

INDEX

Note: *Bold page numbers indicate where a term is defined or given extensive treatment. For classes and functions, bold entries show where they are defined; non-bold entries show where they are used.*

Symbols

& (deeptack operator), 141, 171, 176, 183, 186, 190–192, 194, 200, 215, 230, 235, 242, 256, 269, 286, 308
^ (deeptack operator), 175, 228, 239
>> (deeptack operator), 140, 141, 170, 175, 176, 181, 186, 190, 198, 215, 227–230, 235, 239–242, 256, 269, 272, 286, 308
@ (dot product), **14**, 17, 23, 28, 32, 33, 58, 62, 63, 64, 69

A

A100 GPU, 386
Abbe, Ernst, 463
Abbe's diffraction limit, 463.
See also diffraction limit
accuracy, **39**, 203, 211, 533. *See also*
 binary accuracy
 vs. precision, 136
Accuracy (torchmetrics), 548, 550–552
accuracy_score() (sklearn), 203
action (reinforcement learning), **558**
 in deep Q-learning, 586
 legal, 565
 in Q-table, 578–579
action-value function, **559**
activation function, 2, 3, 19–20
 Heaviside step, 3, 20
 hyperbolic tangent, 19, 20
 leaky ReLU, 19, 20
 linear, 19, 20

ReLU, 19, 20, 43
sigmoid, 19, 20
softmax, 42
activation potential. **2**, **3**, 23, 26–28, 33, 63
active learning, **530**, 556
 with multiple groups, 539–546
 with two groups, 531–539
ActiveLearningDataset (deeplay), 550–552
ADALINE (adaptive linear element), 48
Adam (deeplay, torch), 87, 135, 173, 200, 217, 231, 244, 259, 264, 267, 273, 289, 292, 293, 295, 297, 316, 363, 368, 372, 383, 402, 416, 490, 496, 508, 522, 548, 588
Adam optimizer, **45**
 β_1 and β_2 parameters, 383
 with parameter list, 289
 regularization with weight decay, 363
AdamW (deeplay, torch), 356, 440, 458, 461
adaptive average pooling, 148
AdaptiveAvgPool2d (torch), 118, 148, 547
adaptive linear element (ADALINE), 48
adaptive switching circuits, 48
Add (deeplay), 351
Add (deeptack), 227, 228, 241, 272
additive attention, 335–336
AdditiveAttention, **335**
adjacency matrix, **476–477**, **611**
adversarial learning/training, **376–379**, 423. *See also* generative adversarial networks
adversarial sampling, 549, **552–553**
agent (reinforcement learning), **558**
 in deep Q-learning, 585–586
 in Q-learning, 574–585, 597
AI (artificial intelligence), xxii
AlexNet, xxii, 165

`al_loop()`, 549, 550–552
`all_words_in_vocab()`, 306, 307
`alpha` (parameter LodeSTAR), 275–276
 AlphaFold, xxii, 326, 528
 AlphaGo, 557, 599
`animate()`, 510, 511
 Anki decks, 299
 annotated data (active learning),
 533–538, 541–545, 548–553
 AnnotatedDataset, 133, 134
 anomaly detection, 195, 375
 with autoencoder, 195–206
 using latent space, 204–206
 using reconstruction error,
 201–203
`append()` (deepplay), 353, 486, 487,
 495, 547
 Application (deepplay), 258, 315,
 587–592, 594
 area under the ROC curve
 (AUROC), 117
 ArgmaxJI, 217, 217
 artificial intelligence (AI), xxii
 artificial neuron, 2–3, 48. *See also*
 single neuron
 AsType (deeptrack), 228, 229, 235,
 241–242
 Atari games, 599
 attention, 325, 326–336, 373, 528
 additive, 335–336
 concatenative, 373
 dot-product, 326–331, 373
 standard scaled, 333
 trainable, 333–335
 general, 373
 attention head, 343. *See also* multi-head
 attention
 attention matrix, 327, 328–332
 interpretation, 340–342
 visualization, 331–332
 attention outputs, 327, 329–331
 attention U-Net, 438–439
 architecture, 438
 conditional, 447–448, 457–458, 467
 AttentionUNet (deepplay), 439, 448,
 458, 467
 AUROC (area under the
 ROC curve), 117

autoencoder, 167, 168, 179–207
 for anomaly detection, 195–206
 variational, 179–188
 Wasserstein, 188–195
 average pooling, 119, 148, 266,
 487–489
 axon, 2

B

Ba, Jimmy, 45
 Bachimanchi, Harshith, 472
 backbone, 257–258, 265, 335, 552–553
 convolutional, 257–258, 265, 547–548
 message-passing, 501–502
 ViT as, 370–371
 backpropagation algorithm, 25–27,
 48, 145
 implementation, 27–28, 33, 63
 mathematical derivation, 25–27
`backward()` (torch), 125, 150, 154, 157,
 162, 289, 384, 403, 404, 417, 441,
 462, 593
 `sum().backward()`, 125
 Bahdanau, Dzmitry, 373
 Barcelona, 158
 batch (in PyTorch), 103
 batch processing (graphs), 488–489
 batch training/learning, 45, 65–73
 implementation, 67–68
 increasing batch size, 93
 randomization, 71–73
 BCELoss (torch), 182, 383
 BCEWithLogitsLoss (torch), 244
 Bellman, Richard, 557, 598
 Bellman equation, 559, 598
 BERT, 374
 beta (parameter diffusion model), 433
 beta (parameter LodeSTAR), 275–276
 beta (parameter VAE), 182–183
 beta (regularization factor style
 transfer), 160, 163
 Beta distribution, 366
 betas (parameter Adam), 363, 368, 383,
 402, 416
 bias
 in CNN, 102
 in single neuron, 2, 3, 17–18
 bilinear upsampling, 107

