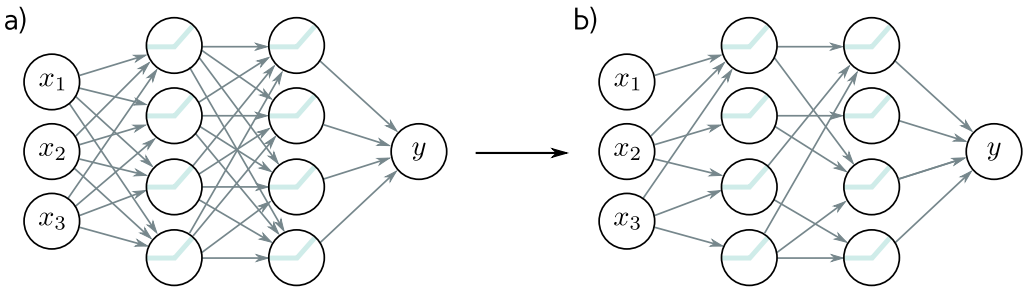


Figure 20.15 Pruning neural networks. The goal is to remove as many weights as possible without decreasing performance. This is often done just based on the magnitude of the weights. Typically, the network is fine-tuned after pruning. a) Example fully connected network. b) After pruning.

fine-tuned after pruning, and sometimes this process is repeated.

For example, Han et al. (2016) maintained good performance for the VGG network on ImageNet classification when 8% of the weights were retained. This significantly decreases the model size but isn't enough to show that overparameterization is not required; the VGG network has ∼100 times as many parameters as there are ImageNet training data (disregarding augmentation).

Pruning is a form of architecture search. In their work on lottery tickets (see section 20.2.7), Frankle & Carbin (2019) (i) trained a network, (ii) pruned the weights with the smallest magnitudes, and (iii) retrained the remaining network from the same initial weights. By iterating this procedure, they reduced the size of the VGG-19 network (originally 138 million parameters) by 98.5% on the CIFAR-10 database (60,000 examples) while maintaining good performance. For ResNet-50 (25.6 million parameters), they reduced the parameters by 80% without reducing the performance on ImageNet (1.28 million examples). These demonstrations are impressive but (disregarding data augmentation) these networks are still over-parameterized after pruning.

20.5.2 Knowledge distillation

The parameters can also be reduced by training a smaller network (the student) to replicate the performance of a larger one (the teacher). This is known as *knowledge distillation* and dates back to at least Buciluǎ et al. (2006). Hinton et al. (2015) showed that the pattern of information across the output classes is important and trained a smaller network to approximate the pre-softmax logits of the larger one (figure 20.16).

Zagoruyko & Komodakis (2017) further encouraged the spatial maps of the activations of the student network to be similar to the teacher network at various points. They use this *attention transfer* method to approximate the performance of a 34-layer residual network (∼63 million parameters) with an 18-layer residual network (∼11 million param-

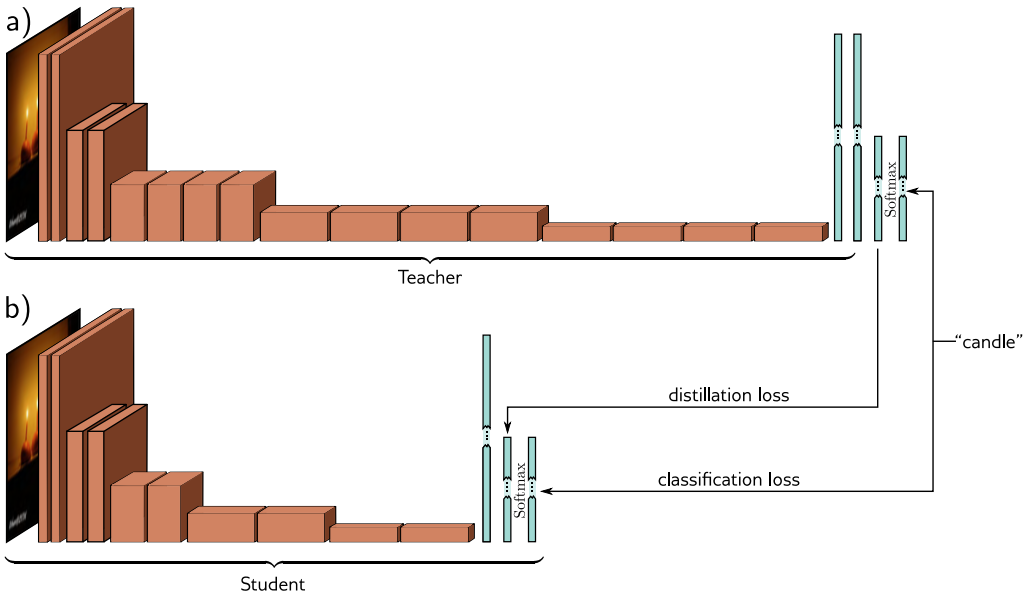


Figure 20.16 Knowledge distillation. a) A teacher network for image classification is trained as usual, using a multiclass cross-entropy classification loss. b) A smaller student network is trained with the same loss, plus also a distillation loss that encourages the pre-softmax activations to be the same as for the teacher.

eters) on the ImageNet classification task. However, this is still larger than the number of training examples (∼1 million images). Modern methods (e.g. Chen et al., 2021a) can improve on this result, but distillation has not yet provided convincing evidence that under-parameterized models can perform well.

20.5.3 Discussion

Current evidence suggests that overparameterization *is* needed for generalization — at least for the size and complexity of datasets that are currently used. There are no demonstrations of state-of-the-art performance on complex datasets where there are significantly fewer parameters than training examples. Attempts to reduce model size by pruning or distilling trained networks have not changed this picture.

Moreover, recent theory shows that there is a trade-off between the model's Lipschitz constant and overparameterization; Bubeck & Sellke (2021) proved that in D dimensions, *smooth* interpolation requires D times more parameters than mere interpolation. They argue that current models for large datasets (e.g., ImageNet) aren't overparameterized *enough*; increasing model capacity further may be key to improving performance.

20.6 Do networks have to be deep?

Chapter 3 discussed the universal approximation theorem. This states that shallow neural networks can approximate any function to arbitrary accuracy given enough hidden units. This raises the obvious question of whether networks *need* to be deep.

First, let's consider the evidence that depth *is* required. Historically, there has been a definite correlation between performance and depth. For example, performance on the ImageNet benchmark initially improved as a function of network depth until training became difficult. Subsequently, residual connections and batch normalization (chapter 11) allowed training of deeper networks with commensurate gains in performance. At the time of writing, almost all state-of-the-art applications, including image classification (e.g., the vision transformer), text generation (e.g., GPT3), and text-guided image synthesis (e.g., DALL·E-2), are based on deep networks with tens or hundreds of layers.

Despite this trend, there have been efforts to use shallower networks. Zagoruyko & Komodakis (2016) constructed shallower but wider residual neural networks and achieved similar performance to ResNet. More recently, Goyal et al. (2021) constructed a network that used parallel convolutional channels and achieved performance similar to deeper networks with only 12 layers. Furthermore, Veit et al. (2016) showed that it is predominantly shorter paths of 5–17 layers that drive performance in residual networks.

Nonetheless, the balance of evidence suggests that depth is critical; even the shallowest networks with good image classification performance require >10 layers. However, there is no definitive explanation for why. Three possible explanations are that (i) deep networks can represent more complex functions than shallow ones, (ii) deep networks are easier to train, and (iii) deep networks impose better inductive biases.

20.6.1 Complexity of modeled function

Chapter 4 showed that deep networks make functions with many more linear regions than shallow ones for the same parameter count. We also saw that "pathological" functions have been identified that require exponentially more hidden units to model with a shallow network than a deep one (e.g., Eldan & Shamir, 2016; Telgarsky, 2016). Indeed Liang & Srikant (2016) found quite general families of functions that are more efficiently modeled by deep networks. However, Nye & Saxe (2018) found that some of these functions cannot easily be fit by deep networks in practice. Moreover, there is little evidence that the real-world functions that we are approximating have these pathological properties.

20.6.2 Tractability of training

An alternative explanation is that shallow networks with a practical number of hidden units could support state-of-the-art performance, but it is just difficult to find a good solution that both fits the training data well and interpolates sensibly.

One way to show this is to distill successful deep networks into shallower (but wider) student models and see if performance can be maintained. Urban et al. (2017) dis-

tilled an ensemble of 16 convolutional networks for image classification on the CIFAR-10 dataset into student models of varying depths. They found that shallow networks could not replicate the performance of the deeper teacher and that the student performance increased as a function of depth for a constant parameter budget.

20.6.3 Inductive bias

Most current models rely on convolutional blocks or transformers. These networks share parameters for local regions of the input data, and often they gradually integrate this information across the whole input. These constraints mean that the functions that these networks can represent are not general. One explanation for the supremacy of deep networks, then, is that these constraints have a good inductive bias and that it is difficult to force shallow networks to obey these constraints.

Multi-layer convolutional architectures seem to be inherently helpful, even without training. Ulyanov et al. (2018) demonstrated that the structure of an untrained CNN can be used as a prior in low-level tasks such as denoising and super-resolution. Frankle et al. (2021) achieved good performance in image classification by initializing the kernels randomly, fixing their values, and just training the batch normalization offset and scaling factors. Zhang et al. (2017a) show that features from randomly initialized convolutional filters can support subsequent image classification using a kernel model.

Additional evidence that convolutional networks provide a useful inductive bias comes from Urban et al. (2017), who attempted to distill convolutional networks into shallower networks. They found that distilling into convolutional architectures systematically worked better than distilling into fully connected networks. This suggests that the convolutional architecture has some inherent advantages. Since the sequential local processing of convolutional networks cannot easily be replicated by shallower networks, this argues that depth is indeed important.

20.7 Summary

This chapter has made the case that the success of deep learning is surprising. We discussed the challenges of optimizing high-dimensional loss functions and argued that overparameterization and the choice of activation function are the two most important factors that make this tractable in deep networks. We saw that, during training, the parameters move through a low-dimensional subspace to one of a family of connected global minima and that local minima are not apparent.

Generalization of neural networks also improves with overparameterization, although other factors, such as the flatness of the minimum and the inductive bias of the architecture, are also important. It appears that both a large number of parameters and multiple network layers are required for good generalization, although we do not yet know why.

Many questions remain unanswered. We do not currently have any prescriptive theory that will allow us to predict the circumstances in which training and generalization will

succeed or fail. We do not know the limits of learning in deep networks or whether much more efficient models are possible. We do not know if there are parameters that would generalize better within the same model. The study of deep learning is still driven by empirical demonstrations. These are undeniably impressive, but they are not yet matched by our understanding of deep learning mechanisms.

Problems

Problem 20.1 Consider the ImageNet image classification task in which the input images contain $224\times224\times3$ RGB values. Consider coarsely quantizing these inputs into ten bins per RGB value and training with $\sim 10^7$ training examples. How many possible inputs are there per training data point?

Problem 20.2 Consider figure 20.1. Why do you think that the algorithm fits the data faster when the pixels are randomized relative to when the labels are randomized?

Problem 20.3 Figure 20.2 shows a non-stochastic fitting process with a fixed learning rate successfully fitting random data. Does this imply that the loss function has no local minima? Does this imply that the function is convex? Justify your answer and give a counter-example if you think either statement is false.

Chapter 21

Deep learning and ethics

This chapter was written by Travis LaCroix and Simon J.D. Prince.

AI is poised to change society for better or worse. These technologies have enormous potential for social good (Taddeo & Floridi, 2018; Tomašev et al., 2020), including important roles in healthcare (Rajpurkar et al., 2022) and the fight against climate change (Rolnick et al., 2023). However, they also have the potential for misuse and unintended harm. This has led to the emergence of the field of *AI ethics*.

The modern era of deep learning started in 2012 with AlexNet, but sustained interest in AI ethics did not follow immediately. Indeed, a workshop on fairness in machine learning was rejected from NeurIPS 2013 for want of material. It wasn't until 2016 that AI Ethics had its "AlexNet" moment, with ProPublica's exposé on bias in the COMPAS recidivism-prediction model (Angwin et al., 2016) and Cathy O'Neil's book *Weapons of Math Destruction* (O'Neil, 2016). Interest has swelled ever since; submissions to the Conference on *Fairness, Accountability, and Transparency* (FAccT) have increased nearly ten-fold in the five years since its inception in 2018.

In parallel, many organizations have proposed policy recommendations for responsible AI. Jobin et al. (2019) found 84 documents containing AI ethics principles, with 88% released since 2016. This proliferation of non-legislative policy agreements, which depend on voluntary, non-binding cooperation, calls into question their efficacy (McNamara et al., 2018; Hagendorff, 2020; LaCroix & Mohseni, 2022). In short, AI Ethics is in its infancy, and ethical considerations are often reactive rather than proactive.

This chapter considers potential harms arising from the design and use of AI systems. These include algorithmic bias, lack of explainability, data privacy violations, militarization, fraud, and environmental concerns. The aim is not to provide advice on being more ethical. Instead, the goal is to express ideas and start conversations in key areas that have received attention in philosophy, political science, and the broader social sciences.

21.1 Value alignment

Problem 21.1

When we design AI systems, we wish to ensure that their "values" (objectives) are aligned with those of humanity. This is sometimes called the *value alignment problem* (Russell,

2019; Christian, 2020; Gabriel, 2020). This is challenging for three reasons. First, it's difficult to define our values completely and correctly. Second, it is hard to encode these values as objectives of an AI model, and third, it is hard to ensure that the model learns to carry out these objectives.

In a machine learning model, the loss function is a proxy for our true objectives, and a misalignment between the two is termed the *outer alignment problem* (Hubinger et al., 2019). To the extent that this proxy is inadequate, there will be "loopholes" that the system can exploit to minimize its loss function while failing to satisfy the intended objective. For example, consider training an RL agent to play chess. If the agent is rewarded for capturing pieces, this may result in many drawn games rather than the desired behavior (to win the game). In contrast, the *inner alignment problem* is to ensure that the behavior of an AI system does not diverge from the intended objectives even when the loss function is well specified. If the learning algorithm fails to find the global minimum or the training data are unrepresentative, training can converge to a solution that is misaligned with the true objective resulting in undesirable behavior (Goldberg, 1987; Mitchell et al., 1992; Lehman & Stanley, 2008).

Problem 21.2

Gabriel (2020) divides the value alignment problem into *technical* and *normative* components. The technical component concerns how we encode values into the models so that they reliably do what they should. Some concrete problems, such as avoiding reward hacking and safe exploration, may have purely technical solutions (Amodei et al., 2016). In contrast, the normative component concerns what the correct values are in the first place. There may be no single answer to this question, given the range of things that different cultures and societies value. It's important that the encoded values are representative of everyone and not just culturally dominant subsets of society.

Another way to think about value alignment is as a *structural* problem that arises when a human *principal* delegates tasks to an artificial *agent* (LaCroix, 2022). This is similar to the principal-agent problem in economics (Laffont & Martimort, 2002), which allows that there are competing incentives inherent in any relationship where one party is expected to act in another's best interests. In the AI context, such conflicts of interest can arise when either (i) the objectives are misspecified or (ii) there is an informational asymmetry between the principal and the agent (figure 21.1).

Many topics in AI ethics can be understood in terms of this structural view of value alignment. The following sections discuss problems of bias and fairness and artificial moral agency (both pertaining to specifying objectives) and transparency and explainability (both related to informational asymmetry).

21.1.1 Bias and fairness

From a purely scientific perspective, bias refers to statistical deviation from some norm. In AI, it can be pernicious when this deviation depends on *illegitimate* factors that impact an output. For example, gender is irrelevant to job performance, so it is illegitimate to use gender as a basis for hiring a candidate. Similarly, race is irrelevant to criminality, so it is illegitimate to use race as a feature for recidivism prediction.

Bias in AI models can be introduced in various ways (Fazelpour & Danks, 2021):

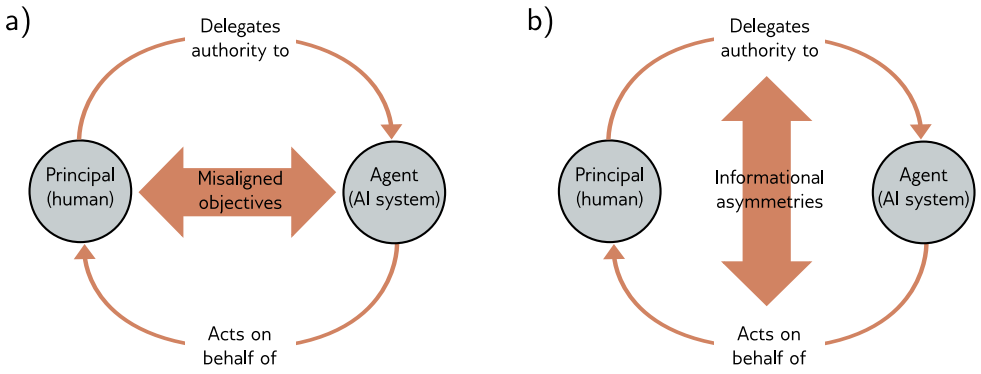


Figure 21.1 Structural description of the value alignment problem. a) Problems arise from a) misaligned objectives (e.g., bias) or b) informational asymmetries between a (human) principal and an (artificial) agent (e.g., lack of explainability). Adapted from LaCroix (2023).

- **Problem specification:** Choosing a model's goals requires a value judgment about what is important to us, which allows for the creation of biases (Fazelpour & Danks, 2021). Further biases may emerge if we fail to operationalize these choices successfully and the problem *specification* fails to capture our intended goals (Mitchell et al., 2021).
- **Data:** Algorithmic bias can result when the dataset is unrepresentative or incomplete (Danks & London, 2017). For example, the PULSE face super-resolution algorithm (Menon et al., 2020) was trained on a database of photos of predominantly white celebrities. When applied to a low-resolution portrait of Barack Obama, it generated a photo of a white man (Vincent, 2020).

  If the society in which training data are generated is structurally biased against marginalized communities, even complete and representative datasets will elicit biases (Mayson, 2018). For example, Black individuals in the US have been policed and jailed more frequently than white individuals. Hence, historical data used to train recidivism prediction models are already biased against Black communities.
- **Modeling and validation:** Choosing a mathematical definition to measure model fairness requires a value judgment. There exist distinct but equally-intuitive definitions that are logically inconsistent (Kleinberg et al., 2017; Chouldechova, 2017; Berk et al., 2017). This suggests the need to move from a purely mathematical conceptualization of fairness toward a more substantive evaluation of whether algorithms promote justice in practice (Green, 2022).
- **Deployment:** Deployed algorithms may interact with other algorithms, structures, or institutions in society to create complex feedback loops that entrench extant biases (O'Neil, 2016). For example, large language models like GPT3 (Brown et al., 2020) are trained on web data. However, when GPT3 outputs are published

Problem 21.3

Data collection	Pre-processing	Training	Post-processing
• Identify lack of examples or variates and collect	• Modify labels • Modify input data • Modify input/ output pairs	• Adversarial training • Regularize for fairness • Constrain to be fair	• Change thresholds • Trade-off accuracy for fairness

Figure 21.2 Bias mitigation. Methods have been proposed to compensate for bias at all stages of the training pipeline, from data collection to post-processing of already trained models. See Barocas et al. (2023) and Mehrabi et al. (2022).

online, the training data for future models is degraded. This may exacerbate biases and generate novel societal harm (Falbo & LaCroix, 2022).

Unfairness can be exacerbated by considerations of *intersectionality*; social categories can combine to create overlapping and interdependent systems of oppression. For example, the discrimination experienced by a queer woman of color is not merely the sum of the discrimination she might experience as queer, as gendered, or as racialized (Crenshaw, 1991). Within AI, Buolamwini & Gebru (2018) showed that face analysis algorithms trained primarily on lighter-skinned faces underperform for darker-skinned faces. However, they perform even worse on combinations of features such as skin color and gender than might be expected by considering those features independently.

Of course, steps can be taken to ensure that data are diverse, representative, and complete. But if the society in which the training data are generated is structurally biased against marginalized communities, even completely accurate datasets will elicit biases. In light of the potential for algorithmic bias and the lack of representation in training datasets described above, it is also necessary to consider how failure rates for the outputs of these systems are likely to exacerbate discrimination against already-marginalized communities (Buolamwini & Gebru, 2018; Raji & Buolamwini, 2019; Raji et al., 2022). The resulting models may codify and entrench systems of power and oppression, including capitalism and classism; sexism, misogyny, and patriarchy; colonialism and imperialism; racism and white supremacy; ableism; and cis- and heteronormativity. A perspective on bias that maintains sensitivity to power dynamics requires accounting for historical inequities and labor conditions encoded in data (Micelli et al., 2022).

To prevent this, we must actively ensure that our algorithms are fair. A naïve approach is *fairness through unawareness* which simply removes the *protected attributes* (e.g., race, gender) from the input features. Unfortunately, this is ineffective; the remaining features can still carry information about the protected attributes. More practical approaches first define a mathematical criterion for fairness. For example, the *separation* measure in binary classification requires that the prediction $\hat{y}$ is conditionally independent of the protected variable a (e.g., race) given the true label y. Then they intervene in various ways to minimize the deviation from this measure (figure 21.2).

Notebook 21.1
Bias mitigation

A further complicating factor is that we cannot tell if an algorithm is unfair to a community or take steps to avoid this unless we can establish community membership. Most research on algorithmic bias and fairness has focused on ostensibly *observable* features

that might be present in training data (e.g., gender). However, features of marginalized communities may be *unobservable*, making bias mitigation even more difficult. Examples include queerness (Tomasev et al., 2021), disability status, neurotype, class, and religion. A similar problem occurs when observable features have been excised from the training data to prevent models from exploiting them.

21.1.2 Artificial moral agency

Many decision spaces do not include actions that carry moral weight. For example, choosing the next chess move has no obvious moral consequence. However, elsewhere actions can carry moral weight. Examples include decision-making in autonomous vehicles (Awad et al., 2018; Evans et al., 2020), lethal autonomous weapons systems (Arkin, 2008a,b), and professional service robots for childcare, elderly care, and health care (Anderson & Anderson, 2008; Sharkey & Sharkey, 2012). As these systems become more autonomous, they may need to make moral decisions independent of human input.

This leads to the notion of *artificial moral agency*. An artificial moral agent is an autonomous AI system capable of making moral judgments. Moral agency can be categorized in terms of increasing complexity (Moor, 2006):

1. **Ethical impact agents** are agents whose actions have ethical impacts. Hence, almost any technology deployed in society might count as an ethical impact agent.
2. **Implicit ethical agents** are ethical impact agents that include some in-built safety features.
3. **Explicit ethical agents** can contextually follow general moral principles or rules of ethical conduct.
4. **Full ethical agents** are agents with beliefs, desires, intentions, free will, and consciousness of their actions.

The field of machine ethics seeks approaches to creating artificial moral agents. These approaches can be categorized as *top-down*, *bottom-up*, or *hybrid* (Allen et al., 2005). Top-down (theory-driven) methods directly implement and hierarchically arrange concrete rules based on some moral theory to guide ethical behavior. Asimov's "Three Laws of Robotics" are a trivial example of this approach.

In bottom-up (learning-driven) approaches, a model learns moral regularities from data without explicit programming (Wallach et al., 2008). For example, Noothigattu et al. (2018) designed a voting-based system for ethical decision-making that uses data collected from human preferences in moral dilemmas to learn social preferences; the system then summarizes and aggregates the results to render an "ethical" decision. Hybrid approaches combine top-down and bottom-up approaches.

Some researchers have questioned the very idea of artificial moral agency and argued that moral agency is unnecessary for ensuring safety (van Wynsberghe & Robbins, 2019). See Cervantes et al. (2019) for a recent survey of artificial moral agency and Tolmeijer et al. (2020) for a recent survey on technical approaches to artificial moral agency.

21.1.3 Transparency and opacity

A complex computational system is *transparent* if all of the details of its operation are known. A system is *explainable* if humans can understand how it makes decisions. In the absence of transparency or explainability, there is an asymmetry of information between the user and the AI system, which makes it hard to ensure value alignment.

Creel (2020) characterizes transparency at several levels of granularity. *Functional transparency* refers to knowledge of the algorithmic functioning of the system (i.e., the logical rules that map inputs to outputs). The methods in this book are described at this level of detail. *Structural transparency* entails knowing *how* a program executes the algorithm. This can be obscured when commands written in high-level programming languages are executed by machine code. Finally, *run transparency* requires understanding how a program was executed in a particular instance. For deep networks, this includes knowledge about the hardware, input data, training data, and interactions thereof. None of these can be ascertained by scrutinizing code.

Problem 21.4

For example, GPT3 is functionally transparent; its architecture is described in Brown et al. (2020). However, it does not exhibit structural transparency as we do not have access to the code, and it does not exhibit run transparency as we have no access to the learned parameters, hardware, or training data. The subsequent version GPT4 is not transparent at all. The details of how this commercial product works are unknown.

21.1.4 Explainability and interpretability

Even if a system is transparent, this does not imply that we can understand how a decision is made or what information this decision is based on. Deep networks may contain billions of parameters, so there is no way we can understand how they work based on examination alone. However, in some jurisdictions, the public may have a right to an explanation. Article 22 of the EU General Data Protection Regulation suggests all data subjects should have the right to "obtain an explanation of the decision reached" in cases where a decision is based solely on automated processes.[1]

These difficulties have led to the sub-field of explainable AI. One moderately successful area is producing local explanations. Although we can't explain the entire system, we can sometimes describe how a particular input was classified. For example, *Local interpretable model-agnostic explanations* or *LIME* (Ribeiro et al., 2016) samples the model output at nearby inputs and uses these samples to construct a simpler model (figure 21.3). This provides insight into the classification decision, even if the original model is neither transparent nor explainable.

Notebook 21.2 Explainability

It remains to be seen whether it is possible to build complex decision-making systems that are fully understandable to their users or even their creators. There is also an ongoing debate about what it means for a system to be explainable, understandable, or interpretable (Erasmus et al., 2021); there is currently no concrete definition of these concepts. See Molnar (2022) for more information.

[1] Whether Article 22 *actually* mandates such a right is debatable (see Wachter et al., 2017).

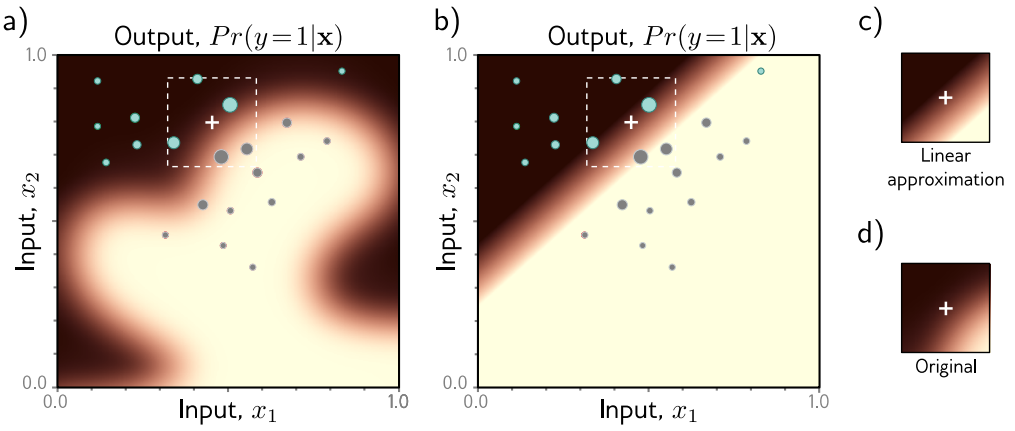


Figure 21.3 LIME. Output functions of deep networks are complex; in high dimensions, it's hard to know why a decision was made or how to modify the inputs to change it without access to the model. a) Consider trying to understand why $Pr(y = 1|\mathbf{x})$ is low at the white cross. LIME probes the network at nearby points to see if it identifies these as $Pr(y = 1|\mathbf{x}) < 0.5$ (cyan points) or $Pr(y = 1|\mathbf{x}) \geq 0.5$ (gray points). It weights these points by proximity to the point of interest (weight indicated by circle size). b) The weighted points are used to train a simpler model (here, logistic regression — a linear function passed through a sigmoid). c) Near the white cross, this approximation is close to d) the original function. Even though we did not have access to the original model, we can deduce from the parameters of this approximate model, that if we increase x_1 or decrease x_2, $Pr(y = 1|\mathbf{x})$ will increase, and the output class will change. Adapted from Prince (2022).

21.2 Intentional misuse

The problems in the previous section arise from poorly specified objectives and informational asymmetries. However, even when a system functions correctly, it can entail unethical behavior or be intentionally misused. This section highlights some specific ethical concerns arising from the misuse of AI systems.

Problem 21.5

21.2.1 Face recognition and analysis

Face recognition technologies have an especially high risk for misuse. Authoritarian states can use them to identify and silence protesters, thus risking democratic ideals of free speech and the right to protest. Smith & Miller (2022) argue that there is a mismatch between the values of liberal democracy (e.g., security, privacy, autonomy, and accountability) and the potential use cases for these technologies (e.g., border security, criminal investigation and policing, national security, and the commercialization

of personal data). Thus, some researchers, activists, and policymakers have questioned whether this technology should exist (Barrett, 2020).

Moreover, these technologies often do not do what they purport to (Raji et al., 2022). For example, the New York Metropolitan Transportation Authority moved forward with and expanded its use of facial recognition despite a proof-of-concept trial reporting a 100% failure rate to detect faces within acceptable parameters (Berger, 2019). Similarly, facial analysis tools often oversell their abilities (Raji & Fried, 2020), dubiously claiming to be able to infer individuals' sexual orientation (Leuner, 2019), emotions (Stark & Hoey, 2021), hireability (Fetscherin et al., 2020), or criminality (Wu & Zhang, 2016). Stark & Hutson (2022) highlight that computer vision systems have created a resurgence in the "scientifically baseless, racist, and discredited pseudoscientific fields" of physiognomy and phrenology.

21.2.2 Militarization and political interference

Governments have a vested interest in funding AI research in the name of national security and state building. This risks an arms race between nation-states, which carries with it "high rates of investment, a lack of transparency, mutual suspicion and fear, and a perceived intent to deploy first" (Sisson et al., 2020).

Lethal autonomous weapons systems receive significant attention because they are easy to imagine, and indeed many such systems are under development (Heikkilä, 2022). However, AI also facilitates cyber-attacks and disinformation campaigns (i.e., inaccurate or misleading information that is shared with the intent to deceive). AI systems allow the creation of highly realistic fake content and facilitate the dissemination of information, often to targeted audiences (Akers et al., 2018) and at scale (Bontridder & Poullet, 2021).

Problem 21.6

Kosinski et al. (2013) suggest that sensitive variables, including sexual orientation, ethnicity, religious and political views, personality traits, intelligence, happiness, use of addictive substances, parental separation, age, and gender, can be predicted by "likes" on social media alone. From this information, personality traits like "openness" can be used for manipulative purposes (e.g., to change voting behavior).

21.2.3 Fraud

Unfortunately, AI is a useful tool for automating fraudulent activities (e.g., sending mass emails or text messages that trick people into revealing sensitive information or sending money). Generative AI can be used to deceive people into thinking they are interacting with a legitimate entity or generate fake documents that mislead or deceive people. Additionally, AI could increase the sophistication of cyber-attacks, such as by generating more convincing phishing emails or adapting to the defenses of targeted organizations.

This highlights the downside of calls for transparency in machine learning systems: the more open and transparent these systems are, the more vulnerable they may be to security risks or use by bad-faith actors. For example, generative language models, like ChatGPT, have been used to write software and emails that could be used for espionage,

Problem 21.7

ransomware, and other malware (Goodin, 2023).

The tendency to anthropomorphize computer behaviors and particularly the projection of meaning onto strings of symbols is termed the *ELIZA effect* (Hofstadter, 1995). This leads to a false sense of security when interacting with sophisticated chatbots, making people more susceptible to text-based fraud such as romance scams or business email compromise schemes (Abrahams, 2023). Véliz (2023) highlights how emoji use in some chatbots is inherently manipulative, exploiting instinctual responses to emotive images.

21.2.4 Data privacy

Modern deep learning methods rely on huge crowd-sourced datasets, which may contain sensitive or private information. Even when sensitive information is removed, auxiliary knowledge and redundant encodings can be used to de-anonymize datasets (Narayanan & Shmatikov, 2008). Indeed, this famously happened to the Governor of Massachusetts, William Weld, in 1997. After an insurance group released health records that had been stripped of obvious personal information like patient name and address, an aspiring graduate student was able to "de-anonymize" which records belonged to Governor Weld by cross-referencing with public voter rolls.

Problem 21.8

Hence, privacy-first design is important for ensuring the security of individuals' information, especially when applying deep learning techniques to high-risk areas such as healthcare and finance. Differential privacy and semantic security (homomorphic encryption or secure multi-party computation) methods can be used to ensure data security during model training (see Mireshghallah et al., 2020; Boulemtafes et al., 2020).

21.3 Other social, ethical, and professional issues

The previous section identified areas where AI can be deliberately misused. This section describes other potential side effects of the widespread adoption of AI.

21.3.1 Intellectual property

Intellectual property (IP) can be characterized as non-physical property that is the product of original thought (Moore & Himma, 2022). In practice, many AI models are trained on copyrighted material. Consequently, these models' deployment can pose legal and ethical risks and run afoul of intellectual property rights (Henderson et al., 2023).

Sometimes, these issues are explicit. When language models are prompted with excerpts of copyrighted material, their outputs may include copyrighted text verbatim, and similar issues apply in the context of image generation in diffusion models (Henderson et al., 2023; Carlini et al., 2022, 2023). Even if the training falls under "fair use," this may violate the moral rights of content creators in some cases (Weidinger et al., 2022).

More subtly, generative models (chapters 12,14–18) raise novel questions regarding AI

and intellectual property. Can the output of a machine learning model (e.g., art, music, code, text) be copyrighted or patented? Is it morally acceptable or legal to fine-tune a model on a particular artist's work to reproduce that artist's style? IP law is one area that highlights how existing legislation was not created with machine learning models in mind. Although governments and courts may set precedents in the near future, these questions are still open at the time of writing. Problem 21.9

21.3.2 Automation bias and moral deskilling

As society relies more on AI systems, there is an increased risk of automation bias (i.e., expectations that the model outputs are correct because they are "objective"). This leads to the view that quantitative methods are better than qualitative ones. However, as we shall see in section 21.5, purportedly objective endeavors are rarely value-free.

The sociological concept of deskilling refers to the redundancy and devaluation of skills in light of automation (Braverman, 1974). For example, off-loading cognitive skills like memory onto technology may cause a decrease in our capacity to remember things. Analogously, the automation of AI in morally-loaded decision-making may lead to a decrease in our moral abilities (Vallor, 2015). For example, in the context of war, the automation of weapons systems may lead to the dehumanization of victims of war (Asaro, 2012; Heyns, 2017). Similarly, care robots in elderly-, child-, or healthcare settings may reduce our ability to care for one another (Vallor, 2011).

21.3.3 Environmental impact

Training deep networks requires significant computational power and hence consumes a large amount of energy. Strubell et al. (2019, 2020) estimate that training a transformer model with 213 million parameters emitted around 284 tonnes of CO_2.[2] Luccioni et al. (2022) have provided similar estimates for the emissions produced from training the BLOOM language model. Unfortunately, the increasing prevalence of closed, proprietary models means that we know nothing about their environmental impacts (Luccioni, 2023).

21.3.4 Employment and society

The history of technological innovation is a history of job displacement. In 2018, the McKinsey Global Institute estimated that AI may increase economic output by approximately US $13 trillion by 2030, primarily from the substitution of labor by automation (Bughin et al., 2018). Another study from the McKinsey Global Institute suggests that up to 30% of the global workforce (10-800 million people) could have their jobs displaced due to AI between 2016 and 2030 (Manyika et al., 2017; Manyika & Sneader, 2018).

[2] As a baseline, it is estimated that the average human is responsible for around 5 tonnes of CO_2 per year, with individuals from major oil-producing countries responsible for three times this amount. See https://ourworldindata.org/co2-emissions.

Problem 21.10

However, forecasting is inherently difficult, and although automation by AI may lead to short-term job losses, the concept of *technological unemployment* has been described as a "temporary phase of maladjustment" (Keynes, 2010). This is because gains in wealth can offset gains in productivity by creating increased demand for products and services. In addition, new technologies can create new types of jobs.

Even if automation doesn't lead to a net loss of overall employment in the long term, new social programs may be required in the short term. Therefore, regardless of whether one is optimistic (Brynjolfsson & McAfee, 2016; Danaher, 2019), neutral (Metcalf et al., 2016; Calo, 2018; Frey, 2019), or pessimistic (Frey & Osborne, 2017) about the possibility of unemployment in light of AI, it is clear that society will be changed significantly.

21.3.5 Concentration of power

As deep networks increase in size, there is a corresponding increase in the amount of data and computing power required to train these models. In this regard, smaller companies and start-ups may not be able to compete with large, established tech companies. This may give rise to a feedback loop whereby the power and wealth become increasingly concentrated in the hands of a small number of corporations. A recent study finds an increasing discrepancy between publications at major AI venues by large tech firms and "elite" universities versus mid- or lower-tier universities (Ahmed & Wahed, 2016). In many views, such a concentration of wealth and power is incompatible with just distributions in society (Rawls, 1971).

Problem 21.11

This has led to calls to democratize AI by making it possible for everyone to create such systems (Li, 2018; Knight, 2018; Kratsios, 2019; Riedl, 2020). Such a process requires making deep learning technologies more widely available and easier to use via open source and open science so that more people can benefit from them. This reduces barriers to entry and increases access to AI while cutting down costs, ensuring model accuracy, and increasing participation and inclusion (Ahmed et al., 2020).

21.4 Case study

We now describe a case study that speaks to many of the issues that we have discussed in this chapter. In 2018, the popular media reported on a controversial facial analysis model—dubbed "gaydar AI" (Wang & Kosinski, 2018)—with sensationalist headlines like *AI Can Tell If You're Gay: Artificial Intelligence Predicts Sexuality From One Photo with Startling Accuracy* (Ahmed, 2017); *A Frightening AI Can Determine Whether a Person Is Gay With 91 Percent Accuracy* (Matsakis, 2017); and *Artificial Intelligence System Can Tell If You're Gay* (Fernandez, 2017).

There are a number of problems with this work. First, the training dataset was highly biased and unrepresentative, being comprised mostly of Caucasian images. Second, modeling and validation are also questionable, given the fluidity of gender and sexuality. Third, the most obvious use case for such a model is the targeted discrimination and

persecution of LGBTQ+ individuals in countries where queerness is criminalized. Fourth, with regard to transparency, explainability, and value alignment more generally, the "gaydar" model appears to pick up on spurious correlations due to patterns in grooming, presentation, and lifestyle rather than facial structure, as the authors claimed (Agüera y Arcas et al., 2018). Fifth, with regard to data privacy, questions arise regarding the ethics of scraping "public" photos and sexual orientation labels from a dating website. Finally, with regard to scientific communication, the researchers communicated their results in a way that was sure to generate headlines: even the title of the paper is an overstatement of the model's abilities: *Deep Neural Networks Can Detect Sexual Orientation from Faces*. (They cannot.)

It should also be apparent that a facial-analysis model for determining sexual orientation does *nothing* whatsoever to benefit the LGBTQ+ community. If it is to benefit society, the most important question is whether a particular study, experiment, model, application, or technology serves the interests of the community to which it pertains.

21.5 The value-free ideal of science

This chapter has enumerated a number of ways that the objectives of AI systems can unintentionally, or through misuse, diverge from the values of humanity. We now argue that scientists are not neutral actors; their values inevitably impinge on their work.

Perhaps this is surprising. There is a broad belief that science is—or ought to be—objective. This is codified by the *value-free ideal of science*. Many would argue that machine learning is objective because algorithms are just mathematics. However, analogous to algorithmic bias (section 21.1.1), there are four stages at which the values of AI practitioners can affect their work (Reiss & Sprenger, 2017):

1. The choice of research problem.
2. Gathering evidence related to a research problem.
3. Accepting a scientific hypothesis as an answer to a problem.
4. Applying the results of scientific research.