BiLingual Evaluation Understudy (BLEU) score, **321–322**, 339
 binary accuracy, **116**, 119
 BinaryClassifier (deeplay), 115, 119, 356, 522
 binary cross-entropy loss, **115**, 116, 119, 182, 244, 356, 377, 379, 383
 BioSR (super-resolution) dataset, 463–464
 BioSRDataset, **464**, 465
 bisectrix, 66, 67, 69, 262, 265, 267
 BLEU (BiLingual Evaluation Understudy) score, **321–322**, 339
 BLEUScore (torchmetrics), 321
 bmm() (torch), 160
 boredom (reinforcement learning), 561
 bottleneck, **168**, 180, 210, 439
 Brightfield (deeptack), 138, 169
 bright-field microscopy, 138, 170, 178, 394, 409
 Broad Bioimage Benchmark Collection, 234
 Bronte Ciriza, David, 80, 93
 Brown, Robert, 426
 Brownian motion, 128, 426
 Bruna, Joan, 527
 buffer (for dataset), 172
 buffer (to replay experiences), 591.
 See also replay buffer
 build() (deeplay), 267, 273, 363, 368, 372, 381, 382, 391, 392, 400, 401, 414, 415, 439, 467, 501, 548, 550–552, 588
 build_vocab_from_iterator(), **304**, 305, 348, 453
 butterfly effect, 602, 607–610
 bwd_hook_func(), **125**

C
 Caicedo, Juan C., 234
 Callegari, Agnese, 82
 candidate gate (LSTM), 297
 Cat (deeplay), 217
 CategoricalClassifier (deeplay), 548
 categorical cross-entropy loss, **43**, **212**
 cell counting dataset, 234–235
 cell counting with U-Net, 234–247
 cell gate (LSTM), 297
 cell localization with LodeSTAR, 268–277
 CellTracingDataset, **519**, 521
 Cell Tracking Challenge dataset
 BF-C2DL-HSC, 268–269
 DIC-C2DH-HeLa, 512–514
 cell trajectory identification with
 MAGIK, 511–526
 centroid (position), 232, 270, 515
 chaos, 602, 607, 608
 chaotic system, 602
 chatbots, 299
 ChatGPT, xxii
 Chen, Ting, 278
 Cho, Kyunghyun, 323
 Christiansen, Eric M., 394
 Çiçek, Özgün, 248
 CIFAR-10 dataset, 358–359
 clamp_() (torch), 150
 classification head, 362, 370, 522, 548
 classification of malaria-infected blood smears, 110–128
 classification of MNIST digits
 with active learning, 547–555
 with DNN, 34–47
 Classifier (deeplay), 37, 363, 367, 372
 classifier-free guidance. 446–453, 458–460
 classifier guidance, 446
 CLIP (Contrastive Language-Image Pre-training), 326, **450**
 Clip (deeptack), 235, 241
 CLIP text encoder, 460–463
 CLIPTextModel (transformers), 461
 CLIP tokenizer, 460–463
 CLIPTokenizer (transformers), 461
 clone() (torch), 121, 123, 125, 151, 184, 365, 520
 closure(), 161, **161**
 closure function, 161–163
 code examples
 1-1: *Classifying 1D Data with a Single Neuron*, 9
 1-2: *Classifying 2D Data with a Single Neuron*, 16
 1-3: *Classifying 2D Data with a Two-Layer Neural Network*, 29
 1-A: *Classifying the MNIST Digits*, 47

- code examples (continued)
- 2-1: *Regressing 1D Data with a Single Neuron*, 55
 - 2-2: *Regressing 2D Data with a Single Neuron*, 60
 - 2-3: *Regressing 2D Data with a Two-Layer Neural Network*, 64
 - 2-4: *Fitting Data with a Neural Network Trained Using Batch Training*, 73
 - 2-5: *Training a Neural Network Splitting the Data*, 79
 - 2-A: *Simulating the Forces Acting on an Optically Trapped Particle*, 92
 - 3-1: *Implementing Neural Networks in PyTorch*, 110
 - 3-A: *Classifying Blood Smears with a Convolutional Neural Network*, 128
 - 3-B: *Localizing Microscopic Particles*, 145
 - 3-C: *Creating DeepDreams*, 158
 - 3-D: *Transferring Image Styles*, 164
 - 4-1: *Denoising Images*, 178
 - 4-A: *Generating Digit Images*, 188
 - 4-B: *Interpolating Between Images*, 195
 - 4-C: *Detecting ECG Anomalies*, 206
 - 5-1: *Segmenting Biological Tissue Images*, 223
 - 5-A: *Detecting Quantum Dots*, 234
 - 5-B: *Counting Cells with a U-Net*, 247
 - 6-1: *Localizing Particles Using LodeSTAR*, 267
 - 6-A: *Localizing Multiple Cells Using LodeSTAR*, 277
 - 7-1: *Predicting Temperatures using Recurrent Neural Networks*, 298
 - 7-A: *Translating with a Recurrent Neural Network*, 322
 - 8-1: *Understanding Attention*, 336
 - 8-A: *Translating with Attention*, 342
 - 8-B: *Predicting Sentiment Using a Transformer*, 357
 - 8-C: *Classifying Images with a Vision Transformer*, 372
 - 9-1: *Generating New MNIST Digits with a GAN*, 389
 - 9-A: *Generating MNIST Digits On Demand*, 394
 - 9-B: *Virtually Straining a Biological Tissue with a Conditional GAN*, 409
 - 9-C: *Converting Microscopy Images with a CycleGAN*, 422
 - 10-1: *Generating Digits with a Diffusion Model*, 445
 - 10-A: *Generating Bespoke Digits with a Conditional Diffusion Model*, 449
 - 10-B: *Generating Images of Digits from Text Prompts*, 463
 - 10-C: *Generating Super-Resolution Images*, 471
 - 11-1: *Predicting Molecular Properties*, 497
 - 11-A: *Simulating Complex Physical Phenomena*, 511
 - 11-B: *Identifying Trajectories with MAGIK*, 526
 - 12-1: *Training a Binary Classifier with Active Learning*, 539
 - 12-2: *Training a Three-Class Classifier with Active Learning*, 546
 - 12-A: *Training a MNIST Digits Classifier with Active Learning*, 555
 - 13-1: *Teaching a Deep Q-Learning Agent to Play Tetris*, 597
 - 14-1: *Training a Reservoir Computer to Predict the Lorenz System*, 619
- Cohn, David, 556
- `collate()`, 349, 349
- comb filter, 280–281
- commonsense benchmark, **287**
- compiling (a neural network), **37**
- Compose (torchvision), 113, 149, 360, 361, 371, 380, 398, 412, 432, 465, 521
- computational device, 287–288
- computational graph, 91, 121, 184, 385
- `compute_connectivity()`, **503**, 503
- `compute_graph()`, **503**, 506, 509
- `compute_node_attr()`, **502**, 503
- ComputeTrajectories, **523**, 524
- Compute Unified Device Architecture (CUDA), 288
- CompVis, 425
- concatenative attention, 373
- `configure()` (deeplay), 36, 42, 44, 45, 118, 199, 353, 400, 401, 522, 547
- `configure_optimizers()` (deeplay, Application method), 368
- confusion matrix, **40**
- plotting, 40–41