It is perhaps uncontroversial that values play a significant role in the first and last of these stages. The initial selection of research problems and the choice of subsequent applications are influenced by the interests of scientists, institutions, and funding agencies. However, the value-free ideal of science prescribes minimizing the influence of moral, personal, social, political, and cultural values on the intervening scientific process. This idea presupposes the *value-neutrality thesis*, which suggests that scientists can (at least in principle) attend to stages (2) and (3) without making these value judgments.

However, whether intentional or not, values are embedded in machine learning research. Most of these values would be classed as *epistemic* (e.g., performance, generalization, building on past work, efficiency, novelty). But deciding the set of values is itself a value-laden decision; few papers explicitly discuss societal need, and fewer still discuss potential negative impacts (Birhane et al., 2022b). Philosophers of science have

questioned whether the value-free ideal of science is attainable or desirable. For example, Longino (1990, 1996) argues that these epistemic values are not *purely* epistemic. Kitcher (2011a,b) argues that scientists don't typically care about *truth* itself; instead, they pursue truths relevant to their goals and interests.

Machine learning depends on inductive inference and is hence prone to inductive risk. Models are only constrained at the training data points, and the curse of dimensionality means this is a tiny proportion of the input space; outputs can always be wrong, regardless of how much data we use to train the model. It follows that choosing to accept or reject a model prediction requires a value judgment: that the risks if we are wrong in acceptance are lower than the risks if we are wrong in rejection.

Hence, the use of inductive inference implies that machine learning models are deeply value-laden (Johnson, 2022). In fact, if they were not, they would have no application: it is precisely *because* they are value-laden that they are useful. Thus, accepting that algorithms are used for ranking, sorting, filtering, recommending, categorizing, labeling, predicting, etc., in the real world implies that these processes will have real-world effects. As machine learning systems become increasingly commercialized and applied, they become more entrenched in the things we care about.

These insights have implications for researchers who believe that algorithms are somehow more objective than human decision-makers (and, therefore, ought to replace human decision-makers in areas where we think objectivity matters).

21.6 Responsible AI research as a collective action problem

It is easy to defer responsibility. Students and professionals who read this chapter might think their work is so far removed from the real world or a small part of a larger machine that their actions could not make a difference. However, this is a mistake. Researchers often have a choice about the projects to which they devote their time, the companies or institutions for which they work, the knowledge they seek, the social and intellectual circles in which they interact, and the way they communicate.

Doing the right thing, whatever that may comprise, often takes the form of a social dilemma; the best outcomes depend upon cooperation, although it isn't necessarily in any individual's interest to cooperate: responsible AI research is a collective action problem.

Problem 21.12

21.6.1 Scientific communication

One positive step is to communicate responsibly. Misinformation spreads faster and persists more readily than the truth in many types of social networks (LaCroix et al., 2021; Ceylan et al., 2023). As such, it is important not to overstate machine learning systems' abilities (see case study above) and to avoid misleading anthropomorphism. It is also important to be aware of the potential for the misapplication of machine learning techniques. For example, pseudoscientific practices like phrenology and physiognomy have found a surprising resurgence in AI (Stark & Hutson, 2022).

21.6.2 Diversity and heterogeneity

A second positive step is to encourage diversity. When social groups are homogeneous (composed mainly of similar members) or homophilous (comprising members that tend to associate with similar others), the dominant group tends to have its conventions recapitulated and stabilized (O'Connor & Bruner, 2019). One way to mitigate systems of oppression is to ensure that diverse views are considered. This might be achieved through equity, diversity, inclusion, and accessibility initiatives (at an institutional level), participatory and community-based approaches to research (at the research level), and increased awareness of social, political, and moral issues (at an individual level).

The theory of *standpoint epistemology* (Harding, 1986) suggests that knowledge is socially situated (i.e., depends on one's social position in society). Homogeneity in tech circles can give rise to biased tech (Noble, 2018; Eubanks, 2018; Benjamin, 2019; Broussard, 2023). Lack of diversity implies that the perspectives of the individuals who create these technologies will seep into the datasets, algorithms, and code as the default perspective. Broussard (2023) argues that because much technology is developed by able-bodied, white, cisgender, American men, that technology is *optimized for* able-bodied, white, cisgender, American men, the perspective of whom is taken as the status quo. Ensuring technologies benefit historically marginalized communities requires researchers to understand the needs, wants, and perspectives of those communities (Birhane et al., 2022a). *Design justice* and participatory- and community-based approaches to AI research contend that the communities affected by technologies should be actively involved in their design (Constanza-Chock, 2020).

21.7 Ways forward

It is undeniable that AI will radically change society for better or worse. However, optimistic visions of a future Utopian society driven by AI should be met with caution and a healthy dose of critical reflection. Many of the touted benefits of AI are beneficial only in certain contexts and only to a subset of society. For example, Green (2019) highlights that one project developed using AI to enhance police accountability and alternatives to incarceration and another developed to increase security through predictive policing are both advertised as "AI for Social Good." Assigning this label is a value judgment that lacks any grounding principles; one community's good is another's harm.

When considering the potential for emerging technologies to benefit society, it is necessary to reflect on whether those benefits will be equally or equitably distributed. It is often assumed that the most technologically advanced solution is the best one—so-called *technochauvinism* (Broussard, 2018). However, many social issues arise from underlying social problems and do not warrant technological solutions.

Some common themes emerged throughout this chapter, and we would like to impress four key points upon the reader:

1. **Research in machine learning cannot avoid ethics.** Historically, researchers could focus on fundamental aspects of their work in a controlled laboratory set-

ting. However, this luxury is dwindling due to the vast economic incentives to commercialize AI and the degree to which academic work is funded by industry (see Abdalla & Abdalla, 2021); even theoretical studies may have social impacts, so researchers must engage with the social and ethical dimensions of their work.

2. **Even purely technical decisions can be value-laden.** There is still a widely-held view that AI is fundamentally just mathematics and, therefore, it is "objective," and ethics are irrelevant. This assumption is not true when we consider the creation of AI systems or their deployment.
3. **We should question the structures within which AI work takes place.** Much research on AI ethics focuses on specific situations rather than questioning the larger social structures within which AI will be deployed. For example, there is considerable interest in ensuring algorithmic fairness, but it may not always be possible to instantiate conceptions of fairness, justice, or equity within extant social and political structures. Therefore, technology is inherently political.
4. **Social and ethical problems don't necessarily require technical solutions.** Many potential ethical problems surrounding AI technologies are primarily social and structural, so technical innovation alone cannot solve these problems; if scientists are to effect positive change with new technology, they must take a political and moral position.

Problem 21.13

Where does this leave the average scientist? Perhaps with the following imperative: it is necessary to reflect upon the moral and social dimensions of one's work. This might require actively engaging those communities that are likely to be most affected by new technologies, thus cultivating relationships between researchers and communities and empowering those communities. Likewise, it might involve engagement with the literature beyond one's own discipline. For philosophical questions, the *Stanford Encyclopedia of Philosophy* is an invaluable resource. Interdisciplinary conferences are also useful in this regard. Leading work is published at both the Conference on Fairness, Accountability, and Transparency (FAccT) and the Conference on AI, Ethics, and Society (AIES).

21.8 Summary

This chapter considered the ethical implications of deep learning and AI. The value alignment problem is the task of ensuring that the objectives of AI systems are aligned with human objectives. Bias, explainability, artificial moral agency, and other topics can be viewed through this lens. AI can be intentionally misused, and this chapter detailed some ways this can happen. Progress in AI has further implications in areas as diverse as IP law and climate change.

Ethical AI is a collective action problem, and the chapter concludes with an appeal to scientists to consider the moral and ethical implications of their work. Every ethical issue is not within the control of every individual computer scientist. However, this does not imply that researchers have no responsibility whatsoever to consider—and mitigate where they can—the potential for misuse of the systems they create.

Problems

Problem 21.1 It was suggested that the most common specification of the value alignment problem for AI is "the problem of ensuring that the values of AI systems are aligned with the values of humanity." Discuss the ways in which this statement of the problem is underspecified. *Discussion Resource*: LaCroix (2023).

Problem 21.2 Goodhart's law states that "when a measure becomes a target, it ceases to be a good measure." Consider how this law might be reformulated to apply to value alignment for artificial intelligence, given that the loss function is a mere proxy for our true objectives.

Problem 21.3 Suppose a university uses data from past students to build models for predicting "student success," where those models can support informed changes in policies and practices. Consider how biases might affect each of the four stages of the development and deployment of this model.
Discussion Resource: Fazelpour & Danks (2021).

Problem 21.4 We might think of functional transparency, structural transparency, and run transparency as orthogonal. Provide an example of how an increase in one form of transparency may not lead to a concomitant increase in another form of transparency.
Discussion Resource: Creel (2020).

Problem 21.5 If a computer scientist writes a research paper on AI or pushes code to a public repository, do you consider them responsible for future misuse of their work?

Problem 21.6 To what extent do you think the militarization of AI is inevitable?

Problem 21.7 In light of the possible misuse of AI highlighted in section 21.2, make arguments both for and against the open-source culture of research in deep learning.

Problem 21.8 Some have suggested that personal data is a source of power for those who own it. Discuss the ways personal data is valuable to companies that utilize deep learning and consider the claim that losses to privacy are experienced collectively rather than individually.
Discussion Resource: Véliz (2020).

Problem 21.9 What are the implications of generative AI for the creative industries? How do you think IP laws should be modified to cope with this new development?

Problem 21.10 A good forecast must (i) be specific enough to know when it is wrong, (ii) account for possible cognitive biases, and (iii) allow for rationally updating beliefs. Consider any claim in the recent media about future AI and discuss whether it satisfies these criteria.
Discussion Resource: Tetlock & Gardner (2016).

Problem 21.11 Some critics have argued that calls to democratize AI have focused too heavily on the *participatory* aspects of democracy, which can increase risks of errors in collective perception, reasoning, and agency, leading to morally-bad outcomes. Reflect on each of the following: *What* aspects of AI should be democratized? *Why* should AI be democratized? *How* should AI be democratized?
Discussion Resource: Himmelreich (2022).

Problem 21.12 In March 2023, the Future of Life Institute published a letter, "Pause Giant AI Experiments," in which they called on all AI labs to immediately pause for at least six months the training of AI systems more powerful than GPT-4. Discuss the motivations of the authors in writing this letter, the public reaction, and the implications of such a pause. Relate this episode to the view that AI ethics can be considered a collective action problem (section 21.6).
Discussion Resource: Gebru et al. (2023).

Problem 21.13 Discuss the merits of the four points in section 21.7. Do you agree with them?

Appendix A

Notation

This appendix details the notation used in this book. This mostly adheres to standard conventions in computer science, but deep learning is applicable to many different areas, so it is explained in full. In addition, there are several notational conventions that are unique to this book, including notation for functions and the systematic distinction between parameters and variables.

Scalars, vectors, matrices, and tensors

Scalars are denoted by either small or capital letters a, A, α. Column vectors (i.e., 1D arrays of numbers) are denoted by small bold letters $\mathbf{a}, \boldsymbol{\phi}$, and row vectors as the transpose of column vectors $\mathbf{a}^T, \boldsymbol{\phi}^T$. Matrices and tensors (i.e., 2D and ND arrays of numbers, respectively) are both represented by bold capital letters $\mathbf{B}, \boldsymbol{\Phi}$.

Variables and parameters

Variables (usually the inputs and outputs of functions or intermediate calculations) are always denoted by Roman letters $a, \mathbf{b}, \mathbf{C}$. Parameters (which are internal to functions or probability distributions) are always denoted by Greek letters $\alpha, \boldsymbol{\beta}, \boldsymbol{\Gamma}$. Generic, unspecified parameters are denoted by $\boldsymbol{\phi}$. This distinction is retained throughout the book except for the policy in reinforcement learning, which is denoted by π according to the usual convention.

Sets

Sets are denoted by curly brackets, so $\{0, 1, 2\}$ denotes the numbers 0, 1, and 2. The notation $\{0, 1, 2, \ldots\}$ denotes the set of non-negative integers. Sometimes, we want to specify a set of variables and $\{\mathbf{x}_i\}_{i=1}^{I}$ denotes the I variables $\mathbf{x}_1, \ldots \mathbf{x}_I$. When it's not necessary to specify how many items are in the set, this is shortened to $\{\mathbf{x}_i\}$. The notation $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{I}$ denotes the set of I pairs $\mathbf{x}_i, \mathbf{y}_i$. The convention for naming sets is to use calligraphic letters. Notably, $\mathcal{B}_t$ is used to denote the set of indices in a batch at iteration t during training. The number of elements in a set $\mathcal{S}$ is denoted by $|\mathcal{S}|$.

The set $\mathbb{R}$ denotes the set of real numbers. The set $\mathbb{R}^+$ denotes the set of non-negative real numbers. The notation $\mathbb{R}^D$ denotes the set of D-dimensional vectors containing real

numbers. The notation $\mathbb{R}^{D_1 \times D_2}$ denotes the set of matrices of dimension $D_1 \times D_2$. The notation $\mathbb{R}^{D_1 \times D_2 \times D_3}$ denotes the set of tensors of size $D_1 \times D_2 \times D_3$ and so on.

The notation $[a, b]$ denotes the real numbers from a to b, including a and b themselves. When the square brackets are replaced by round brackets, this means that the adjacent value is not included in the set. For example, the set $(-\pi, \pi]$ denotes the real numbers from $-\pi$ to π, but excluding $-\pi$.

Membership of sets is denoted by the symbol $\in$, so $x \in \mathbb{R}^+$ means that the variable x is a non-negative real number, and the notation $\boldsymbol{\Sigma} \in \mathbb{R}^{D \times D}$ denotes that $\boldsymbol{\Sigma}$ is a matrix of size $D \times D$. Sometimes, we want to work through each element of a set systematically, and the notation $\forall\ \{1, \ldots, K\}$ means "for all" the integers from 1 to K.

Functions

Functions are expressed as a name, followed by square brackets that contain the arguments of the function. For example, $\log[x]$ returns the logarithm of the variable x. When the function returns a vector, it is written in bold and starts with a small letter. For example, the function $\mathbf{y} = \mathbf{mlp}[\mathbf{x}, \boldsymbol{\phi}]$ returns a vector $\mathbf{y}$ and has vector arguments $\mathbf{x}$ and $\boldsymbol{\phi}$. When a function returns a matrix or tensor, it is written in bold and starts with a capital letter. For example, the function $\mathbf{Y} = \mathbf{Sa}[\mathbf{X}, \boldsymbol{\phi}]$ returns a matrix $\mathbf{Y}$ and has arguments $\mathbf{X}$ and $\boldsymbol{\phi}$. When we want to leave the arguments of a function deliberately ambiguous, we use the bullet symbol (e.g., $\mathbf{mlp}[\bullet, \boldsymbol{\phi}]$).

Minimizing and maximizing

Some special functions are used repeatedly throughout the text:

- The function $\min_x[\mathrm{f}[x]]$ returns the minimum value of the function $\mathrm{f}[x]$ over all possible values of the variable x. This notation is often used without specifying the details of how this minimum might be found.
- The function $\operatorname{argmin}_x[\mathrm{f}[x]]$ returns the value of x that minimizes $\mathrm{f}[x]$, so if $y = \operatorname{argmin}_x[\mathrm{f}[x]]$, then $\min_x[\mathrm{f}[x]] = \mathrm{f}[y]$.
- The functions $\max_x[\mathrm{f}[x]]$ and $\operatorname{argmax}_x[\mathrm{f}[x]]$ perform the equivalent operations for maximizing functions.

Probability distributions

Probability distributions should be written as $Pr(x = a)$, denoting that the random variable x takes the value of a. However, this notation is cumbersome. Hence, we usually simplify this and just write $Pr(x)$, where x denotes either the random variable or the value it takes according to the sense of the equation. The conditional probability of y given x is written as $Pr(y|x)$. The joint probability of y and x is written as $Pr(y, x)$. These two forms can be combined, so $Pr(\mathbf{y}|\mathbf{x}, \boldsymbol{\phi})$ denotes the probability of the variable $\mathbf{y}$, given that we know $\mathbf{x}$ and $\boldsymbol{\phi}$. Similarly, $Pr(\mathbf{y}, \mathbf{x}|\boldsymbol{\phi})$ denotes the probability of variables $\mathbf{y}$ and $\mathbf{x}$ given that we know $\boldsymbol{\phi}$. When we need two probability distributions over the same variable, we write $Pr(x)$ for the first distribution and $q(x)$ for the second. More information about probability distributions can be found in appendix C.

Asymptotic notation

Asymptotic notation is used to compare the amount of work done by different algorithms as the size D of the input increases. This can be done in various ways, but this book only uses *big-O* notation, which represents an upper bound on the growth of computation in an algorithm. A function $\mathrm{f}[n]$ is $\mathcal{O}[\mathrm{g}[n]]$ if there exists a constant $c > 0$ and integer n_0 such that $\mathrm{f}[n] < c \cdot \mathrm{g}[n]$ for all $n > n_0$.

This notation provides a bound on the worst-case running time of an algorithm. For example, when we say that inversion of a $D \times D$ matrix is $\mathcal{O}[D^3]$, we mean that the computation will increase no faster than some constant times D^3 once D is large enough. This gives us an idea of how feasible it is to invert matrices of different sizes. If $D = 10^3$, then it may take of the order of 10^9 operations to invert it.

Miscellaneous

A small dot in a mathematical equation is intended to improve ease of reading and has no real meaning (or just implies multiplication). For example, $\alpha \cdot \mathrm{f}[x]$ is the same as $\alpha \mathrm{f}[x]$ but is easier to read. To avoid ambiguity, dot products are written as $\mathbf{a}^T\mathbf{b}$ (see appendix B.3.4). A left arrow symbol $\leftarrow$ denotes assignment, so $x \leftarrow x + 2$ means that we are adding two to the current value of x.

Appendix B

Mathematics

This appendix reviews mathematical concepts that are used in the main text.

B.1 Functions

A *function* defines a mapping from a set $\mathcal{X}$ (e.g., the set of real numbers) to another set $\mathcal{Y}$. An *injection* is a one-to-one function where *every* element in the first set maps to a unique position in the second set (but there may be elements of the second set that are not mapped to). A *surjection* is a function where every element in the second set receives a mapping from the first (but there may be elements of the first set that are not mapped). A *bijection* or *bijective mapping* is a function that is both injective and surjective. It provides a one-to-one correspondence between all members of the two sets. A *diffeomorphism* is a special case of a bijection where both the forward and reverse mapping are differentiable.

B.1.1 Lipschitz constant

A function $\mathrm{f}[z]$ is *Lipschitz continuous* if for all z_1, z_2:

$$||\mathrm{f}[z_1] - \mathrm{f}[z_2]|| \leq \beta||z_1 - z_2||, \tag{B.1}$$

where β is known as the Lipschitz constant and determines the maximum gradient of the function (i.e., how fast the function can change) with respect to the distance metric. If the Lipschitz constant is less than one, the function is a contraction mapping, and we can use Banach's theorem to find the inverse for any point (see figure 16.9).

Composing two functions with Lipschitz constants β_1 and β_2 creates a new Lipschitz continuous function with a constant that is less than or equal to $\beta_1\beta_2$. Adding two functions with Lipschitz constants β_1 and β_2 creates a new Lipschitz continuous function with a constant that is less than or equal to $\beta_1+\beta_2$. The Lipschitz constant of a linear transformation $\mathbf{f}[\mathbf{z}] = \mathbf{A}\mathbf{z}+\mathbf{b}$ with respect to a Euclidean distance measure is the maximum eigenvalue of $\mathbf{A}$.

B.1.2 Convexity

A function is *convex* if we can draw a straight line between any two points on the function, and this line always lies above the function. Similarly, a function is *concave* if a straight line between any two points always lies below the function. By definition, convex (concave) functions have at most one minimum (maximum).

A region of $\mathbb{R}^D$ is convex if we can draw a straight line between any two points on the boundary of the region without intersecting the boundary in another place. Gradient descent guarantees to find the global minimum of any function that is both convex and defined on a convex region.

B.1.3 Special functions

The following functions are used in the main text:

- The *exponential function* $y = \exp[x]$ (figure B.1a) maps a real variable $x \in \mathbb{R}$ to a non-negative number $y \in \mathcal{R}^+$ as $y = e^x$.
- The *logarithm* $x = \log[y]$ (figure B.1b) is the inverse of the exponential function and maps a non-negative number $y \in \mathcal{R}^+$ to a real variable $x \in \mathbb{R}$. Note that all logarithms in this book are natural (i.e., in base e).
- The *gamma function* $\Gamma[x]$ (figure B.1c) is defined as:

$$\Gamma[x] = \int_0^\infty t^{x-1} e^{-t} dt. \tag{B.2}$$

  This extends the factorial function to continuous values so that $\Gamma[x] = (x-1)!$ for $x \in \{1, 2, \ldots\}$.
- The *Dirac delta function* $\delta[\mathbf{z}]$ has a total area of one, all of which is at position $\mathbf{z} = \mathbf{0}$. A dataset with N elements can be thought of as a probability distribution consisting of a sum of N delta functions centered at each data point $\mathbf{x}_i$ and scaled by $1/N$. The delta function is usually drawn as an arrow (e.g., figure 5.12). The delta function has the key property that:

$$\int \mathrm{f}[\mathbf{x}]\delta[\mathbf{x} - \mathbf{x}_0]d\mathbf{x} = \mathrm{f}[\mathbf{x}_0]. \tag{B.3}$$

B.1.4 Stirling's formula

Stirling's formula (figure B.2) approximates the factorial function (and hence the Gamma function) using the formula:

$$x! \approx \sqrt{2\pi x}\left(\frac{x}{e}\right)^x. \tag{B.4}$$

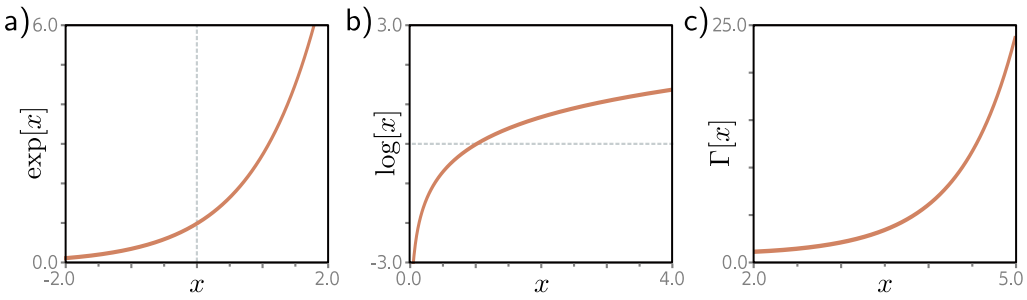


Figure B.1 Exponential, logarithm, and gamma functions. a) The exponential function maps a real number to a positive number. It is a convex function. b) The logarithm is the inverse of the exponential and maps a positive number to a real number. It is a concave function. c) The Gamma function is a continuous extension of the factorial function so that $\Gamma[x] = (x-1)!$ for $x \in \{1, 2, \ldots\}$.

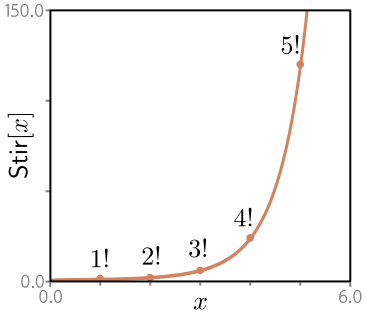


Figure B.2 Stirling's formula. The factorial function $x!$ can be approximated by Stirling's formula $\text{Stir}[x]$ which is defined for every real value.

B.2 Binomial coefficients

Binomial coefficients are written as $\binom{n}{k}$ and pronounced as "n choose k." They are positive integers that represent the number of ways of choosing an unordered subset of k items from a set of n items without replacement. Binomial coefficients can be computed using the simple formula:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}. \tag{B.5}$$

B.2.1 Autocorrelation

The autocorrelation $r[\tau]$ of a continuous function $\text{f}[z]$ is defined as:

$$r[\tau] = \int_{-\infty}^{\infty} \text{f}[t+\tau]\text{f}[t]dt, \tag{B.6}$$

where τ is the time lag. Sometimes, this is normalized by $r[0]$ so that the autocorrelation at time lag zero is one. The autocorrelation function is a measure of the correlation of the function with itself as a function of an offset (i.e., the time lag). If a function changes slowly and predictably, then the autocorrelation function will decrease slowly as the time lag increases from zero. If the function changes fast and unpredictably, then it will decrease quickly to zero.

B.3 Vector, matrices, and tensors

In machine learning, a vector $\mathbf{x} \in \mathbb{R}^D$ is a one-dimensional array of D numbers, which we will assume are organized in a column. Similarly, a matrix $\mathbf{Y} \in \mathbb{R}^{D_1 \times D_2}$ is a two-dimensional array of numbers with D_1 rows and D_2 columns. A tensor $\mathbf{z} \in \mathbb{R}^{D_1 \times D_2 \ldots \times D_N}$ is an N-dimensional array of numbers. Confusingly, all three of these quantities are stored in objects known as "tensors" in deep learning APIs such as PyTorch and TensorFlow.

B.3.1 Transpose

The transpose $\mathbf{A}^T \in \mathbb{R}^{D_2 \times D_1}$ of a matrix $\mathbf{A} \in \mathbb{R}^{D_1 \times D_2}$ is formed by reflecting it around the principal diagonal so that the k^{th} column becomes the k^{th} row and vice-versa. If we take the transpose of a matrix product $\mathbf{AB}$, then we take the transpose of the original matrices but reverse the order so that

$$(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T. \tag{B.7}$$

The transpose of a column vector $\mathbf{a}$ is a row vector $\mathbf{a}^T$ and vice-versa.

B.3.2 Vector and matrix norms

For a vector $\mathbf{z}$, the ℓ_p norm is defined as:

$$||\mathbf{z}||_p = \left(\sum_{d=1}^{D} |z_d|^p\right)^{1/p}. \tag{B.8}$$

When $p = 2$, this returns the length of the vector, and this is known as the *Euclidean norm*. It is this case that is most commonly used in deep learning, and often the exponent p is omitted, and the Euclidean norm is just written as $||\mathbf{z}||$. When $p = \infty$, the operator returns the maximum absolute value in the vector.

Norms can be computed in a similar way for matrices. For example, the ℓ_2 norm of a matrix $\mathbf{Z}$ (known as the *Frobenius norm*) is calculated as:

$$||\mathbf{Z}||_F = \left(\sum_{i=1}^{I} \sum_{j=1}^{J} |z_{ij}|^2 \right)^{1/2}. \tag{B.9}$$

B.3.3 Product of matrices

The product $\mathbf{C} = \mathbf{AB}$ of two matrices $\mathbf{A} \in \mathbb{R}^{D_1 \times D_2}$ and $\mathbf{B} \in \mathbb{R}^{D_2 \times D_3}$ is a third matrix $\mathbf{C} \in \mathbb{R}^{D_1 \times D_3}$ where:

$$C_{ij} = \sum_{d=1}^{D_2} A_{id} B_{dj}. \tag{B.10}$$

B.3.4 Dot product of vectors

The dot product $\mathbf{a}^T\mathbf{b}$ of two vectors $\mathbf{a} \in \mathbb{R}^D$ and $\mathbf{b} \in \mathbb{R}^D$ is a scalar and is defined as:

$$\mathbf{a}^T\mathbf{b} = \mathbf{b}^T\mathbf{a} = \sum_{d=1}^{D} a_d b_d. \tag{B.11}$$

It can be shown that the dot product is proportional to the Euclidean norm of the first vector times the Euclidean norm of the second vector times the angle θ between them:

$$\mathbf{a}^T\mathbf{b} = ||\mathbf{a}||\,||\mathbf{b}||\cos[\theta]. \tag{B.12}$$

B.3.5 Inverse

A square matrix $\mathbf{A}$ may or may not have an inverse $\mathbf{A}^{-1}$ such that $\mathbf{A}^{-1}\mathbf{A} = \mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$. If a matrix does not have an inverse, it is called *singular*. If we take the inverse of a matrix product $\mathbf{AB}$ then we can equivalently take the inverse of each matrix individually and reverse the order of multiplication.

$$(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}. \tag{B.13}$$

In general, it takes $\mathcal{O}[D^3]$ operations to invert a $D \times D$ matrix. However, inversion is more efficient for special types of matrices, including diagonal, orthogonal, and triangular matrices (see section B.4).

B.3.6 Subspaces

Consider a matrix $\mathbf{A} \in \mathbb{R}^{D_1 \times D_2}$. If the number of columns D_2 of the matrix is fewer than the number of rows D_1 (i.e., the matrix is "portrait"), the product $\mathbf{Ax}$ cannot reach all

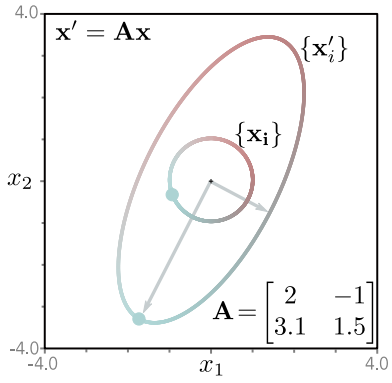


Figure B.3 Eigenvalues. When the points $\{\mathbf{x}_i\}$ on the unit circle are transformed to points $\{\mathbf{x}_i'\}$ by a linear transformation $\mathbf{x}_i' = \mathbf{A}\mathbf{x}_i$, they are mapped to an ellipse. For example, the light blue point on the unit circle is mapped to the light blue point on the ellipse. The length of the major (longest) axis of the ellipse (long gray arrow) is the magnitude of the first eigenvalue of the matrix, and the length of the minor (shortest) axis of the ellipse (short gray arrow) is the magnitude of the second eigenvalue.

possible positions in the D_1-dimensional output space. This product consists of the D_2 columns of $\mathbf{A}$ weighted by the D_2 elements of $\mathbf{x}$ and can only reach the *linear subspace* that is spanned by these columns. This is known as the *column space* of the matrix. Conversely, for a landscape matrix $\mathbf{A}$, the part of the input space that maps to zero (i.e., those $\mathbf{x}$ where $\mathbf{A}\mathbf{x} = \mathbf{0}$) is termed the *nullspace* of the matrix.

B.3.7 Eigenspectrum

If we multiply the set of 2D points on a unit circle by a 2×2 matrix $\mathbf{A}$, they map to an ellipse (figure B.3). The radii of the major and minor axes of this ellipse (i.e., the longest and shortest directions) correspond to the magnitude of the *eigenvalues* λ_1 and λ_2 of the matrix. The eigenvalues also have a sign, which relates to whether the matrix reflects the inputs about the origin. The same idea applies in higher dimensions. A $D-$dimensional spheroid is mapped by a $D \times D$ matrix $\mathbf{A}$ to a D-dimensional ellipsoid. The radii of the D principal axes of this ellipsoid determine the magnitude of the eigenvalues.

The *spectral norm* of a square matrix is the largest absolute eigenvalue. It captures the largest possible change in magnitude when the matrix is applied to a vector of unit length. As such, it tells us about the Lipschitz constant of the transformation. The set of eigenvalues is sometimes called the *eigenspectrum* and tells us about the magnitude of the scaling applied by the matrix across all directions. This information can be summarized using the *determinant* and *trace* of the matrix.

B.3.8 Determinant and trace

Every square matrix $\mathbf{A}$ has a scalar associated with it called the determinant and denoted by $|\mathbf{A}|$ or $\det[\mathbf{A}]$, which is the product of the eigenvalues. It is hence related to the average scaling applied by the matrix for different inputs. Matrices with small absolute determinants tend to decrease the norm of vectors upon multiplication. Matrices with large absolute determinants tend to increase the norm. If a matrix is *singular*, the determinant will be zero, and there will be at least one direction in space that is mapped

to the origin when the matrix is applied. Determinants of matrix expressions obey the following rules:

$$\begin{aligned} |\mathbf{A}^T| &= |\mathbf{A}| \\ |\mathbf{AB}| &= |\mathbf{A}||\mathbf{B}| \\ |\mathbf{A}^{-1}| &= 1/|\mathbf{A}|. \end{aligned} \tag{B.14}$$

The trace of a square matrix is the sum of the diagonal values (the matrix itself need not be diagonal) or the sum of the eigenvalues. Traces obey these rules:

$$\begin{aligned} \text{trace}[\mathbf{A}^\mathbf{T}] &= \text{trace}[\mathbf{A}] \\ \text{trace}[\mathbf{AB}] &= \text{trace}[\mathbf{BA}] \\ \text{trace}[\mathbf{A}+\mathbf{B}] &= \text{trace}[\mathbf{A}] + \text{trace}[\mathbf{B}] \\ \text{trace}[\mathbf{ABC}] &= \text{trace}[\mathbf{BCA}] = \text{trace}[\mathbf{CAB}], \end{aligned} \tag{B.15}$$

where in the last relation, the trace is invariant for cyclic permutations only, so in general, $\text{trace}[\mathbf{ABC}] \neq \text{trace}[\mathbf{BAC}]$.

B.4 Special types of matrix

Calculating the inverse of a square matrix $\mathbf{A} \in \mathbb{R}^{D\times D}$ has a complexity of $\mathcal{O}[D^3]$, as does the computation of the determinant. However, for some matrices with special properties, these computations can be more efficient.

B.4.1 Diagonal matrices

A *diagonal matrix* has zeros everywhere except on the principal diagonal. If these diagonal entries are all non-zero, the inverse is also a diagonal matrix, with each diagonal entry d_{ii} replaced by $1/d_{ii}$. The determinant is the product of the values on the diagonal. A special case of this is the *identity matrix*, which has ones on the diagonal. Consequently, its inverse is also the identity matrix, and its determinant is one.

B.4.2 Triangular matrices

A *lower triangular matrix* contains only non-zero values on the principal diagonal and the positions below this. An *upper* triangular matrix contains only non-zero values on the principal diagonal and the positions above this. In both cases, the matrix can be inverted in $\mathcal{O}[D^2]$ (see problem 16.4), and the determinant is just the product of the values on the diagonal.

B.4.3 Orthogonal matrices

Orthogonal matrices represent rotations and reflections around the origin, so in figure B.3, the circle would be mapped to another circle of unit radius but rotated and possibly reflected. Accordingly, the eigenvalues must all have magnitude one, and the determinant must be either one or minus one. The inverse of an orthogonal matrix is its transpose, so $\mathbf{A}^{-1} = \mathbf{A}^T$.

B.4.4 Permutation matrices

A permutation matrix has exactly one non-zero entry in each row and column, and all of these entries take the value one. It is a special case of an orthogonal matrix, so its inverse is its own transpose, and its determinant is always one. As the name suggests, it has the effect of permuting the entries of a vector. For example:

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} b \\ c \\ a \end{bmatrix}. \tag{B.16}$$

B.4.5 Linear algebra

Linear algebra is the mathematics of linear functions, which have the form:

$$\mathrm{f}[z_1, z_2, \ldots z_D] = \phi_1 z_1 + \phi_2 z_2 + \ldots \phi_D z_D, \tag{B.17}$$

where $\phi_1, \ldots, \phi_D$ are parameters that define the function. We often add a constant term ϕ_0 to the right-hand side. This is technically an *affine* function but is commonly referred to as linear in machine learning. We adopt this convention throughout.

B.4.6 Linear equations in matrix form

Consider a collection of linear functions:

$$\begin{aligned} y_1 &= \phi_{10} + \phi_{11} z_1 + \phi_{12} z_2 + \phi_{13} z_3 \\ y_2 &= \phi_{20} + \phi_{21} z_1 + \phi_{22} z_2 + \phi_{23} z_3 \\ y_3 &= \phi_{30} + \phi_{31} z_1 + \phi_{32} z_2 + \phi_{33} z_3. \end{aligned} \tag{B.18}$$

These can be written in matrix form as:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} \phi_{10} \\ \phi_{20} \\ \phi_{30} \end{bmatrix} + \begin{bmatrix} \phi_{11} & \phi_{12} & \phi_{13} \\ \phi_{21} & \phi_{22} & \phi_{23} \\ \phi_{31} & \phi_{32} & \phi_{33} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix}, \tag{B.19}$$

or as $\mathbf{y} = \boldsymbol{\phi}_0 + \boldsymbol{\Phi}\mathbf{z}$ for short, where $y_i = \phi_{i0} + \sum_{j=1}^{3} \phi_{ij} z_j$.

B.5 Matrix calculus

Most readers of this book will be accustomed to the idea that if we have a function $y = \text{f}[x]$, we can compute the derivative $\partial y/\partial x$, and this represents how y changes when we make a small change in x. This idea extends to functions $y = \text{f}[\mathbf{x}]$ mapping a vector $\mathbf{x}$ to a scalar y, functions $\mathbf{y} = \mathbf{f}[\mathbf{x}]$ mapping a vector $\mathbf{x}$ to a vector $\mathbf{y}$, functions $\mathbf{y} = \mathbf{f}[\mathbf{X}]$ mapping a matrix $\mathbf{X}$ to a vector $\mathbf{y}$, and so on. The rules of *matrix calculus* help us compute derivatives of these quantities. The derivatives take the following forms:

- For a function $y = \text{f}[\mathbf{x}]$ where $y \in \mathbb{R}$ and $\mathbf{x} \in \mathbb{R}^D$, the derivative $\partial y/\partial \mathbf{x}$ is also a D-dimensional vector, where the i^{th} element is computed as $\partial y/\partial x_i$.
- For a function $\mathbf{y} = \mathbf{f}[\mathbf{x}]$ where $\mathbf{y} \in \mathbb{R}^{D_y}$ and $\mathbf{x} \in \mathbb{R}^{D_x}$, the derivative $\partial \mathbf{y}/\partial \mathbf{x}$ is a $D_x \times D_y$ matrix where element (i, j) contains the derivative $\partial y_j/\partial x_i$. This is known as a *Jacobian* and is sometimes written as $\nabla_{\mathbf{x}}\mathbf{y}$ in other documents.
- For a function $\mathbf{y} = \mathbf{f}[\mathbf{X}]$ where $\mathbf{y} \in \mathbb{R}^{D_y}$ and $\mathbf{X} \in \mathbb{R}^{D_1 \times D_2}$, the derivative $\partial \mathbf{y}/\partial \mathbf{X}$ is a 3D tensor containing the derivatives $\partial y_i/\partial x_{jk}$.

Often these matrix and vector derivatives have superficially similar forms to the scalar case. For example, we have:

$$y = ax \quad \longrightarrow \quad \frac{\partial y}{\partial x} = a, \tag{B.20}$$

and

$$\mathbf{y} = \mathbf{A}\mathbf{x} \quad \longrightarrow \quad \frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \mathbf{A}^T. \tag{B.21}$$

Appendix C

Probability

Probability is critical to deep learning. In supervised learning, deep networks implicitly rely on a probabilistic formulation of the loss function. In unsupervised learning, generative models aim to produce samples that are drawn from the same probability distribution as the training data. Reinforcement learning occurs within Markov decision processes, and these are defined in terms of probability distributions. This appendix provides a primer for probability as used in machine learning.

C.1 Random variables and probability distributions

A random variable x denotes a quantity that is uncertain. It may be *discrete* (take only certain values, for example integers) or *continuous* (take any value on a continuum, for example real numbers). If we observe several instances of a random variable x, it will take different values, and the relative propensity to take different values is described by a *probability distribution* $Pr(x)$.

For a discrete variable, this distribution associates a *probability* $Pr(x=k) \in [0,1]$ with each potential outcome k, and the sum of these probabilities is one. For a continuous variable, there is a non-negative *probability density* $Pr(x=a) \geq 0$ associated with each value a in the domain of x, and the integral of this probability density function (PDF) over this domain must be one. This density can be greater than one for any point a. From here on, we assume that the random variables are continuous. The ideas are exactly the same for discrete distributions but with sums replacing integrals.

C.1.1 Joint probability

Consider the case where we have two random variables x and y. The *joint distribution* $Pr(x,y)$ tells us about the propensity that x and y take particular combinations of values (figure C.1a). Now there is a non-negative probability density $Pr(x=a, y=b)$ associated with each pair of values $x=a$ and $y=b$ and this must satisfy:

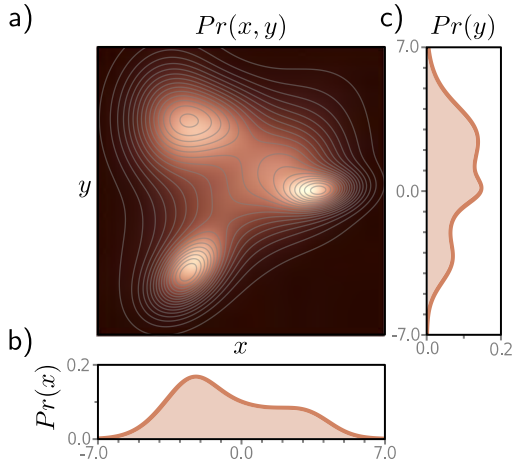


Figure C.1 Joint and marginal distributions. a) The joint distribution $Pr(x, y)$ captures the propensity of variables x and y to take different combinations of values. Here, the probability density is represented by the color map, so brighter positions are more probable. For example, the combination $x=6, y=6$ is much less likely to be observed than the combination $x = 5, y = 0$. b) The marginal distribution $Pr(x)$ of variable x can be recovered by integrating over y. c) The marginal distribution $Pr(y)$ of variable y can be recovered by integrating over x.

$$\iint Pr(x, y) \cdot dxdy = 1. \tag{C.1}$$

This idea extends to more than two variables, so the joint density of x, y, and z is written as $Pr(x, y, z)$. Sometimes, we store multiple random variables in a vector $\mathbf{x}$, and we write their joint density as $Pr(\mathbf{x})$. Extending this, we can write the joint density of all of the variables in two vectors $\mathbf{x}$ and $\mathbf{y}$ as $Pr(\mathbf{x}, \mathbf{y})$.

C.1.2 Marginalization

If we know the joint distribution $Pr(x, y)$ over two variables, we can recover the *marginal* distributions $Pr(x)$ and $Pr(y)$ by integrating over the other variable (figure C.1b-c):

$$\begin{aligned} \int Pr(x, y) \cdot dx &= Pr(y) \\ \int Pr(x, y) \cdot dy &= Pr(x). \end{aligned} \tag{C.2}$$

This process is called *marginalization* and has the interpretation that we are computing the distribution of one variable *regardless* of the value the other one took. The idea of marginalization extends to higher dimensions, so if we have a joint distribution $Pr(x, y, z)$, we can recover the joint distribution $Pr(x, z)$ by integrating over y.

C.1.3 Conditional probability and likelihood

The *conditional probability* $Pr(x|y)$ is the probability of variable x taking a certain value, assuming we know the value of y. The vertical line is read as the English word "given,"

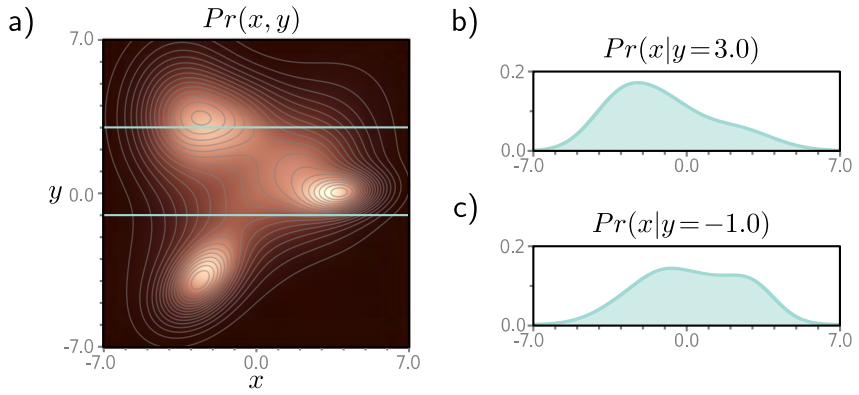


Figure C.2 Conditional distributions. a) Joint distribution $Pr(x, y)$ of variables x and y. b) The conditional probability $Pr(x|y = 3.0)$ of variable x, given that y takes the value 3.0, is found by taking the horizontal "slice" $Pr(x, y = 3.0)$ of the joint probability (top cyan line in panel a), and dividing this by the total area $Pr(y = 3.0)$ in that slice so that it forms a valid probability distribution that integrates to one. c) The joint probability $Pr(x, y=-1.0)$ is found similarly using the slice at $y=-1.0$.

so $Pr(x|y)$ is the probability of x given y. The conditional probability $Pr(x|y)$ can be found by taking a slice through the joint distribution $Pr(x, y)$ for a fixed y. This slice is then divided by the probability of that value y occurring (the total area under the slice) so that the conditional distribution sums to one (figure C.2):

$$Pr(x|y) = \frac{Pr(x, y)}{Pr(y)}. \tag{C.3}$$

Similarly,

$$Pr(y|x) = \frac{Pr(x, y)}{Pr(x)}. \tag{C.4}$$

When we consider the conditional probability $Pr(x|y)$ as a function of x, it must sum to one. When we consider the same quantity $Pr(x|y)$ as a function of y, it is termed the *likelihood* of x given y and does not have to sum to one.

C.1.4 Bayes' rule

From equations C.3 and C.4, we get two expressions for the joint probability $Pr(x, y)$:

$$Pr(x, y) = Pr(x|y)Pr(y) = Pr(y|x)Pr(x), \tag{C.5}$$

which we can rearrange to get:

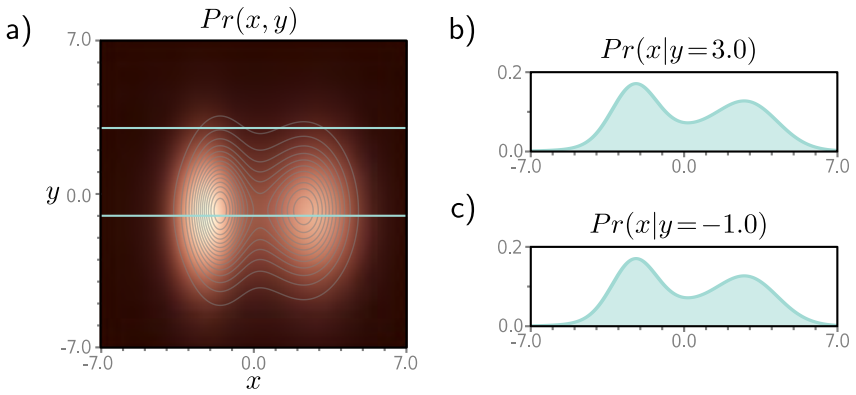


Figure C.3 Independence. a) When two variables x and y are independent, the joint distribution factors into the product of marginal distributions, so $Pr(x,y) = Pr(x)Pr(y)$. Independence implies that knowing the value of one variable tells us nothing about the other. b–c) Accordingly, all of the conditional distributions $Pr(x|y=\bullet)$ are the same and are equal to the marginal distribution $Pr(x)$.

$$Pr(x|y) = \frac{Pr(y|x)Pr(x)}{Pr(y)}. \tag{C.6}$$

This expression relates the conditional probability $Pr(x|y)$ of x given y to the conditional probability $Pr(y|x)$ of y given x and is known as *Bayes' rule*.

Each term in this Bayes' rule has a name. The term $Pr(y|x)$ is the *likelihood* of y given x, and the term $Pr(x)$ is the *prior probability* of x. The denominator $Pr(y)$ is known as the *evidence*, and the left-hand side $Pr(x|y)$ is termed the *posterior probability* of x given y. The equation maps from the prior $Pr(x)$ (what we know about x before observing y) to the posterior $Pr(x|y)$ (what we know about x after observing y).

C.1.5 Independence

If the value of the random variable y tells us nothing about x and vice-versa, we say that x and y are *independent*, and we can write $Pr(x|y) = Pr(x)$ and $Pr(y|x) = Pr(y)$. It follows that all of the conditional distributions $Pr(y|x=\bullet)$ are identical, as are the conditional distributions $Pr(x|y=\bullet)$.

Starting from the first expression for the joint probability in equation C.5, we see that the joint distribution becomes the product of the marginal distributions:

$$Pr(x,y) \quad = \quad Pr(x|y)Pr(y) = Pr(x)Pr(y) \tag{C.7}$$

when the variables are independent (figure C.3).

C.2 Expectation

Consider a function $\mathrm{f}[x]$ and a probability distribution $Pr(x)$ defined over x. The *expected value* of a function $\mathrm{f}[\bullet]$ of a random variable x with respect to the probability distribution $Pr(x)$ is defined as:

$$\mathbb{E}_x\big[\mathrm{f}[x]\big] = \int \mathrm{f}[x] Pr(x) dx. \tag{C.8}$$

As the name suggests, this is the expected or average value of $\mathrm{f}[x]$ after taking into account the probability of seeing different values of x. This idea generalizes to functions $\mathbf{f}[\bullet, \bullet]$ of more than one random variable:

$$\mathbb{E}_{x,y}\big[\mathrm{f}[x,y]\big] = \iint \mathrm{f}[x,y] Pr(x,y) dx dy. \tag{C.9}$$

An expectation is always taken with respect to a distribution over one or more variables. However, we don't usually make this explicit when the choice of distribution is obvious and write $\mathbb{E}[\mathrm{f}[x]]$ instead of $\mathbb{E}_x[\mathrm{f}[x]]$.

If we drew a large number I of samples $\{x_i\}_{i=1}^I$ from $Pr(x)$, calculated $\mathrm{f}[x_i]$ for each sample and took the average of these values, the result would approximate the expectation $\mathbb{E}[\mathrm{f}[x]]$ of the function:

$$\mathbb{E}_x\big[\mathrm{f}[x]\big] \approx \frac{1}{I}\sum_{i=1}^{I} \mathrm{f}[x_i]. \tag{C.10}$$

C.2.1 Rules for manipulating expectations

There are four rules for manipulating expectations:

$$\begin{aligned}
\mathbb{E}\big[k\big] &= k \\
\mathbb{E}\big[k \cdot \mathrm{f}[x]\big] &= k \cdot \mathbb{E}\big[\mathrm{f}[x]\big] \\
\mathbb{E}\big[\mathrm{f}[x] + \mathrm{g}[x]\big] &= \mathbb{E}\big[\mathrm{f}[x]\big] + \mathbb{E}\big[\mathrm{g}[x]\big] \\
\mathbb{E}_{x,y}\big[\mathrm{f}[x] \cdot \mathrm{g}[y]\big] &= \mathbb{E}_x\big[\mathrm{f}[x]\big] \cdot \mathbb{E}_y\big[\mathrm{g}[y]\big] \qquad \text{if } x, y \text{ independent,}
\end{aligned} \tag{C.11}$$

where k is an arbitrary constant. These are proven below for the continuous case.

Rule 1: The expectation $\mathbb{E}[k]$ of a constant value k is just k.

$$\begin{aligned}
\mathbb{E}\big[k\big] &= \int k \cdot Pr(x) dx \\
&= k \cdot \int Pr(x) dx \\
&= k.
\end{aligned}$$

Rule 2: The expectation $\mathbb{E}[k \cdot \mathrm{f}[x]]$ of a constant k times a function of the variable x is k times the expectation $\mathbb{E}[\mathrm{f}[x]]$ of the function:

$$\begin{aligned}\mathbb{E}\big[k \cdot \mathrm{f}[x]\big] &= \int k \cdot \mathrm{f}[x] Pr(x) dx \\ &= k \cdot \int \mathrm{f}[x] Pr(x) dx \\ &= k \cdot \mathbb{E}\big[\mathrm{f}[x]\big].\end{aligned}$$

Rule 3: The expectation of a sum $\mathbb{E}[\mathrm{f}[x] + \mathrm{g}[x]]$ of terms is the sum $\mathbb{E}[\mathrm{f}[x]] + \mathbb{E}[\mathrm{g}[x]]$ of the expectations:

$$\begin{aligned}\mathbb{E}\big[\mathrm{f}[x] + \mathrm{g}[x]\big] &= \int (\mathrm{f}[x] + \mathrm{g}[x]) \cdot Pr(x) dx \\ &= \int \big(\mathrm{f}[x] \cdot Pr(x) + \mathrm{g}[x] \cdot Pr(x)\big) dx \\ &= \int \mathrm{f}[x] \cdot Pr(x) dx + \int \mathrm{g}[x] \cdot Pr(x) dx \\ &= \mathbb{E}\big[\mathrm{f}[x]\big] + \mathbb{E}\big[\mathrm{g}[x]\big].\end{aligned}$$

Rule 4: The expectation of a product $\mathbb{E}[\mathrm{f}[x] \cdot \mathrm{g}[y]]$ of terms is the product $\mathbb{E}[\mathrm{f}[x]] \cdot \mathbb{E}[\mathrm{g}[y]]$ if x and y are independent.

$$\begin{aligned}\mathbb{E}\big[\mathrm{f}[x] \cdot \mathrm{g}[y]\big] &= \iint \mathrm{f}[x] \cdot \mathrm{g}[y] Pr(x, y) dx dy \\ &= \iint \mathrm{f}[x] \cdot \mathrm{g}[y] Pr(x) Pr(y) dx dy \\ &= \int \mathrm{f}[x] \cdot Pr(x) dx \int \mathrm{g}[y] \cdot Pr(y) dy \\ &= \mathbb{E}\big[\mathrm{f}[x]\big] \mathbb{E}\big[\mathrm{g}[y]\big],\end{aligned}$$

where we used the definition of independence (equation C.7) between the first two lines.

The four rules generalize to the multivariate case:

$$\begin{aligned}\mathbb{E}\big[\mathbf{A}\big] &= \mathbf{A} \\ \mathbb{E}\big[\mathbf{A} \cdot \mathbf{f}[\mathbf{x}]\big] &= \mathbf{A}\mathbb{E}\big[\mathbf{f}[\mathbf{x}]\big] \\ \mathbb{E}\big[\mathbf{f}[\mathbf{x}] + \mathbf{g}[\mathbf{x}]\big] &= \mathbb{E}\big[\mathbf{f}[\mathbf{x}]\big] + \mathbb{E}\big[\mathbf{g}[\mathbf{x}]\big] \\ \mathbb{E}_{\mathbf{x},\mathbf{y}}\big[\mathbf{f}[\mathbf{x}]^T \mathbf{g}[\mathbf{y}]\big] &= \mathbb{E}_{\mathbf{x}}\big[\mathbf{f}[\mathbf{x}]\big]^T \mathbb{E}_{\mathbf{y}}\big[\mathbf{g}[\mathbf{y}]\big] \qquad \text{if } \mathbf{x}, \mathbf{y} \text{ independent,}\end{aligned} \tag{C.12}$$

where now $\mathbf{A}$ is a constant matrix and $\mathbf{f}[\mathbf{x}]$ is a function of the vector $\mathbf{x}$ that returns a vector, and $\mathbf{g}[\mathbf{y}]$ is a function of the vector $\mathbf{y}$ that also returns a vector.

C.2.2 Mean, variance, and covariance

For some choices of function f[•], the expectation is given a special name. These quantities are often used to summarize the properties of complex distributions. For example, when f[x] = x, the resulting expectation $\mathbb{E}[x]$ is termed the *mean*, μ. It is a measure of the center of a distribution. Similarly, the expected squared deviation from the mean $\mathbb{E}[(x-\mu)^2]$ is termed the *variance*, σ^2. This is a measure of the spread of the distribution. The *standard deviation* σ is the positive square root of the variance. It also measures the spread of the distribution but has the merit that it is expressed in the same units as the variable x.

As the name suggests, the *covariance* $\mathbb{E}[(x-\mu_x)(y-\mu_y)]$ of two variables x and y measures the degree to which they co-vary. Here μ_x and μ_y represent the mean of the variables x and y, respectively. The covariance will be large when the variance of both variables is large and when the value of x tends to increase when the value of y increases.

If two variables are independent, then their covariance is zero. However, a covariance of zero does not imply independence. For example, consider a distribution $Pr(x,y)$ where the probability is uniformly distributed on a circle of radius one centered on the origin of the x, y plane. There is no tendency on average for x to increase when y increases or vice-versa. However, knowing the value of $x = 0$ tells us that y has an equal chance of taking the values ± 1, so the variables cannot be independent.

The covariances of multiple random variables stored in a column vector $\mathbf{x} \in \mathbb{R}^D$ can be represented by the $D \times D$ *covariance matrix* $\mathbb{E}[(\mathbf{x}-\boldsymbol{\mu}_x)(\mathbf{x}-\boldsymbol{\mu}_x)^T]$, where the vector $\boldsymbol{\mu}_x$ contains the means $\mathbb{E}[\mathbf{x}]$. The element at position (i,j) of this matrix represents the covariance between variables x_i and x_j.

C.2.3 Variance identity

The rules of expectation (appendix C.2.1) can be used to prove the following identity that allows us to write the variance in a different form:

$$\mathbb{E}\big[(x-\mu)^2\big] = \mathbb{E}\big[x^2\big] - \mathbb{E}\big[x\big]^2. \tag{C.13}$$

Proof:

$$\begin{aligned}
\mathbb{E}\big[(x-\mu)^2\big] &= \mathbb{E}\big[x^2 - 2\mu x + \mu^2\big] \\
&= \mathbb{E}\big[x^2\big] - \mathbb{E}\big[2\mu x\big] + \mathbb{E}\big[\mu^2\big] \\
&= \mathbb{E}\big[x^2\big] - 2\mu \cdot \mathbb{E}\big[x\big] + \mu^2 \\
&= \mathbb{E}\big[x^2\big] - 2\mu^2 + \mu^2 \\
&= \mathbb{E}\big[x^2\big] - \mu^2 \\
&= \mathbb{E}\big[x^2\big] - \mathbb{E}\big[x\big]^2,
\end{aligned} \tag{C.14}$$

where we have used rule 3 between lines 1 and 2, rules 1 and 2 between lines 2 and 3, and the definition $\mu = \mathbb{E}[x]$ in the remaining two lines.

C.2.4 Standardization

Setting the mean of a random variable to zero and the variance to one is known as *standardization*. This is achieved using the transformation:

$$z = \frac{x-\mu}{\sigma}, \tag{C.15}$$

where μ is the mean of x and σ is the standard deviation.

Proof: The mean of the new distribution over z is given by:

$$\begin{aligned}
\mathbb{E}[z] &= \mathbb{E}\left[\frac{x-\mu}{\sigma}\right] \\
&= \frac{1}{\sigma}\mathbb{E}\big[x-\mu\big] \\
&= \frac{1}{\sigma}\left(\mathbb{E}\big[x\big]-\mathbb{E}\big[\mu\big]\right) \\
&= \frac{1}{\sigma}\left(\mu-\mu\right) = 0,
\end{aligned} \tag{C.16}$$

where again, we have used the four rules for manipulating expectations. The variance of the new distribution is given by:

$$\begin{aligned}
\mathbb{E}\big[(z-\mu_z)^2\big] &= \mathbb{E}\big[(z-\mathbb{E}[z])^2\big] \\
&= \mathbb{E}\big[z^2\big] \\
&= \mathbb{E}\left[\left(\frac{x-\mu}{\sigma}\right)^2\right] \\
&= \frac{1}{\sigma^2}\cdot\mathbb{E}[(x-\mu)^2] \\
&= \frac{1}{\sigma^2}\cdot\sigma^2 = 1.
\end{aligned} \tag{C.17}$$

By a similar argument, we can take a standardized variable z with mean zero and unit variance and convert it to a variable x with mean μ and variance σ^2 using:

$$x = \mu + \sigma z. \tag{C.18}$$

In the multivariate case, we can standardize a variable $\mathbf{x}$ with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ using:

$$\mathbf{z} = \boldsymbol{\Sigma}^{-1/2}(\mathbf{x}-\boldsymbol{\mu}). \tag{C.19}$$

The result will have a mean $\mathbb{E}[\mathbf{z}] = \mathbf{0}$ and an identity covariance matrix $\mathbb{E}[(\mathbf{z}-\mathbb{E}[\mathbf{z}])(\mathbf{z}-\mathbb{E}[\mathbf{z}])^T] = \mathbf{I}$. To reverse this process, we use:

$$\mathbf{x} = \boldsymbol{\mu} + \boldsymbol{\Sigma}^{1/2}\mathbf{z}. \tag{C.20}$$

C.3 Normal probability distribution

Probability distributions used in this book include the Bernoulli distribution (figure 5.6), categorical distribution (figure 5.9), Poisson distribution (figure 5.15), von Mises distribution (figure 5.13), and mixture of Gaussians (figures 5.14 and 17.1). However, the most common distribution in machine learning is the normal or Gaussian distribution.

C.3.1 Univariate normal distribution

A univariate normal distribution (figure 5.3) over scalar variable x has two parameters, the mean μ and the variance σ^2, and is defined as:

$$Pr(x) = \text{Norm}_x[\mu, \sigma^2] = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]. \tag{C.21}$$

Unsurprisingly, the mean $\mathbb{E}[x]$ of a normally distributed variable is given by the mean parameter μ and the variance $\mathbb{E}[(x - \mathbb{E}[x])^2]$ by the variance parameter σ^2. When the mean is zero and the variance is one, we refer to this as a *standard normal distribution*.

The shape of the normal distribution can be inferred from the following argument. The term $-(x-\mu)^2/2\sigma^2$ is a quadratic function that falls away from zero when $x = \mu$ at a rate that increases when σ becomes smaller. When we pass this through the exponential function (figure B.1), we get a bell-shaped curve, which has a value of one at $x = \mu$ and falls away to either side. Dividing by the constant $\sqrt{2\pi\sigma^2}$ ensures that the function integrates to one and is a valid distribution. It follows from this argument that the mean μ control the position of the center of the bell curve, and the square root σ of the variance (the standard deviation) controls the width of the bell curve.

C.3.2 Multivariate normal distribution

The multivariate normal distribution generalizes the normal distribution to describe the probability over a vector quantity $\mathbf{x}$ of length D. It is defined by a $D \times 1$ *mean vector* $\boldsymbol{\mu}$ and a symmetric positive definite $D \times D$ *covariance matrix* $\boldsymbol{\Sigma}$:

$$\text{Norm}_{\mathbf{x}}[\boldsymbol{\mu}, \boldsymbol{\Sigma}] = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left[-\frac{(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}{2}\right]. \tag{C.22}$$

The interpretation is similar to the univariate case. The quadratic term $-(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})/2$ returns a scalar that decreases as $\mathbf{x}$ grows further from the mean $\boldsymbol{\mu}$, at a rate that depends on the matrix $\boldsymbol{\Sigma}$. This is turned into a bell-curve shape by the exponential, and dividing by $(2\pi)^{D/2}|\Sigma|^{1/2}$ ensures that the distribution integrates to one.

The covariance matrix can take spherical, diagonal, and full forms:

$$\boldsymbol{\Sigma}_{spher} = \begin{bmatrix}\sigma^2 & 0\\ 0 & \sigma^2\end{bmatrix} \quad \boldsymbol{\Sigma}_{diag} = \begin{bmatrix}\sigma_1^2 & 0\\ 0 & \sigma_2^2\end{bmatrix} \quad \boldsymbol{\Sigma}_{full} = \begin{bmatrix}\sigma_{11}^2 & \sigma_{12}^2\\ \sigma_{21}^2 & \sigma_{22}^2\end{bmatrix}. \tag{C.23}$$

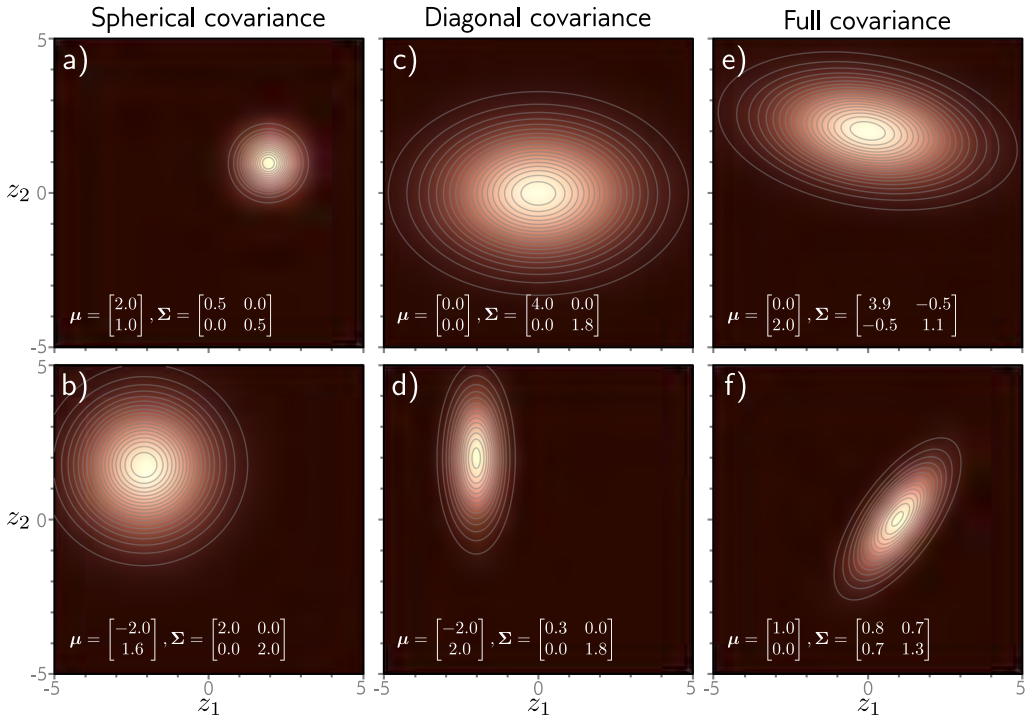


Figure C.4 Bivariate normal distribution. a–b) When the covariance matrix is a multiple of the diagonal matrix, the isocontours are circles, and we refer to this as spherical covariance. c–d) When the covariance is an arbitrary diagonal matrix, the isocontours are axis-aligned ellipses, and we refer to this as diagonal covariance e–f) When the covariance is an arbitrary symmetric positive definite matrix, the iso-contours are general ellipses, and we refer to this as full covariance.

In two dimensions (figure C.4), spherical covariances produce circular iso-density contours, and diagonal covariances produce ellipsoidal iso-contours that are aligned with the coordinate axes. Full covariances produce general ellipsoidal iso-density contours. When the covariance is spherical or diagonal, the individual variables are independent:

$$\begin{aligned} Pr(x_1, x_2) &= \frac{1}{2\pi\sqrt{|\boldsymbol{\Sigma}|}} \exp\left[-0.5\begin{pmatrix}x_1 & x_2\end{pmatrix}\boldsymbol{\Sigma}^{-1}\begin{pmatrix}x_1\\x_2\end{pmatrix}\right] \\ &= \frac{1}{2\pi\sigma_1\sigma_2} \exp\left[-0.5\begin{pmatrix}x_1 & x_2\end{pmatrix}\begin{pmatrix}\sigma_1^{-2} & 0\\0 & \sigma_2^{-2}\end{pmatrix}\begin{pmatrix}x_1\\x_2\end{pmatrix}\right] \\ &= \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left[-\frac{x_1^2}{2\sigma_1^2}\right] \cdot \frac{1}{\sqrt{2\pi\sigma_2^2}} \exp\left[-\frac{x_2^2}{2\sigma_2^2}\right] \\ &= Pr(x_1) \cdot Pr(x_2). \end{aligned} \tag{C.24}$$

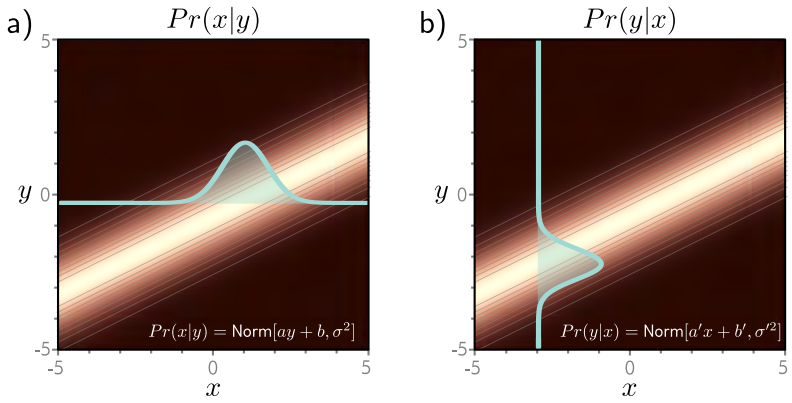


Figure C.5 Change of variables. a) The conditional distribution $Pr(x|y)$ is a normal distribution with constant variance and a mean that depends linearly on y. Cyan distribution shows one example for $y = -0.2$. b) This is proportional to the conditional probability $Pr(y|x)$, which is a normal distribution with constant variance and a mean that depends linearly on x. Cyan distribution shows one example for $x = -3$.

C.3.3 Product of two normal distributions

The product of two normal distributions is proportional to a third normal distribution according to the relation:

$$\text{Norm}_{\mathbf{x}}[\mathbf{a}, \mathbf{A}]\text{Norm}_{\mathbf{x}}[\mathbf{b}, \mathbf{B}] \;\propto\; \text{Norm}_{\mathbf{x}}\Big[(\mathbf{A}^{-1} + \mathbf{B}^{-1})^{-1}(\mathbf{A}^{-1}\mathbf{a} + \mathbf{B}^{-1}\mathbf{b}), (\mathbf{A}^{-1} + \mathbf{B}^{-1})^{-1}\Big].$$

This is easily proved by multiplying out the exponential terms and completing the square (see problem 18.5).

C.3.4 Change of variable

When the mean of a multivariate normal in $\mathbf{x}$ is a linear function $\mathbf{A}\mathbf{y} + \mathbf{b}$ of a second variable $\mathbf{y}$, this is proportional to another normal distribution in $\mathbf{y}$, where the mean is a linear function of $\mathbf{x}$:

$$\text{Norm}_{\mathbf{x}}\left[\mathbf{A}\mathbf{y} + \mathbf{b}, \boldsymbol{\Sigma}\right] \propto \text{Norm}_{\mathbf{y}}[(\mathbf{A}^T\boldsymbol{\Sigma}^{-1}\mathbf{A})^{-1}\mathbf{A}^T\boldsymbol{\Sigma}^{-1}(\mathbf{x} - \mathbf{b}), (\mathbf{A}^T\boldsymbol{\Sigma}^{-1}\mathbf{A})^{-1}]. \quad \text{(C.25)}$$

At first sight, this relation is rather opaque, but figure C.5 shows the case for scalar x and y, which is easy to understand. As for the previous relation, this can be proved by expanding the quadratic product in the exponential term and completing the square to make this a distribution in $\mathbf{y}$. (see problem 18.4).

C.4 Sampling

To sample from a univariate distribution $Pr(x)$, we first compute the cumulative distribution $\mathrm{F}[x]$ (the integral of $Pr(x)$). Then we draw a sample z^* from a uniform distribution over the range $[0, 1]$ and evaluate this against the inverse of the cumulative distribution, so the sample x^* is created as:

$$x^* = \mathrm{F}^{-1}[z^*]. \tag{C.26}$$

C.4.1 Sampling from normal distributions

The method above can be used to generate a sample x^* from a univariate standard normal distribution. A sample from a normal distribution with mean μ and variance σ^2 can then be created using equation C.18. Similarly, a sample $\mathbf{x}^*$ from a D-dimensional multivariate standard distribution can be created by independently sampling D univariate standard normal variables. A sample from a multivariate normal distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ can be then created using equation C.20.

C.4.2 Ancestral sampling

When the joint distribution can be factored into a series of conditional probabilities, we can generate samples using *ancestral sampling*. The basic idea is to generate a sample from the root variable(s) and then sample from the subsequent conditional distributions based on this instantiation. This process is known as *ancestral sampling* and is easiest to understand with an example. Consider a joint distribution over three variables, x, y, and z, where the distribution factors as:

$$Pr(x, y, z) = Pr(x)Pr(y|x)Pr(z|y). \tag{C.27}$$

To sample from this joint distribution, we first draw a sample x^* from $Pr(x)$. Then we draw a sample y^* from $Pr(y|x^*)$. Finally, we draw a sample z^* from $Pr(z|y^*)$.

C.5 Distances between probability distributions

Supervised learning can be framed in terms of minimizing the distance between the probability distribution implied by the model and the discrete probability distribution implied by the samples (section 5.7). Unsupervised learning can often be framed in terms of minimizing the distance between the probability distribution of real examples and the distribution of data from the model. In both cases, we need a measure of distance between two probability distributions. This section considers the properties of several different measures of distance between distributions (see also figure 15.8 for a discussion of the Wasserstein or earth mover's distance).

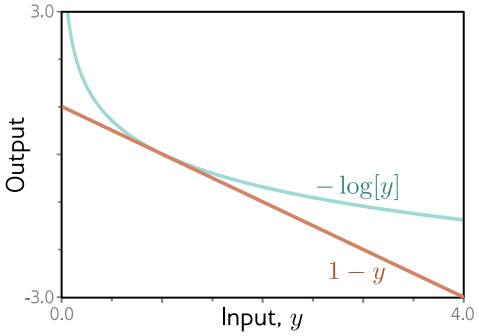


Figure C.6 Lower bound on negative logarithm. The function $1-y$ is always less than the function $-\log[y]$. This relation is used to show that the Kullback-Leibler divergence is always greater than or equal to zero.

C.5.1 Kullback-Leibler divergence

The most common measure of distance between probability distributions $p(x)$ and $q(x)$ is the *Kullback-Leibler* or KL divergence and is defined as:

$$D_{KL}\big[p(x)||q(x)\big] = \int p(x)\log\left[\frac{p(x)}{q(x)}\right]dx. \tag{C.28}$$

This distance is always greater than or equal to zero, which is easily demonstrated by noting that $-\log[y] \geq 1-y$ (figure C.6) so:

$$\begin{aligned} D_{KL}\Big[p(x)\Big|\Big|q(x)\Big] &= \int p(x)\log\left[\frac{p(x)}{q(x)}\right]dx \\ &= -\int p(x)\log\left[\frac{q(x)}{p(x)}\right]dx \\ &\geq \int p(x)\left(1-\frac{q(x)}{p(x)}\right)dx \\ &= \int p(x)-q(x)dx \\ &= 1-1=0. \end{aligned} \tag{C.29}$$

The KL divergence is infinite if there are places where $q(x)$ is zero but $p(x)$ is non-zero. This can lead to problems when we are minimizing a function based on this distance.

C.5.2 Jensen-Shannon divergence

The KL divergence is not symmetric (i.e., $D_{KL}[p(x)||q(x)] \neq D_{KL}[q(x)||p(x)]$). The Jensen-Shannon divergence is a measure of distance that is symmetric by construction:

$$D_{JS}\Big[p(x)\Big|\Big|q(x)\Big] = \frac{1}{2}D_{KL}\left[p(x)\Big|\Big|\frac{p(x)+q(x)}{2}\right] + \frac{1}{2}D_{KL}\left[q(x)\Big|\Big|\frac{p(x)+q(x)}{2}\right]. \tag{C.30}$$

It is the mean divergence of $p(x)$ and $q(x)$ to the average of the two distributions.

C.5.3 Fréchet distance

The Fréchet distance D_{FR} between two distributions $p(x)$ and $q(x)$ is given by:

$$D_{Fr}\Big[p(x)\Big|\Big|q(y)\Big] = \sqrt{\min_{\pi(x,y)}\left[\iint \pi(x,y)|x-y|^2 dxdy\right]}, \tag{C.31}$$

where $\pi(x,y)$ represents the set of joint distributions that are compatible with the marginal distributions $p(x)$ and $q(y)$. The Fréchet distance can also be formulated as a measure of the maximum distance between the cumulative probability curves.

C.5.4 Distances between normal distributions

Often we want to compute the distance between two multivariate normal distributions with means $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ and covariances $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$. In this case, various measures of distance can be written in closed form.