connected_components() (networkx), 524
Connect Four, 560
 connection (graph), 476. *See also* edge
 constants() (deepttrack, Source
 method), 213
 context manager, **153–154**. *See also* with
 context vector (seq2seq model), **299**,
 308–311, 337–338
 Contrastive Language-Image Pre-
 training (CLIP), 326, **450**
 contrastive learning, 252–255, 277,
 278, 450
 contrastive loss, 252
 Conv1d (torch), 199
 Conv2d (torch), 102, 108, 147–148
 convex dataset, 10
 convolution, **96–100**
 1D, 96–98
 2D, 98–100
 graph, **476–479**
 convolutional base, **109**, 110, 119
 ConvolutionalEncoderDecoder2d
 (deeplay), 173, 199
 convolutional layer, **100**, 102–104
 activation, 121–124
 ConvolutionalNeuralNetwork (deeplay),
 118, 134, 257, 401, 547
 ConvTranspose1d (torch), 199
 coregistration, 395, 409
 corpus file (NLP), 299
 corpus iterator (NLP), 302–303
 corpus_iterator(), **302**, 305
 correlation matrix analysis, 284
 Cortes, Corinna, 34
 CosineAnnealingLR (torch), 367–368
 CPU, xxiv, 38, 91, 244, **287–288**, 312,
 381, 386, 387
 cpu() (torch), 91, 245, 386, 393, 405,
 419, 434, 438, 443, 445, 470, 509
 create() (deeplay), 36, 37, 42, 44, 45,
 87, 115, 119, 135, 143, 173, 178,
 182, 191, 200, 217, 221, 231, 244,
 259, 264, 292, 293, 295, 297, 318,
 355, 356, 458, 460, 489, 490, 494,
 496, 508, 522
 create_batch(), **366**, 366, 369
 Crop (deepttrack), 235
 CropTight (deepttrack), 239
 cross-attention, **326**, 326–337, 345, 457,
 458. *See also* attention
 cross-entropy loss, 211
 binary, **115**, 116, 119, 182, 244, 356,
 377, 379, 383
 categorical, **43**, **212**
 sparse, 212
 weighted, 224
 CrossEntropyLoss (torch), 217, 231, 363
 .csv files, 3, 12, 83, 89, 90, 196, 219, 283
 CSVLogger (lightning), 89, 218, 221
 CUDA (Compute Unified Device
 Architecture), 288
 cuda (torch), 288
 curiosity-driven exploration
 (reinforcement learning),
 560–561
 current reward (reinforcement
 learning), 559, 594
 Cursor (matplotlib), 130–131
 custom text encoder implementation
 (NLP), 455–457
 CutMix, **365**, 366
 CutMix augmentation, **364**
 CutMixClassifier, **367**, 368
 cutoff (classifier), 116
 cutoff (parameter LodeSTAR),
 275–276
 cvtColor() (cv2), 128
 Cybenko, George, 49
 Cybenko’s theorem, 49
 cycle consistency, **410**, 418, 420
 CycleGAN, **410**, 424
 CycleGANDiscriminator (deeplay), 415
 CycleGANResnetGenerator (deeplay), 414
 cycle graph, 476–479

D

DALL-E, xxii, 425, 450
 Data (torch_geometric), 349, 505–506,
 509, 516–517, 519
 data augmentation, 214, 252–255, 372,
 412, 504, 519–521
 CutMix, 364
 Mixup, 364
 with VAE, 183
 DataFrame (pandas), 346, 357
 data loader, **38**