The KL divergence can be computed as:

$$\begin{aligned} D_{KL}\Big[\text{Norm}[\boldsymbol{\mu}_1,\boldsymbol{\Sigma}_1]\Big|\Big|\text{Norm}[\boldsymbol{\mu}_2,\boldsymbol{\Sigma}_2]\Big] \;\; = \;\; & \\ \frac{1}{2}\left(\log\left[\frac{|\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|}\right] - D + \text{tr}\left[\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1\right] + (\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)\boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)\right)&. \end{aligned} \tag{C.32}$$

where $tr[\bullet]$ is the trace of the matrix argument. The Fréchet/2-Wasserstein distance is given by:

$$D^2_{Fr/W_2}\Big[\text{Norm}[\boldsymbol{\mu}_1,\boldsymbol{\Sigma}_1]\Big|\Big|\text{Norm}[\boldsymbol{\mu}_2,\boldsymbol{\Sigma}_2]\Big] = |\boldsymbol{\mu}_1-\boldsymbol{\mu}_2|^2 + \text{tr}\left[\boldsymbol{\Sigma}_1+\boldsymbol{\Sigma}_2-2\left(\boldsymbol{\Sigma}_1\boldsymbol{\Sigma}_2\right)^{1/2}\right]. \tag{C.33}$$

Bibliography

Abdal, R., Qin, Y., & Wonka, P. (2019). Image2StyleGAN: How to embed images into the StyleGAN latent space? *IEEE/CVF International Conference on Computer Vision*, 4432–4441. 301

Abdal, R., Qin, Y., & Wonka, P. (2020). Image2StyleGAN++: How to edit the embedded images? *IEEE/CVF Computer Vision & Pattern Recognition*, 8296–8305. 301

Abdal, R., Zhu, P., Mitra, N. J., & Wonka, P. (2021). StyleFlow: Attribute-conditioned exploration of StyleGAN-generated images using conditional continuous normalizing flows. *ACM Transactions on Graphics (ToG)*, *40*(3), 1–21. 300, 322

Abdalla, M., & Abdalla, M. (2021). The grey hoodie project: Big tobacco, big tech, and the threat on academic integrity. *AAAI/ACM Conference on AI, Ethics, and Society*, 287–297. 434

Abdel-Hamid, O., Mohamed, A.-r., Jiang, H., & Penn, G. (2012). Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition. *IEEE International Conference on Acoustics, Speech and Signal Processing*, 4277–4280. 182

Abdelhamed, A., Brubaker, M. A., & Brown, M. S. (2019). Noise flow: Noise modeling with conditional normalizing flows. *IEEE/CVF International Conference on Computer Vision*, 3165–3173. 322

Abeßer, J., Mimilakis, S. I., Gräfe, R., Lukashevich, H., & Fraunhofer, I. (2017). Acoustic scene classification by combining autoencoder-based dimensionality reduction and convolutional neural networks. *Workshop on Detection and Classification of Acoustic Scenes and Events*, 7–11. 160

Abrahams, D. (2023). Let's talk about generative AI and fraud. *Forter Blog, March 27, 2023*. `https://www.forter.com/blog/lets-talk-about-generative-ai-and-fraud/`. 428

Abu-El-Haija, S., Perozzi, B., Kapoor, A., Alipourfard, N., Lerman, K., Harutyunyan, H., Ver Steeg, G., & Galstyan, A. (2019). MixHop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. *International Conference on Machine Learning*, 21–29. 263

Adler, J., & Lunz, S. (2018). Banach Wasserstein GAN. *Neural Information Processing Systems*, *31*, 6755–6764. 299

Agarwal, R., Schuurmans, D., & Norouzi, M. (2020). An optimistic perspective on offline reinforcement learning. *International Conference on Machine Learning*, 104–114. 398

Aggarwal, C. C., Hinneburg, A., & Keim, D. A. (2001). On the surprising behavior of distance metrics in high dimensional space. *International Conference on Database Theory*, 420–434. 135

Agüera y Arcas, B., Todorov, A., & Mitchell, M. (2018). Do algorithms reveal sexual orientation or just expose our stereotypes? Medium, Jan 11, 2018. `https://medium.com/@blaisea/do-algorithms-reveal-sexual-orientation-or-just-expose-our-stereotypes-d998fafdf477`. 431

Ahmed, N., & Wahed, M. (2016). The de-democratization of AI: Deep learning and the compute divide in artificial intelligence research. *arXiv:1606.06565*. 430

Ahmed, S., Mula, R. S., & Dhavala, S. S. (2020). A framework for democratizing AI. *arXiv:2001.00818*. 430

Ahmed, T. (2017). AI can tell if you're gay: Artificial intelligence predicts sexuality from one photo with startling accuracy. Newsweek, 8 Sept 2017. `https://www.newsweek.com/ai-can-tell-if-youre-gay-artificial-intelligence-predicts-sexuality-one-photo-661643`. 430

Aiken, M., & Park, M. (2010). The efficacy of round-trip translation for MT evaluation. *Translation Journal*, *14*(1). 160

Ainslie, J., Ontañón, S., Alberti, C., Cvicek, V., Fisher, Z., Pham, P., Ravula, A., Sanghai, S., Wang, Q., & Yang, L. (2020). ETC: Encoding long and structured inputs in transformers. *ACL Empirical Methods in Natural Language Processing*, 268–284. 237

Akers, J., Bansal, G., Cadamuro, G., Chen, C., Chen, Q., Lin, L., Mulcaire, P., Nandakumar, R., Rockett, M., Simko, L., Toman, J., Wu, T., Zeng, E., Zorn, B., & Roesner, F. (2018). Technology-enabled disinformation: Summary, lessons, and recommendations. *arXiv:1812.09383*. 427

Akuzawa, K., Iwasawa, Y., & Matsuo, Y. (2018). Expressive speech synthesis via modeling expressions with variational autoencoder. *INTERPSPEECH*, 3067–3071. 343

Ali, A., Touvron, H., Caron, M., Bojanowski, P., Douze, M., Joulin, A., Laptev, I., Neverova, N., Synnaeve, G., Verbeek, J., et al. (2021). XCiT: Cross-covariance image transformers. *Neural Information Processing Systems*, *34*, 20014–20027. 238

Allen, C., Smit, I., & Wallach, W. (2005). Artificial morality: Top-down, bottom-up, and hybrid approaches. *Ethics and Information Technology*, *7*, 149–155. 424

Allen-Zhu, Z., Li, Y., & Song, Z. (2019). A convergence theory for deep learning via over-parameterization. *International Conference on Machine Learning*, *97*, 242–252. 404

Alon, U., & Yahav, E. (2021). On the bottleneck of graph neural networks and its practical implications. *International Conference on Learning Representations*. 265

Alvarez, J. M., & Salzmann, M. (2016). Learning the number of neurons in deep networks. *Neural Information Processing Systems*, *29*, 2262–2270. 414

Amari, S.-I. (1998). Natural gradient works efficiently in learning. *Neural Computation*, *10*(2), 251–276. 397

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., & Mané, D. (2016). Concrete problems in AI safety. *arXiv:1606.06565*. 421

An, G. (1996). The effects of adding noise during backpropagation training on a generalization performance. *Neural Computation*, *8*(3), 643–674. 158

An, J., Huang, S., Song, Y., Dou, D., Liu, W., & Luo, J. (2021). ArtFlow: Unbiased image style transfer via reversible neural flows. *IEEE/CVF Computer Vision & Pattern Recognition*, 862–871. 322

Anderson, M., & Anderson, S. L. (2008). Ethical healthcare agents. *Advanced Computational Intelligence Paradigms in Healthcare 3. Studies in Computational Intelligence*, vol. 107, 233–257. 424

Andreae, J. (1969). Learning machines: A unified view. *Encyclopaedia of Linguistics, Information and Control*, 261–270. 396

Angwin, J., Larson, J., Mattu, S., & Kirchner, L. (2016). Machine bias: There's software used across the country to predict future criminals. and it's biased against blacks. ProPublica, May 23, 2016. `https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing`. 420

Ardizzone, L., Kruse, J., Lüth, C., Bracher, N., Rother, C., & Köthe, U. (2020). Conditional invertible neural networks for diverse image-to-image translation. *DAGM German Conference on Pattern Recognition*, 373–387. 322

Arjovsky, M., & Bottou, L. (2017). Towards principled methods for training generative adversarial networks. *International Conference on Learning Representations*. 283, 299

Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein generative adversarial networks. *International Conference on Machine Learning*, 214–223. 280, 299

Arkin, R. C. (2008a). Governing lethal behavior: Embedding ethics in a hybrid deliberative/reactive robot architecture—Part I: Motivation and philosophy. *ACM/IEEE International Conference on Human Robot Interaction*, 121–128. 424

Arkin, R. C. (2008b). Governing lethal behavior: Embedding ethics in a hybrid deliberative/reactive robot architecture—Part II: Formalization for ethical control. *Conference on Artificial General Intelligence*, 51–62. 424

Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lučić, M., & Schmid, C. (2021). ViVit: A video vision transformer. *IEEE/CVF International Conference on Computer Vision*, 6836–6846. 238

Arora, R., Basu, A., Mianjy, P., & Mukherjee, A. (2016). Understanding deep neural networks with rectified linear units. *arXiv:1611.01491*. 52

Arora, S., Ge, R., Liang, Y., Ma, T., & Zhang, Y. (2017). Generalization and equilibrium in generative adversarial nets (GANs). *International Conference on Machine Learning*, 224–232. 300

Arora, S., Li, Z., & Lyu, K. (2018). Theoretical analysis of auto rate-tuning by batch normalization. *arXiv:1812.03981*. 204

Arora, S., & Zhang, Y. (2017). Do GANs actually learn the distribution? An empirical study. *arXiv:1706.08224*. 300

Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, *34*(6), 26–38. 396

Asaro, P. (2012). On banning autonomous weapon systems: human rights, automation, and the dehumanization of lethal decision-making. *International Review of the Red Cross*, *94*(886), 687–709. 429

Atwood, J., & Towsley, D. (2016). Diffusion-convolutional neural networks. *Neural Information Processing Systems*, *29*, 1993–2001. 262

Aubret, A., Matignon, L., & Hassas, S. (2019). A survey on intrinsic motivation in reinforcement learning. *arXiv:1908.06976*. 398

Austin, J., Johnson, D. D., Ho, J., Tarlow, D., & van den Berg, R. (2021). Structured denoising diffusion models in discrete state-spaces. *Neural Information Processing Systems*, *34*, 17981–17993. 369

Awad, E., Dsouza, S., Kim, R., Schulz, J., Henrich, J., Shariff, A., Bonnefon, J.-F., & Rahwan, I. (2018). The moral machine experiment. *Nature*, *563*, 59–64. 424

Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. *arXiv:1607.06450*. 203

Bachlechner, T., Majumder, B. P., Mao, H., Cottrell, G., & McAuley, J. (2021). ReZero is all you need: Fast convergence at large depth. *Uncertainty in Artificial Intelligence*, 1352–1361. 238

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations*. 233, 235

Bahri, Y., Kadmon, J., Pennington, J., Schoenholz, S. S., Sohl-Dickstein, J., & Ganguli, S. (2020). Statistical mechanics of deep learning. *Annual Review of Condensed Matter Physics*, *11*, 501–528. 409, 410

Baldi, P., & Hornik, K. (1989). Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, *2*(1), 53–58. 410

Balduzzi, D., Frean, M., Leary, L., Lewis, J., Ma, K. W.-D., & McWilliams, B. (2017). The shattered gradients problem: If ResNets are the answer, then what is the question? *International Conference on Machine Learning*, 342–350. 188, 202, 203, 205

Bansal, A., Borgnia, E., Chu, H.-M., Li, J. S., Kazemi, H., Huang, F., Goldblum, M., Geiping, J., & Goldstein, T. (2022). Cold diffusion: Inverting arbitrary image transforms without noise. *arXiv:2208.09392*. 369

Bao, F., Li, C., Zhu, J., & Zhang, B. (2022). Analytic-DPM: An analytic estimate of the optimal reverse variance in diffusion probabilistic models. *International Conference on Learning Representations*. 369

Baranchuk, D., Rubachev, I., Voynov, A., Khrulkov, V., & Babenko, A. (2022). Label-efficient semantic segmentation with diffusion models. *International Conference on Learning Representations*. 369

Barber, D., & Bishop, C. (1997). Ensemble learning for multi-layer networks. *Neural Information Processing Systems*, *10*, 395–401. 159

Barocas, S., Hardt, M., & Narayanan, A. (2023). *Fairness and Machine Learning: Limitations and Opportunities*. MIT Press. 423

Barratt, S., & Sharma, R. (2018). A note on the inception score. *Workshop on Theoretical Foundations and Applications of Deep Generative Models*. 274

Barrett, D. G. T., & Dherin, B. (2021). Implicit gradient regularization. *International Conference on Learning Representations*. 157

Barrett, L. (2020). Ban facial recognition technologies for children — and for everyone else. *Boston University Journal of Science and Technology Law*, *26*(2), 223–285. 427

Barron, J. T. (2019). A general and adaptive robust loss function. *IEEE/CVF Computer Vision & Pattern Recognition*, 4331–4339. 73

Bartlett, P. L., Foster, D. J., & Telgarsky, M. J. (2017). Spectrally-normalized margin bounds for neural networks. *Neural Information Processing Systems*, vol. 30, 6240–6249. 156

Bartlett, P. L., Harvey, N., Liaw, C., & Mehrabian, A. (2019). Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear

neural networks. *Journal of Machine Learning Research*, *20*(1), 2285–2301. 134

Barto, A. G. (2013). Intrinsic motivation and reinforcement learning. *Intrinsically Motivated Learning in Natural and Artificial Systems*, 17–47. 398

Bau, D., Zhou, B., Khosla, A., Oliva, A., & Torralba, A. (2017). Network dissection: Quantifying interpretability of deep visual representations. *IEEE/CVF Computer Vision & Pattern Recognition*, 6541–6549. 184

Bau, D., Zhu, J.-Y., Wulff, J., Peebles, W., Strobelt, H., Zhou, B., & Torralba, A. (2019). Seeing what a GAN cannot generate. *IEEE/CVF International Conference on Computer Vision*, 4502–4511. 300

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., & Siskind, J. M. (2018). Automatic differentiation in machine learning: A survey. *Journal of Marchine Learning Research*, *18*, 1–43. 113

Bayer, M., Kaufhold, M.-A., & Reuter, C. (2022). A survey on data augmentation for text classification. *ACM Computing Surveys*, *55*(7), 1–39. 160

Behrmann, J., Grathwohl, W., Chen, R. T., Duvenaud, D., & Jacobsen, J.-H. (2019). Invertible residual networks. *International Conference on Machine Learning*, 573–582. 318, 323

Belinkov, Y., & Bisk, Y. (2018). Synthetic and natural noise both break neural machine translation. *International Conference on Learning Representations*. 160

Belkin, M., Hsu, D., Ma, S., & Mandal, S. (2019). Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, *116*(32), 15849–15854. 130, 134

Bellemare, M. G., Dabney, W., & Munos, R. (2017a). A distributional perspective on reinforcement learning. *International Conference on Machine Learning*, 449–458. 397

Bellemare, M. G., Danihelka, I., Dabney, W., Mohamed, S., Lakshminarayanan, B., Hoyer, S., & Munos, R. (2017b). The Cramer distance as a solution to biased Wasserstein gradients. *arXiv:1705.10743*. 299

Bellman, R. (1966). Dynamic programming. *Science*, *153*(3731), 34–37. 396

Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The long-document transformer. *arXiv:2004.05150*. 237

Bender, E. M., & Koller, A. (2020). Climbing towards NLU: On meaning, form, and understanding in the age of data. *Meeting of the Association for Computational Linguistics*, 5185–5198. 234

Bengio, Y., Ducharme, R., & Vincent, P. (2000). A neural probabilistic language model. *Neural Information Processing Systems*, *13*, 932–938. 274

Benjamin, R. (2019). *Race After Technology: Abolitionist Tools for the New Jim Code*. Polity. 433

Berard, H., Gidel, G., Almahairi, A., Vincent, P., & Lacoste-Julien, S. (2019). A closer look at the optimization landscapes of generative adversarial networks. *arXiv:1906.04848*. 299

Berger, P. (2019). MTA's initial foray into facial recognition at high speed is a bust. April 07, 2019. `https://www.wsj.com/articles/mtas-initial-foray-into-facial-recognition-at-high-speed-is-a-bust-11554642000`. 427

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, *13*(10), 281–305. 136

Bergstra, J. S., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for hyper-parameter optimization. *Neural Information Processing Systems*, vol. 24, 2546–2554. 136

Berk, R., Heidari, H., Jabbari, S., Kearns, M., & Roth, A. (2017). Fairness in criminal justice risk assessments: the state of the art. *Sociological Methods & Research*, *50*(1), 3–44. 422

Berner, C., Brockman, G., Chan, B., Cheung, V., Dębiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., et al. (2019). DOTA 2 with large scale deep reinforcement learning. *arXiv:1912.06680*. 396

Bertasius, G., Wang, H., & Torresani, L. (2021). Is space-time attention all you need for video understanding? *International Conference on Machine Learning*, 3, 813–824. 238

Beyer, K., Goldstein, J., Ramakrishnan, R., & Shaft, U. (1999). When is "nearest neighbor" meaningful? *International Conference on Database Theory*, 217–235. 135

Binns, R. (2018). Algorithmic accountability and public reason. *Philosophy & Technology*, *31*(4), 543–556. 13

Birhane, A., Isaac, W., Prabhakaran, V., Diaz, M., Elish, M. C., Gabriel, I., & Mohamed, S. (2022a). Power to the people? Opportunities and challenges for participatory AI. *Equity and*

Access in Algorithms, Mechanisms, and Optimization. 433

Birhane, A., Kalluri, P., Card, D., Agnew, W., Dotan, R., & Bao, M. (2022b). The values encoded in machine learning research. *ACM Conference on Fairness, Accountability, and Transparency*, 173–184. 431

Bishop, C. (1995). Regularization and complexity control in feed-forward networks. *International Conference on Artificial Neural Networks*, 141–148. 157, 158

Bishop, C. M. (1994). Mixture density networks. *Aston University Technical Report*. 73

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer. 15, 159

Bjorck, N., Gomes, C. P., Selman, B., & Weinberger, K. Q. (2018). Understanding batch normalization. *Neural Information Processing Systems*, *31*, 7705–7716. 204

Blum, A. L., & Rivest, R. L. (1992). Training a 3-node neural network is NP-complete. *Neural Networks*, *5*(1), 117–127. 401

Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D. (2015). Weight uncertainty in neural network. *International Conference on Machine Learning*, 1613–1622. 159

Bond-Taylor, S., Leach, A., Long, Y., & Willcocks, C. G. (2022). Deep generative modelling: A comparative review of VAEs, GANs, normalizing flows, energy-based and autoregressive models. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, *44*(11), 7327–7347. 274

Bontridder, N., & Poullet, Y. (2021). The role of artificial intelligence in disinformation. *Data & Policy*, *3*, E32. 427

Borji, A. (2022). Pros and cons of GAN evaluation measures: New developments. *Computer Vision & Image Understanding*, *215*, 103329. 274

Bornschein, J., Shabanian, S., Fischer, A., & Bengio, Y. (2016). Bidirectional Helmholtz machines. *International Conference on Machine Learning*, 2511–2519. 346

Boscaini, D., Masci, J., Rodolà, E., & Bronstein, M. (2016). Learning shape correspondence with anisotropic convolutional neural networks. *Neural Information Processing Systems*, *29*, 3189–3197. 265

Bottou, L. (2012). Stochastic gradient descent tricks. *Neural Networks: Tricks of the Trade: Second Edition*, 421–436. 91

Bottou, L., Curtis, F. E., & Nocedal, J. (2018). Optimization methods for large-scale machine learning. *SIAM Review*, *60*(2), 223–311. 91

Bottou, L., Soulié, F. F., Blanchet, P., & Liénard, J.-S. (1990). Speaker-independent isolated digit recognition: Multilayer perceptrons vs. dynamic time warping. *Neural Networks*, *3*(4), 453–465. 181

Boulemtafes, A., Derhab, A., & Challal, Y. (2020). A review of privacy-preserving techniques for deep learning. *Neurocomputing*, *384*, 21–45. 428

Bousselham, W., Thibault, G., Pagano, L., Machireddy, A., Gray, J., Chang, Y. H., & Song, X. (2021). Efficient self-ensemble framework for semantic segmentation. *arXiv:2111.13280*. 162

Bowman, S. R., & Dahl, G. E. (2021). What will it take to fix benchmarking in natural language understanding? *ACL Human Language Technologies*, 4843–4855. 234

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., & Bengio, S. (2015). Generating sentences from a continuous space. *ACL Conference on Computational Natural Language Learning*, 10–21. 343, 344, 345

Braverman, H. (1974). *Labor and monopoly capital: the degradation of work in the twentieth century*. Monthly Review Press. 429

Brock, A., Donahue, J., & Simonyan, K. (2019). Large scale GAN training for high fidelity natural image synthesis. *International Conference on Learning Representations*. 287, 299

Brock, A., Lim, T., Ritchie, J. M., & Weston, N. (2016). Neural photo editing with introspective adversarial networks. *International Conference on Learning Representations*. 345

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., & Shah, R. (1993). Signature verification using a "Siamese" time delay neural network. *Neural Information Processing Systems*, *6*, 737–744. 181

Bronstein, M. M., Bruna, J., Cohen, T., & Veličković, P. (2021). Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv:2104.13478*. 262

Broussard, M. (2018). *Artificial Unintelligence: How Computers Misunderstand the World*. The MIT Press. 433

Broussard, M. (2023). *More than a Glitch: Confronting Race, Gender, and Ability Bias in Tech*. The MIT Press. 433

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Neural Information Processing Systems*, *33*, 1877–1901. 9, 159, 234, 237, 422, 425

Brügger, R., Baumgartner, C. F., & Konukoglu, E. (2019). A partially reversible U-Net for memory-efficient volumetric image segmentation. *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 429–437. 322

Bruna, J., Zaremba, W., Szlam, A., & LeCun, Y. (2013). Spectral networks and locally connected networks on graphs. *International Conference on Learning Representations*. 262

Brynjolfsson, E., & McAfee, A. (2016). *The Second Machine Age: Work, Progress, and Prosperity in a Time of Brilliant Technologies*. W. W. Norton. 430

Bryson, A., Ho, Y.-C., & Siouris, G. (1979). Applied optimal control: Optimization, estimation, and control. *IEEE Transactions on Systems, Man & Cybernetics*, *9*, 366–367. 113

Bubeck, S., & Sellke, M. (2021). A universal law of robustness via isoperimetry. *Neural Information Processing Systems*, *34*, 28811–28822. 135, 416

Buciluǎ, C., Caruana, R., & Niculescu-Mizil, A. (2006). Model compression. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 535–541. 415

Bughin, J., Seong, J., Manyika, J., Chui, M., & Joshi, R. (2018). *Notes from the AI Frontier: Modelling the Impact of AI on the World Economy*. McKinsey Global Institute, Sept 4, 2018. 429

Buolamwini, J., & Gebru, T. (2018). Gender shades: Intersectional accuracy disparities in commercial gender classification. *Proceedings of Machine Learning Research*, *81*. 423

Burda, Y., Grosse, R. B., & Salakhutdinov, R. (2016). Importance weighted autoencoders. *International Conference on Learning Representations*. 73, 346

Buschjäger, S., & Morik, K. (2021). There is no double-descent in random forests. *arXiv:2111.04409*. 134

Cai, T., Luo, S., Xu, K., He, D., Liu, T.-y., & Wang, L. (2021). GraphNorm: A principled approach to accelerating graph neural network training. *International Conference on Machine Learning*, 1204–1215. 265

Calimeri, F., Marzullo, A., Stamile, C., & Terracina, G. (2017). Biomedical data augmentation using adversarial neural networks. *International Conference on Artificial Neural Networks*, 626–634. 159

Calo, R. (2018). Artificial intelligence policy: A primer and roadmap. *University of Bologna Law Review*, *3*(2), 180–218. 430

Cao, H., Tan, C., Gao, Z., Chen, G., Heng, P.-A., & Li, S. Z. (2022). A survey on generative diffusion model. *arXiv:2209.02646*. 369

Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., & Li, H. (2007). Learning to rank: From pairwise approach to listwise approach. *International Conference on Machine Learning*, 129–136. 73

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-end object detection with transformers. *European Conference on Computer Vision*, 213–229. 238

Carlini, N., Hayes, J., Nasr, M., Jagielski, M., Sehwag, V., Tramèr, F., Balle, B., Ippolito, D., & Wallace, E. (2023). Extracting training data from diffusion models. *arXiv:2301.13188*. 428

Carlini, N., Ippolito, D., Jagielski, M., Lee, K., Tramer, F., , & Zhang, C. (2022). Quantifying memorization across neural language models. *arXiv:2202.07646*. 428

Cauchy, A. (1847). Methode generale pour la resolution des systemes d'equations simultanees. *Comptes Rendus de l'Académie des Sciences*, *25*. 91

Cervantes, J.-A., López, S., Rodríguez, L.-F., Cervantes, S., Cervantes, F., & Ramos, F. (2019). Artificial moral agents: A survey of the current status. *Science and Engineering Ethics*, *26*, 501–532. 424

Ceylan, G., Anderson, I. A., & Wood, W. (2023). Sharing of misinformation is habitual, not just lazy or biased. *Proceedings of the National Academy of Sciences of the United States of America*, *120*(4). 432

Chami, I., Abu-El-Haija, S., Perozzi, B., Ré, C., & Murphy, K. (2020). Machine learning on graphs: A model and comprehensive taxonomy. *arXiv:2005.03675*. 261

Chang, B., Chen, M., Haber, E., & Chi, E. H. (2019a). AntisymmetricRNN: A dynamical system view on recurrent neural networks. *International Conference on Learning Representations*. 323

Chang, B., Meng, L., Haber, E., Ruthotto, L., Begert, D., & Holtham, E. (2018). Reversible

architectures for arbitrarily deep residual neural networks. *AAAI Conference on Artificial Intelligence*, 2811–2818. 323

Chang, Y.-L., Liu, Z. Y., Lee, K.-Y., & Hsu, W. (2019b). Free-form video inpainting with 3D gated convolution and temporal PatchGAN. *IEEE/CVF International Conference on Computer Vision*, 9066–9075. 181

Chaudhari, P., Choromanska, A., Soatto, S., LeCun, Y., Baldassi, C., Borgs, C., Chayes, J., Sagun, L., & Zecchina, R. (2019). Entropy-SGD: Biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment*, *12*, 124018. 158, 411

Chen, D., Mei, J.-P., Zhang, Y., Wang, C., Wang, Z., Feng, Y., & Chen, C. (2021a). Cross-layer distillation with semantic calibration. *AAAI Conference on Artificial Intelligence*, 7028–7036. 416

Chen, H., Wang, Y., Guo, T., Xu, C., Deng, Y., Liu, Z., Ma, S., Xu, C., Xu, C., & Gao, W. (2021b). Pre-trained image processing transformer. *IEEE/CVF Computer Vision & Pattern Recognition*, 12299–12310. 238

Chen, J., Ma, T., & Xiao, C. (2018a). FastGCN: Fast learning with graph convolutional networks via importance sampling. *International Conference on Learning Representations*. 264, 265

Chen, J., Zhu, J., & Song, L. (2018b). Stochastic training of graph convolutional networks with variance reduction. *International Conference on Machine Learning*, 941–949. 264

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., & Mordatch, I. (2021c). Decision transformer: Reinforcement learning via sequence modeling. *Neural Information Processing Systems*, *34*, 15084–15097. 398

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2018c). DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, *40*(4), 834——848. 181

Chen, M., Radford, A., Child, R., Wu, J., Jun, H., Luan, D., & Sutskever, I. (2020a). Generative pretraining from pixels. *International Conference on Machine Learning*, 1691–1703. 238

Chen, M., Wei, Z., Huang, Z., Ding, B., & Li, Y. (2020b). Simple and deep graph convolutional networks. *International Conference on Machine Learning*, 1725–1735. 266

Chen, N., Zhang, Y., Zen, H., Weiss, R. J., Norouzi, M., Dehak, N., & Chan, W. (2021d). WaveGrad 2: Iterative refinement for text-to-speech synthesis. *INTERSPEECH*, 3765–3769. 369

Chen, R. T., Behrmann, J., Duvenaud, D. K., & Jacobsen, J.-H. (2019). Residual flows for invertible generative modeling. *Neural Information Processing Systems*, *32*, 9913–9923. 324

Chen, R. T., Li, X., Grosse, R. B., & Duvenaud, D. K. (2018d). Isolating sources of disentanglement in variational autoencoders. *Neural Information Processing Systems*, *31*, 2615–2625. 343, 346

Chen, R. T., Rubanova, Y., Bettencourt, J., & Duvenaud, D. K. (2018e). Neural ordinary differential equations. *Neural Information Processing Systems*, *31*, 6572–6583. 324

Chen, T., Fox, E., & Guestrin, C. (2014). Stochastic gradient Hamiltonian Monte Carlo. *International Conference on Machine Learning*, 1683–1691. 159

Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020c). A simple framework for contrastive learning of visual representations. *International Conference on Machine Learning*, 1597–1607. 159

Chen, T., Xu, B., Zhang, C., & Guestrin, C. (2016a). Training deep nets with sublinear memory cost. *arXiv:1604.06174*. 114

Chen, W., Liu, T.-Y., Lan, Y., Ma, Z.-M., & Li, H. (2009). Ranking measures and loss functions in learning to rank. *Neural Information Processing Systems*, *22*, 315–323. 73

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., & Abbeel, P. (2016b). InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. *Neural Information Processing Systems*, *29*, 2172–2180. 291, 301

Chen, X., Kingma, D. P., Salimans, T., Duan, Y., Dhariwal, P., Schulman, J., Sutskever, I., & Abbeel, P. (2017). Variational lossy autoencoder. *International Conference on Learning Representations*. 345

Chen, Y.-C., Li, L., Yu, L., El Kholy, A., Ahmed, F., Gan, Z., Cheng, Y., & Liu, J. (2020d). UNITER: Universal image-text representation learning. *European Conference on Computer Vision*, 104–120. 238

Chiang, W.-L., Liu, X., Si, S., Li, Y., Bengio, S., & Hsieh, C.-J. (2019). Cluster-GCN: An efficient algorithm for training deep and large

graph convolutional networks. *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 257–266. 263, 264, 265

Child, R., Gray, S., Radford, A., & Sutskever, I. (2019). Generating long sequences with sparse transformers. *arXiv:1904.10509*. 237

Chintala, S., Denton, E., Arjovsky, M., & Matheiu, M. (2020). How to train a GAN? Tips and tricks to make GANs work. `https://github.com/soumith/ganhacks`. 299

Cho, K., van Merrienboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *ACL Workshop on Syntax, Semantics and Structure in Statistical Translation*, 103–111. 233

Choi, D., Shallue, C. J., Nado, Z., Lee, J., Maddison, C. J., & Dahl, G. E. (2019). On empirical comparisons of optimizers for deep learning. *arXiv:1910.05446*. 94, 410

Choi, J., Kim, S., Jeong, Y., Gwon, Y., & Yoon, S. (2021). ILVR: Conditioning method for denoising diffusion probabilistic models. *IEEE/CVF International Conference on Computer Vision*, 14347–14356. 370

Choi, J., Lee, J., Shin, C., Kim, S., Kim, H., & Yoon, S. (2022). Perception prioritized training of diffusion models. *IEEE/CVF Computer Vision & Pattern Recognition*, 11472–11481. 369

Choi, Y., Choi, M., Kim, M., Ha, J.-W., Kim, S., & Choo, J. (2018). StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation. *IEEE/CVF Computer Vision & Pattern Recognition*, 8789–8797. 301

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. *IEEE/CVF Computer Vision & Pattern Recognition*, 1251–1258. 405

Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., & LeCun, Y. (2015). The loss surfaces of multilayer networks. *International Conference on Artificial Intelligence and Statistics*. 405

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., et al. (2020). Rethinking attention with Performers. *International Conference on Learning Representations*. 236, 237

Chorowski, J., & Jaitly, N. (2017). Towards better decoding and language model integration in sequence to sequence models. *INTERSPEECH*, 523–527. 158

Chouldechova, A. (2017). Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big data*, *5*(2), 153–163. 422

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. (2022). PaLM: Scaling language modeling with pathways. *arXiv:2204.02311*. 234

Christian, B. (2020). *The Alignment Problem: Machine Learning and Human Values*. W. W. Norton. 421

Christiano, P., Shlegeris, B., & Amodei, D. (2018). Supervising strong learners by amplifying weak experts. *arXiv:1810.08575*. 398

Chu, X., Tian, Z., Wang, Y., Zhang, B., Ren, H., Wei, X., Xia, H., & Shen, C. (2021). Twins: Revisiting the design of spatial attention in vision transformers. *Neural Information Processing Systems*, *34*, 9355–9366. 238

Chung, H., Sim, B., & Ye, J. C. (2022). Come-closer-diffuse-faster: Accelerating conditional diffusion models for inverse problems through stochastic contraction. *IEEE/CVF Computer Vision & Pattern Recognition*, 12413–12422. 369

Chung, H., & Ye, J. C. (2022). Score-based diffusion models for accelerated MRI. *Medical Image Analysis*, *80*, 102479. 369

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *Deep Learning and Representation Workshop*. 233

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A. C., & Bengio, Y. (2015). A recurrent latent variable model for sequential data. *Neural Information Processing Systems*, *28*, 2980–2988. 344, 345

Çiçek, Ö., Abdulkadir, A., Lienkamp, S. S., Brox, T., & Ronneberger, O. (2016). 3D U-Net: Learning dense volumetric segmentation from sparse annotation. *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 424–432. 205

Clark, M. (2022). The engineer who claimed a Google AI is sentient has been fired. The Verge, July 22, 2022. `https://www.theverge.com/2022/7/22/23274958/google-ai-engineer-blake-lemoine-chatbot-lamda-2-sentience`. 234

Clevert, D.-A., Unterthiner, T., & Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (ELUs). *arXiv:1511.07289*. 38

Cohen, J. M., Kaur, S., Li, Y., Kolter, J. Z., & Talwalkar, A. (2021). Gradient descent on neural networks typically occurs at the edge of stability. *International Conference on Learning Representations*. 157

Cohen, N., Sharir, O., & Shashua, A. (2016). On the expressive power of deep learning: A tensor analysis. *PMLR Conference on Learning Theory*, 698–728. 53

Cohen, T., & Welling, M. (2016). Group equivariant convolutional networks. *International Conference on Machine Learning*, 2990–2999. 183

Collins, E., Bala, R., Price, B., & Susstrunk, S. (2020). Editing in style: Uncovering the local semantics of GANs. *IEEE/CVF Computer Vision & Pattern Recognition*, 5771–5780. 300

Conneau, A., Schwenk, H., Barrault, L., & Lecun, Y. (2017). Very deep convolutional networks for text classification. *Meeting of the Association for Computational Linguistics*, 1107–1116. 182

Constanza-Chock, S. (2020). *Design Justice: Community-Led Practices to Build the Worlds We Need*. Cambridge, MA: The MIT Press. 433

Cordonnier, J.-B., Loukas, A., & Jaggi, M. (2020). On the relationship between self-attention and convolutional layers. *International Conference on Learning Representations*. 236

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., & Schiele, B. (2016). The Cityscapes dataset for semantic urban scene understanding. *IEEE/CVF Computer Vision & Pattern Recognition*, 1877–1901. 6, 153

Coulombe, C. (2018). Text data augmentation made simple by leveraging NLP cloud APIs. *arXiv:1812.04718*. 160

Creel, K. A. (2020). Transparency in complex computational systems. *Philosophy of Science*, *87*(4), 568–589. 425, 435

Crenshaw, K. (1991). Mapping the margins: Intersectionality, identity politics, and violence against women of color. *Stanford Law Review*, *43*(6), 1241–1299. 423

Creswell, A., & Bharath, A. A. (2018). Inverting the generator of a generative adversarial network. *IEEE Transactions on Neural Networks and Learning Systems*, *30*(7), 1967–1974. 301

Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., & Bharath, A. A. (2018). Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, *35*(1), 53–65. 298

Cristianini, M., & Shawe-Taylor, J. (2000). *An Introduction to support vector machines*. CUP. 74

Croitoru, F.-A., Hondru, V., Ionescu, R. T., & Shah, M. (2022). Diffusion models in vision: A survey. *arXiv:2209.04747*. 369

Cubuk, E. D., Zoph, B., Mané, D., Vasudevan, V., & Le, Q. V. (2019). Autoaugment: Learning augmentation strategies from data. *IEEE/CVF Computer Vision & Pattern Recognition*, 113–123. 405

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, *2*(4), 303–314. 38

Dabney, W., Rowland, M., Bellemare, M., & Munos, R. (2018). Distributional reinforcement learning with quantile regression. *AAAI Conference on Artificial Intelligence*. 397

Dai, H., Dai, B., & Song, L. (2016). Discriminative embeddings of latent variable models for structured data. *International Conference on Machine Learning*, 2702–2711. 262

Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., & Wei, Y. (2017). Deformable convolutional networks. *IEEE/CVF International Conference on Computer Vision*, 764–773. 183

Daigavane, A., Balaraman, R., & Aggarwal, G. (2021). Understanding convolutions on graphs. Distill, `https://distill.pub/2021/understanding-gnns/`. 261

Danaher, J. (2019). *Automation and Utopia: Human Flourishing in a World without Work*. Harvard University Press. 430

Daniluk, M., Rocktäschel, T., Welbl, J., & Riedel, S. (2017). Frustratingly short attention spans in neural language modeling. *International Conference on Learning Representations*. 235

Danks, D., & London, A. J. (2017). Algorithmic bias in autonomous systems. *International Joint Conference on Artificial Intelligence*, 4691–4697. 422

Dao, D. (2021). *Awful AI*. Github. Retrieved January 17, 2023. `https://github.com/daviddao/awful-ai`. 14

Dar, Y., Muthukumar, V., & Baraniuk, R. G. (2021). A farewell to the bias-variance tradeoff? An overview of the theory of overparameterized machine learning. *arXiv:2109.02355*. 135

Das, H. P., Abbeel, P., & Spanos, C. J. (2019). Likelihood contribution based multi-scale architecture for generative flows. *arXiv:1908.01686*. 323

Dauphin, Y. N., Pascanu, R., Gülçehre, Ç., Cho, K., Ganguli, S., & Bengio, Y. (2014). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *Neural Information Processing Systems*, vol. 27, 2933–2941. 409, 410

David, H. (2015). Why are there still so many jobs? The history and future of workplace automation. *Journal of Economic Perspectives*, *29*(3), 3–30. 14

De, S., & Smith, S. (2020). Batch normalization biases residual blocks towards the identity function in deep networks. *Neural Information Processing Systems*, *33*, 19964–19975. 205

De Cao, N., & Kipf, T. (2018). MolGAN: An implicit generative model for small molecular graphs. *ICML Workshop on Theoretical Foundations and Applications of Deep Generative Models*. 299

Dechter, R. (1986). Learning while searching in constraint-satisfaction-problems. *AAAI Conference on Artificial Intelligence*, 178—183. 52

Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. *Neural Information Processing Systems*, *29*, 3837–3845. 262

Dehghani, M., Tay, Y., Gritsenko, A. A., Zhao, Z., Houlsby, N., Diaz, F., Metzler, D., & Vinyals, O. (2021). The benchmark lottery. *arXiv:2107.07002*. 234

Deisenroth, M. P., Faisal, A. A., & Ong, C. S. (2020). *Mathematics for machine learning*. Cambridge University Press. 15

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B*, *39*(1), 1–22. 346

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. *IEEE Computer Vision & Pattern Recognition*, 248–255. 181, 272

Denton, E. L., Chintala, S., Fergus, R., et al. (2015). Deep generative image models using a Laplacian pyramid of adversarial networks. *Neural Information Processing Systems*, *28*, 1486–1494. 300, 301

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2019). BERT: pre-training of deep bidirectional transformers for language understanding. *ACL Human Language Technologies*, 4171–4186. 159, 234

DeVries, T., & Taylor, G. W. (2017a). Dataset augmentation in feature space. *arXiv:1702.05538*. 158

DeVries, T., & Taylor, G. W. (2017b). Improved regularization of convolutional neural networks with Cutout. *arXiv:1708.04552*. 183

Dhariwal, P., & Nichol, A. (2021). Diffusion models beat GANs on image synthesis. *Neural Information Processing Systems*, *34*, 8780–8794. 367, 368, 370

Ding, M., Xiao, B., Codella, N., Luo, P., Wang, J., & Yuan, L. (2022). DaViT: Dual attention vision transformers. *European Conference on Computer Vision*, 74–92. 238

Dinh, L., Krueger, D., & Bengio, Y. (2015). NICE: Non-linear independent components estimation. *International Conference on Learning Representations Workshop*. 323

Dinh, L., Pascanu, R., Bengio, S., & Bengio, Y. (2017). Sharp minima can generalize for deep nets. *International Conference on Machine Learning*, 1019–1028. 411

Dinh, L., Sohl-Dickstein, J., & Bengio, S. (2016). Density estimation using Real NVP. *International Conference on Learning Representations*. 322, 323

Dinh, L., Sohl-Dickstein, J., Larochelle, H., & Pascanu, R. (2019). A RAD approach to deep mixture models. *ICLR Workshop on Deep Generative Models for Highly Structured Data*. 323

Dockhorn, T., Vahdat, A., & Kreis, K. (2022). Score-based generative modeling with critically-damped Langevin diffusion. *International Conference on Learning Representations*. 370

Doersch, C., Gupta, A., & Efros, A. A. (2015). Unsupervised visual representation learning by context prediction. *IEEE International Conference on Computer Vision*, 1422–1430. 159

Domingos, P. (2000). A unified bias-variance decomposition. *International Conference on Machine Learning*, 231–238. 133

Domke, J. (2010). Statistical machine learning. `https://people.cs.umass.edu/~domke/`. 116

Donahue, C., Lipton, Z. C., Balsubramani, A., & McAuley, J. (2018a). Semantically decomposing the latent spaces of generative adversarial

networks. *International Conference on Learning Representations*. 301

Donahue, C., McAuley, J., & Puckette, M. (2018b). Adversarial audio synthesis. *International Conference on Learning Representations*. 299, 301

Dong, X., Bao, J., Chen, D., Zhang, W., Yu, N., Yuan, L., Chen, D., & Guo, B. (2022). CSWin transformer: A general vision transformer backbone with cross-shaped windows. *IEEE/CVF Computer Vision & Pattern Recognition*, 12124–12134. 238

Dorta, G., Vicente, S., Agapito, L., Campbell, N. D., & Simpson, I. (2018). Structured uncertainty prediction networks. *IEEE/CVF Computer Vision & Pattern Recognition*, 5477–5485. 73, 340, 344

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations*. 234, 238

Dozat, T. (2016). Incorporating Nesterov momentum into Adam. *International Conference on Learning Representations — Workshop track*. 94

Draxler, F., Veschgini, K., Salmhofer, M., & Hamprecht, F. A. (2018). Essentially no barriers in neural network energy landscape. *International Conference on Machine Learning*, 1308–1317. 408, 409

Du, N., Huang, Y., Dai, A. M., Tong, S., Lepikhin, D., Xu, Y., Krikun, M., Zhou, Y., Yu, A. W., Firat, O., et al. (2022). GLaM: Efficient scaling of language models with mixture-of-experts. *International Conference on Machine Learning*, 5547–5569. 234

Du, S. S., Lee, J. D., Li, H., Wang, L., & Zhai, X. (2019a). Gradient descent finds global minima of deep neural networks. *International Conference on Machine Learning*, 1675–1685. 404, 405

Du, S. S., Zhai, X., Poczos, B., & Singh, A. (2019b). Gradient descent provably optimizes over-parameterized neural networks. *International Conference on Learning Representations*. 404

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, *12*, 2121–2159. 93

Dufter, P., Schmitt, M., & Schütze, H. (2021). Position information in transformers: An overview. *Computational Linguistics*, 1–31. 236

Dumoulin, V., Belghazi, I., Poole, B., Mastropietro, O., Lamb, A., Arjovsky, M., & Courville, A. (2017). Adversarially learned inference. *International Conference on Learning Representations*. 301, 345

Dumoulin, V., & Visin, F. (2016). A guide to convolution arithmetic for deep learning. *arXiv:1603.07285*. 180

Dupont, E., Doucet, A., & Teh, Y. W. (2019). Augmented neural ODEs. *Neural Information Processing Systems*, *32*, 3134–3144. 324

Durkan, C., Bekasov, A., Murray, I., & Papamakarios, G. (2019a). Cubic-spline flows. *ICML Invertible Neural Networks and Normalizing Flows*. 323