DataLoader (deeplay, torch), 38, 39, 45, 88, 115, 134, 142, 172, 177, 183, 186, 191, 200, 218, 222, 232, 235, 244, 260, 267, 272, 286, 308, 349, 361, 364, 372, 383, 403, 416, 440, 468, 547
 DataLoader (torch_geometric), 489, 507, 521
 data pipeline
 with augmentations, 214–215
 with labels, 186, 190, 215
 to load an image crop, 272
 to load images, 181
 to load text, 307–308
 to load traces, 196–199, 286
 with segmentations, 235, 269
 dataset
 BioSR (super-resolution), 463–464
 cell counting, 234–235
 Cell Tracking Challenge
 BF-C2DL-HSC, 268–269
 DIC-C2DH-HeLa, 512–514
 CIFAR-10, 358–359
 electrocardiogram (ECG), 196–197
 English-to-Spanish translations, 299–300
 Fashion-MNIST, 189–191
 Holo2Bright, 411
 Human Motor Neuron, 394–395
 IMDb Large Movie Review, 345–347
 Jena Climate, 282–284
 malaria-infected blood smears, 111–112
 microscopic particle videos, 128–130
 MNIST, 1, 34–35, 181, 380–381, 432–433, 547
 with sentences, 451–452
 optical forces, 80–81
 quantum dot image, 224–225
 SAND (particle simulation), 497–500
 segmented tissue images, 213, 216
 ZINC (molecular properties), 483–486
 Dataset (deeptack), 183, 186, 191, 200, 215, 222, 232, 235, 242, 256, 272, 286, 308
 Dataset (torch), 88, 133, 142, 172, 395, 397, 411, 464, 504, 505, 519
 Dayan, Peter, 598
 DCGANDiscriminator (deeplay), 382, 392
 DCGANGenerator (deeplay), 381, 391
 DDPM (denoising diffusion probabilistic models), **427–432**
 decision boundary, 15, 20, 535–538, 543, 544
 decoder, **168**, 180, 182–185, 188, 199, 218, 299, 310–315, 336–339, 345, 376, 400–401, 501
 decoder-only transformer, 345
 deep belief nets, 49
 deep learning, **xxii**
Deep Learning: A Visual Approach (Glassner), xxiii
 deep learning revolution, xxi
 deepcopy() (copy), 360, 371
 deepdream(), **152**, 153, **154**, 154, **156**, 157
 DeepDreams, **145–158**
 Deeplay, xxiv, 36, 87, 279, 287, 292, 315, 362, 401, 414, 415, 439, 588
 DeeplayModule (deeplay), 309–311, 313, 314, 330, 334, 335, 337, 338, 343, 350, 351, 353, 481, 482, 486–488, 495
 DeepTrack2, 137, 286, 308
 denoising diffusion probabilistic models (DDPM), **427–432**
 denoising encoder-decoder, 169–179
 dense neural networks, 20–33, 61–65
 with three layers, 31–33
 with two layers, 20–31, 61–65
 dense top, 109, 112, 118–119, 124, 134, 257, 265–266, 353, 361, 362, 487–488, 496
 depth (of a neural network), 21
 detach() (torch), 91, 103–105, 107, 121, 123, 125, 126, 136, 143, 151, 160, 174, 175, 177, 178, 184, 186, 192–194, 200, 201, 203–205, 232, 245, 260, 262, 264, 267, 273, 312–314, 331, 369, 386, 393, 417, 419, 523, 590
 deterministic chaos, 602. *See also* chaos
 detransformation, 254
 Dhariwal, Prafulla, 472
 diffraction limit, 255, 463
 diffraction-limited particle images, 256

Diffusion, **433**, 434, **435**, 440, **444**, **447**,
 452, **467**, 468
 diffusion (in physics), 426
 diffusion equations, 426
 diffusion trajectory, 432
 digital twin, 52, 79
 of a physical system, 79–92
 directed graph, 477
 discount factor (reinforcement
 learning), 559, 580–581, 594
 discriminator, **377–378**, 382, 423
 conditional, **390**, 391–392
 distance_matrix() (scipy), 276
 Divide (deeptack), 228, 235, 241
 dnn2_clas(), **23**, 24, 29
 dnn2_reg(), **62**, 62, 67, 68, 75, 77
 dnn3_clas(), **32**, 32
 Doersch, Carl, 278
 Dosovitskiy, Alexey, 374
 dot-product attention, 328–331, 373
 standard scaled, 333
 trainable, 333–335
 DotProductAttention, **330**, 330, 337, 338
 downsampling layer, 105
 DQAgent, **587–595**, 596, 597
 dropout, 148, **295–298**, 355, 387,
 551, 615
 Dropout (torch), 148, 351, 353
 dropout rate, 295
 d_sigmoid(), **28**, 28, 33, 63, 68
 dying ReLU problem, 105
 dynamically learned loss function, 377

E
 early stopping, 73, 220–222
 EarlyStopping (lightning), 221
 Earth mover’s distance, 188
 eccentricity (of object), 515
 edge (graph), 476–479, 611
 edge attributes, 491
 edge index, 492
 eigenvalues (graph), 611
 Einstein, Albert, 426
 ElasticTransformation (deeptack), 239
 electrocardiogram (ECG) dataset,
 196–197
 Ellipsoid (deeptack), 238, 239
 Elman, Jeffrey L., 323

Embedding (torch), 309, 310, 353, 455,
 487, 495
 embedding dimension, 328, 329, 354,
 390, 391, 447, 457, 458, 487
 embedding layer, 390
 encoded distribution, 188
 encoder, **168**, 180, 182, 186–188, 199,
 218, 299, 308–315, 336, 345,
 351–355, 376, 501
 encoder-decoder, **168–169**
 encoder-only transformer, 345
 encoding
 contrastive learning, 252
 geometric learning, 254
 non-contrastive learning, 253
 English-to-Spanish translations dataset,
 299–300
 entropy sampling, 545
 environment (reinforcement
 learning), 558
 epoch, **38**
 epsilon-greedy policy/strategy, 579, 585,
 587, 590, 591, 597
 equivariance, 254
 error backpropagation.
 See backpropagation algorithm
 estimated optimal reward
 (reinforcement learning), 559
 Euclidean norm, 19
 Euler integration scheme, 500, 509–510
 eval() (torch), 147, 191, 356, 369, 386,
 392, 405, 419, 442, 444, 448, 459,
 462, 469, 509, 523. *See also* train()

evaluate_model()
 for CycleGAN, **419**, 420
 for virtual staining, **405**, 406
 experience buffer, 591.
 See also replay buffer
 exploration vs. exploitation
 (reinforcement learning), 560–561,
 579, 587, 591
 exposure bias, 314