Durkan, C., Bekasov, A., Murray, I., & Papamakarios, G. (2019b). Neural spline flows. *Neural Information Processing Systems*, *32*, 7509–7520. 323

Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., & Adams, R. P. (2015). Convolutional networks on graphs for learning molecular fingerprints. *Neural Information Processing Systems*, *28*, 2224–2232. 262

D'Amour, A., Heller, K., Moldovan, D., Adlam, B., Alipanahi, B., Beutel, A., Chen, C., Deaton, J., Eisenstein, J., Hoffman, M. D., et al. (2020). Underspecification presents challenges for credibility in modern machine learning. *Journal of Machine Learning Research*, 1–61. 413

Ebrahimi, J., Rao, A., Lowd, D., & Dou, D. (2018). HotFlip: White-box adversarial examples for text classification. *Meeting of the Association for Computational Linguistics*, 31–36. 160

El Asri, L., & Prince, J. D., Simon (2020). Tutorial #6: Neural natural language generation – decoding algorithms. `https://www.borealisai.com/research-blogs/tutorial-6-neural-natural-language-generation-decoding-algorithms/`. 235

Eldan, R., & Shamir, O. (2016). The power of depth for feedforward neural networks. *PMLR Conference on Learning Theory*, 907–940. 53, 417

Elfwing, S., Uchibe, E., & Doya, K. (2018). Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, *107*, 3–11. 38

Erasmus, A., Brunet, T. D. P., & Fisher, E. (2021). What is interpretability? *Philosophy & Technology*, *34*, 833–862. 425

Eren, L., Ince, T., & Kiranyaz, S. (2019). A generic intelligent bearing fault diagnosis system using compact adaptive 1D CNN classifier. *Journal of Signal Processing Systems*, *91*(2), 179–189. 182

Erhan, D., Bengio, Y., Courville, A., & Vincent, P. (2009). Visualizing higher-layer features of a deep network. *Technical Report, University of Montreal*, *1341*(3). 184

Errica, F., Podda, M., Bacciu, D., & Micheli, A. (2019). A fair comparison of graph neural networks for graph classification. *International Conference on Learning Representations*. 262

Eslami, S., Heess, N., Weber, T., Tassa, Y., Szepesvari, D., Hinton, G. E., et al. (2016). Attend, infer, repeat: Fast scene understanding with generative models. *Neural Information Processing Systems*, *29*, 3225–3233. 344

Eslami, S. A., Jimenez Rezende, D., Besse, F., Viola, F., Morcos, A. S., Garnelo, M., Ruderman, A., Rusu, A. A., Danihelka, I., Gregor, K., et al. (2018). Neural scene representation and rendering. *Science*, *360*(6394), 1204–1210. 344

Esling, P., Masuda, N., Bardet, A., Despres, R., et al. (2019). Universal audio synthesizer control with normalizing flows. *International Conference on Digital Audio Effects*. 322

Esser, P., Rombach, R., & Ommer, B. (2021). Taming transformers for high-resolution image synthesis. *IEEE/CVF Computer Vision & Pattern Recognition*, 12873–12883. 301

Esteves, C., Allen-Blanchette, C., Zhou, X., & Daniilidis, K. (2018). Polar transformer networks. *International Conference on Learning Representations*. 183

Etmann, C., Ke, R., & Schönlieb, C.-B. (2020). iunets: Fully invertible U-Nets with learnable up-and downsampling. *IEEE International Workshop on Machine Learning for Signal Processing*. 322

Eubanks, V. (2018). *Automating Inequality: How High-Tech Tools Profile, Police, and Punish the Poor*. New York: St. Martin's Press. 433

Evans, K., de Moura, N., Chauvier, S., Chatila, R., & Dogan, E. (2020). Ethical decision making in autonomous vehicles: the AV ethics project. *Science and Engineering Ethics*, *26*(6), 3285–3312. 424

FAIR (2022). Human-level play in the game of Diplomacy by combining language models with strategic reasoning. *Science*, *378*(6624), 1067–1074. 396

Falbo, A., & LaCroix, T. (2022). Est-ce que vous compute? Code-switching, cultural identity, and AI. *Feminist Philosophy Quarterly*, *8*(3/4). 423

Falk, T., Mai, D., Bensch, R., Çiçek, Ö., Abdulkadir, A., Marrakchi, Y., Böhm, A., Deubner, J., Jäckel, Z., Seiwald, K., et al. (2019). U-Net: Deep learning for cell counting, detection, and morphometry. *Nature Methods*, *16*(1), 67–70. 199

Falkner, S., Klein, A., & Hutter, F. (2018). BOHB: Robust and efficient hyperparameter optimization at scale. *International Conference on Machine Learning*, 1437–1446. 136

Fallah, N., Gu, H., Mohammad, K., Seyyedsalehi, S. A., Nourijelyani, K., & Eshraghian, M. R. (2009). Nonlinear Poisson regression using neural networks: A simulation study. *Neural Computing and Applications*, *18*(8), 939–943. 74

Fan, A., Lewis, M., & Dauphin, Y. N. (2018). Hierarchical neural story generation. *Meeting of the Association for Computational Linguistics*, 889–898. 235

Fan, H., Xiong, B., Mangalam, K., Li, Y., Yan, Z., Malik, J., & Feichtenhofer, C. (2021). Multiscale vision transformers. *IEEE/CVF International Conference on Computer Vision*, 6824–6835. 238

Fan, K., Li, B., Wang, J., Zhang, S., Chen, B., Ge, N., & Yan, Z. (2020). Neural zero-inflated quality estimation model for automatic speech recognition system. *Interspeech*, 606–610. 73

Fang, F., Yamagishi, J., Echizen, I., & Lorenzo-Trueba, J. (2018). High-quality nonparallel voice conversion based on cycle-consistent adversarial network. *International Conference on Acoustics, Speech and Signal Processing*, 5279–5283. 299

Fang, Y., Liao, B., Wang, X., Fang, J., Qi, J., Wu, R., Niu, J., & Liu, W. (2021). You only look at one sequence: Rethinking transformer in vision through object detection. *Neural Information Processing Systems*, *34*, 26183–26197. 238

Farnia, F., & Ozdaglar, A. (2020). Do GANs always have Nash equilibria? *International Conference on Machine Learning*, 3029–3039. 299

Fawzi, A., Balog, M., Huang, A., Hubert, T., Romera-Paredes, B., Barekatain, M., Novikov,

A., R Ruiz, F. J., Schrittwieser, J., Swirszcz, G., et al. (2022). Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, *610*(7930), 47–53. 396

Fazelpour, S., & Danks, D. (2021). Algorithmic bias: Senses, sources, solutions. *Philosophy Compass*, *16*. 421, 422, 435

Fedus, W., Goodfellow, I., & Dai, A. M. (2018). MaskGAN: Better text generation via filling in the_. *International Conference on Learning Representations*. 299

Feng, S. Y., Gangal, V., Kang, D., Mitamura, T., & Hovy, E. (2020). GenAug: Data augmentation for finetuning text generators. *ACL Deep Learning Inside Out*, 29–42. 160

Feng, Z., Zhang, Z., Yu, X., Fang, Y., Li, L., Chen, X., Lu, Y., Liu, J., Yin, W., Feng, S., et al. (2022). ERNIE-ViLG 2.0: Improving text-to-image diffusion model with knowledge-enhanced mixture-of-denoising-experts. *arXiv:2210.15257*. 371

Fernandez, C. (2017). Can a computer tell if you're gay? Artificial intelligence system guesses your sexuality with 91% accuracy just by looking at a photo of your face. Daily Mail, 7 Sept, 2017. `https://www.dailymail.co.uk/sciencetech/article-4862676/Artificial-intelligence-tell-gay.html`. 430

Fernández-Madrigal, J.-A., & González, J. (2002). Multihierarchical graph search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(1), 103–113. 242

Fetscherin, M., Tantleff-Dunn, S., & Klumb, A. (2020). Effects of facial features and styling elements on perceptions of competence, warmth, and hireability of male professionals. *The Journal of Social Psychology*, *160*(3), 332–345. 427

Finlay, C., Jacobsen, J., Nurbekyan, L., & Oberman, A. M. (2020). How to train your neural ODE: The world of Jacobian and kinetic regularization. *International Conference on Machine Learning*, 3154–3164. 324

Fort, S., Hu, H., & Lakshminarayanan, B. (2019). Deep ensembles: A loss landscape perspective. *arXiv:1912.02757*. 158

Fort, S., & Jastrzębski, S. (2019). Large scale structure of neural network loss landscapes. *Neural Information Processing Systems*, vol. 32, 6706–6714. 408

Fort, S., & Scherlis, A. (2019). The Goldilocks zone: Towards better understanding of neural network loss landscapes. *AAAI Conference on Artificial Intelligence*, 3574–3581. 409, 410, 412, 413

Fortunato, M., Azar, M. G., Piot, B., Menick, J., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., et al. (2018). Noisy networks for exploration. *International Conference on Learning Representations*. 397

François-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G., Pineau, J., et al. (2018). An introduction to deep reinforcement learning. *Foundations and Trends in Machine Learning*, *11*(3-4), 219–354. 396

Frankle, J., & Carbin, M. (2019). The lottery ticket hypothesis: Finding sparse, trainable neural networks. *International Conference on Learning Representations*. 406, 415

Frankle, J., Dziugaite, G. K., Roy, D. M., & Carbin, M. (2020). Linear mode connectivity and the lottery ticket hypothesis. *International Conference on Machine Learning*, 3259–3269. 158, 408

Frankle, J., Schwab, D. J., & Morcos, A. S. (2021). Training BatchNorm and only BatchNorm: On the expressive power of random features in CNNs. *International Conference on Learning Representations*. 418

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, *55*(1), 119–139. 74

Frey, C. B. (2019). *The Technology Trap: Capital, Labour, and Power in the Age of Automation*. Princeton University Press. 430

Frey, C. B., & Osborne, M. A. (2017). The future of employment: How susceptible are jobs to computerisation? *Technological forecasting and social change*, *114*, 254–280. 430

Friedman, J. H. (1997). On bias, variance, 0/1—loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, *1*(1), 55–77. 133

Fujimoto, S., Hoof, H., & Meger, D. (2018). Addressing function approximation error in actor-critic methods. *International Conference on Machine Learning*, 1587–1596. 397

Fujimoto, S., Meger, D., & Precup, D. (2019). Off-policy deep reinforcement learning without exploration. *International Conference on Machine Learning*, 2052–2062. 398

Fukushima, K. (1969). Visual feature extraction by a multilayered network of analog threshold elements. *IEEE Transactions on Systems Science and Cybernetics*, *5*(4), 322–333. 37

Fukushima, K., & Miyake, S. (1982). Neocognitron: A self-organizing neural network model

for a mechanism of visual pattern recognition. *Competition and Cooperation in Neural Nets*, 267–285. 180

Gabriel, I. (2020). Artificial intelligence, values, and alignment. *Minds and Machines*, *30*, 411–437. 421

Gal, Y., & Ghahramani, Z. (2016). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. *International Conference on Machine Learning*, 1050—1059. 158

Gales, M. J. (1998). Maximum likelihood linear transformations for HMM-based speech recognition. *Computer Speech & Language*, *12*(2), 75–98. 160

Gales, M. J., Ragni, A., AlDamarki, H., & Gautier, C. (2009). Support vector machines for noise robust ASR. *2009 IEEE Workshop on Automatic Speech Recognition & Understanding*, 205–210. 160

Ganaie, M., Hu, M., Malik, A., Tanveer, M., & Suganthan, P. (2022). Ensemble deep learning: A review. *Engineering Applications of Artificial Intelligence*, *115*. 158

Gao, H., & Ji, S. (2019). Graph U-Nets. *International Conference on Machine Learning*, 2083–2092. 265

Gao, R., Song, Y., Poole, B., Wu, Y. N., & Kingma, D. P. (2021). Learning energy-based models by diffusion recovery likelihood. *International Conference on Learning Representations*. 370

Garg, R., Bg, V. K., Carneiro, G., & Reid, I. (2016). Unsupervised CNN for single view depth estimation: Geometry to the rescue. *European Conference on Computer Vision*, 740–756. 205

Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D., & Wilson, A. G. (2018). Loss surfaces, mode connectivity, and fast ensembling of DNNs. *Neural Information Processing Systems*, vol. 31, 8803—8812. 158, 408

Gastaldi, X. (2017a). Shake-shake regularization. *arXiv:1705.07485*. 203

Gastaldi, X. (2017b). Shake-shake regularization of 3-branch residual networks. 203

Gebru, T., Bender, E. M., McMillan-Major, A., & Mitchell, M. (2023). Statement from the listed authors of stochastic parrots on the "AI pause" letter. `https://www.dair-institute.org/blog/letter-statement-March2023`. 435

Gemici, M. C., Rezende, D., & Mohamed, S. (2016). Normalizing flows on Riemannian manifolds. *NIPS Workshop on Bayesian Deep Learning*. 324

Germain, M., Gregor, K., Murray, I., & Larochelle, H. (2015). MADE: Masked autoencoder for distribution estimation. *International Conference on Machine Learning*, 881–889. 323

Ghosh, A., Kulharia, V., Namboodiri, V. P., Torr, P. H., & Dokania, P. K. (2018). Multi-agent diverse generative adversarial networks. *IEEE/CVF Computer Vision & Pattern Recognition*, 8513–8521. 300

Gidaris, S., Singh, P., & Komodakis, N. (2018). Unsupervised representation learning by predicting image rotations. *International Conference on Learning Representations*. 159

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017). Neural message passing for quantum chemistry. *International Conference on Machine Learning*, 1263–1272. 262

Girdhar, R., Carreira, J., Doersch, C., & Zisserman, A. (2019). Video action transformer network. *IEEE/CVF Computer Vision & Pattern Recognition*, 244–253. 238

Girshick, R. (2015). Fast R-CNN. *IEEE International Conference on Computer Vision*, 1440–1448. 183

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *IEEE Computer Vision & Pattern Recognition*, 580–587. 183

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *International Conference on Artificial Intelligence and Statistics*, *9*, 249–256. 113, 183

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. *International Conference on Artificial Intelligence and Statistics*, 315–323. 37, 38

Goh, G. (2017). Why momentum really works. Distill, `http://distill.pub/2017/momentum`. 92

Goldberg, D. E. (1987). Simple genetic algorithms and the minimal deceptive problem. *Genetic Algorithms and Simulated Annealing*, 74–88. Morgan Kaufmann. 421

Gomez, A. N., Ren, M., Urtasun, R., & Grosse, R. B. (2017). The reversible residual network: Backpropagation without storing activations. *Neural Information Processing Systems*, *30*, 2214–2224. 114, 322, 323

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., & Aspuru-Guzik, A. (2018). Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, *4*(2), 268–276. 343, 344

Gong, S., Bahri, M., Bronstein, M. M., & Zafeiriou, S. (2020). Geometrically principled connections in graph neural networks. *IEEE/CVF Computer Vision & Pattern Recognition*, 11415–11424. 266

Goodfellow, I. (2016). Generative adversarial networks. *NIPS 2016 Tutorial*. 298

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. 15, 157

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial networks. *Communications of the ACM*, *63*(11), 139–144. 273, 298, 300

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2015a). Explaining and harnessing adversarial examples. *International Conference on Learning Representations*. 159, 413

Goodfellow, I. J., Vinyals, O., & Saxe, A. M. (2015b). Qualitatively characterizing neural network optimization problems. *International Conference on Learning Representations*. 407, 408

Goodin, D. (2023). ChatGPT is enabling script kiddies to write functional malware. ars Technica, June 1, 2023. `https://arstechnica.com/information-technology/2023/01/chatgpt-is-enabling-script-kiddies-to-write-functional-malware/`. 428

Gordon, G. J. (1995). Stable fitted reinforcement learning. *Neural Information Processing Systems*, *8*, 1052–1058. 396

Gori, M., Monfardini, G., & Scarselli, F. (2005). A new model for learning in graph domains. *IEEE International Joint Conference on Neural Networks*, 2005, 729–734. 262

Gouk, H., Frank, E., Pfahringer, B., & Cree, M. J. (2021). Regularisation of neural networks by enforcing Lipschitz continuity. *Machine Learning*, *110*(2), 393—416. 156

Goyal, A., Bochkovskiy, A., Deng, J., & Koltun, V. (2021). Non-deep networks. *arXiv:2110.07641*. 417

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., & He, K. (2018). Accurate, large minibatch SGD: Training ImageNet in 1 hour. *arXiv:1706.02677*. 92, 93, 237, 410

Graesser, L., & Keng, W. L. (2019). *Foundations of deep reinforcement learning*. Addison-Wesley Professional. 16, 396

Grathwohl, W., Chen, R. T., Bettencourt, J., Sutskever, I., & Duvenaud, D. (2019). Ffjord: Free-form continuous dynamics for scalable reversible generative models. *International Conference on Learning Representations*. 324

Grattarola, D., Zambon, D., Bianchi, F. M., & Alippi, C. (2022). Understanding pooling in graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*. 265

Green, B. (2019). "Good" isn't good enough. *NeurIPS Workshop on AI for Social Good*. 433

Green, B. (2022). Escaping the impossibility of fairness: From formal to substantive algorithmic fairness. *Philosophy & Technology*, *35*(90). 422

Greensmith, E., Bartlett, P. L., & Baxter, J. (2004). Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, *5*(9), 1471–1530. 397

Gregor, K., Besse, F., Jimenez Rezende, D., Danihelka, I., & Wierstra, D. (2016). Towards conceptual compression. *Neural Information Processing Systems*, *29*, 3549–3557. 343, 344

Gregor, K., Papamakarios, G., Besse, F., Buesing, L., & Weber, T. (2019). Temporal difference variational auto-encoder. *International Conference on Learning Representations*. 344

Grennan, L., Kremer, A., Singla, A., & Zipparo, P. (2022). *Why businesses need explainable AI—and how to deliver it*. McKinsey, September 29, 2022. `https://www.mckinsey.com/capabilities/quantumblack/our-insights/why-businesses-need-explainable-ai-and-how-to-deliver-it/`. 13

Greydanus, S. (2020). Scaling down deep learning. *arXiv:2011.14439*. 119

Griewank, A., & Walther, A. (2008). *Evaluating derivatives: Principles and techniques of algorithmic differentiation*. SIAM. 113

Gu, J., Kwon, H., Wang, D., Ye, W., Li, M., Chen, Y.-H., Lai, L., Chandra, V., & Pan, D. Z. (2022). Multi-scale high-resolution vision transformer for semantic segmentation. *IEEE/CVF Computer Vision & Pattern Recognition*, 12094–12103. 238

Guan, S., Tai, Y., Ni, B., Zhu, F., Huang, F., & Yang, X. (2020). Collaborative learning for faster StyleGAN embedding. *arXiv:2007.01758*. 301

Gui, J., Sun, Z., Wen, Y., Tao, D., & Ye, J. (2021). A review on generative adversarial networks: Algorithms, theory, and applications. *IEEE Transactions on Knowledge and Data Engineering*. 299

Guimaraes, G. L., Sanchez-Lengeling, B., Outeiral, C., Farias, P. L. C., & Aspuru-Guzik, A. (2017). Objective-reinforced generative adversarial networks (ORGAN) for sequence generation models. *arXiv:1705.10843*. 299

Gulrajani, I., Kumar, K., Ahmed, F., Taiga, A. A., Visin, F., Vazquez, D., & Courville, A. (2016). PixelVAE: A latent variable model for natural images. *International Conference on Learning Representations*. 299, 343, 344, 345

Ha, D., Dai, A., & Le, Q. V. (2017). Hypernetworks. *International Conference on Learning Representations*. 235

Haarnoja, T., Hartikainen, K., Abbeel, P., & Levine, S. (2018a). Latent space policies for hierarchical reinforcement learning. *International Conference on Machine Learning*, 1851–1860. 322

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018b). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *International Conference on Machine Learning*, 1861–1870. 398

Hagendorff, T. (2020). The ethics of AI ethics: An evaluation of guidelines. *Minds and Machines*, *30*(1), 99–120. 420

Hamilton, W., Ying, Z., & Leskovec, J. (2017a). Inductive representation learning on large graphs. *Neural Information Processing Systems*, *30*, 1024–1034. 262, 263, 264, 265, 267

Hamilton, W. L. (2020). Graph representation learning. *Synthesis Lectures on Artifical Intelligence and Machine Learning*, *14*(3), 1–159. 15, 261

Hamilton, W. L., Ying, R., & Leskovec, J. (2017b). Representation learning on graphs: Methods and applications. *IEEE Data Engineering Bulletin*, *40*(3), 52–74. 263

Han, S., Mao, H., & Dally, W. J. (2016). Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. *International Conference on Learning Representations*. 414, 415

Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. *Neural Information Processing Systems*, vol. 28, 1135–1143. 414

Hannun, A. Y., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., & Ng, A. Y. (2014). Deep speech: Scaling up end-to-end speech recognition. *arXiv:1412.5567*. 160

Hanson, S. J., & Pratt, L. Y. (1988). Comparing biases for minimal network construction with back-propagation. *Neural Information Processing Systems*, vol. 2, 177—185. 155

Harding, S. (1986). *The Science Question in Feminism*. Cornell University Press. 433

Härkönen, E., Hertzmann, A., Lehtinen, J., & Paris, S. (2020). GANSpace: Discovering interpretable GAN controls. *Neural Information Processing Systems*, *33*, 9841–9850. 300

Hartmann, K. G., Schirrmeister, R. T., & Ball, T. (2018). EEG-GAN: Generative adversarial networks for electroencephalograhic (EEG) brain signals. *arXiv:1806.01875*. 299

Harvey, W., Naderiparizi, S., Masrani, V., Weilbach, C., & Wood, F. (2022). Flexible diffusion modeling of long videos. *Neural Information Processing Systems*, *35*. 369

Hasanzadeh, A., Hajiramezanali, E., Boluki, S., Zhou, M., Duffield, N., Narayanan, K., & Qian, X. (2020). Bayesian graph neural networks with adaptive connection sampling. *International Conference on Machine Learning*, 4094–4104. 265

Hassibi, B., & Stork, D. G. (1993). Second order derivatives for network pruning: Optimal brain surgeon. *Neural Information Processing Systems*, vol. 6, 164–171. 414

Hausknecht, M., & Stone, P. (2015). Deep recurrent Q-learning for partially observable MDPs. *AAAI Fall Symposia*, 29–37. 397

Hayou, S., Clerico, E., He, B., Deligiannidis, G., Doucet, A., & Rousseau, J. (2021). Stable ResNet. *International Conference on Artificial Intelligence and Statistics*, 1324–1332. 205

He, F., Liu, T., & Tao, D. (2019). Control batch size and learning rate to generalize well: Theoretical and empirical evidence. *Neural Information Processing Systems*, *32*, 1143–1152. 92, 410, 411

He, J., Neubig, G., & Berg-Kirkpatrick, T. (2018). Unsupervised learning of syntactic structure

with invertible neural projections. *ACL Empirical Methods in Natural Language Processing*, 1292–1302. 322

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. *IEEE International Conference on Computer Vision*, 1026–1034. 38, 113, 183

He, K., Zhang, X., Ren, S., & Sun, J. (2016a). Deep residual learning for image recognition. *IEEE/CVF Computer Vision & Pattern Recognition*, 770–778. 188, 201, 323, 405

He, K., Zhang, X., Ren, S., & Sun, J. (2016b). Identity mappings in deep residual networks. *European Conference on Computer Vision*, 630–645. 202, 405

He, P., Liu, X., Gao, J., & Chen, W. (2021). DeBERTa: Decoding-enhanced BERT with disentangled attention. *International Conference on Learning Representations*. 236

He, X., Haffari, G., & Norouzi, M. (2020). Dynamic programming encoding for subword segmentation in neural machine translation. *Meeting of the Association for Computational Linguistics*, 3042–3051. 234

He, Y., Zhang, X., & Sun, J. (2017). Channel pruning for accelerating very deep neural networks. *IEEE/CVF International Conference on Computer Vision*, 1389–1397. 414

Heess, N., Wayne, G., Silver, D., Lillicrap, T., Erez, T., & Tassa, Y. (2015). Learning continuous control policies by stochastic value gradients. *Neural Information Processing Systems*, *28*, 2944–2952. 344

Heikkilä, M. (2022). *Why business is booming for military AI startups*. MIT Technology Review, July 7 2022. `https://www.technologyreview.com/2022/07/07/1055526/why-business-is-booming-for-military-ai-startups/`. 13, 427

Henaff, M., Bruna, J., & LeCun, Y. (2015). Deep convolutional networks on graph-structured data. *arXiv:1506.05163*. 262

Henderson, P., Li, X., Jurafsky, D., Hashimoto, T., Lemley, M. A., & Liang, P. (2023). Foundation models and fair use. *arXiv:2303.15715*. 428

Hendrycks, D., & Gimpel, K. (2016). Gaussian error linear units (GELUs). *arXiv:1606.08415*. 38

Hermann, V. (2017). Wasserstein GAN and the Kantorovich-Rubinstein duality. `https://vincentherrmann.github.io/blog/wasserstein/`. 284, 299

Hernández, C. X., Wayment-Steele, H. K., Sultan, M. M., Husic, B. E., & Pande, V. S. (2018). Variational encoding of complex dynamics. *Physical Review E*, *97*(6), 062412. 344

Hertz, A., Mokady, R., Tenenbaum, J., Aberman, K., Pritch, Y., & Cohen-Or, D. (2022). Prompt-to-prompt image editing with cross attention control. *arXiv:2208.01626*. 369

Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., & Silver, D. (2018). Rainbow: Combining improvements in deep reinforcement learning. *AAAI Conference on Artificial Intelligence*, 3215–3222. 397

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017). GANs trained by a two time-scale update rule converge to a local Nash equilibrium. *Neural Information Processing Systems*, *30*, 6626–6637. 274

Heyns, C. (2017). Autonomous weapons in armed conflict and the right to a dignified life: An African perspective. *South African Journal of Human Rights*, *33*(1), 46–71. 429

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., & Lerchner, A. (2017). Beta-VAE: Learning basic visual concepts with a constrained variational framework. *International Conference on Learning Representations*. 345

Himmelreich, J. (2022). Against 'democratizing AI'. *AI & Society*. 435

Hindupur, A. (2022). The GAN zoo. GitHub Retrieved January 17, 2023. `https://github.com/hindupuravinash/the-gan-zoo`. 299

Hinton, G., Srivastava, N., & Swersky, K. (2012a). Neural networks for machine learning: Lecture 6a – Overview of mini-batch gradient descent. `https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf`. 93

Hinton, G., & van Camp, D. (1993). Keeping neural networks simple by minimising the description length of weights. *Computational learning theory*, 5–13. 159

Hinton, G., Vinyals, O., Dean, J., et al. (2015). Distilling the knowledge in a neural network. *arXiv:1503.02531*, *2*(7). 415

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504–507. 344

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R.

(2012b). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*. 158

Ho, J., Chen, X., Srinivas, A., Duan, Y., & Abbeel, P. (2019). Flow++: Improving flow-based generative models with variational dequantization and architecture design. *International Conference on Machine Learning*, 2722–2730. 322, 323

Ho, J., Jain, A., & Abbeel, P. (2020). Denoising diffusion probabilistic models. *Neural Information Processing Systems*, *33*, 6840–6851. 274, 367, 369

Ho, J., Saharia, C., Chan, W., Fleet, D. J., Norouzi, M., & Salimans, T. (2022a). Cascaded diffusion models for high fidelity image generation. *Journal of Machine Learning Research*, *23*, 47–1. 369, 370

Ho, J., & Salimans, T. (2022). Classifier-free diffusion guidance. *NeurIPS Workshop on Deep Generative Models and Downstream Applications*. 370

Ho, J., Salimans, T., Gritsenko, A., Chan, W., Norouzi, M., & Fleet, D. J. (2022b). Video diffusion models. *International Conference on Learning Representations*. 369

Hochreiter, S., & Schmidhuber, J. (1997a). Flat minima. *Neural Computation*, *9*(1), 1–42. 411

Hochreiter, S., & Schmidhuber, J. (1997b). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. 233

Hoffer, E., Hubara, I., & Soudry, D. (2017). Train longer, generalize better: Closing the generalization gap in large batch training of neural networks. *Neural Information Processing Systems*, *30*, 1731–1741. 203, 204

Hoffman, M. D., & Johnson, M. J. (2016). ELBO surgery: Yet another way to carve up the variational evidence lower bound. *NIPS Workshop in Advances in Approximate Bayesian Inference*, 2. 346

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. (2023). Training compute-optimal large language models. *arXiv:2203.15556*. 234

Hofstadter, D. R. (1995). The ineradicable Eliza effect and its dangers (preface 4). *Fluid Concepts and Creative Analogies: Computer Models Of The Fundamental Mechanisms Of Thought*, 155–168. Basic Books. 428

Holland, C. A., Ebner, N. C., Lin, T., & Samanez-Larkin, G. R. (2019). Emotion identification across adulthood using the dynamic faces database of emotional expressions in younger, middle aged, and older adults. *Cognition and Emotion*, *33*(2), 245–257. 9

Holtzman, A., Buys, J., Du, L., Forbes, M., & Choi, Y. (2020). The curious case of neural text degeneration. *International Conference on Learning Representations*. 235

Hoogeboom, E., Nielsen, D., Jaini, P., Forré, P., & Welling, M. (2021). Argmax flows and multinomial diffusion: Learning categorical distributions. *Neural Information Processing Systems*, *34*, 12454–12465. 369

Hoogeboom, E., Peters, J., Van Den Berg, R., & Welling, M. (2019a). Integer discrete flows and lossless compression. *Neural Information Processing Systems*, *32*, 12134–12144. 324

Hoogeboom, E., Van Den Berg, R., & Welling, M. (2019b). Emerging convolutions for generative normalizing flows. *International Conference on Machine Learning*, 2771–2780. 322

Höppe, T., Mehrjou, A., Bauer, S., Nielsen, D., & Dittadi, A. (2022). Diffusion models for video prediction and infilling. *ECCV Workshop on AI for Creative Video Editing and Understanding*. 369

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, *4*(2), 251–257. 38

Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al. (2019). Searching for MobileNetV3. *IEEE/CVF International Conference on Computer Vision*, 1314–1324. 38

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). MobileN ets: Efficient convolutional neural networks for mobile vision applications. *arXiv:1704.04861*. 181

Howard, R. A. (1960). *Dynamic programming and Narkov processes*. Wiley. 396

Hsu, C.-C., Hwang, H.-T., Wu, Y.-C., Tsao, Y., & Wang, H.-M. (2017a). Voice conversion from unaligned corpora using variational autoencoding Wasserstein generative adversarial networks. *INTERSPEECH*, 3364–3368. 345

Hsu, W.-N., Zhang, Y., & Glass, J. (2017b). Learning latent representations for speech generation and transformation. *INTERSPEECH*, 1273–1277. 343

Hu, H., Gu, J., Zhang, Z., Dai, J., & Wei, Y. (2018a). Relation networks for object detection. *IEEE/CVF Computer Vision & Pattern Recognition*, 3588–3597. 238

Hu, H., Zhang, Z., Xie, Z., & Lin, S. (2019). Local relation networks for image recognition. *IEEE/CVF International Conference on Computer Vision*, 3464–3473. 238

Hu, J., Shen, L., & Sun, G. (2018b). Squeeze-and-excitation networks. *IEEE/CVF Computer Vision & Pattern Recognition*, 7132–7141. 181, 235

Hu, W., Pang, J., Liu, X., Tian, D., Lin, C.-W., & Vetro, A. (2022). Graph signal processing for geometric data and beyond: Theory and applications. *IEEE Transactions on Multimedia*, *24*, 3961–3977. 242

Hu, Z., Yang, Z., Liang, X., Salakhutdinov, R., & Xing, E. P. (2017). Toward controlled generation of text. *International Conference on Machine Learning*, 1587–1596. 343

Huang, C.-W., Krueger, D., Lacoste, A., & Courville, A. (2018a). Neural autoregressive flows. *International Conference on Machine Learning*, 2078–2087. 323, 324

Huang, G., Li, Y., Pleiss, G., Liu, Z., Hopcroft, J. E., & Weinberger, K. Q. (2017a). Snapshot ensembles: Train 1, get M for free. *International Conference on Learning Representations*. 158

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017b). Densely connected convolutional networks. *IEEE/CVF Computer Vision & Pattern Recognition*, 4700–4708. 205, 405

Huang, G., Sun, Y., Liu, Z., Sedra, D., & Weinberger, K. Q. (2016). Deep networks with stochastic depth. *European Conference on Computer Vision*, 646–661. 202

Huang, W., Zhang, T., Rong, Y., & Huang, J. (2018b). Adaptive sampling towards fast graph representation learning. *Neural Information Processing Systems*, *31*, 4563–4572. 264, 265

Huang, X., Li, Y., Poursaeed, O., Hopcroft, J., & Belongie, S. (2017c). Stacked generative adversarial networks. *IEEE/CVF Computer Vision & Pattern Recognition*, 5077–5086. 300

Huang, X. S., Perez, F., Ba, J., & Volkovs, M. (2020a). Improving transformer optimization through better initialization. *International Conference on Machine Learning*, 4475–4483. 114, 237

Huang, Y., Cheng, Y., Bapna, A., Firat, O., Chen, D., Chen, M., Lee, H., Ngiam, J., Le, Q. V., Wu, Y., et al. (2019). GPipe: Efficient training of giant neural networks using pipeline parallelism. *Neural Information Processing Systems*, *32*, 103–112. 114

Huang, Z., Liang, D., Xu, P., & Xiang, B. (2020b). Improve transformer models with better relative position embeddings. *Empirical Methods in Natural Language Processing*. 236

Huang, Z., & Wang, N. (2018). Data-driven sparse structure selection for deep neural networks. *European Conference on Computer Vision*, 304–320. 414

Hubinger, E., van Merwijk, C., Mikulik, V., Skalse, J., & Garrabrant, S. (2019). Risks from learned optimization in advanced machine learning systems. *arXiv:1906.01820*. 421

Hussein, A., Gaber, M. M., Elyan, E., & Jayne, C. (2017). Imitation learning: A survey of learning methods. *ACM Computing Surveys*, *50*(2), 1–35. 398

Huszár, F. (2019). Exponentially growing learning rate? Implications of scale invariance induced by batch normalization. `https://www.inference.vc/exponentially-growing-learning-rate-implications-of-scale-invariance-induced-by-BatchNorm/`. 204

Hutchinson, M. F. (1989). A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, *18*(3), 1059–1076. 324

Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. *International Conference on Learning and Intelligent Optimization*, 507–523. 136

Iglovikov, V., & Shvets, A. (2018). TernausNet: U-Net with VGG11 encoder pre-trained on ImageNet for image segmentation. *arXiv:1801.05746*. 205

Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., & Madry, A. (2019). Adversarial examples are not bugs, they are features. *Neural Information Processing Systems*, *32*, 125–136. 414

Inoue, H. (2018). Data augmentation by pairing samples for images classification. *arXiv:1801.02929*. 159

Inoue, T., Choudhury, S., De Magistris, G., & Dasgupta, S. (2018). Transfer learning from synthetic to real images using variational autoen-

coders for precise position detection. *IEEE International Conference on Image Processing*, 2725–2729. 344

Ioffe, S. (2017). Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. *Neural Information Processing Systems*, *30*, 1945–1953. 203

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International Conference on Machine Learning*, 448–456. 114, 203, 204

Ishida, T., Yamane, I., Sakai, T., Niu, G., & Sugiyama, M. (2020). Do we need zero training loss after achieving zero training error? *International Conference on Machine Learning*, 4604–4614. 134, 159

Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. *IEEE/CVF Computer Vision & Pattern Recognition*, 1125–1134. 205, 293, 301

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., & Wilson, A. G. (2018). Averaging weights leads to wider optima and better generalization. *Uncertainly in Artificial Intelligence*, 876–885. 158, 411

Jackson, P. T., Abarghouei, A. A., Bonner, S., Breckon, T. P., & Obara, B. (2019). Style augmentation: Data augmentation via style randomization. *IEEE Computer Vision and Pattern Recognition Workshops*, 10–11. 159

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, *3*(1), 79–87. 73

Jacobsen, J.-H., Smeulders, A., & Oyallon, E. (2018). i-RevNet: Deep invertible networks. *International Conference on Learning Representations*. 322, 323

Jaini, P., Kobyzev, I., Yu, Y., & Brubaker, M. A. (2020). Tails of Lipschitz triangular flows. *International Conference on Machine Learning*, 4673–4681. 324

Jaini, P., Selby, K. A., & Yu, Y. (2019). Sum-of-squares polynomial flow. *International Conference on Machine Learning*, 3009–3018. 323

Jaitly, N., & Hinton, G. E. (2013). Vocal tract length perturbation (VTLP) improves speech recognition. *ICML Workshop on Deep Learning for Audio, Speech and Language*. 160

Jarrett, K., Kavukcuoglu, K., Ranzato, M., & LeCun, Y. (2009). What is the best multi-stage architecture for object recognition? *IEEE International Conference on Computer Vision*, 2146–2153. 37

Jastrzębski, S., Arpit, D., Astrand, O., Kerg, G. B., Wang, H., Xiong, C., Socher, R., Cho, K., & Geras, K. J. (2021). Catastrophic fisher explosion: Early phase fisher matrix impacts generalization. *International Conference on Machine Learning*, 4772–4784. 157

Jastrzębski, S., Kenton, Z., Arpit, D., Ballas, N., Fischer, A., Bengio, Y., & Storkey, A. (2018). Three factors influencing minima in SGD. *arXiv:1711.04623*. 92, 410

Ji, S., Xu, W., Yang, M., & Yu, K. (2012). 3D convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, *35*(1), 221–231. 182

Jia, X., De Brabandere, B., Tuytelaars, T., & Gool, L. V. (2016). Dynamic filter networks. *Neural Information Processing Systems*, *29*. 183

Jiang, Z., Zheng, Y., Tan, H., Tang, B., & Zhou, H. (2016). Variational deep embedding: An unsupervised and generative approach to clustering. *International Joint Conference on Artificial Intelligence*, 1965–1972. 344

Jin, C., Netrapalli, P., & Jordan, M. (2020). What is local optimality in nonconvex-nonconcave minimax optimization? *International Conference on Machine Learning*, 4880–4889. 299

Jin, L., Doshi-Velez, F., Miller, T., Schwartz, L., & Schuler, W. (2019). Unsupervised learning of PCFGs with normalizing flow. *Meeting of the Association for Computational Linguistics*, 2442–2452. 322

Jing, L., & Tian, Y. (2020). Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, *43*(11), 4037–4058. 159

Jobin, A., Ienca, M., & Vayena, E. (2019). The global landscape of AI ethics guidelines. *Nature Machine Intelligence*, *1*, 389–399. 420

Johnson, G. M. (2022). Are algorithms value-free? feminist theoretical virtues in machine learning. *198*. 432

Johnson, R., & Zhang, T. (2013). Accelerating stochastic gradient descent using predictive variance reduction. *Neural Information Processing Systems*, *26*, 315–323. 91

Jolicoeur-Martineau, A. (2019). The relativistic discriminator: A key element missing from

standard GAN. *International Conference on Learning Representations*. 299

Jurafsky, D., & Martin, J. H. (2000). *Speech and Language Processing, 2nd Edition*. Pearson. 233

Kakade, S. M. (2001). A natural policy gradient. *Neural Information Processing Systems*, *14*, 1531–1538. 397

Kanazawa, A., Sharma, A., & Jacobs, D. (2014). Locally scale-invariant convolutional neural networks. *Neural Information Processing Systems Workshop*. 183

Kanda, N., Takeda, R., & Obuchi, Y. (2013). Elastic spectral distortion for low resource speech recognition with deep neural networks. *IEEE Workshop on Automatic Speech Recognition and Understanding*, 309–314. 160

Kaneko, T., & Kameoka, H. (2017). Parallel-data-free voice conversion using cycle-consistent adversarial networks. *arXiv:1711.11293*. 299

Kang, G., Dong, X., Zheng, L., & Yang, Y. (2017). PatchShuffle regularization. *arXiv:1707.07103*. 159

Kanwar, G., Albergo, M. S., Boyda, D., Cranmer, K., Hackett, D. C., Racaniere, S., Rezende, D. J., & Shanahan, P. E. (2020). Equivariant flow-based sampling for lattice gauge theory. *Physical Review Letters*, *125*(12), 121601. 322

Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2018). Progressive growing of GANs for improved quality, stability, and variation. *International Conference on Learning Representations*. 286, 287, 299, 300, 319, 345

Karras, T., Aittala, M., Aila, T., & Laine, S. (2022). Elucidating the design space of diffusion-based generative models. *Neural Information Processing Systems*. 369, 370

Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J., & Aila, T. (2020a). Training generative adversarial networks with limited data. *Neural Information Processing Systems*, *33*, 12104–12114. 300

Karras, T., Aittala, M., Laine, S., Härkönen, E., Hellsten, J., Lehtinen, J., & Aila, T. (2021). Alias-free generative adversarial networks. *Neural Information Processing Systems*, *34*, 852–863. 300

Karras, T., Laine, S., & Aila, T. (2019). A style-based generator architecture for generative adversarial networks. *IEEE/CVF Computer Vision & Pattern Recognition*, 4401–4410. 299, 300

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., & Aila, T. (2020b). Analyzing and improving the image quality of StyleGAN. *IEEE/CVF Computer Vision & Pattern Recognition*, 8110–8119. 8, 300, 301

Katharopoulos, A., Vyas, A., Pappas, N., & Fleuret, F. (2020). Transformers are RNNs: Fast autoregressive transformers with linear attention. *International Conference on Machine Learning*, 5156–5165. 237

Kawaguchi, K., Huang, J., & Kaelbling, L. P. (2019). Effect of depth and width on local minima in deep learning. *Neural Computation*, *31*(7), 1462–1498. 405

Ke, G., He, D., & Liu, T.-Y. (2021). Rethinking positional encoding in language pre-training. *International Conference on Learning Representations*. 236

Kearnes, S., McCloskey, K., Berndl, M., Pande, V., & Riley, P. (2016). Molecular graph convolutions: Moving beyond fingerprints. *Journal of computer-aided molecular design*, *30*(8), 595–608. 264

Kendall, A., & Gal, Y. (2017). What uncertainties do we need in Bayesian deep learning for computer vision? *Neural Information Processing Systems*, *30*, 5574–5584. 158

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. T. P. (2017). On large-batch training for deep learning: Generalization gap and sharp minima. *International Conference on Learning Representations*. 158, 403, 410, 411

Keskar, N. S., & Socher, R. (2017). Improving generalization performance by switching from Adam to SGD. *arXiv:1712.07628*. 94, 410

Keynes, J. M. (2010). Economic possibilities for our grandchildren. *Essays in Persuasion*, 321–332. Palgrave Macmillan. 430

Khan, S., Naseer, M., Hayat, M., Zamir, S. W., Khan, F. S., & Shah, M. (2022). Transformers in vision: A survey. *ACM Computing Surveys*, *54*(10), 200:1–200:41. 238

Killoran, N., Lee, L. J., Delong, A., Duvenaud, D., & Frey, B. J. (2017). Generating and designing DNA with deep generative models. *NIPS 2017 Workshop on Computational Biology*. 299

Kim, H., & Mnih, A. (2018). Disentangling by factorising. *International Conference on Machine Learning*, 2649–2658. 345, 346

Kim, I., Han, S., Baek, J.-w., Park, S.-J., Han, J.-J., & Shin, J. (2021). Quality-agnostic image recognition via invertible de-

coder. *IEEE/CVF Computer Vision & Pattern Recognition*, 12257–12266. 322

Kim, S., Lee, S.-g., Song, J., Kim, J., & Yoon, S. (2018). FloWaveNet: A generative flow for raw audio. *International Conference on Machine Learning*, 3370–3378. 322, 323

Kingma, D., Salimans, T., Poole, B., & Ho, J. (2021). Variational diffusion models. *Neural Information Processing Systems*, *34*, 21696–21707. 369

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. *International Conference on Learning Representations*. 93, 237

Kingma, D. P., & Dhariwal, P. (2018). Glow: Generative flow with invertible 1x1 convolutions. *Neural Information Processing Systems*, *31*, 10236–10245. 319, 322, 323

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., & Welling, M. (2016). Improved variational inference with inverse autoregressive flow. *Neural Information Processing Systems*, *29*, 4736–4744. 323, 344

Kingma, D. P., Salimans, T., & Welling, M. (2015). Variational dropout and the local reparameterization trick. *Advances in neural information processing systems*, *28*, 2575–2583. 346

Kingma, D. P., & Welling, M. (2014). Auto-encoding variational Bayes. *International Conference on Learning Representations*. 273, 343

Kingma, D. P., Welling, M., et al. (2019). An introduction to variational autoencoders. *Foundations and Trends in Machine Learning*, *12*(4), 307–392. 343

Kipf, T. N., & Welling, M. (2016). Variational graph auto-encoders. *NIPS Bayesian Deep Learning Workshop*. 159, 344

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*. 262, 263, 264, 265

Kiranyaz, S., Avci, O., Abdeljaber, O., Ince, T., Gabbouj, M., & Inman, D. J. (2021). 1D convolutional neural networks and applications: A survey. *Mechanical Systems and Signal Processing*, *151*, 107398. 182

Kiranyaz, S., Ince, T., Hamila, R., & Gabbouj, M. (2015). Convolutional neural networks for patient-specific ECG classification. *International Conference of the IEEE Engineering in Medicine and Biology Society*, vol. 37, 2608–2611. 182

Kitaev, N., Kaiser, Ł., & Levskaya, A. (2020). Reformer: The efficient transformer. *International Conference on Learning Representations*. 237

Kitcher, P. (2011a). *The Ethical Project*. Harvard University Press. 432

Kitcher, P. (2011b). *Science in a Democratic Society*. Prometheus Books. 432

Klambauer, G., Unterthiner, T., Mayr, A., & Hochreiter, S. (2017). Self-normalizing neural networks. *Neural Information Processing Systems*, vol. 30, 972–981. 38, 113

Kleinberg, J., Mullainathan, S., & Raghavan, M. (2017). Inherent trade-offs in the fair determination of risk scores. *Innovations in Theoretical Computer Science Conference*, vol. 67, 1–23. 422

Kleinberg, R., Li, Y., & Yuan, Y. (2018). An alternative view: When does SGD escape local minima? *International Conference on Machine Learning*, 2703–2712. 411

Knight, W. (2018). One of the fathers of AI is worried about its future. MIT Technology Review, Nov 20, 2018. `https://www.technologyreview.com/2018/11/17/66372/one-of-the-fathers-of-ai-is-worried-about-its-future/`. 430

Kobyzev, I., Prince, S. J., & Brubaker, M. A. (2020). Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, *43*(11), 3964–3979. xii, 321, 324

Koenker, R., & Hallock, K. F. (2001). Quantile regression. *Journal of Economic Perspectives*, *15*(4), 143–156. 73

Köhler, J., Klein, L., & Noé, F. (2020). Equivariant flows: Exact likelihood generative learning for symmetric densities. *International Conference on Machine Learning*, 5361–5370. 322, 324

Koller, D., & Friedman, N. (2009). *Probabilistic graphical models: Principles and techniques*. MIT Press. 15

Kolomiyets, O., Bethard, S., & Moens, M.-F. (2011). Model-portability experiments for textual temporal analysis. *Meeting of the Association for Computational Linguistics*, 271–276. 160

Konda, V., & Tsitsiklis, J. (1999). Actor-critic algorithms. *Neural Information Processing Systems*, *12*, 1008–1014. 397

Kong, Z., Ping, W., Huang, J., Zhao, K., & Catanzaro, B. (2021). DiffWave: A versatile diffusion

model for audio synthesis. *International Conference on Learning Representations*. 369

Kool, W., van Hoof, H., & Welling, M. (2019). Attention, learn to solve routing problems! *International Conference on Learning Representations*. 396

Kosinski, M., Stillwell, D., & Graepel, T. (2013). Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the National Academy of Sciences of the United States of America*, *110*(15), 5802–5805. 427

Kratsios, M. (2019). The national artificial intelligence research and development strategic plan: 2019 update. Tech. rep., Networking and Information Technology Research and Development. `https://www.nitrd.gov/pubs/National-AI-RD-Strategy-2019.pdf`. 430

Krizhevsky, A., & Hinton, G. (2009). Learning multiple layers of features from tiny images. *Technical Report, University of Toronto*. 188

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Neural Information Processing Systems*, *25*, 1097–1105. 52, 113, 159, 176, 181

Kruse, J., Detommaso, G., Köthe, U., & Scheichl, R. (2021). HINT: Hierarchical invertible neural transport for density estimation and Bayesian inference. *AAAI Conference on Artificial Intelligence*, 8191–8199. 323

Kudo, T. (2018). Subword regularization: Improving neural network translation models with multiple subword candidates. *Meeting of the Association for Computational Linguistics*, 66–75. 234

Kudo, T., & Richardson, J. (2018). SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *Empirical Methods in Natural Language Processing*, 66–71. 234

Kukačka, J., Golkov, V., & Cremers, D. (2017). Regularization for deep learning: A taxonomy. *arXiv:1710.10686*. 155

Kulikov, I., Miller, A. H., Cho, K., & Weston, J. (2018). Importance of search and evaluation strategies in neural dialogue modeling. *ACL International Conference on Natural Language Generation*, 76–87. 235

Kumar, A., Fu, J., Soh, M., Tucker, G., & Levine, S. (2019a). Stabilizing off-policy Q-learning via bootstrapping error reduction. *Neural Information Processing Systems*, *32*, 11761–11771. 398

Kumar, A., Sattigeri, P., & Balakrishnan, A. (2018). Variational inference of disentangled latent concepts from unlabeled observations. *International Conference on Learning Representations*. 345

Kumar, A., Singh, S. S., Singh, K., & Biswas, B. (2020a). Link prediction techniques, applications, and performance: A survey. *Physica A: Statistical Mechanics and its Applications*, *553*, 124289. 262

Kumar, A., Zhou, A., Tucker, G., & Levine, S. (2020b). Conservative Q-learning for offline reinforcement learning. *Neural Information Processing Systems*, *33*, 1179–1191. 398

Kumar, M., Babaeizadeh, M., Erhan, D., Finn, C., Levine, S., Dinh, L., & Kingma, D. (2019b). VideoFlow: A flow-based generative model for video. *ICML Workshop on Invertible Neural Networks and Normalizing Flows*. 322

Kumar, M., Weissenborn, D., & Kalchbrenner, N. (2021). Colorization transformer. *International Conference on Learning Representations*. 238

Kurach, K., Lučić, M., Zhai, X., Michalski, M., & Gelly, S. (2019). A large-scale study on regularization and normalization in GANs. *International Conference on Machine Learning*, 3581–3590. 299

Kurenkov, A. (2020). *A Brief History of Neural Nets and Deep Learning*. `https://www.skynettoday.com/overviews/neural-net-history`. 37

Kynkäänniemi, T., Karras, T., Laine, S., Lehtinen, J., & Aila, T. (2019). Improved precision and recall metric for assessing generative models. *Neural Information Processing Systems*, *32*, 3929–3938. 274

LaCroix, T. (2022). The linguistic blind spot of value-aligned agency, natural and artificial. *arXiv:2207.00868*. 421

LaCroix, T. (2023). *Artificial Intelligence and the Value-Alignment Problem: A Philosophical Introduction*. `https://value-alignment.github.io`. 422, 435

LaCroix, T., Geil, A., & O'Connor, C. (2021). The dynamics of retraction in epistemic networks. *Philosophy of Science*, *88*(3), 415–438. 432

LaCroix, T., & Mohseni, A. (2022). The tragedy of the AI commons. *Synthese*, *200*(289). 420

Laffont, J.-J., & Martimort, D. (2002). *The Theory of Incentives: The Principal-Agent Model*. Princeton University Press. 421

Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. *Neural Information Processing Systems*, *30*, 6402–6413. 158

Lamb, A., Dumoulin, V., & Courville, A. (2016). Discriminative regularization for generative models. *arXiv:1602.03220*. 344

Lample, G., & Charton, F. (2020). Deep learning for symbolic mathematics. *International Conference on Learning Representations*. 234

Larsen, A. B. L., Sønderby, S. K., Larochelle, H., & Winther, O. (2016). Autoencoding beyond pixels using a learned similarity metric. *International Conference on Machine Learning*, 1558–1566. 344, 345

Lasseck, M. (2018). Acoustic bird detection with deep convolutional neural networks. *Detection and Classification of Acoustic Scenes and Events*, 143–147. 160

Lattimore, T., & Szepesvári, C. (2020). *Bandit algorithms*. Cambridge University Press. 136

Lawrence, S., Giles, C. L., Tsoi, A. C., & Back, A. D. (1997). Face recognition: A convolutional neural-network approach. *IEEE Transactions on Neural Networks*, *8*(1), 98–113. 181

LeCun, Y. (1985). Une procedure d'apprentissage pour reseau a seuil asymmetrique. *Proceedings of Cognitiva*, 599–604. 113

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. 52

LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., & Jackel, L. (1989a). Handwritten digit recognition with a back-propagation network. *Neural Information Processing Systems*, *2*, 396–404. 180, 181

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989b). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, *1*(4), 541–551. 180

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. 159, 181

LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., & Huang, F. (2006). A tutorial on energy-based learning. *Predicting structured data*, *1*(0). 274

LeCun, Y., Denker, J. S., & Solla, S. A. (1990). Optimal brain damage. *Neural Information Processing Systems*, vol. 3, 598–605. 414

LeCun, Y. A., Bottou, L., Orr, G. B., & Müller, K.-R. (2012). Efficient backprop. *Neural Networks: Tricks of the trade*, 9–48. Springer. 113, 410

Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al. (2017). Photo-realistic single image super-resolution using a generative adversarial network. *IEEE/CVF Computer Vision & Pattern Recognition*, 4681–4690. 294, 301

Lee, J., Lee, I., & Kang, J. (2019). Self-attention graph pooling. *International Conference on Machine Learning*, 3734–3743. 265

Lee, J. B., Rossi, R. A., Kong, X., Kim, S., Koh, E., & Rao, A. (2018). Higher-order graph convolutional networks. *arXiv:1809.07697*. 263

Lehman, J., & Stanley, K. O. (2008). Exploiting open-endedness to solve problems through the search for novelty. *International Conference on Artificial Life*, 329–336. 421

Leuner, J. (2019). A replication study: Machine learning models are capable of predicting sexual orientation from facial images. *arXiv:1902.10739*. 427

Li, C., Chen, C., Carlson, D., & Carin, L. (2016a). Preconditioned stochastic gradient Langevin dynamics for deep neural networks. *AAAI Conference on Artificial Intelligence*, 1788–1794. 159

Li, C., Farkhoor, H., Liu, R., & Yosinski, J. (2018a). Measuring the intrinsic dimension of objective landscapes. *International Conference on Learning Representations*. 407, 408

Li, F.-F. (2018). How to make A.I. that's good for people. The New York Times, March 7, 2018. `https://www.nytimes.com/2018/03/07/opinion/artificial-intelligence-human.html`. 430

Li, G., Müller, M., Ghanem, B., & Koltun, V. (2021a). Training graph neural networks with 1000 layers. *International Conference on Machine Learning*, 6437–6449. 266, 322

Li, G., Müller, M., Qian, G., Perez, I. C. D., Abualshour, A., Thabet, A. K., & Ghanem, B. (2021b). DeepGCNs: Making GCNs go as deep as CNNs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 266

Li, G., Xiong, C., Thabet, A., & Ghanem, B. (2020a). DeeperGCN: All you need to train deeper GCNs. *arXiv:2006.07739*. 266

Li, H., Kadav, A., Durdanovic, I., Samet, H., & Graf, H. P. (2017a). Pruning filters for efficient ConvNets. *International Conference on Learning Representations*. 414

Li, H., Xu, Z., Taylor, G., Studer, C., & Goldstein, T. (2018b). Visualizing the loss landscape of neural nets. *Neural Information Processing Systems*, *31*, 6391–6401. 201, 202, 407

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2017b). Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, *18*(1), 6765–6816. 136

Li, L. H., Yatskar, M., Yin, D., Hsieh, C.-J., & Chang, K.-W. (2019). VisualBERT: A simple and performant baseline for vision and language. *arXiv:1908.03557*. 238

Li, Q., Han, Z., & Wu, X.-M. (2018c). Deeper insights into graph convolutional networks for semi-supervised learning. *AAAI Conference on Artificial Intelligence*, 3438–3545. 265

Li, S., Zhao, Y., Varma, R., Salpekar, O., Noordhuis, P., Li, T., Paszke, A., Smith, J., Vaughan, B., Damania, P., & Chintala, S. (2020b). Pytorch distributed: Experiences on accelerating data parallel training. *International Conference on Very Large Databases*. 114

Li, W., Lin, Z., Zhou, K., Qi, L., Wang, Y., & Jia, J. (2022). MAT: Mask-aware transformer for large hole image inpainting. *IEEE/CVF Computer Vision & Pattern Recognition*, 10758–10768. 238

Li, Y. (2017). Deep reinforcement learning: An overview. *arXiv:1701.07274*. 396

Li, Y., Cohn, T., & Baldwin, T. (2017c). Robust training under linguistic adversity. *Meeting of the Association for Computational Linguistics*, 21–27. 160

Li, Y., & Liang, Y. (2018). Learning overparameterized neural networks via stochastic gradient descent on structured data. *Neural Information Processing Systems*, *31*, 8168–8177. 407

Li, Y., Tarlow, D., Brockschmidt, M., & Zemel, R. (2016b). Gated graph sequence neural networks. *International Conference on Learning Representations*. 262

Li, Y., & Turner, R. E. (2016). Rényi divergence variational inference. *Neural Information Processing Systems*, *29*, 1073–1081. 346

Li, Z., & Arora, S. (2019). An exponential learning rate schedule for deep learning. *International Conference on Learning Representations*. 204

Liang, D., Krishnan, R. G., Hoffman, M. D., & Jebara, T. (2018). Variational autoencoders for collaborative filtering. *World Wide Web Conference*, 689–698. 344

Liang, J., Zhang, K., Gu, S., Van Gool, L., & Timofte, R. (2021). Flow-based kernel prior with application to blind super-resolution. *IEEE/CVF Computer Vision & Pattern Recognition*, 10601–10610. 322

Liang, S., & Srikant, R. (2016). Why deep neural networks for function approximation? *International Conference on Learning Representations*. 53, 417

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2016). Continuous control with deep reinforcement learning. *International Conference on Learning Representations*. 397

Lin, K., Li, D., He, X., Zhang, Z., & Sun, M.-T. (2017a). Adversarial ranking for language generation. *Neural Information Processing Systems*, *30*, 3155–3165. 299

Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, *8*, 293–321. 396

Lin, M., Chen, Q., & Yan, S. (2014). Network in network. *International Conference on Learning Representations*. 181

Lin, T., Wang, Y., Liu, X., & Qiu, X. (2022). A survey of transformers. *AI Open*, *3*, 111–132. 233

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017b). Feature pyramid networks for object detection. *IEEE Computer Vision & Pattern Recognition*, 2117–2125. 184

Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017c). Focal loss for dense object detection. *IEEE/CVF International Conference on Computer Vision*, 2980–2988. 73

Lin, Z., Khetan, A., Fanti, G., & Oh, S. (2018). PacGAN: The power of two samples in generative adversarial networks. *Neural Information Processing Systems*, *31*, 1505–1514. 300

Ling, H., Kreis, K., Li, D., Kim, S. W., Torralba, A., & Fidler, S. (2021). EditGAN: High-precision semantic image editing. *Neural Information Processing Systems*, *34*, 16331–16345. 302

Lipman, Y., Chen, R. T., Ben-Hamu, H., Nickel, M., & Le, M. (2022). Flow matching for generative modeling. *arXiv:2210.02747*. 369

Lipton, Z. C., & Tripathi, S. (2017). Precise recovery of latent vectors from generative adversarial networks. *International Conference on Learning Representations*. 301

Liu, G., Reda, F. A., Shih, K. J., Wang, T.-C., Tao, A., & Catanzaro, B. (2018a). Image inpainting for irregular holes using partial convolutions. *European Conference on Computer Vision*, 85–100. 181

Liu, H., Simonyan, K., & Yang, Y. (2019a). DARTS: Differentiable architecture search. *International Conference on Learning Representations*. 414

Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., & Han, J. (2021a). On the variance of the adaptive learning rate and beyond. *International Conference on Learning Representations*. 93

Liu, L., Liu, X., Gao, J., Chen, W., & Han, J. (2020). Understanding the difficulty of training transformers. *Empirical Methods in Natural Language Processing*, 5747–5763. 237, 238

Liu, L., Luo, Y., Shen, X., Sun, M., & Li, B. (2019b). Beta-dropout: A unified dropout. *IEEE Access*, *7*, 36140–36153. 158

Liu, P. J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., & Shazeer, N. (2018b). Generating Wikipedia by summarizing long sequences. *International Conference on Learning Representations*. 237

Liu, X., Zhang, F., Hou, Z., Mian, L., Wang, Z., Zhang, J., & Tang, J. (2023a). Self-supervised learning: Generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering*, *35*(1), 857–876. 159

Liu, Y., Qin, Z., Anwar, S., Ji, P., Kim, D., Caldwell, S., & Gedeon, T. (2021b). Invertible denoising network: A light solution for real noise removal. *IEEE/CVF Computer Vision & Pattern Recognition*, 13365–13374. 322

Liu, Y., Zhang, Y., Wang, Y., Hou, F., Yuan, J., Tian, J., Zhang, Y., Shi, Z., Fan, J., & He, Z. (2023b). A survey of visual transformers. *IEEE Transactions on Neural Networks and Learning Systems*. 238

Liu, Z., Hu, H., Lin, Y., Yao, Z., Xie, Z., Wei, Y., Ning, J., Cao, Y., Zhang, Z., Dong, L., Wei, F., & Guo, B. (2022). Swin transformer V2: Scaling up capacity and resolution. *IEEE/CVF Computer Vision & Pattern Recognition*, 12009–12019. 238

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., & Guo, B. (2021c). Swin transformer: Hierarchical vision transformer using shifted windows. *IEEE/CVF International Conference on Computer Vision*, 10012–10022. 231, 238

Liu, Z., Luo, P., Wang, X., & Tang, X. (2015). Deep learning face attributes in the wild. *IEEE International Conference on Computer Vision*, 3730–3738. 345

Liu, Z., Michaud, E. J., & Tegmark, M. (2023c). Omnigrok: Grokking beyond algorithmic data. *International Conference on Learning Representations*. 405, 406, 412, 413

Liu, Z., Sun, M., Zhou, T., Huang, G., & Darrell, T. (2019c). Rethinking the value of network pruning. *International Conference on Learning Representations*. 235

Livni, R., Shalev-Shwartz, S., & Shamir, O. (2014). On the computational efficiency of training neural networks. *Neural Information Processing Systems*, *27*, 855–863. 405

Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., & Kipf, T. (2020). Object-centric learning with slot attention. *Neural Information Processing Systems*, *33*, 11525–11538. 238

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. *IEEE/CVF Computer Vision & Pattern Recognition*, 3431–3440. 181

Longino, H. E. (1990). *Science as Social Knowledge: Values and Objectivity in Scientific Inquiry*. Princeton University Press. 432

Longino, H. E. (1996). Cognitive and non-cognitive values in science: Rethinking the dichotomy. *Feminism, Science, and the Philosophy of Science*, 39–58. 432

Loshchilov, I., & Hutter, F. (2019). Decoupled weight decay regularization. *International Conference on Learning Representations*. 94, 156

Louizos, C., Welling, M., & Kingma, D. P. (2018). Learning sparse neural networks through l_0 regularization. *International Conference on Learning Representations*. 156

Loukas, A. (2020). What graph neural networks cannot learn: Depth vs width. *International Conference on Learning Representations*. 262

Lu, J., Batra, D., Parikh, D., & Lee, S. (2019). VilBERT: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *Neural Information Processing Systems*, *32*, 13–23. 238

Lu, S.-P., Wang, R., Zhong, T., & Rosin, P. L. (2021). Large-capacity image steganography based on invertible neural networks. *IEEE/CVF Computer Vision & Pattern Recognition*, 10816–10825. 322

Lu, Z., Pu, H., Wang, F., Hu, Z., & Wang, L. (2017). The expressive power of neural networks: A view from the width. *Neural Information Processing Systems*, *30*, 6231–6239. 53

Lubana, E. S., Dick, R., & Tanaka, H. (2021). Beyond BatchNorm: Towards a unified understanding of normalization in deep learning. *Neural Information Processing Systems*, *34*, 4778–4791. 204

Lucas, J., Tucker, G., Grosse, R., & Norouzi, M. (2019a). Understanding posterior collapse in generative latent variable models. *ICLR Workshop on Deep Generative Models for Highly Structured Data*. 345

Lucas, J., Tucker, G., Grosse, R. B., & Norouzi, M. (2019b). Don't blame the ELBO! A linear VAE perspective on posterior collapse. *Neural Information Processing Systems*, *32*, 9403–9413. 345

Luccioni, A. S. (2023). The mounting human and environmental costs of generative AI. ars Technica, April 12, 2023.`https://arstechnica.com/gadgets/2023/04/generative-ai-is-cool-but-lets-not-forget-its-human-and-environmental-costs`. 429

Luccioni, A. S., Viguier, S., & Ligozat, A.-L. (2022). Estimating the carbon footprint of bloom, a 176b parameter language model. *arXiv:2211.02001*. 429

Lucic, M., Kurach, K., Michalski, M., Gelly, S., & Bousquet, O. (2018). Are GANs created equal? A large-scale study. *Neural Information Processing Systems*, *31*, 698–707. 299

Lücke, J., Forster, D., & Dai, Z. (2020). The evidence lower bound of variational autoencoders converges to a sum of three entropies. *arXiv:2010.14860*. 346

Luo, C. (2022). Understanding diffusion models: A unified perspective. *arXiv:2208.11970*. 369

Luo, G., Heide, M., & Uecker, M. (2022). MRI reconstruction via data driven Markov chain with joint uncertainty estimation. *arXiv:2202.01479*. 369

Luo, J., Xu, Y., Tang, C., & Lv, J. (2017a). Learning inverse mapping by autoencoder based generative adversarial nets. *Neural Information Processing Systems*, vol. 30, 207–216. 301

Luo, J.-H., Wu, J., & Lin, W. (2017b). ThiNet: A filter level pruning method for deep neural network compression. *IEEE/CVF International Conference on Computer Vision*, 5058–5066. 414

Luo, P., Wang, X., Shao, W., & Peng, Z. (2018). Towards understanding regularization in batch normalization. *International Conference on Learning Representations*. 205

Luo, S., & Hu, W. (2021). Diffusion probabilistic models for 3D point cloud generation. *IEEE/CVF Computer Vision & Pattern Recognition*, 2837–2845. 369

Luong, M.-T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *Empirical Methods in Natural Language Processing*, 1412–1421. 235

Luther, K. (2020). Why BatchNorm causes exploding gradients. `https://kyleluther.github.io/2020/02/18/BatchNorm-exploding-gradients.html`. 203

Ma, Y., & Tang, J. (2021). *Deep learning on graphs*. Cambridge University Press. 261

Ma, Y.-A., Chen, T., & Fox, E. (2015). A complete recipe for stochastic gradient MCMC. *Neural Information Processing Systems*, *28*, 2917–2925. 159

Maaløe, L., Sønderby, C. K., Sønderby, S. K., & Winther, O. (2016). Auxiliary deep generative models. *International Conference on Machine Learning*, 1445–1453. 344, 345

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. *ICML Workshop on Deep Learning for Audio, Speech, and Language Processing*. 38

MacKay, D. J. (1995). Ensemble learning and evidence maximization. *Neural Information Processing Systems*, vol. 8, 4083–4090. 159

MacKay, M., Vicol, P., Ba, J., & Grosse, R. B. (2018). Reversible recurrent neural networks. *Neural Information Processing Systems*, *31*, 9043–9054. 322

Mackowiak, R., Ardizzone, L., Kothe, U., & Rother, C. (2021). Generative classifiers as a basis for trustworthy image classification. *IEEE/CVF Computer Vision & Pattern Recognition*, 2971–2981. 322

Madhawa, K., Ishiguro, K., Nakago, K., & Abe, M. (2019). GraphNVP: An invertible flow model for generating molecular graphs. *arXiv:1905.11600*. 322

Mahendran, A., & Vedaldi, A. (2015). Understanding deep image representations by inverting them. *IEEE/CVF Computer Vision & Pattern Recognition*, 5188–5196. 184

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., & Frey, B. (2015). Adversarial autoencoders. *arXiv:1511.05644*. 345

Mangalam, K., Fan, H., Li, Y., Wu, C.-Y., Xiong, B., Feichtenhofer, C., & Malik, J. (2022). Reversible vision transformers. *IEEE/CVF Computer Vision & Pattern Recognition*, 10830–10840. 322

Manning, C., & Schutze, H. (1999). *Foundations of statistical natural language processing*. MIT Press. 233

Manyika, J., Lund, S., Chui, M., Bughin, J., Woetzel, J., Batra, P., Ko, R., & Sanghvi, S. (2017). *Jobs Lost, Jobs Gained: Workforce Transitions in a Time of Automation*. McKinsey Global Institute. 429

Manyika, J., & Sneader, K. (2018). *AI, automation, and the future of work: Ten things to solve for*. McKinsey Global Institute. 429

Mao, Q., Lee, H.-Y., Tseng, H.-Y., Ma, S., & Yang, M.-H. (2019). Mode seeking generative adversarial networks for diverse image synthesis. *IEEE/CVF Computer Vision & Pattern Recognition*, 1429–1437. 300

Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z., & Paul Smolley, S. (2017). Least squares generative adversarial networks. *IEEE/CVF International Conference on Computer Vision*, 2794–2802. 299

Marchesi, M. (2017). Megapixel size image creation using generative adversarial networks. *arXiv:1706.00082*. 299

Martin, G. L. (1993). Centered-object integrated segmentation and recognition of overlapping handprinted characters. *Neural Computation*, *5*(3), 419–429. 181

Masci, J., Boscaini, D., Bronstein, M., & Vandergheynst, P. (2015). Geodesic convolutional neural networks on Riemannian manifolds. *IEEE International Conference on Computer Vision Workshop*, 832–840. 265

Masrani, V., Le, T. A., & Wood, F. (2019). The thermodynamic variational objective. *Neural Information Processing Systems*, *32*, 11521–11530. 346

Mathieu, E., Rainforth, T., Siddharth, N., & Teh, Y. W. (2019). Disentangling disentanglement in variational autoencoders. *International Conference on Machine Learning*, 4402–4412. 346

Matsakis, L. (2017). A frightening AI can determine whether a person is gay with 91 percent accuracy. Vice, Sept 8, 2017. `https://www.vice.com/en/article/a33xb4/a-frightening-ai-can-determine-a-persons-sexuality-with-91-accuracy`. 430

Maturana, D., & Scherer, S. (2015). VoxNet: A 3D convolutional neural network for real-time object recognition. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 922–928. 182

Mayson, S. G. (2018). Bias in bias out. *Yale Law Journal*, *128*, 2122–2473. 422

Mazoure, B., Doan, T., Durand, A., Pineau, J., & Hjelm, R. D. (2020). Leveraging exploration in off-policy algorithms via normalizing flows. *Conference on Robot Learning*, 430–444. 322

Mazyavkina, N., Sviridov, S., Ivanov, S., & Burnaev, E. (2021). Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research*, *134*, 105400. 396

McCoy, R. T., Pavlick, E., & Linzen, T. (2019). Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. *Meeting of the Association for Computational Linguistics*, 2428–3448. 234

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, *5*(4), 115–133. 37

McNamara, A., Smith, J., & Murphy-Hill, E. (2018). Does ACM's code of ethics change ethical decision making in software development? *ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 729–733. 420

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., & Galstyan, A. (2022). A survey on bias and fairness in machine learning. *ACM Computing Surveys*, *54*(6), 1–35. 423

Mei, J., Chung, W., Thomas, V., Dai, B., Szepesvári, C., & Schuurmans, D. (2022). The role of baselines in policy gradient optimization. *Neural Information Processing Systems*, vol. 35, 17818–17830. 397

Meng, C., Song, Y., Song, J., Wu, J., Zhu, J.-Y., & Ermon, S. (2021). SDEdit: Image synthesis and editing with stochastic differential equations. *International Conference on Learning Representations*. 369

Menon, S., Damian, A., Hu, S., Ravi, N., & Rudin, C. (2020). PULSE: self-supervised photo upsampling via latent space exploration of generative models. *IEEE/CVF Computer Vision & Pattern Recognition*, 2434–2442. 422

Metcalf, J., Keller, E. F., & Boyd, D. (2016). Perspectives on big data, ethics, and society. *Council for Big Data, Ethics, and Society*. `https://bdes.datasociety.net/council-output/perspectives-on-big-data-ethics-and-society/`. 430

Metz, L., Poole, B., Pfau, D., & Sohl-Dickstein, J. (2017). Unrolled generative adversarial networks. *International Conference on Learning Representations*. 299

Mézard, M., & Mora, T. (2009). Constraint satisfaction problems and neural networks: A statistical physics perspective. *Journal of Physiology-Paris*, *103*(1-2), 107–113. 94

Micelli, M., Posada, J., & Yang, T. (2022). Studying up machine learning data: Why talk about bias when we mean power? *Proceedngs of ACM on Human-Computer Interaction*, *6*. 423

Milletari, F., Navab, N., & Ahmadi, S.-A. (2016). V-Net: Fully convolutional neural networks for volumetric medical image segmentation. *International Conference on 3D Vision*, 565–571. 205

Min, J., McCoy, R. T., Das, D., Pitler, E., & Linzen, T. (2020). Syntactic data augmentation increases robustness to inference heuristics. *Meeting of the Association for Computational Linguistics*, 2339–2352. 160

Minaee, S., Boykov, Y. Y., Porikli, F., Plaza, A. J., Kehtarnavaz, N., & Terzopoulos, D. (2021). Image segmentation using deep learning: A survey. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, *44*(7), 3523–3542. 184

Minsky, M., & Papert, S. A. (1969). *Perceptrons: An introduction to computational geometry*. MIT Press. 37, 233

Mireshghallah, F., Taram, M., Vepakomma, P., Singh, A., Raskar, R., & Esmaeilzadeh, H. (2020). Privacy in deep learning: A survey. *arXiv:2004.12254*. 428

Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets. *arXiv:1411.1784*. 301

Mishkin, D., & Matas, J. (2016). All you need is a good init. *International Conference on Learning Representations*. 113

Mitchell, M., Forrest, S., & Holland, J. H. (1992). The royal road for genetic algorithms: Fitness landscapes and GA performance. *European Conference on Artificial Life*. 421

Mitchell, S., Potash, E., Barocas, S., D'Amour, A., & Lum, K. (2021). Algorithmic fairness: Choices, assumptions, and definitions. *Annual Review of Statistics and Its Application*, *8*, 141–163. 422

Miyato, T., Kataoka, T., Koyama, M., & Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. *International Conference on Learning Representations*. 299

Miyato, T., & Koyama, M. (2018). cGANs with projection discriminator. *International Conference on Learning Representations*. 301

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., & Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. *International Conference on Machine Learning*, 1928–1937. 398

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533. 396

Moerland, T. M., Broekens, J., Plaat, A., Jonker, C. M., et al. (2023). Model-based reinforcement learning: A survey. *Foundations and Trends in Machine Learning*, *16*(1), 1–118. 398

Mogren, O. (2016). C-RNN-GAN: Continuous recurrent neural networks with adversarial training. *NIPS 2016 Constructive Machine Learning Workshop*. 299

Molnar, C. (2022). *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*. `https://christophm.github.io/interpretable-ml-book`. 425

Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., & Bronstein, M. M. (2017). Geometric deep learning on graphs and manifolds using mixture model CNNs. *IEEE/CVF Computer Vision & Pattern Recognition*, 5115–5124. 263, 265

Monti, F., Shchur, O., Bojchevski, A., Litany, O., Günnemann, S., & Bronstein, M. M. (2018). Dual-primal graph convolutional networks. *arXiv:1806.00770*. 264

Montúfar, G. (2017). Notes on the number of linear regions of deep neural networks. 52, 53

Montúfar, G. F., Pascanu, R., Cho, K., & Bengio, Y. (2014). On the number of linear regions

of deep neural networks. *Neural Information Processing Systems*, *27*, 2924–2932. 52, 53

Moor, J. (2006). The nature, importance, and difficulty of machine ethics. *Intelligence Systems*, *21*(4), 18–21. 424

Moore, A., & Himma, K. (2022). Intellectual Property. *The Stanford Encyclopedia of Philosophy*. 428

Moreno-Torres, J. G., Raeder, T., Alaiz-Rodríguez, R., Chawla, N. V., & Herrera, F. (2012). A unifying view on dataset shift in classification. *Pattern Recognition*, *45*(1), 521–530. 135

Morimura, T., Sugiyama, M., Kashima, H., Hachiya, H., & Tanaka, T. (2010). Nonparametric return distribution approximation for reinforcement learning. *International Conference on Machine Learning*, 799–806. 397

Müller, R., Kornblith, S., & Hinton, G. E. (2019a). When does label smoothing help? *Neural Information Processing Systems*, *32*, 4696–4705. 158

Müller, T., McWilliams, B., Rousselle, F., Gross, M., & Novák, J. (2019b). Neural importance sampling. *ACM Transactions on Graphics (TOG)*, *38*(5), 1–19. 322, 323

Mun, S., Shon, S., Kim, W., Han, D. K., & Ko, H. (2017). Deep neural network based learning and transferring mid-level audio features for acoustic scene classification. *IEEE International Conference on Acoustics, Speech and Signal Processing*, 796–800. 160

Murphy, K. P. (2022). *Probabilistic machine learning: An introduction*. MIT Press. 15

Murphy, K. P. (2023). *Probabilistic machine learning: Advanced topics*. MIT Press. 15

Murphy, R. L., Srinivasan, B., Rao, V., & Ribeiro, B. (2018). Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs. *International Conference on Learning Representations*. 263

Murty, K. G., & Kabadi, S. N. (1987). Some NP-complete problems in quadratic and nonlinear programming. *Mathematical Programming*, *39*(2), 117–129. 401

Mutlu, E. C., Oghaz, T., Rajabi, A., & Garibay, I. (2020). Review on learning and extracting graph features for link prediction. *Machine Learning and Knowledge Extraction*, *2*(4), 672–704. 262

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. *International Conference on Machine Learning*, 807–814. 37

Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., & Sutskever, I. (2021). Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, *2021*(12), 124003. 130, 134

Narang, S., Chung, H. W., Tay, Y., Fedus, W., Fevry, T., Matena, M., Malkan, K., Fiedel, N., Shazeer, N., Lan, Z., et al. (2021). Do transformer modifications transfer across implementations and applications? *Empirical Methods in Natural Language Processing*, 5758–5773. 233

Narayanan, A., & Shmatikov, V. (2008). Robust de-anonymization of large sparse datasets. *IEEE Symposium on Security and Privacy*, 111–125. 428

Narayanan, D., Phanishayee, A., Shi, K., Chen, X., & Zaharia, M. (2021a). Memory-efficient pipeline-parallel DNN training. *International Conference on Machine Learning*, 7937–7947. 114

Narayanan, D., Shoeybi, M., Casper, J., LeGresley, P., Patwary, M., Korthikanti, V., Vainbrand, D., Kashinkunti, P., Bernauer, J., Catanzaro, B., et al. (2021b). Efficient large-scale language model training on GPU clusters using Megatron-LM. *International Conference for High Performance Computing, Networking, Storage and Analysis*, 1–15. 114

Nash, C., Menick, J., Dieleman, S., & Battaglia, P. W. (2021). Generating images with sparse representations. *International Conference on Machine Learning*, 7958–7968. 238, 274

Neal, R. M. (1995). *Bayesian learning for neural networks*. Springer. 159

Neimark, D., Bar, O., Zohar, M., & Asselmann, D. (2021). Video transformer network. *IEEE/CVF International Conference on Computer Vision*, 3163–3172. 238

Nesterov, Y. E. (1983). A method for solving the convex programming problem with convergence rate. *Doklady Akademii Nauk SSSR*, vol. 269, 543–547. 93