F
 F1 score, 276, 523
 f1_score() (sklearn), 523
 failure analysis, 46–47
 fallout, 117

false-positive rate (FPR), 116
 Fashion-MNIST dataset, 189–191
 fast forward diffusion process, 429–434
 feature map, **99–100**, 103–110, 160, 182, 210–211, 363, 402, 415, 438, 439, 457, 467, 548
 feedback (reinforcement learning), 558, 573
 feedback loop/mechanism (RNN), 280, 281
 filter, 96
 2D, 98–99
 Prewitt, 99
 Sobel, 99
 Gaussian, 97, 98, 99
 matching pattern, 98, 99
 Prewitt, 98
 rectangular, 97
 RGB, 100
 smoothing, 98
 Sobel, 97, 98
 valid complete placements of, 97, 99
 filter() (deeptrack), 196
 find_contours() (skimage), 525
 fit() (torch), 38, 42, 44, 45, 89, 116, 119, 135, 143, 173, 178, 183, 191, 200, 204, 218, 221, 232, 244, 260, 264, 267, 273, 292, 293, 295, 297, 319, 356, 363, 368, 372, 490, 496, 508, 522, 532, 548, 549
 flanging, 280
 Flatten (torch), 109, 134, 257, 547
 FlipLR (deeptrack), 215
 flip_transform(), **263**, 263
 FlipUD (deeptrack), 215
 Fluorescence (deeptrack), 226, 228, 238, 256
 fluorescence microscopy, 226, 227, 234, 238, 256, 394
 forget gate (LSTM), 297
 forward diffusion process, **427**, 427–429
 fast, 429–430
 implementation, 433–435
 forward filling, 219
 FPR (false-positive rate), 116
 from_numpy() (torch), 91, 232, 273, 275, 276
 from_pretrained() (transformers), 371, 461
 fully connected neural network, 21.
 See also dense neural networks
 FuncAnimation (matplotlib), 499, 510, 525
 future reward, 558, 559, 578, 581, 594
 Fwd_Hook, **153**, 154
 fwd_hook_func(), **125**, 125
 Fwd_Hooks, **156**, 157, 160, 162

G
 GANs. *See* generative adversarial networks
 GATs (graph attention networks), 528
 gated recurrent unit (GRU), **294–296**
 gather() (torch), 315
 Gatys, Leon A., 165
 Gaudí, Antoni, 158
 Gaussian (deeptrack), 241, 256
 Gaussian blur, 240
 GaussianBlur (deeptrack), 240
 Gaussian Error Linear Unit (GeLU), 588
 Gaussian filter, 97–99
 Gaussian sampling, **179**, **180**
 GCL, **481–482**, 486, 487
 GCN, **486–488**, 489
 GCNs (graph convolutional networks), **486–489**, 527
 GeLU (Gaussian Error Linear Unit), 588
 GELU (torch), 456
 general attention, 335, 373
 generative adversarial networks (GANs), 375, 377, **376–379**
 conditional, 390
 training
 algorithm, 379
 implementation, 384–386
 generator, **377–379**, 381–382, 423
 conditional, **390**, 390–391
 geometric contrastive learning, 278
 geometrical–optics approximation (optical tweezers), 79
 get_device(), **288**, 288, 381, 400, 414, 433, 466, 508
 get_glove_embeddings(), **317**, 318, 328, 355

`get_mask()`, 242, 242
 Gilmer, Justin, 527
 Glassner, Andrew, xxiii
 global average pooling, 266, 487–489
 Global Vectors for Word Representation
 (GloVe), 316–318, 328, 332, 355
 GNNs (graph neural networks), 475
`gnp_random_graph()` (networkx), 612
Go (game), xxii, 557, 599
 G0Dataset, 88, 88
 Goodfellow, Ian, 379, 423
 Google, 145, 325, 425
 Google Colab, xxiv
 Google Imagen, 425
 GoogLeNet, 165
 Google Research, 358
 GPT, 374
 GPT-4, xxi
 GPU, xxiv, 38, 91, 244, **287–288**, 312,
 381, 386
 A100, 386
 gradient ascent, 149–151
 gradient descent, 25
 gradient-weighted class activation
 mapping (Grad-CAM), 124–127
 GradualWarmupScheduler
 (warmup_scheduler), 367
`gram()`, 160, 162
 Gram matrix, 160–163
 Graph (networkx), 524
 graph, 476, 611
 directed, 477
 undirected, 477
 graph attention network (GAT), 528
 graph convolution, **476–479**
 GraphConvolution, 480, 481
`graph_convolution()`, 478, 478
 graph convolutional network,
 486–489, 527
 graph convolution layer, **480–482**
 GraphFromSegmentations, **514–516**, 517
 graph network-based simulator, 497,
 500–502
 graph neural networks (GNNs), 475
 graph theory, 611
 GraphToEdgeMAGIK (deeplay), 522
 GraphToNodeMPM (deeplay), 501
 Grill, Jean-Bastien, 278

GRU (gated recurrent unit), **294–296**
 GRU (torch), 309, 310
 guidance strength, 446
 GUI for particle position, 130–132