Newell, A., Yang, K., & Deng, J. (2016). Stacked hourglass networks for human pose estimation. *European Conference on Computer Vision*, 483–499. 200, 205

Neyshabur, B., Bhojanapalli, S., McAllester, D., & Srebro, N. (2017). Exploring generalization in deep learning. *Neural Information Processing Systems*, *30*, 5947–5956. 134, 412

Neyshabur, B., Bhojanapalli, S., & Srebro, N. (2018). A PAC-Bayesian approach to

spectrally-normalized margin bounds for neural networks. *International Conference on Learning Representations*. 156

Ng, N. H., Gabriel, R. A., McAuley, J., Elkan, C., & Lipton, Z. C. (2017). Predicting surgery duration with neural heteroscedastic regression. *PMLR Machine Learning for Healthcare Conference*, 100–111. 74

Nguyen, Q., & Hein, M. (2017). The loss surface of deep and wide neural networks. *International Conference on Machine Learning*, 2603–2612. 405

Nguyen, Q., & Hein, M. (2018). Optimization landscape and expressivity of deep CNNs. *International Conference on Machine Learning*, 3730–3739. 405

Nichol, A. Q., & Dhariwal, P. (2021). Improved denoising diffusion probabilistic models. *International Conference on Machine Learning*, 8162–8171. 369

Nichol, A. Q., Dhariwal, P., Ramesh, A., Shyam, P., Mishkin, P., McGrew, B., Sutskever, I., & Chen, M. (2022). GLIDE: towards photorealistic image generation and editing with text-guided diffusion models. *International Conference on Machine Learning*, 16784–16804. 369, 370

Nie, W., Guo, B., Huang, Y., Xiao, C., Vahdat, A., & Anandkumar, A. (2022). Diffusion models for adversarial purification. *International Conference on Machine Learning*, 16805–16827. 369

Nix, D. A., & Weigend, A. S. (1994). Estimating the mean and variance of the target probability distribution. *IEEE International Conference on Neural Networks*, 55–60. 73

Noble, S. (2018). *Algorithms of Oppression*. New York: NYU Press. 433

Noci, L., Roth, K., Bachmann, G., Nowozin, S., & Hofmann, T. (2021). Disentangling the roles of curation, data-augmentation and the prior in the cold posterior effect. *Neural Information Processing Systems*, *34*, 12738–12748. 159

Noé, F., Olsson, S., Köhler, J., & Wu, H. (2019). Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, *365*(6457). 322

Noh, H., Hong, S., & Han, B. (2015). Learning deconvolution network for semantic segmentation. *IEEE International Conference on Computer Vision*, 1520–1528. 6, 179, 180, 184

Noothigattu, R., Gaikwad, S. N., Awad, E., Dsouza, S., Rahwan, I., Ravikumar, P., & Procaccia, A. D. (2018). A voting-based system for ethical decision making. *AAAI Portuguese Conference on Artificial Intelligence*, 1587–1594. 424

Noroozi, M., & Favaro, P. (2016). Unsupervised learning of visual representations by solving jigsaw puzzles. *European Conference on Computer Vision*, 69–84. 159

Nowozin, S., Cseke, B., & Tomioka, R. (2016). f-GAN: Training generative neural samplers using variational divergence minimization. *Neural Information Processing Systems*, *29*, 271–279. 299

Nye, M., & Saxe, A. (2018). Are efficient deep representations learnable? *International Conference on Learning Representations (Workshop)*. 417

O'Connor, C., & Bruner, J. (2019). Dynamics and diversity in epistemic communities. *Erkenntnis*, *84*, 101–119. 433

Odena, A. (2019). Open questions about generative adversarial networks. Distill, `https://distill.pub/2019/gan-open-problems`. 299

Odena, A., Dumoulin, V., & Olah, C. (2016). Deconvolution and checkerboard artifacts. Distill, `https://distill.pub/2016/deconv-checkerboard/`. 181

Odena, A., Olah, C., & Shlens, J. (2017). Conditional image synthesis with auxiliary classifier GANs. *International Conference on Machine Learning*, 2642–2651. 290, 301

O'Neil, C. (2016). *Weapons of Math Destruction*. Crown. 420, 422

Oono, K., & Suzuki, T. (2019). Graph neural networks exponentially lose expressive power for node classification. *International Conference on Learning Representations*. 265

Orhan, A. E., & Pitkow, X. (2017). Skip connections eliminate singularities. *International Conference on Learning Representations*. 202

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. (2022). Training language models to follow instructions with human feedback. *Neural Information Processing Systems*, *35*, 27730–27744. 398

Papamakarios, G., Nalisnick, E. T., Rezende, D. J., Mohamed, S., & Lakshminarayanan, B. (2021). Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, *22*(57), 1–64. 321

Papamakarios, G., Pavlakou, T., & Murray, I. (2017). Masked autoregressive flow for density estimation. *Neural Information Processing Systems*, *30*, 2338–2347. 323

Park, D., Hoshi, Y., & Kemp, C. C. (2018). A multimodal anomaly detector for robot-assisted feeding using an LSTM-based variational autoencoder. *IEEE Robotics and Automation Letters*, *3*(3), 1544–1551. 344

Park, D. S., Chan, W., Zhang, Y., Chiu, C.-C., Zoph, B., Cubuk, E. D., & Le, Q. V. (2019). SpecAugment: A simple data augmentation method for automatic speech recognition. *INTERSPEECH*. 160

Park, S., & Kwak, N. (2016). Analysis on the dropout effect in convolutional neural networks. *Asian Conference on Computer Vision*, 189–204. 183

Park, S.-W., Ko, J.-S., Huh, J.-H., & Kim, J.-C. (2021). Review on generative adversarial networks: Focusing on computer vision and its applications. *Electronics*, *10*(10), 1216. 299

Parker, D. B. (1985). *Learning-logic: Casting the cortex of the human brain in silicon*. Alfred P. Sloan School of Management, MIT. 113

Parmar, N., Ramachandran, P., Vaswani, A., Bello, I., Levskaya, A., & Shlens, J. (2019). Stand-alone self-attention in vision models. *Neural Information Processing Systems*, *32*, 68–80. 238

Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A., & Tran, D. (2018). Image transformer. *International Conference on Machine Learning*, 4055–4064. 238

Pascanu, R., Dauphin, Y. N., Ganguli, S., & Bengio, Y. (2014). On the saddle point problem for non-convex optimization. *arXiv:1405.4604*. 405

Pascanu, R., Montúfar, G., & Bengio, Y. (2013). On the number of response regions of deep feed forward networks with piece-wise linear activations. *arXiv:1312.6098*. 53

Paschalidou, D., Katharopoulos, A., Geiger, A., & Fidler, S. (2021). Neural parts: Learning expressive 3D shape abstractions with invertible neural networks. *IEEE/CVF Computer Vision & Pattern Recognition*, 3204–3215. 322

Patashnik, O., Wu, Z., Shechtman, E., Cohen-Or, D., & Lischinski, D. (2021). StyleCLIP: Text-driven manipulation of StyleGAN imagery. *IEEE/CVF International Conference on Computer Vision*, 2085–2094. 300

Pateria, S., Subagdja, B., Tan, A.-h., & Quek, C. (2021). Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Surveys*, *54*(5), 1–35. 398

Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., & Efros, A. A. (2016). Context encoders: Feature learning by inpainting. *IEEE/CVF Computer Vision & Pattern Recognition*, 2536–2544. 159

Patrick, M., Campbell, D., Asano, Y., Misra, I., Metze, F., Feichtenhofer, C., Vedaldi, A., & Henriques, J. F. (2021). Keeping your eye on the ball: Trajectory attention in video transformers. *Neural Information Processing Systems*, *34*, 12493–12506. 238

Peluchetti, S., & Favaro, S. (2020). Infinitely deep neural networks as diffusion processes. *International Conference on Artificial Intelligence and Statistics*, 1126–1136. 324

Peng, C., Guo, P., Zhou, S. K., Patel, V., & Chellappa, R. (2022). Towards performant and reliable undersampled MR reconstruction via diffusion model sampling. *Medical Image Computing and Computer Assisted Intervention*, *13436*, 623–633. 369

Pennington, J., & Bahri, Y. (2017). Geometry of neural network loss surfaces via random matrix theory. *International Conference on Machine Learning*, 2798–2806. 405

Perarnau, G., Van De Weijer, J., Raducanu, B., & Álvarez, J. M. (2016). Invertible conditional GANs for image editing. *NIPS 2016 Workshop on Adversarial Training*. 301

Pereyra, G., Tucker, G., Chorowski, J., Kaiser, ., & Hinton, G. (2017). Regularizing neural networks by penalizing confident output distributions. *International Conference on Learning Representations Workshop*. 158

Peters, J., & Schaal, S. (2008). Reinforcement learning of motor skills with policy gradients. *Neural Networks*, *21*(4), 682–697. 397

Peyré, G., Cuturi, M., et al. (2019). Computational optimal transport with applications to data science. *Foundations and Trends in Machine Learning*, *11*(5-6), 355–607. 299

Pezeshki, M., Mitra, A., Bengio, Y., & Lajoie, G. (2022). Multi-scale feature learning dynamics: Insights for double descent. *International Conference on Machine Learning*, 17669–17690. 134

Pham, T., Tran, T., Phung, D., & Venkatesh, S. (2017). Column networks for collective classification. *AAAI Conference on Artificial Intelligence*, 2485–2491. 263

Phuong, M., & Hutter, M. (2022). Formal algorithms for transformers. *Technical Report, DeepMind*. 233

Pieters, M., & Wiering, M. (2018). Comparing generative adversarial network techniques for image creation and modification. *arXiv:1803.09093*. 299

Pintea, S. L., Tömen, N., Goes, S. F., Loog, M., & van Gemert, J. C. (2021). Resolution learning in deep convolutional networks using scale-space theory. *IEEE Transactions on Image Processing*, *30*, 8342–8353. 183

Poggio, T., Mhaskar, H., Rosasco, L., Miranda, B., & Liao, Q. (2017). Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review. *International Journal of Automation and Computing*, *14*(5), 503–519. 53

Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, *4*(5), 1–17. 92

Poole, B., Jain, A., Barron, J. T., & Mildenhall, B. (2023). DreamFusion: Text-to-3D using 2D diffusion. *International Conference on Learning Representations*. 369

Power, A., Burda, Y., Edwards, H., Babuschkin, I., & Misra, V. (2022). Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv:2201.02177*. 412

Prenger, R., Valle, R., & Catanzaro, B. (2019). Waveglow: A flow-based generative network for speech synthesis. *IEEE International Conference on Acoustics, Speech and Signal Processing*, 3617–3621. 322, 323

Prince, S. J. D. (2012). *Computer vision: Models, learning, and inference*. Cambridge University Press. 15, 159

Prince, S. J. D. (2021a). Transformers II: Extensions. `https://www.borealisai.com/en/blog/tutorial-16-transformers-ii-extensions/`. 236, 237

Prince, S. J. D. (2021b). Transformers III: Training. `https://www.borealisai.com/en/blog/tutorial-17-transformers-iii-training/`. 238

Prince, S. J. D. (2022). Explainability I: local post-hoc explanations. `https://www.borealisai.com/research-blogs/explainability-i-local-post-hoc-explanations/`. 426

Prokudin, S., Gehler, P., & Nowozin, S. (2018). Deep directional statistics: Pose estimation with uncertainty quantification. *European Conference on Computer Vision*, 534–551. 74

Provilkov, I., Emelianenko, D., & Voita, E. (2020). BPE-Dropout: Simple and effective subword regularization. *Meeting of the Association for Computational Linguistics*, 1882–1892. 234

Qi, G.-J. (2020). Loss-sensitive generative adversarial networks on Lipschitz densities. *International Journal of Computer Vision*, *128*(5), 1118–1140. 299

Qi, J., Du, J., Siniscalchi, S. M., Ma, X., & Lee, C.-H. (2020). On mean absolute error for deep neural network based vector-to-vector regression. *IEEE Signal Processing Letters*, *27*, 1485—1489. 73

Qin, Z., Yu, F., Liu, C., & Chen, X. (2018). How convolutional neural network see the world — A survey of convolutional neural network visualization methods. *arXiv:1804.11191*. 184

Qiu, S., Xu, B., Zhang, J., Wang, Y., Shen, X., De Melo, G., Long, C., & Li, X. (2020). EasyAug: An automatic textual data augmentation platform for classification tasks. *Companion Proceedings of the Web Conference 2020*, 249–252. 160

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. *International Conference on Machine Learning*, 8748–8763. 238, 370

Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *International Conference on Learning Representations*. 280, 299

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI Blog*, *1*(8), 9. 159, 234

Rae, J. W., Borgeaud, S., Cai, T., Millican, K., Hoffmann, J., Song, F., Aslanides, J., Henderson, S., Ring, R., Young, S., et al. (2021). Scaling language models: Methods, analysis & insights from training Gopher. *arXiv:2112.11446*. 234

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P. J., et al. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, *21*(140), 1–67. 236

Raji, I. D., & Buolamwini, J. (2019). Actionable auditing: Investigating the impact of publicly naming biased performance results of commercial AI products. *AAAI/ACM Conference on AI, Ethics, and Society*, 429–435. 423

Raji, I. D., & Fried, G. (2020). About face: A survey of facial recognition evaluation. *AAAI Workshop on AI Evaluation*. 427

Raji, I. D., Kumar, I. E., Horowitz, A., & Selbst, A. (2022). The fallacy of AI functionality. *ACM Conference on Fairness, Accountability, and Transparency*, 959–972. 423, 427

Rajpurkar, P., Chen, E., Banerjee, O., & Topol, E. J. (2022). AI in health and medicine. *Nature Medicine*, *28*(1), 31–38. 420

Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). SQuAD: 100,000+ questions for machine comprehension of text. *Empirical Methods in Natural Language Processing*, 2383–2392. 234

Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for activation functions. *arXiv:1710.05941*. 38

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., & Chen, M. (2022). Hierarchical text-conditional image generation with CLIP latents. *arXiv:2204.06125*. 10, 11, 238, 369, 370

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., & Sutskever, I. (2021). Zero-shot text-to-image generation. *International Conference on Machine Learning*, 8821–8831. 238, 370

Ramsauer, H., Schäfl, B., Lehner, J., Seidl, P., Widrich, M., Adler, T., Gruber, L., Holzleitner, M., Pavlović, M., Sandve, G. K., et al. (2021). Hopfield networks is all you need. *International Conference on Learning Representations*. 236

Ranganath, R., Tran, D., & Blei, D. (2016). Hierarchical variational models. *International Conference on Machine Learning*, 324–333. 345

Ravanbakhsh, S., Lanusse, F., Mandelbaum, R., Schneider, J., & Poczos, B. (2017). Enabling dark energy science with deep generative models of galaxy images. *AAAI Conference on Artificial Intelligence*, 1488–1494. 344

Rawat, W., & Wang, Z. (2017). Deep convolutional neural networks for image classification: A comprehensive review. *Neural Computation*, *29*(9), 2352–2449. 181

Rawls, J. (1971). *A Theory of Justice*. Belknap Press. 430

Razavi, A., Oord, A. v. d., Poole, B., & Vinyals, O. (2019a). Preventing posterior collapse with delta-VAEs. *International Conference on Learning Representations*. 345

Razavi, A., Van den Oord, A., & Vinyals, O. (2019b). Generating diverse high-fidelity images with VQ-VAE-2. *Neural Information Processing Systems*, *32*, 14837–14847. 344, 345

Recht, B., Re, C., Wright, S., & Niu, F. (2011). Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. *Neural Information Processing Systems*, *24*, 693–701. 114

Reddi, S. J., Kale, S., & Kumar, S. (2018). On the convergence of Adam and beyond. *International Conference on Learning Representations*. 93

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *IEEE/CVF Computer Vision & Pattern Recognition*, 779–788. 178, 184

Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., & Lee, H. (2016a). Generative adversarial text to image synthesis. *International Conference on Machine Learning*, 1060–1069. 301

Reed, S. E., Akata, Z., Mohan, S., Tenka, S., Schiele, B., & Lee, H. (2016b). Learning what and where to draw. *Neural Information Processing Systems*, *29*, 217–225. 301

Reiss, J., & Sprenger, J. (2017). Scientific Objectivity. *The Stanford Encyclopedia of Philosophy*. 431

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. *Neural Information Processing Systems*, *28*. 183

Rezende, D. J., & Mohamed, S. (2015). Variational inference with normalizing flows. *International Conference on Machine Learning*, 1530–1538. 273, 321, 322, 344

Rezende, D. J., Mohamed, S., & Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. *International Conference on Machine Learning*, 1278–1286. 346

Rezende, D. J., Racanière, S., Higgins, I., & Toth, P. (2019). Equivariant Hamiltonian flows. *arXiv:1909.13739*. 324

Rezende Jimenez, D., Eslami, S., Mohamed, S., Battaglia, P., Jaderberg, M., & Heess, N. (2016). Unsupervised learning of 3D structure

from images. *Neural Information Processing Systems*, *29*, 4997–5005. 344

Riad, R., Teboul, O., Grangier, D., & Zeghidour, N. (2022). Learning strides in convolutional neural networks. *International Conference on Learning Representations*. 183

Ribeiro, M., Singh, S., & Guestrin, C. (2016). "Why should I trust you?": Explaining the predictions of any classifier. *Meeting of the Association for Computational Linguistics*, 97–101. 425

Ribeiro, M. T., Wu, T., Guestrin, C., & Singh, S. (2021). Beyond accuracy: Behavioral testing of NLP models with CheckList. 4824–4828. 234

Richardson, E., Alaluf, Y., Patashnik, O., Nitzan, Y., Azar, Y., Shapiro, S., & Cohen-Or, D. (2021). Encoding in style: A StyleGAN encoder for image-to-image translation. *IEEE/CVF Computer Vision & Pattern Recognition*, 2287–2296. 301

Riedl, M. (2020). AI democratization in the era of GPT-3. The Gradient, Sept 25, 2020. `https://thegradient.pub/ai-democratization-in-the-era-of-gpt-3/`. 430

Riedmiller, M. (2005). Neural fitted Q iteration — first experiences with a data efficient neural reinforcement learning method. *European Conference on Machine Learning*, 317–328. 396

Rippel, O., & Adams, R. P. (2013). High-dimensional probability estimation with deep density models. *arXiv:1302.5125*. 321

Rissanen, J. (1983). A universal prior for integers and estimation by minimum description length. *The Annals of Statistics*, *11*(2), 416–431. 411

Rissanen, S., Heinonen, M., & Solin, A. (2022). Generative modelling with inverse heat dissipation. *arXiv:2206.13397*. 369

Rives, A., Meier, J., Sercu, T., Goyal, S., Lin, Z., Liu, J., Guo, D., Ott, M., Zitnick, C. L., Ma, J., et al. (2021). Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, *118*(15). 234

Robbins, H., & Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, *22*(3), 400–407. 91

Rodrigues, F., & Pereira, F. C. (2020). Beyond expectation: Deep joint mean and quantile regression for spatiotemporal problems. *IEEE Transactions on Neural Networks and Learning Systems*, *31*(12), 5377–5389. 73

Roeder, G., Wu, Y., & Duvenaud, D. K. (2017). Sticking the landing: Simple, lower-variance gradient estimators for variational inference. *Neural Information Processing Systems*, *30*, 6925–6934. 346

Roich, D., Mokady, R., Bermano, A. H., & Cohen-Or, D. (2022). Pivotal tuning for latent-based editing of real images. *ACM Transactions on Graphics (TOG)*, *42*(1), 1–13. 300, 301

Rolfe, J. T. (2017). Discrete variational autoencoders. *International Conference on Learning Representations*. 344

Rolnick, D., Donti, P. L., Kaack, L. H., Kochanski, K., Lacoste, A., Sankaran, K., Ross, A. S., Milojevic-Dupont, N., Jaques, N., Waldman-Brown, A., Luccioni, A. S., Maharaj, T., Sherwin, E. D., Mukkavilli, S. K., Kording, K. P., Gomes, C. P., Ng, A. Y., Hassabis, D., Platt, J. C., Creutzig, F., Chayes, J. T., & Bengio, Y. (2023). Tackling climate change with machine learning. *ACM Computing Surveys*, *55*(2), 1–42. 420

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. *IEEE/CVF Computer Vision & Pattern Recognition*, 10684–10695. 370

Romero, D. W., Bruintjes, R.-J., Tomczak, J. M., Bekkers, E. J., Hoogendoorn, M., & van Gemert, J. C. (2021). FlexConv: Continuous kernel convolutions with differentiable kernel sizes. *International Conference on Learning Representations*. 183

Rong, Y., Huang, W., Xu, T., & Huang, J. (2020). DropEdge: Towards deep graph convolutional networks on node classification. *International Conference on Learning Representations*. 264

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 234–241. 184, 198, 205

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, *65*(6), 386. 37

Rossi, E., Frasca, F., Chamberlain, B., Eynard, D., Bronstein, M., & Monti, F. (2020). SIGN: Scalable inception graph neural networks. *ICML Graph Representation Learning and Beyond Workshop*, *7*, 15. 263

Roy, A., Saffar, M., Vaswani, A., & Grangier, D. (2021). Efficient content-based sparse attention with routing transformers. *Transactions*

of the Association for Computational Linguistics, *9*, 53–68. 237

Rozemberczki, B., Kiss, O., & Sarkar, R. (2020). Little ball of fur: A Python library for graph sampling. *ACM International Conference on Information & Knowledge Management*, 3133–3140. 264

Rubin, D. B., & Thayer, D. T. (1982). EM algorithms for ML factor analysis. *Psychometrika*, *47*(1), 69–76. 344

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv:1609.04747*. 91

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). Learning internal representations by error propagation. *Techical Report, La Jolla Institute for Cognitive Science, UCSD*. 113, 233, 344

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. 113

Rummery, G. A., & Niranjan, M. (1994). *On-line Q-learning using connectionist systems*. Technical Report, University of Cambridge. 396

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, *115*(3), 211–252. 175, 181

Russell, S. (2019). *Human Compatible: Artificial Intelligence and the Problem of Control*. Viking. 420

Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic routing between capsules. *Neural Information Processing Systems*, *30*, 3856–3866. 235

Safran, I., & Shamir, O. (2017). Depth-width tradeoffs in approximating natural functions with neural networks. *International Conference on Machine Learning*, 2979–2987. 53

Saha, S., Singh, G., Sapienza, M., Torr, P. H., & Cuzzolin, F. (2016). Deep learning for detecting multiple space-time action tubes in videos. *British Machine Vision Conference*. 182

Saharia, C., Chan, W., Chang, H., Lee, C., Ho, J., Salimans, T., Fleet, D., & Norouzi, M. (2022a). Palette: Image-to-image diffusion models. *ACM SIGGRAPH*. 8, 369

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S. K. S., Ayan, B. K., Mahdavi, S. S., Lopes, R. G., et al. (2022b). Photorealistic text-to-image diffusion models with deep language understanding. *arXiv:2205.11487*. 366, 368, 369, 371

Saharia, C., Ho, J., Chan, W., Salimans, T., Fleet, D. J., & Norouzi, M. (2022c). Image super-resolution via iterative refinement. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 1–14. 369

Sainath, T. N., Kingsbury, B., Mohamed, A.-r., Dahl, G. E., Saon, G., Soltau, H., Beran, T., Aravkin, A. Y., & Ramabhadran, B. (2013). Improvements to deep convolutional neural networks for LVCSR. *IEEE Workshop on Automatic Speech Recognition and Understanding*, 315–320. 182

Saito, Y., Takamichi, S., & Saruwatari, H. (2017). Statistical parametric speech synthesis incorporating generative adversarial networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *26*(1), 84–96. 299, 301

Salamon, J., & Bello, J. P. (2017). Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, *24*(3), 279–283. 160

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016). Improved techniques for training GANs. *Neural Information Processing Systems*, *29*, 2226–2234. 274, 299, 300

Salimans, T., & Ho, J. (2022). Progressive distillation for fast sampling of diffusion models. *International Conference on Learning Representations*. 370

Salimans, T., Kingma, D., & Welling, M. (2015). Markov chain Monte Carlo and variational inference: Bridging the gap. *International Conference on Machine Learning*, 1218–1226. 345

Salimans, T., & Kingma, D. P. (2016). Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Neural Information Processing Systems*, *29*, 901–909. 204

Sanchez-Lengeling, B., Reif, E., Pearce, A., & Wiltschko, A. B. (2021). A gentle introduction to graph neural networks. Distill, `https://distill.pub/2021/gnn-intro/`. 261

Sankararaman, K. A., De, S., Xu, Z., Huang, W. R., & Goldstein, T. (2020). The impact of neural network overparameterization on gradient confusion and stochastic gradient descent. *International Conference on Machine Learning*, 8469–8479. 202

Santurkar, S., Tsipras, D., Ilyas, A., & Madry, A. (2018). How does batch normalization help optimization? *Neural Information Processing Systems*, *31*, 2488–2498. 204

Sauer, A., Schwarz, K., & Geiger, A. (2022). StyleGAN-XL: Scaling StyleGAN to large diverse datasets. *ACM SIGGRAPH*. 10

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2008). The graph neural network model. *IEEE Transactions on Neural Networks*, *20*(1), 61–80. 262

Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2016). Prioritized experience replay. *International Conference on Learning Representations*. 396

Scherer, D., Müller, A., & Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. *International Conference on Artificial Neural Networks*, 92–101. 181

Schlag, I., Irie, K., & Schmidhuber, J. (2021). Linear transformers are secretly fast weight programmers. *International Conference on Machine Learning*, 9355–9366. 235

Schlichtkrull, M., Kipf, T. N., Bloem, P., Berg, R. v. d., Titov, I., & Welling, M. (2018). Modeling relational data with graph convolutional networks. *European Semantic Web Conference*, 593–607. 265

Schmidhuber, J. (2022). Annotated history of modern AI and deep learning. *arXiv:2212.11279*. 37

Schneider, S., Baevski, A., Collobert, R., & Auli, M. (2019). wav2vec: Unsupervised pre-training for speech recognition. *INTERSPEECH*, 3465–3469. 159

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. (2020). Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, *588*(7839), 604–609. 398

Schroecker, Y., Vecerik, M., & Scholz, J. (2019). Generative predecessor models for sample-efficient imitation learning. *International Conference on Learning Representations*. 322

Schuhmann, C., Vencu, R., Beaumont, R., Kaczmarczyk, R., Mullis, C., Katta, A., Coombes, T., Jitsev, J., & Komatsuzaki, A. (2021). Laion-400m: Open dataset of clip-filtered 400 million image-text pairs. *NeurIPS Workshop on Data-centric AI*. 238

Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015). Trust region policy optimization. *International Conference on Machine Learning*, 1889–1897. 397

Schulman, J., Moritz, P., Levine, S., Jordan, M., & Abbeel, P. (2016). High-dimensional continuous control using generalized advantage estimation. *International Conference on Learning Representations*. 398

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv:1707.06347*. 397

Schuster, M., & Nakajima, K. (2012). Japanese and Korean voice search. *IEEE International Conference on Acoustics, Speech and Signal Processing*, 5149–5152. 234

Schwarz, J., Jayakumar, S., Pascanu, R., Latham, P., & Teh, Y. (2021). Powerpropagation: A sparsity inducing weight reparameterisation. *Neural Information Processing Systems*, *34*, 28889–28903. 156

Sejnowski, T. J. (2018). *The deep learning revolution*. MIT press. 37

Sejnowski, T. J. (2020). The unreasonable effectiveness of deep learning in artificial intelligence. *Proceedings of the National Academy of Sciences*, *117*(48), 30033–30038. 404

Selsam, D., Lamm, M., Bünz, B., Liang, P., de Moura, L., & Dill, D. L. (2019). Learning a SAT solver from single-bit supervision. *International Conference on Learning Representations*. 262

Selva, J., Johansen, A. S., Escalera, S., Nasrollahi, K., Moeslund, T. B., & Clapés, A. (2022). Video transformers: A survey. *arXiv:2201.05991*. 238

Sennrich, R., Haddow, B., & Birch, A. (2015). Neural machine translation of rare words with subword units. *Meeting of the Association for Computational Linguistics*. 234

Serra, T., Tjandraatmadja, C., & Ramalingam, S. (2018). Bounding and counting linear regions of deep neural networks. *International Conference on Machine Learning*, 4558–4566. 52

Shang, W., Sohn, K., Almeida, D., & Lee, H. (2016). Understanding and improving convolutional neural networks via concatenated rectified linear units. *International Conference on Machine Learning*, 2217–2225. 38

Sharif Razavian, A., Azizpour, H., Sullivan, J., & Carlsson, S. (2014). CNN features off-the-shelf: An astounding baseline for recognition. *IEEE*

Conference on Computer Vision and Pattern Recognition Workshop, 806–813. 159

Sharkey, A., & Sharkey, N. (2012). Granny and the robots: Ethical issues in robot care for the elderly. *Ethics and Information Technology*, *14*(1), 27–40. 424

Shaw, P., Uszkoreit, J., & Vaswani, A. (2018). Self-attention with relative position representations. *ACL Human Language Technologies*, 464–468. 236

Shen, S., Yao, Z., Gholami, A., Mahoney, M., & Keutzer, K. (2020a). PowerNorm: Rethinking batch normalization in transformers. *International Conference on Machine Learning*, 8741–8751. 237

Shen, X., Tian, X., Liu, T., Xu, F., & Tao, D. (2017). Continuous dropout. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(9), 3926–3937. 158

Shen, Y., Gu, J., Tang, X., & Zhou, B. (2020b). Interpreting the latent space of GANs for semantic face editing. *IEEE/CVF Computer Vision & Pattern Recognition*, 9243–9252. 300

Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A. P., Bishop, R., Rueckert, D., & Wang, Z. (2016). Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. *IEEE/CVF Computer Vision & Pattern Recognition*, 1874–1883. 182

Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., & Catanzaro, B. (2019). Megatron-LM: Training multi-billion parameter language models using model parallelism. *arXiv:1909.08053*. 114

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, *6*(1), 1–48. 159

Siddique, N., Paheding, S., Elkin, C. P., & Devabhaktuni, V. (2021). U-Net and its variants for medical image segmentation: A review of theory and applications. *IEEE Access*, 82031–82057. 205

Sifre, L., & Mallat, S. (2013). Rotation, scaling and deformation invariant scattering for texture discrimination. *IEEE/CVF Computer Vision & Pattern Recognition*, 1233–1240. 183

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*(7587), 484–489. 396, 398

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014). Deterministic policy gradient algorithms. *International Conference on Machine Learning*, 387–395. 397

Simonovsky, M., & Komodakis, N. (2018). GraphVAE: Towards generation of small graphs using variational autoencoders. *International Conference on Artificial Neural Networks*, 412–422. 344

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations*. 177, 181

Singh, S. P., & Sutton, R. S. (1996). Reinforcement learning with replacing eligibility traces. *Machine learning*, *22*(1), 123–158. 396

Sinha, S., Zhao, Z., Goyal, A., Raffel, C., & Odena, A. (2020). Top-k training of GANs: Improving GAN performance by throwing away bad samples. *Neural Information Processing Systems*, *33*, 14638–14649. 299

Sisson, M., Spindel, J., Scharre, P., & Kozyulin, V. (2020). The militarization of artificial intelligence. *United Nations Office for Disarmament Affairs*. 427

Sjöberg, J., & Ljung, L. (1995). Overtraining, regularization and searching for a minimum, with application to neural networks. *International Journal of Control*, *62*(6), 1391–1407. 157

Smith, M., & Miller, S. (2022). The ethical application of biometric facial recognition technology. *AI & Society*, *37*, 167–175. 426

Smith, S., Elsen, E., & De, S. (2020). On the generalization benefit of noise in stochastic gradient descent. *International Conference on Machine Learning*, 9058–9067. 157

Smith, S., Patwary, M., Norick, B., LeGresley, P., Rajbhandari, S., Casper, J., Liu, Z., Prabhumoye, S., Zerveas, G., Korthikanti, V., et al. (2022). Using DeepSpeed and Megatron to train Megatron-Turing NLG 530B, a large-scale generative language model. *arXiv:2201.11990*. 234

Smith, S. L., Dherin, B., Barrett, D. G. T., & De, S. (2021). On the origin of implicit regularization in stochastic gradient descent. *International Conference on Learning Representations*. 157

Smith, S. L., Kindermans, P., Ying, C., & Le, Q. V. (2018). Don't decay the learning rate, increase the batch size. *International Conference on Learning Representations*. 92

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. *Neural Information Processing Systems*, vol. 25, 2951–2959. 136

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., & Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. *International Conference on Machine Learning*, 2256–2265. 274, 367

Sohn, K., Lee, H., & Yan, X. (2015). Learning structured output representation using deep conditional generative models. *Neural Information Processing Systems*, *28*, 3483–3491. 344

Sohoni, N. S., Aberger, C. R., Leszczynski, M., Zhang, J., & Ré, C. (2019). Low-memory neural network training: A technical report. *arXiv:1904.10631*. 114

Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., & Winther, O. (2016a). How to train deep variational autoencoders and probabilistic ladder networks. *arXiv:1602.02282*. 344

Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., & Winther, O. (2016b). Ladder variational autoencoders. *Neural Information Processing Systems*, *29*, 738–3746. 369

Song, J., Meng, C., & Ermon, S. (2021a). Denoising diffusion implicit models. *International Conference on Learning Representations*. 370

Song, Y., & Ermon, S. (2019). Generative modeling by estimating gradients of the data distribution. *Neural Information Processing Systems*, *32*, 11895–11907. 367, 371

Song, Y., & Ermon, S. (2020). Improved techniques for training score-based generative models. *Neural Information Processing Systems*, *33*, 12438–12448. 371

Song, Y., Meng, C., & Ermon, S. (2019). MintNet: Building invertible neural networks with masked convolutions. *Neural Information Processing Systems*, *32*, 11002–11012. 322

Song, Y., Shen, L., Xing, L., & Ermon, S. (2021b). Solving inverse problems in medical imaging with score-based generative models. *International Conference on Learning Representations*. 369

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., & Poole, B. (2021c). Score-based generative modeling through stochastic differential equations. *International Conference on Learning Representations*. 369, 370, 371

Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2015). Striving for simplicity: The all convolutional net. *International Conference on Learning Representations*. 182

Srivastava, A., Rastogi, A., Rao, A., Shoeb, A. A. M., Abid, A., Fisch, A., Brown, A. R., Santoro, A., Gupta, A., Garriga-Alonso, A., et al. (2022). Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv:2206.04615*. 234

Srivastava, A., Valkov, L., Russell, C., Gutmann, M. U., & Sutton, C. (2017). VEEGAN: Reducing mode collapse in GANs using implicit variational learning. *Neural Information Processing Systems*, *30*, 3308–3318. 300

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(1), 1929–1958. 158

Srivastava, R. K., Greff, K., & Schmidhuber, J. (2015). Highway networks. *arXiv:1505.00387*. 202

Stark, L., & Hoey, J. (2021). The ethics of emotions in artificial intelligence systems. *ACM Conference on Fairness, Accountability, and Transparency*, 782–793. 427

Stark, L., & Hutson, J. (2022). Physiognomic artificial intelligence. *Fordham Intellectual Property, Media & Entertainment Law Journal*, *XXXII*(4), 922–978. 427, 432

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D., Lowe, R., Voss, C., Radford, A., Amodei, D., & Christiano, P. F. (2020). Learning to summarize with human feedback. *Neural Information Processing Systems*, *33*, 3008–3021. 398

Strubell, E., Ganesh, A., & McCallum, A. (2019). Energy and policy considerations for deep learning in NLP. *Meeting of the Association for Computational Linguistics*, 3645–3650. 429

Strubell, E., Ganesh, A., & McCallum, A. (2020). Energy and policy considerations for modern deep learning research. *Meeting of the Association for Computational Linguistics*, 13693–13696. 429

Su, H., Jampani, V., Sun, D., Gallo, O., Learned-Miller, E., & Kautz, J. (2019a). Pixel-adaptive convolutional neural networks. *IEEE/CVF Computer Vision & Pattern Recognition*, 11166–11175. 183

Su, J., Lu, Y., Pan, S., Wen, B., & Liu, Y. (2021). Roformer: Enhanced transformer with rotary position embedding. *arXiv:2104.09864*. 236

Su, W., Zhu, X., Cao, Y., Li, B., Lu, L., Wei, F., & Dai, J. (2019b). VL-BERT: Pre-training of generic visual-linguistic representations. *International Conference on Learning Representations*. 238

Sultan, M. M., Wayment-Steele, H. K., & Pande, V. S. (2018). Transferable neural networks for enhanced sampling of protein dynamics. *Journal of Chemical Theory and Computation*, *14*(4), 1887–1894. 344

Summers, C., & Dinneen, M. J. (2019). Improved mixed-example data augmentation. *Winter Conference on Applications of Computer Vision*, 1262–1270. 159

Sun, C., Myers, A., Vondrick, C., Murphy, K., & Schmid, C. (2019). VideoBERT: A joint model for video and language representation learning. *IEEE/CVF International Conference on Computer Vision*, 7464–7473. 238

Sun, C., Shrivastava, A., Singh, S., & Gupta, A. (2017). Revisiting unreasonable effectiveness of data in deep learning era. *IEEE/CVF International Conference on Computer Vision*, 843–852. 238

Sun, R.-Y. (2020). Optimization for deep learning: An overview. *Journal of the Operations Research Society of China*, *8*(2), 249–294. 91

Susmelj, I., Agustsson, E., & Timofte, R. (2017). ABC-GAN: Adaptive blur and control for improved training stability of generative adversarial networks. *ICML Workshop on Implicit Models*. 299

Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013). On the importance of initialization and momentum in deep learning. *International Conference on Machine Learning*, 1139–1147. 93

Sutton, R. S. (1984). *Temporal credit assignment in reinforcement learning*. Ph.D., University of Massachusetts Amherst. 396

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine learning*, *3*(1), 9–44. 396

Sutton, R. S., & Barto, A. G. (1999). *Reinforcement learning: An introduction*. MIT press. 396

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction, 2nd Edition*. MIT Press. 16, 396

Sutton, R. S., McAllester, D., Singh, S., & Mansour, Y. (1999). Policy gradient methods for reinforcement learning with function approximation. *Neural Information Processing Systems*, *12*, 1057–1063. 397

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017). Inception-v4, Inception-Resnet and the impact of residual connections on learning. *AAAI Conference on Artificial Intelligence*, 4278–4284. 181, 183, 405

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the Inception architecture for computer vision. *IEEE/CVF Computer Vision & Pattern Recognition*, 2818–2826. 155, 158, 274

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2014). Intriguing properties of neural networks. *International Conference on Learning Representations*. 414

Szeliski, R. (2022). *Computer vision: Algorithms and applications, 2nd Edition*. Springer. 15

Tabak, E. G., & Turner, C. V. (2013). A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, *66*(2), 145–164. 321

Tabak, E. G., & Vanden-Eijnden, E. (2010). Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, *8*(1), 217–233. 321

Taddeo, M., & Floridi, L. (2018). How AI can be a force for good. *Science*, *361*(6404), 751–752. 420

Tan, H., & Bansal, M. (2019). LXMERT: Learning cross-modality encoder representations from transformers. *Empirical Methods in Natural Language Processing*, 5099–5110. 238

Tan, M., & Le, Q. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. *International Conference on Machine Learning*, 6105–6114. 405

Tay, Y., Bahri, D., Metzler, D., Juan, D.-C., Zhao, Z., & Zheng, C. (2021). Synthesizer: Rethinking self-attention for transformer models. *International Conference on Machine Learning*, 10183–10192. 235

Tay, Y., Bahri, D., Yang, L., Metzler, D., & Juan, D.-C. (2020). Sparse Sinkhorn attention. *International Conference on Machine Learning*, 9438–9447. 237

Tay, Y., Dehghani, M., Bahri, D., & Metzler, D. (2023). Efficient transformers: A survey. *ACM Computing Surveys*, *55*(6), 109:1–109:28. 237

Tegmark, M. (2018). *Life 3.0: Being human in the age of artificial intelligence*. Vintage. 14

Telgarsky, M. (2016). Benefits of depth in neural networks. *PMLR Conference on Learning Theory*, 1517–1539. 53, 417