H

Haas, Harald, 620
 Hadamard product, 294, 297
 handling contractions (NLP), 301
 hash table (analogy with dot-product
 attention), 330
 Hassabis, Demis, 528
 He, Kaiming, 165
 heatmap
 attention, 331–332, 340–342
 with Grad-CAM, 124–127
 Heaviside step function, 3, 19, 20
 Helgadóttir, Saga, 128, 165
 hidden layer, 20, 27
 hidden state, 281, 289, 299, 309,
 336, 337
 Hinton, Geoffrey E., 49, 207
`hist()`, **202**, 202–205
 history (deeplay, Trainer field), 292,
 319, 363
 Ho, Jonathan, 431, 472
 Hochreiter, Sepp, 323
 Hoff, Marcian E., 48
 holdout dataset, 74
 Holo2Bright dataset, 411
 Holo2BrightDataset, **411**, 413
 holographic microscopy, 409
`hook_func()`, **121**, 122, **123**, **150**, 150
 hook function, 121
 backward, 125
 forward, 125
 Hopfield, John J., 620
 Hopfield networks, 620
 Hornik, Kurt, 49
 HTML (IPython), 499, 510, 525
 Hugging Face (library), 345, 370, 372
 Human Motor Neuron dataset, 394–395
 hyperbolic tangent function, 19, 20, 105
 hyperparameters, 40, 73

I

IBM Watson, xxii
 Identity (numpy), 614

- Identity (torch), 134, 199
- Image (PIL), 121, 146, 151, 158, 159, 224, 395, 396, 411, 412
- image classification with a ViT, 358–372
- image conversion with CycleGAN, 409–422
- image embedding, 450
- ImageFolder (deepttrack), 181, 189, 213, 235
- ImageFolder (torchvision), 111, 113
- image generation
 - conditional
 - with diffusion model, 446–450
 - with GAN, 389–394
 - with diffusion model, 432–445
 - with GAN, 379–389
 - with VAE, 183–185
 - with WAE, 192–193
- Imagen (Google), 425
- ImageNet Large Scale Visual Recognition Challenge, xxii, 147, 148, 165, 364, 370
- image patch embeddings, 362
- image segmentation with U-Net, 211–224
- image super-resolution, 463–471
- image_to_tensor(), 149, 150, 153, 154, 160
- image_translation(), 258, 259, 263
- imbalanced data classes (training), 196
- imdb_iterator(), 348, 348
- IMDb Large Movie Review dataset, 345–347
- immediate reward, 559, 594
- imread() (cv2), 513
- imread() (matplotlib), 34, 39, 452
- imread() (tiff), 464
- Inception (architecture), 165
- Inception* (movie), 145
- inceptionism, 145
- increasing batch size (for training), 93
- IndexedPositionalEmbedding (deeplay), 353
- inductive bias, 364
- input (neuron), 2
- input gate (LSTM), 297
- InstanceNorm2d (torch), 400, 401
- instance segmentation, 211
- interobserver variability, 137
- intersection over union (IoU), 216
- intraobserver variability, 137
- invariance, 254
- inverse_flip_transform(), 263, 264
- inverse_translation(), 258, 259, 264
- IOU (intersection over union), 216
- Isensee, Fabian, 248
- isotropic_erosion() (skimage), 242

J

- Jaccard index, 216–217
- Jaeger, Herbert, 620
- Jena Climate dataset, 282–284
- Johnson, Justin, 423
- Join (deepttrack), 181, 189, 213
- Jumper, John, 528
- Jupyter Notebooks, xxiv

K

- Kaggle Notebooks, xxiv
- Kaiming (deeplay), 552
- Kaiming initializer, 553
- kernel, 102. *See also* filter
- Kingma, Diederik P., 45, 207
- Kipf, Thomas N., 527
- KL (Kullback–Leibler) divergence, 180, 189
- k*-nearest neighbors algorithm, 204
- Kneusel, Ronald T., xxiii
- Krizhevsky, Alex, 165
- Kullback–Leibler (KL) divergence, 180, 189

L

- L1Loss (torch), 173, 200, 259, 264, 289, 402, 415, 468, 490, 496
- l1_loss() (torch), 201, 203
- L_1 norm, 19, 416
- L_2 norm, 19, 416
- L_2 -norm pooling, 106
- label() (skimage), 232, 236, 245
- labeling oracle, 530
- label_trans(), 113, 113, 121
- Lambda (deepttrack), 215
- large language models (LLMs), xxii
- large margin sampling, 545
- Latent Diffusion Models, 425, 472