Teru, K., Denis, E., & Hamilton, W. (2020). Inductive relation prediction by subgraph reasoning. *International Conference on Machine Learning*, 9448–9457. 265

Tetlock, P. E., & Gardner, D. (2016). *Superforecasting: The Art and Science of Prediction*. Toronto: Signal, McClelland & Stewart. 435

Tewari, A., Elgharib, M., Bharaj, G., Bernard, F., Seidel, H.-P., Pérez, P., Zollhofer, M., & Theobalt, C. (2020). StyleRig: Rigging StyleGAN for 3D control over portrait images. *IEEE/CVF Computer Vision & Pattern Recognition*, 6142–6151. 300

Teye, M., Azizpour, H., & Smith, K. (2018). Bayesian uncertainty estimation for batch normalized deep networks. *International Conference on Machine Learning*, 4907–4916. 204

Theis, L., Oord, A. v. d., & Bethge, M. (2016). A note on the evaluation of generative models. *International Conference on Learning Representations*. 322

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, *25*(3-4), 285–294. 396

Thompson, W. R. (1935). On the theory of apportionment. *American Journal of Mathematics*, *57*(2), 450–456. 396

Thoppilan, R., De Freitas, D., Hall, J., Shazeer, N., Kulshreshtha, A., Cheng, H.-T., Jin, A., Bos, T., Baker, L., Du, Y., et al. (2022). LaMDA: Language models for dialog applications. *arXiv:2201.08239*. 234

Tipping, M. E., & Bishop, C. M. (1999). Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B*, *61*(3), 611–622. 344

Tolmeijer, S., Kneer, M., Sarasua, C., Christen, M., & Bernstein, A. (2020). Implementations in machine ethics: A survey. *ACM Computing Surveys*, *53*(6), 1–38. 424

Tolstikhin, I., Bousquet, O., Gelly, S., & Schoelkopf, B. (2018). Wasserstein auto-encoders. *International Conference on Learning Representations*. 345

Tomašev, N., Cornebise, J., Hutter, F., Mohamed, S., Picciariello, A., Connelly, B., Belgrave, D. C., Ezer, D., Haert, F. C. v. d., Mugisha, F., et al. (2020). AI for social good: Unlocking the opportunity for positive impact. *Nature Communications*, *11*(1), 2468. 420

Tomasev, N., McKee, K. R., Kay, J., & Mohamed, S. (2021). Fairness for unobserved characteristics: Insights from technological impacts on queer communities. *AAAI/ACM Conference on AI, Ethics, and Society*, 254–265. 424

Tomczak, J. M., & Welling, M. (2016). Improving variational auto-encoders using Householder flow. *NIPS Workshop on Bayesian Deep Learning*. 322

Tompson, J., Goroshin, R., Jain, A., LeCun, Y., & Bregler, C. (2015). Efficient object localization using convolutional networks. *IEEE/CVF Computer Vision & Pattern Recognition*, 648–656. 183

Torralba, A., Freeman, W., & Isola, P. (2024). *Foundations of Computer Vision*. MIT Press. 15

Touati, A., Satija, H., Romoff, J., Pineau, J., & Vincent, P. (2020). Randomized value functions via multiplicative normalizing flows. *Uncertainty in Artificial Intelligence*, 422–432. 322

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., & Jégou, H. (2021). Training data-efficient image transformers & distillation through attention. *International Conference on Machine Learning*, 10347–10357. 238

Tran, D., Bourdev, L., Fergus, R., Torresani, L., & Paluri, M. (2015). Learning spatiotemporal features with 3D convolutional networks. *IEEE International Conference on Computer Vision*, 4489–4497. 182

Tran, D., Vafa, K., Agrawal, K., Dinh, L., & Poole, B. (2019). Discrete flows: Invertible generative models of discrete data. *Neural Information Processing Systems*, *32*, 14692–14701. 322, 324

Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., & Paluri, M. (2018). A closer look at spatiotemporal convolutions for action recognition. *IEEE/CVF Computer Vision & Pattern Recognition*, 6450–6459. 181

Tsitsulin, A., Palowitch, J., Perozzi, B., & Müller, E. (2020). Graph clustering with graph neural networks. *arXiv:2006.16904*. 262

Tzen, B., & Raginsky, M. (2019). Neural stochastic differential equations: Deep latent Gaussian models in the diffusion limit. *arXiv:1905.09883*. 324

Ulku, I., & Akagündüz, E. (2022). A survey on deep learning-based architectures for semantic

segmentation on 2D images. *Applied Artificial Intelligence*, *36*(1). 184

Ulyanov, D., Vedaldi, A., & Lempitsky, V. (2016). Instance normalization: The missing ingredient for fast stylization. *arXiv:1607.08022*. 203

Ulyanov, D., Vedaldi, A., & Lempitsky, V. (2018). Deep image prior. *IEEE/CVF Computer Vision & Pattern Recognition*, 9446–9454. 418

Urban, G., Geras, K. J., Kahou, S. E., Aslan, O., Wang, S., Caruana, R., Mohamed, A., Philipose, M., & Richardson, M. (2017). Do deep convolutional nets really need to be deep and convolutional? *International Conference on Learning Representations*. 417, 418

Vahdat, A., Andriyash, E., & Macready, W. (2018a). DVAE#: Discrete variational autoencoders with relaxed Boltzmann priors. *Neural Information Processing Systems*, *31*, 1869–1878. 344

Vahdat, A., Andriyash, E., & Macready, W. (2020). Undirected graphical models as approximate posteriors. *International Conference on Machine Learning*, 9680–9689. 344

Vahdat, A., & Kautz, J. (2020). NVAE: A deep hierarchical variational autoencoder. *Neural Information Processing Systems*, *33*, 19667–19679. 340, 345, 369

Vahdat, A., Kreis, K., & Kautz, J. (2021). Score-based generative modeling in latent space. *Neural Information Processing Systems*, *34*, 11287–11302. 370

Vahdat, A., Macready, W., Bian, Z., Khoshaman, A., & Andriyash, E. (2018b). DVAE++: Discrete variational autoencoders with overlapping transformations. *International Conference on Machine Learning*, 5035–5044. 344

Vallor, S. (2011). Carebots and caregivers: Sustaining the ethical ideal of care in the 21st century. *Philosophy and Technology*, *24*(3), 251–268. 429

Vallor, S. (2015). Moral deskilling and upskilling in a new machine age: Reflections on the ambiguous future of character. *Philosophy & Technology*, *28*, 107–124. 429

Van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., & Kavukcuoglu, K. (2016a). WaveNet: A generative model for raw audio. *ISCA Speech Synthesis Workshop*. 323

Van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., et al. (2016b). Conditional image generation with PixelCNN decoders. *Neural Information Processing Systems*, *29*, 4790–4798. 274

Van den Oord, A., Kalchbrenner, N., & Kavukcuoglu, K. (2016c). Pixel recurrent neural networks. *International Conference on Machine Learning*, 1747–1756. 233, 345

Van den Oord, A., Li, Y., Babuschkin, I., Simonyan, K., Vinyals, O., Kavukcuoglu, K., Driessche, G., Lockhart, E., Cobo, L., Stimberg, F., et al. (2018). Parallel WaveNet: Fast high-fidelity speech synthesis. *International Conference on Machine Learning*, 3918–3926. 323

Van Den Oord, A., Vinyals, O., et al. (2017). Neural discrete representation learning. *Neural Information Processing Systems*, *30*, 6306–6315. 344, 345

Van Hasselt, H. (2010). Double Q-learning. *Neural Information Processing Systems*, *23*, 2613–2621. 397

Van Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double Q-learning. *AAAI Conference on Artificial Intelligence*, 2094–2100. 397

Van Hoof, H., Chen, N., Karl, M., van der Smagt, P., & Peters, J. (2016). Stable reinforcement learning with autoencoders for tactile and visual data. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3928–3934. IEEE. 344

van Wynsberghe, A., & Robbins, S. (2019). Critiquing the reasons for making artificial moral agents. *Science and Engineering Ethics*, *25*, 719–735. 424

Vapnik, V. (1995). *The nature of statistical learning theory*. New York: Springer Verlag. 74

Vapnik, V. N., & Chervonenkis, A. Y. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Measures of Complexity*, 11–30. 134

Vardi, G., Yehudai, G., & Shamir, O. (2022). Width is less important than depth in ReLU neural networks. *PMRL Conference on Learning Theory*, 1–33. 53

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Neural Information Processing Systems*, *30*, 5998–6008. 158, 233, 234, 235, 236, 237

Veit, A., Wilber, M. J., & Belongie, S. (2016). Residual networks behave like ensembles of relatively shallow networks. *Neural Information Processing Systems*, *29*, 550–558. 202, 417

Veličković, P. (2023). Everything is connected: Graph neural networks. *Current Opinion in Structural Biology*, *79*, 102538. 261

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2019). Graph attention networks. *International Conference on Learning Representations*. 234, 263, 265

Véliz, C. (2020). *Privacy is Power: Why and How You Should Take Back Control of Your Data*. Bantam Press. 435

Véliz, C. (2023). Chatbots shouldn't use emojis. *Nature*, *615*, 375. 428

Vijayakumar, A. K., Cogswell, M., Selvaraju, R. R., Sun, Q., Lee, S., Crandall, D., & Batra, D. (2016). Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv:1610.02424*. 235

Vincent, J. (2020). What a machine learning tool that turns Obama white can (and can't) tell us about AI bias / a striking image that only hints at a much bigger problem. The Verge, June 23, 2020. `https://www.theverge.com/21298762/face-depixelizer-ai-machine-learning-tool-pulse-stylegan-obama-bias`. 422

Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. *International Conference on Machine Learning*, 1096–1103. 344

Voita, E., Talbot, D., Moiseev, F., Sennrich, R., & Titov, I. (2019). Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *Meeting of the Association for Computational Linguistics*, 5797–5808. 235

Voleti, V., Jolicoeur-Martineau, A., & Pal, C. (2022). MCVD: Masked conditional video diffusion for prediction, generation, and interpolation. *Neural Information Processing Systems*, *35*. 369

Vondrick, C., Pirsiavash, H., & Torralba, A. (2016). Generating videos with scene dynamics. *Neural Information Processing Systems*, *29*, 613–621. 299

Wachter, S., Mittelstadt, B., & Floridi, L. (2017). Why a right to explanation of automated decision-making does not exist in the general data protection regulation. *International Data Privacy Law*, *7*(2), 76–99. 425

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., & Lang, K. J. (1989). Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *37*(3), 328–339. 181

Wallach, W., Allen, C., & Smit, I. (2008). Machine morality: Bottom-up and top-down approaches for modeling human moral faculties. *AI & Society*, *22*(4), 565–582. 424

Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., & Fergus, R. (2013). Regularization of neural networks using DropConnect. *International Conference on Machine Learning*, 1058–1066. 158

Wan, Z., Zhang, J., Chen, D., & Liao, J. (2021). High-fidelity pluralistic image completion with transformers. *IEEE/CVF International Conference on Computer Vision*, 4692–4701. 238

Wang, A., Pruksachatkun, Y., Nangia, N., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. (2019a). SuperGLUE: A stickier benchmark for general-purpose language understanding systems. *Neural Information Processing Systems*, *32*, 3261–3275. 234

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. (2019b). GLUE: A multi-task benchmark and analysis platform for natural language understanding. *International Conference on Learning Representations*. 234

Wang, B., Shang, L., Lioma, C., Jiang, X., Yang, H., Liu, Q., & Simonsen, J. G. (2020a). On position embeddings in BERT. *International Conference on Learning Representations*. 236

Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2022a). Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv:2207.02696*. 184

Wang, P. Z., & Wang, W. Y. (2019). Riemannian normalizing flow on variational Wasserstein autoencoder for text modeling. *ACL Human Language Technologies*, 284–294. 324

Wang, S., Li, B. Z., Khabsa, M., Fang, H., & Ma, H. (2020b). Linformer: Self-attention with linear complexity. *arXiv:2006.04768*. 237

Wang, T., Liu, M., Zhu, J., Yakovenko, N., Tao, A., Kautz, J., & Catanzaro, B. (2018a). Video-to-video synthesis. *Neural Information Processing Systems*, vol. 31, 1152–1164. 299

Wang, T.-C., Liu, M.-Y., Zhu, J.-Y., Tao, A., Kautz, J., & Catanzaro, B. (2018b). High-resolution image synthesis and semantic manipulation with conditional GANs. *IEEE/CVF Computer Vision & Pattern Recognition*, 8798–8807. 300, 301

Wang, W., Xie, E., Li, X., Fan, D.-P., Song, K., Liang, D., Lu, T., Luo, P., & Shao, L.

(2021). Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. *IEEE/CVF International Conference on Computer Vision*, 568–578. 238

Wang, W., Yao, L., Chen, L., Lin, B., Cai, D., He, X., & Liu, W. (2022b). Crossformer: A versatile vision transformer hinging on cross-scale attention. *International Conference on Learning Representations*. 238

Wang, X., Girshick, R., Gupta, A., & He, K. (2018c). Non-local neural networks. *IEEE/CVF Computer Vision & Pattern Recognition*, 7794–7803. 238

Wang, X., Wang, S., Liang, X., Zhao, D., Huang, J., Xu, X., Dai, B., & Miao, Q. (2022c). Deep reinforcement learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*. 396

Wang, Y., & Kosinski, M. (2018). Deep neural networks are more accurate than humans at detecting sexual orientation from facial images. *Journal of Personality and Social Psychology*, *114*(2), 246–257. 430

Wang, Y., Mohamed, A., Le, D., Liu, C., Xiao, A., Mahadeokar, J., Huang, H., Tjandra, A., Zhang, X., Zhang, F., et al. (2020c). Transformer-based acoustic modeling for hybrid speech recognition. *IEEE International Conference on Acoustics, Speech and Signal Processing*, 6874–6878. 234

Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., & de Freitas, N. (2017). Sample efficient actor-critic with experience replay. *International Conference on Learning Representations*. 398

Wang, Z., Schaul, T., Hessel, M., van Hasselt, H., Lanctot, M., & Freitas, N. (2016). Dueling network architectures for deep reinforcement learning. *International Conference on Machine Learning*, 1995–2003. 397

Ward, P. N., Smofsky, A., & Bose, A. J. (2019). Improving exploration in soft-actor-critic with normalizing flows policies. *ICML Workshop on Invertible Neural Networks and Normalizing Flows*. 322

Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine learning*, *8*(3-4), 279–292. 396

Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. Ph.D., University of Cambridge. 396

Wehenkel, A., & Louppe, G. (2019). Unconstrained monotonic neural networks. *Neural Information Processing Systems*, *32*, 1543–1553. 323

Wei, J., Ren, X., Li, X., Huang, W., Liao, Y., Wang, Y., Lin, J., Jiang, X., Chen, X., & Liu, Q. (2019). NEZHA: Neural contextualized representation for Chinese language understanding. *arXiv:1909.00204*. 236

Wei, J., & Zou, K. (2019). EDA: Easy data augmentation techniques for boosting performance on text classification tasks. *ACL Empirical Methods in Natural Language Processing*, 6382–6388. 160

Weidinger, L., Uesato, J., Rauh, M., Griffin, C., Huang, P.-S., Mellor, J., Glaese, A., Cheng, M., Balle, B., Kasirzadeh, A., Biles, C., Brown, S., Kenton, Z., Hawkins, W., Stepleton, T., Birhane, A., Hendricks, L. A., Rimell, L., Isaac, W., Haas, J., Legassick, S., Irving, G., & Gabriel, I. (2022). Taxonomy of risks posed by language models. *ACM Conference on Fairness, Accountability, and Transparency*, 214–229. 428

Weisfeiler, B., & Leman, A. (1968). The reduction of a graph to canonical form and the algebra which appears therein. *NTI, Series*, *2*(9), 12–16. 264

Welling, M., & Teh, Y. W. (2011). Bayesian learning via stochastic gradient Langevin dynamics. *International Conference on Machine Learning*, 681–688. 159

Wen, Y.-H., Yang, Z., Fu, H., Gao, L., Sun, Y., & Liu, Y.-J. (2021). Autoregressive stylized motion synthesis with generative flow. *IEEE/CVF Computer Vision & Pattern Recognition*, 13612–13621. 322

Wenzel, F., Roth, K., Veeling, B. S., Świątkowski, J., Tran, L., Mandt, S., Snoek, J., Salimans, T., Jenatton, R., & Nowozin, S. (2020a). How good is the Bayes posterior in deep neural networks really? *International Conference on Machine Learning*, 10248–10259. 159

Wenzel, F., Snoek, J., Tran, D., & Jenatton, R. (2020b). Hyperparameter ensembles for robustness and uncertainty quantification. *Neural Information Processing Systems*, *33*, 6514–6527. 158

Werbos, P. (1974). Beyond regression: New tools for prediction and analysis in the behavioral sciences. *Ph.D. dissertation, Harvard University*. 113

White, T. (2016). Sampling generative networks. *arXiv:1609.04468*. 342, 344

Whitney, H. (1932). Congruent graphs and the connectivity of graphs. *Hassler Whitney Collected Papers*, 61–79. 264

Wightman, R., Touvron, H., & Jégou, H. (2021). ResNet strikes back: An improved training procedure in timm. *Neural Information Processing Systems Workshop*. 202

Williams, C. K., & Rasmussen, C. E. (2006). *Gaussian processes for machine learning*. MIT Press. 15

Williams, P. M. (1996). Using neural networks to model conditional multivariate densities. *Neural Computation*, *8*(4), 843–854. 73

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, *8*(3), 229–256. 397

Wilson, A. C., Roelofs, R., Stern, M., Srebro, N., & Recht, B. (2017). The marginal value of adaptive gradient methods in machine learning. *Neural Information Processing Systems*, *30*, 4148–4158. 94, 410

Wirnsberger, P., Ballard, A. J., Papamakarios, G., Abercrombie, S., Racanière, S., Pritzel, A., Jimenez Rezende, D., & Blundell, C. (2020). Targeted free energy estimation via learned mappings. *The Journal of Chemical Physics*, *153*(14), 144112. 322

Wolf, S. (2021). ProGAN: How NVIDIA generated images of unprecedented quality. `https://towardsdatascience.com/progan-how-nvidia-generated-images-of-unprecedented-quality-51c98ec2cbd2`. 286

Wolf, V., Lugmayr, A., Danelljan, M., Van Gool, L., & Timofte, R. (2021). DeFlow: Learning complex image degradations from unpaired data with conditional flows. *IEEE/CVF Computer Vision & Pattern Recognition*, 94–103. 322

Wolfe, C. R., Yang, J., Chowdhury, A., Dun, C., Bayer, A., Segarra, S., & Kyrillidis, A. (2021). GIST: Distributed training for large-scale graph convolutional networks. *NeurIPS Workshop on New Frontiers in Graph Learning*. 264

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, *5*(2), 241–259. 158

Wong, K. W., Contardo, G., & Ho, S. (2020). Gravitational-wave population inference with deep flow-based generative network. *Physical Review D*, *101*(12), 123005. 322

Worrall, D. E., Garbin, S. J., Turmukhambetov, D., & Brostow, G. J. (2017). Harmonic networks: Deep translation and rotation equivariance. *IEEE/CVF Computer Vision & Pattern Recognition*, 5028–5037. 183

Wu, B., Xu, C., Dai, X., Wan, A., Zhang, P., Yan, Z., Tomizuka, M., Gonzalez, J., Keutzer, K., & Vajda, P. (2020a). Visual transformers: Token-based image representation and processing for computer vision. *arXiv:2006.03677*. 238

Wu, F., Fan, A., Baevski, A., Dauphin, Y. N., & Auli, M. (2019). Pay less attention with lightweight and dynamic convolutions. *International Conference on Learning Representations*. 235

Wu, H., & Gu, X. (2015). Max-pooling dropout for regularization of convolutional neural networks. *Neural Information Processing Systems*, vol. 18, 46–54. 183

Wu, J., Huang, Z., Thoma, J., Acharya, D., & Van Gool, L. (2018a). Wasserstein divergence for GANs. *European Conference on Computer Vision*, 653–668. 299

Wu, J., Zhang, C., Xue, T., Freeman, B., & Tenenbaum, J. (2016). Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. *Neural Information Processing Systems*, *29*, 82–90. 299

Wu, N., Green, B., Ben, X., & O'Banion, S. (2020b). Deep transformer models for time series forecasting: The influenza prevalence case. *arXiv:2001.08317*. 234

Wu, R., Yan, S., Shan, Y., Dang, Q., & Sun, G. (2015a). Deep image: Scaling up image recognition. *arXiv:1501.02876*, *7*(8). 154

Wu, S., Sun, F., Zhang, W., Xie, X., & Cui, B. (2023). Graph neural networks in recommender systems: A survey. *ACM Computing Surveys*, *55*(5), 97:1–97:37. 262

Wu, X., & Zhang, X. (2016). Automated inference on criminality using face images. *arXiv:1611.04135*. 427

Wu, Y., Burda, Y., Salakhutdinov, R., & Grosse, R. (2017). On the quantitative analysis of decoder-based generative models. *International Conference on Learning Representations*. 300

Wu, Y., & He, K. (2018). Group normalization. *European Conference on Computer Vision*, 3–19. 203, 204

Wu, Z., Lischinski, D., & Shechtman, E. (2021). Stylespace analysis: Disentangled controls for StyleGAN image generation. *IEEE/CVF Computer Vision & Pattern Recognition*, 12863–12872. 300

Wu, Z., Nagarajan, T., Kumar, A., Rennie, S., Davis, L. S., Grauman, K., & Feris, R. (2018b).

BlockDrop: Dynamic inference paths in residual networks. *IEEE/CVF Computer Vision & Pattern Recognition*, 8817–8826. 203

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Philip, S. Y. (2020c). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(1), 4–24. 261

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., & Xiao, J. (2015b). 3D ShapeNets: A deep representation for volumetric shapes. *IEEE/CVF Computer Vision & Pattern Recognition*, 1912–1920. 182

Xia, F., Liu, T.-Y., Wang, J., Zhang, W., & Li, H. (2008). Listwise approach to learning to rank: theory and algorithm. *International Conference on Machine Learning*, 1192–1199. 73

Xia, W., Zhang, Y., Yang, Y., Xue, J.-H., Zhou, B., & Yang, M.-H. (2022). GAN inversion: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–17. 301

Xiao, L., Bahri, Y., Sohl-Dickstein, J., Schoenholz, S., & Pennington, J. (2018a). Dynamical isometry and a mean field theory of CNNs: How to train 10,000-layer vanilla convolutional neural networks. *International Conference on Machine Learning*, 5393–5402. 114, 183

Xiao, S., Wang, S., Dai, Y., & Guo, W. (2022a). Graph neural networks in node classification: Survey and evaluation. *Machine Vision and Applications*, *33*(1), 1–19. 262

Xiao, T., Hong, J., & Ma, J. (2018b). DNA-GAN: Learning disentangled representations from multi-attribute images. *International Conference on Learning Representations*. 301

Xiao, Z., Kreis, K., & Vahdat, A. (2022b). Tackling the generative learning trilemma with denoising diffusion GANs. *International Conference on Learning Representations*. 370

Xie, E., Wang, W., Yu, Z., Anandkumar, A., Alvarez, J. M., & Luo, P. (2021). SegFormer: Simple and efficient design for semantic segmentation with transformers. *Neural Information Processing Systems*, *34*, 12077–12090. 238

Xie, L., Wang, J., Wei, Z., Wang, M., & Tian, Q. (2016). DisturbLabel: Regularizing CNN on the loss layer. *IEEE/CVF Computer Vision & Pattern Recognition*, 4753–4762. 158

Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. *IEEE/CVF Computer Vision & Pattern Recognition*, 1492–1500. 181, 202, 405

Xie, Y., & Li, Q. (2022). Measurement-conditioned denoising diffusion probabilistic model for under-sampled medical image reconstruction. *Medical Image Computing and Computer Assisted Intervention*, vol. 13436, 655–664. 369

Xing, E. P., Ho, Q., Dai, W., Kim, J. K., Wei, J., Lee, S., Zheng, X., Xie, P., Kumar, A., & Yu, Y. (2015). Petuum: A new platform for distributed machine learning on big data. *IEEE Transactions on Big Data*, *1*(2), 49–67. 114

Xing, Y., Qian, Z., & Chen, Q. (2021). Invertible image signal processing. *IEEE/CVF Computer Vision & Pattern Recognition*, 6287–6296. 322

Xiong, R., Yang, Y., He, D., Zheng, K., Zheng, S., Xing, C., Zhang, H., Lan, Y., Wang, L., & Liu, T. (2020a). On layer normalization in the transformer architecture. *International Conference on Machine Learning*, 10524–10533. 237

Xiong, Z., Yuan, Y., Guo, N., & Wang, Q. (2020b). Variational context-deformable convnets for indoor scene parsing. *IEEE/CVF Computer Vision & Pattern Recognition*, 3992–4002. 183

Xu, B., Wang, N., Chen, T., & Li, M. (2015). Empirical evaluation of rectified activations in convolutional network. *arXiv:1505.00853*. 158, 160

Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2019). How powerful are graph neural networks? *International Conference on Learning Representations*. 264

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.-i., & Jegelka, S. (2018). Representation learning on graphs with jumping knowledge networks. *International Conference on Machine Learning*, 5453–5462. 263, 265, 266

Xu, K., Zhang, M., Jegelka, S., & Kawaguchi, K. (2021a). Optimization of graph neural networks: Implicit acceleration by skip connections and more depth. *International Conference on Machine Learning*, 11592–11602. 266

Xu, P., Cheung, J. C. K., & Cao, Y. (2020). On variational learning of controllable representations for text without supervision. *International Conference on Machine Learning*, 10534–10543. 343, 345

Xu, P., Kumar, D., Yang, W., Zi, W., Tang, K., Huang, C., Cheung, J. C. K., Prince, S. J. D., & Cao, Y. (2021b). Optimizing deeper transformers on small datasets. *Meeting of the Association for Computational Linguistics*. 114, 234, 238

Yamada, Y., Iwamura, M., Akiba, T., & Kise, K. (2019). Shakedrop regularization for deep residual learning. *IEEE Access*, *7*, 186126–186136. 202, 203

Yamada, Y., Iwamura, M., & Kise, K. (2016). Deep pyramidal residual networks with separated stochastic depth. *arXiv:1612.01230*. 202

Yan, X., Yang, J., Sohn, K., & Lee, H. (2016). Attribute2Image: Conditional image generation from visual attributes. *European Conference on Computer Vision*, 776–791. 301

Yang, F., Yang, H., Fu, J., Lu, H., & Guo, B. (2020a). Learning texture transformer network for image super-resolution. *IEEE/CVF Computer Vision & Pattern Recognition*, 5791–5800. 238

Yang, G., Pennington, J., Rao, V., Sohl-Dickstein, J., & Schoenholz, S. S. (2019). A mean field theory of batch normalization. *International Conference on Learning Representations*. 203

Yang, K., Goldman, S., Jin, W., Lu, A. X., Barzilay, R., Jaakkola, T., & Uhler, C. (2021). Mol2Image: Improved conditional flow models for molecule to image synthesis. *IEEE/CVF Computer Vision & Pattern Recognition*, 6688–6698. 322

Yang, Q., Zhang, Y., Dai, W., & Pan, S. J. (2020b). *Transfer learning*. Cambridge University Press. 159

Yang, R., Srivastava, P., & Mandt, S. (2022). Diffusion probabilistic modeling for video generation. *arXiv:2203.09481*. 369, 371

Yao, W., Zeng, Z., Lian, C., & Tang, H. (2018). Pixel-wise regression using U-Net and its application on pansharpening. *Neurocomputing*, *312*, 364–371. 205

Ye, H., & Young, S. (2004). High quality voice morphing. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1–9. 160

Ye, L., Rochan, M., Liu, Z., & Wang, Y. (2019). Cross-modal self-attention network for referring image segmentation. *IEEE/CVF Computer Vision & Pattern Recognition*, 10502–10511. 238

Ye, W., Liu, S., Kurutach, T., Abbeel, P., & Gao, Y. (2021). Mastering Atari games with limited data. *Neural Information Processing Systems*, *34*, 25476–25488. 396

Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., & Leskovec, J. (2018a). Graph convolutional neural networks for web-scale recommender systems. *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 974–983. 264, 265

Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., & Leskovec, J. (2018b). Hierarchical graph representation learning with differentiable pooling. *Neural Information Processing Systems*, *31*, 4805–4815. 265

Yoshida, Y., & Miyato, T. (2017). Spectral norm regularization for improving the generalizability of deep learning. *arXiv:1705.10941*. 156

You, Y., Chen, T., Wang, Z., & Shen, Y. (2020). When does self-supervision help graph convolutional networks? *International Conference on Machine Learning*, 10871–10880. 159

Yu, F., & Koltun, V. (2015). Multi-scale context aggregation by dilated convolutions. *International Conference on Learning Representations*. 181

Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., & Huang, T. S. (2019). Free-form image inpainting with gated convolution. *IEEE/CVF International Conference on Computer Vision*, 4471–4480. 181

Yu, J., Zheng, Y., Wang, X., Li, W., Wu, Y., Zhao, R., & Wu, L. (2021). FastFlow: Unsupervised anomaly detection and localization via 2D normalizing flows. *arXiv:2111.07677*. 322

Yu, J. J., Derpanis, K. G., & Brubaker, M. A. (2020). Wavelet flow: Fast training of high resolution normalizing flows. *Neural Information Processing Systems*, *33*, 6184–6196. 322

Yu, L., Zhang, W., Wang, J., & Yu, Y. (2017). SeqGAN: Sequence generative adversarial nets with policy gradient. *AAAI Conference on Artificial Intelligence*, 2852–2858. 299

Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., & Yoo, Y. (2019). CutMix: Regularization strategy to train strong classifiers with localizable features. *IEEE/CVF International Conference on Computer Vision*, 6023–6032. 160

Zagoruyko, S., & Komodakis, N. (2016). Wide residual networks. *British Machine Vision Conference*. 202, 417

Zagoruyko, S., & Komodakis, N. (2017). Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *International Conference on Learning Representations*. 415

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., & Smola, A. J. (2017). Deep sets. *Neural Information Processing Systems*, *30*, 3391–3401. 263

Zaheer, M., Reddi, S., Sachan, D., Kale, S., & Kumar, S. (2018). Adaptive methods for nonconvex optimization. *Neural Information Processing Systems*, *31*, 9815–9825. 93

Zaslavsky, T. (1975). *Facing up to arrangements: Face-count formulas for partitions of space by hyperplanes: Face-count formulas for partitions of space by hyperplanes*. Memoirs of the American Mathematical Society. 38, 40

Zeiler, M. D. (2012). ADADELTA: An adaptive learning rate method. *arXiv:1212.5701*. 93

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. *European Conference on Computer Vision*, 818–833. 181, 184

Zeiler, M. D., Taylor, G. W., & Fergus, R. (2011). Adaptive deconvolutional networks for mid and high level feature learning. *IEEE International Conference on Computer Vision*, 2018–2025. 181

Zeng, H., Zhou, H., Srivastava, A., Kannan, R., & Prasanna, V. (2020). GraphSAINT: Graph sampling based inductive learning method. *International Conference on Learning Representations*. 264

Zeng, Y., Fu, J., Chao, H., & Guo, B. (2019). Learning pyramid-context encoder network for high-quality image inpainting. *IEEE/CVF Computer Vision & Pattern Recognition*, 1486–1494. 205

Zhai, S., Talbott, W., Srivastava, N., Huang, C., Goh, H., Zhang, R., & Susskind, J. (2021). An attention free transformer. 235

Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2023). *Dive into deep learning*. Cambridge University Press. 15

Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2017a). Understanding deep learning requires rethinking generalization. *International Conference on Learning Representations*. 156, 403, 418

Zhang, C., Ouyang, X., & Patras, P. (2017b). ZipNet-GAN: Inferring fine-grained mobile traffic patterns via a generative adversarial neural network. *International Conference on emerging Networking EXperiments and Technologies*, 363–375. 299

Zhang, H., Cisse, M., Dauphin, Y. N., & Lopez-Paz, D. (2017c). mixup: Beyond empirical risk minimization. *International Conference on Learning Representations*. 160

Zhang, H., Dauphin, Y. N., & Ma, T. (2019a). Fixup initialization: Residual learning without normalization. *International Conference on Learning Representations*. 114, 205

Zhang, H., Goodfellow, I., Metaxas, D., & Odena, A. (2019b). Self-attention generative adversarial networks. *International Conference on Machine Learning*, 7354–7363. 299

Zhang, H., Hsieh, C.-J., & Akella, V. (2016a). Hogwild++: A new mechanism for decentralized asynchronous stochastic gradient descent. *IEEE International Conference on Data Mining*, 629–638. 114

Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., & Metaxas, D. N. (2017d). StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks. *IEEE/CVF International Conference on Computer Vision*, 5907–5915. 300, 301

Zhang, J., & Meng, L. (2019). GResNet: Graph residual network for reviving deep gnns from suspended animation. *arXiv:1909.05729*. 263

Zhang, J., Shi, X., Xie, J., Ma, H., King, I., & Yeung, D.-Y. (2018a). GaAN: Gated attention networks for learning on large and spatiotemporal graphs. *Uncertainty in Artificial Intelligence*, 339–349. 263

Zhang, J., Zhang, H., Xia, C., & Sun, L. (2020). Graph-Bert: Only attention is needed for learning graph representations. *arXiv:2001.05140*. 263

Zhang, K., Yang, Z., & Başar, T. (2021a). Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control*, 321–384. 398

Zhang, L., & Agrawala, M. (2023). Adding conditional control to text-to-image diffusion models. *arXiv:2302.05543*. 370

Zhang, M., & Chen, Y. (2018). Link prediction based on graph neural networks. *Neural Information Processing Systems*, *31*, 5171–5181. 262

Zhang, M., Cui, Z., Neumann, M., & Chen, Y. (2018b). An end-to-end deep learning architecture for graph classification. *AAAI Conference on Artificial Intelligence*, 4438–4445. 262, 265

Zhang, Q., & Chen, Y. (2021). Diffusion normalizing flow. *Neural Information Processing Systems*, *34*, 16280–16291. 371

Zhang, R. (2019). Making convolutional networks shift-invariant again. *International Conference on Machine Learning*, 7324–7334. 182, 183

Zhang, R., Isola, P., & Efros, A. A. (2016b). Colorful image colorization. *European Conference on Computer Vision*, 649–666. 159

Zhang, S., Tong, H., Xu, J., & Maciejewski, R. (2019c). Graph convolutional networks: A comprehensive review. *Computational Social Networks*, *6*(1), 1–23. 262

Zhang, S., Zhang, C., Kang, N., & Li, Z. (2021b). iVPF: Numerical invertible volume preserving flow for efficient lossless compression. *IEEE/CVF Computer Vision & Pattern Recognition*, 620–629. 322

Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. *Neural Information Processing Systems*, *28*, 649–657. 182

Zhao, H., Jia, J., & Koltun, V. (2020a). Exploring self-attention for image recognition. *IEEE/CVF Computer Vision & Pattern Recognition*, 10076–10085. 238

Zhao, J., Mathieu, M., & LeCun, Y. (2017a). Energy-based generative adversarial network. *International Conference on Learning Representations*. 299

Zhao, L., & Akoglu, L. (2020). PairNorm: Tackling oversmoothing in GNNs. *International Conference on Learning Representations*. 265

Zhao, L., Mo, Q., Lin, S., Wang, Z., Zuo, Z., Chen, H., Xing, W., & Lu, D. (2020b). UCTGAN: Diverse image inpainting based on unsupervised cross-space translation. *IEEE/CVF Computer Vision & Pattern Recognition*, 5741–5750. 238

Zhao, S., Song, J., & Ermon, S. (2017b). InfoVAE: Balancing learning and inference in variational autoencoders. *AAAI Conference on Artificial Intelligence*, 5885–5892. 345

Zhao, S., Song, J., & Ermon, S. (2017c). Towards deeper understanding of variational autoencoding models. *arXiv:1702.08658*. 345

Zheng, C., Cham, T.-J., & Cai, J. (2021). TFill: Image completion via a transformer-based architecture. *arXiv:2104.00845*. 238

Zheng, G., Yang, Y., & Carbonell, J. (2017). Convolutional normalizing flows. *arXiv:1711.02255*. 322

Zheng, Q., Zhang, A., & Grover, A. (2022). Online decision transformer. *International Conference on Machine Learning*, *162*, 27042–27059. 398

Zhong, Z., Zheng, L., Kang, G., Li, S., & Yang, Y. (2020). Random erasing data augmentation. *AAAI Conference on Artificial Intelligence*, 13001–13008. 159

Zhou, C., Ma, X., Wang, D., & Neubig, G. (2019). Density matching for bilingual word embedding. *ACL Human Language Technologies*, 1588–1598. 322

Zhou, H., Alvarez, J. M., & Porikli, F. (2016a). Less is more: Towards compact CNNs. *European Conference on Computer Vision*, 662–677. 414

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., & Sun, M. (2020a). Graph neural networks: A review of methods and applications. *AI Open*, *1*, 57–81. 261

Zhou, K., Huang, X., Li, Y., Zha, D., Chen, R., & Hu, X. (2020b). Towards deeper graph neural networks with differentiable group normalization. *Neural Information Processing Systems*, *33*, 4917–4928. 265

Zhou, L., Du, Y., & Wu, J. (2021). 3D shape generation and completion through point-voxel diffusion. *IEEE/CVF International Conference on Computer Vision*, 5826–5835. 369

Zhou, T., Krahenbuhl, P., Aubry, M., Huang, Q., & Efros, A. A. (2016b). Learning dense correspondence via 3D-guided cycle consistency. *IEEE/CVF Computer Vision & Pattern Recognition*, 117–126. 301

Zhou, Y.-T., & Chellappa, R. (1988). Computation of optical flow using a neural network. *IEEE International Conference on Neural Networks*, 71–78. 181

Zhou, Z., & Li, X. (2017). Graph convolution: A high-order and adaptive approach. *arXiv:1706.09916*. 263

Zhou, Z., Rahman Siddiquee, M. M., Tajbakhsh, N., & Liang, J. (2018). UNet++: A nested U-Net architecture for medical image segmentation. *Deep Learning in Medical Image Analysis Workshop*, 3–11. 205

Zhu, C., Ni, R., Xu, Z., Kong, K., Huang, W. R., & Goldstein, T. (2021). GradInit: Learning to initialize neural networks for stable and efficient training. *Neural Information Processing Systems*, *34*, 16410–16422. 113

Zhu, J., Krähenbühl, P., Shechtman, E., & Efros, A. A. (2016). Generative visual manipulation on the natural image manifold. *European Conference on Computer Vision*, 597–613. 301

Zhu, J., Shen, Y., Zhao, D., & Zhou, B. (2020a). In-domain GAN inversion for real image editing. *European Conference on Computer Vision*, 592–608. 301

Zhu, J.-Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. *IEEE/CVF International Conference on Computer Vision*, 2223–2232. 296, 301

Zhu, X., Su, W., Lu, L., Li, B., Wang, X., & Dai, J. (2020b). Deformable DETR: Deformable transformers for end-to-end object detection. *International Conference on Learning Representations*. 238

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., & He, Q. (2020). A comprehensive survey on transfer learning. *Proceedings of the IEEE*, *109*(1), 43–76. 159

Ziegler, Z., & Rush, A. (2019). Latent normalizing flows for discrete sequences. *International Conference on Machine Learning*, 7673–7682. 322, 323

Zong, B., Song, Q., Min, M. R., Cheng, W., Lumezanu, C., Cho, D., & Chen, H. (2018). Deep autoencoding Gaussian mixture model for unsupervised anomaly detection. *International Conference on Learning Representations*. 344

Zou, D., Cao, Y., Zhou, D., & Gu, Q. (2020). Gradient descent optimizes over-parameterized deep ReLU networks. *Machine Learning*, *109*, 467–492. 404

Zou, D., Hu, Z., Wang, Y., Jiang, S., Sun, Y., & Gu, Q. (2019). Layer-dependent importance sampling for training deep and large graph convolutional networks. *Neural Information Processing Systems*, *32*, 11247–11256. 264

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B*, *67*(2), 301–320. 156

Zou, Z., Chen, K., Shi, Z., Guo, Y., & Ye, J. (2023). Object detection in 20 years: A survey. *Proceedings of the IEEE*. 184

Index