latent representation, 168, 180, 390
 latent space, **168**, 180, 188, 204–206, 377
 Layer (deeplay), 118, 134, 309, 310, 337, 343, 351, 353, 354, 481, 487, 494, 495, 547
 LayerList (deeplay), 36, 353, 486, 487, 495
 LayerNorm (torch), 455
 LayerNorm (torch_geometric), 351
 LazyLinear (torch), 257, 492, 494
 LBFGS (torch), 161
 L-BFGS (Limited-memory Broyden-Fletcher-Goldfarb-Shanno) algorithm, 161–163
 leaky ReLU, **19**, 20, 105
 LeakyReLU (torch), 400, 401
 learned perceptual image patch similarity (LPIPS), 402
 LearnedPerceptualImagePatchSimilarity (torchmetrics), 402
 learning rate, 7, 8
 LeCun, Yann, 34, 165
 legal action (reinforcement learning), 565
 LeNet-5 (architecture), 165
 Lenton, Isaac C.D., 93
 lerp() (torch), 447
 Lightning, xxiv, 287
 Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) algorithm, 161–163
 Linear (torch), 36, 109, 148, 288, 310, 334, 335, 343, 351, 353, 371, 456, 481, 487
 LinearBlock (deeplay), 36
 linear function, **19**, 20
 linear_sum_assignment() (scipy), 276
 link (graph), 476
 LLMs (large language models), xxi
 load() (json), 451, 498
 load() (numpy), 82–83, 133, 498
 load() (torch), 508, 597
 load_data(), **12**, 12, 22, 30, 52, 56, 57, 61, 65, 74, 77
 load_data_1d(), **3**, 4, 10
 load_data_file(), **84**, 84, **85**, 86
 loader.py, 4, 9, 12, 16, 29, 52, 55, 60, 64, 73, 79
 load_from_checkpoint() (lightning), 491, 496
 load_glove_embeddings(), **317**, 318, 328, 355
 LoadImage (deeptack), 181, 190, 215, 235, 269
 load_images(), **512**, 513, 522
 load_npz_data(), **498**, 498
 load pretrained embeddings (NLP), 316–318
 load_state_dict() (torch), 508, 595, 597
 load_video(), **128**, 129
 LodeSTAR (deeplay), 267, 273
 LodeSTAR (Localization and Detection from Symmetries, Translations, and Rotations), **255–268**, 278
 alpha parameter, 275
 beta parameter, 275
 cutoff parameter, 275
 displacement channels, 266
 mode parameter, 275
 probability channel, 266
 training with flipping, 262–265
 translation equivariant training algorithm, 257
 implementation, 258–260
 log() (lightning), 259, 368
 logic gates, 22, 32
 LogisticRegression (sklearn), 532
 logistic regression model, 532–533
 logits, 42, 212
 log_metrics() (lightning), 368
 log_output() (deeplay, Application method), 368
 logP (penalized water-octanol partition coefficient), 483
 Long, Jonathan, 207
 long short-term memory (LSTM), **296–298**, 323
 Lorenz, Edward, 602
 lorenz(), **604**, 604, 607, 613
 Lorenz attractor, 603, 606–607
 lorenz_step(), **603**, 604
 Lorenz system
 definition, 602–603
 numerical integration, 603–604
 time evolution, 604–605

loss() (deepplay, Application method), 259, 367, 593
 loss function, **25**
 dynamically learned, 377
 weighted, 231
 LPIPS (learned perceptual image patch similarity), 402
 L_p norm, 19
 LSTM (long short-term memory), **296–298**, 323
 Luong, Minh-Thang, 373
 Lyapunov exponent, 608, 610, 618

M

machine learning, **xxii**
Machine Learning with Neural Networks: An Introduction for Scientists and Engineers (Mehlig), xxiii
 MAE. *See* mean absolute error
 MAGIK, 511, 521–522, 528
 malaria-infected blood smears dataset, 111–112
 ManualAnnotation, **130**, 131
 manual annotation (of particle positions), 130–134
ManyThings.org website, 299
 Margin (deepplay), 551, 552
 Markov decision process, 598
 Markov process, 427
 maskedNLL(), **315**, 316
 mask_to_positions(), **232**, 232
Math for Deep Learning (Kneusel), xxiii
 Matplotlib, xxiv
 %matplotlib inline (standard backend), 132
 %matplotlib ipynpl (interactive backend), 131
 Matthes, Eric, xxiii
 Max Planck Institute for Biogeochemistry, 282
 MaxPool1d (torch), 199
 MaxPool2d (torch), 105, 108, 109, 118, 134, 147–148, 257, 547
 max pooling, 105
 McCulloch, Warren S., 48
 MeanAbsoluteError (torchmetrics), 87, 135
 mean absolute error (MAE), 87, 135, 143, 245, 287, 402, 416, 491, 509
 mean percentage error (MPE), 245
 mean squared error (MSE), 26, 37, 68, 87, 162, 183, 402, 416, 431, 508, 613, 615
 Mehlig, Bernhard, xxiii
 memory, 280
 message (GNN), 492
 message-passing layer, **492–495**
 message-passing network, **495–496**, 527
 Metal Performance Shaders (MPS), 288. *See also* GPU
 MetricCollection (deepplay), 37–38
metrics.csv file, 89, 90, 219
 microscopic particle videos dataset, 128–130
 Midtvedt, Benjamin, 255, 278
 MieSphere (deepplay), 137, 139
 mini-batch training. *See* batch training/learning
 min pooling, 106
 Mixup augmentation, 364
 MNIST handwritten digit database, 1, 34–35, 181, 380–381, 432–433, 547
 with sentences, 451–452
 mode (parameter LodeSTAR), 275–276
 mode collapse, **175–176**, 253, 382, 389, 393, 394, 444
 ModelCheckpoint (lightning), 490, 496, 508
 model-free reinforcement learning, 559, 598
 Module (torch), 370, 455, 480, 492–494
 molecular property prediction with GNN, 479–497
 MolecularRegressor, **490**, 490, 491, 496
 momentum (of an optimizer), 45
 Mordvintsev, Alexander, 145
 morphing images with WAE, 193–195
 MoveAxis (deeptrack), 170, 175, 176, 181, 190, 215, 230, 235, 242, 256, 272
 moving average, 96
 MPE (mean percentage error), 245
 MPN, **495**, 496
 MPS (Metal Performance Shaders), 288. *See also* GPU

mps (torch), 288
MSE. *See* mean squared error
MSELoss (torch), 37, 87, 135, 161, 191,
402, 415, 440, 508, 588
MulticlassAccuracy (torch), 37–38
MulticlassJaccardIndex
(torchmetrics), 217
multi-head attention, **342–344**, 345
MultiHeadAttentionLayer, **343**, **350**, 351
multilayer perceptron, 21. *See also* dense
neural networks
MultiLayerPerceptron (deeplay), 36, 38,
87, 115, 118, 134, 547, 552, 588
multimodal learning, 358
Multiply (deeptack), 272

N

natural language processing. *See* NLP
NearestNeighbors (sklearn), 204
nearest neighbor upsampling, 107
negative log-likelihood loss, 315
negative pairs (contrastive learning), 252
NetworkX, 612
neural style transfer, **158–164**, 165
neuron. *See also* single neuron
artificial, **2–3**, 48
biological, **2**, 48
neuron_clas_1d(), 5, 6, 8
neuron_clas_2d(), 14, 14, 15
neuron_clas_2d_bias(), **17**, 18
neuron_reg_1d(), **53**, 53, 54
neuron_reg_2d(), **58**, 59
new() (deeplay), 548, 550–552
NFC (Normalization Form C), 302
NFD (Normalization Form D), 302
n-gram, 321
Nichol, Alexander, 472
NLP (natural language processing), 299
corpus file, 299
corpus iterator, 302–303
custom text encoder implementation,
455–457
handling contractions, 301
pretrained embeddings loading,
316–318
removing noise, 301–302
tokenization, 300
vocabulary building, 303–306

nnU-Net, 248
Nobel Prize in Chemistry
Hassabis, Demis, 528
Jumper, John, 528
Nobel Prize in Physics
Hinton, Geoffrey E., 49
Hopfield, John J., 620
node (graph), 476, 611
node attributes, 476, 477
no_grad() (torch), 290, 314, 338, 339,
405, 435, 444, 509, 590, 592, 593
non-contrastive learning, 277, 278
NonOverlapping (deeptack), 241
normalization
output, 42
weight, 18–19
Normalization Form C (NFC), 302
Normalization Form D (NFD), 302
normalize() (cv2), 128
Normalize (torchvision), 149, 151, 360,
361, 371, 380, 398, 412, 432, 465
normalize() (unicodedata), 301, 302,
307, 320, 348
normalized() (deeplay), 400, 401
normalize_image(), **366**, 366, 369
NormalizeMinMax (deeptack), 141, 170,
175, 176, 181, 190, 215, 256
.npy files, 82, 83, 133
NumPy, xxiv
numpy() (torch), 91, 114, 122, 126, 136,
143, 151, 184, 186, 260, 262, 264,
267, 273, 369, 386, 393, 399, 405,
413, 419, 434, 438, 443, 445, 470,
484, 509, 523, 524, 590

O

observe_tetris_with_gui(), **575**, 576,
579, 583, 597
OneHot (deeptack), 229
one-hot encoding, 37, **212**, 229
online network (non-contrastive
learning), 253
online training, **65**
open() (PIL, Image method), 121, 146,
158, 159, 224, 396, 411, 412
open() (Python), 3, 12, 81, 84, 85, 302,
307, 317, 451, 453, 498
OpenAI, 425, 450

optical forces dataset, 80–81
 optical tweezer, 79
 optimal transport theory, 188, 207
 optimizer, **37**, 45
 Adam, 45
 momentum of, 45
 RMSprop, 45, 115
 stochastic gradient descent, 37
 OTGO (toolbox for optical tweezers
 in the geometrical optics regime),
 82, 84
 output (neuron), 2, 3
 output gate (LSTM), 297
 output layer, **20**
 output normalization, 42
 overfitting, 39, 73, **77**, 78, 93, 148,
 219–222, 284, 290, 293, 296, 297,
 352, 360, 386, 443, 550, 586, 602,
 614, 615

P

pad(), **306**, 307, 320
 Pad (deepttrack), 239
 pad_and_process(), **454**, 455, 457, 459
 pairwise_distances() (sklearn), 205
 Pajitnov, Alexey, 561
 Pandas, 89
 read metrics file with, 89
 panoptic segmentation, **211**
 Parameter (torch), 102
 parameters() (torch), 37, 87, 289, 312,
 314, 320, 321, 383, 402, 416, 440,
 458, 461, 462, 588
 Parc Güell, 158
 ParticleDataset, **504–505**, 507
 particle localization (microscopy)
 with convolutional network, 128–145
 with LodeSTAR, 255–268
 with U-Net, 224–234
 ParticleLocalizer, **258**, 259, 263
 ParticleLocalizerWithFlips, **263**, 264
 penalty (reinforcement learning), 558
 perceptron, 48
 perceptual loss, 402, 404, 407, 423
 permute() (torch), 142, 171, 215, 216,
 236, 243, 273, 275, 276, 344, 366,
 369, 405, 413, 419, 443, 445, 465, 470
 phase-space plot, 606

Pineda, Jesús, 511, 528
 pipeline, **138**. *See also* data pipeline;
 simulation pipeline
 pitch shifting, 280
 Pitts, Walter, 48
 pixel-wise multiclass classification, 211
 playing *Tetris*
 with command line, 568–569
 with deep Q-learning, 585–597
 with GUI, 569–572
 with Q-learning, 573–585
 play_tetris_with_gui(), **570**, 572
 plot_activations(), **122**, 123
 plot_attention(), **331**, 331, 340
 plot_blood_smears(), **111**, 112, **113**, 114
 plot_channels(), **103**, 103–105, **107**
 plot_class_examples(), **358**, 359
 plot_data_1d(), **4**, 5, 52
 plot_data_2d(), **13**, 13, 57
 plot_image() (Code Example 3-1),
 101, 101
 plot_image() (Code Example 4-1), **171**,
 171, 174, 175, 177
 plot_model(), **534**, 535, 537
 plot_model3(), **542**, 543, 544
 plot_mse(), **70**, 70, **72**, 72
 plot_mse_train_vs_val(), **76**, 76
 plot_position_comparison(), **260**, 261,
 262, 264, 267
 plot_pred_1d(), **6**, 6, 54
 plot_pred_2d(), **14**, 15, 24, 29, 59, 62,
 75, 77
 plot_pred_vs_gt(), **66**, 67, 75, 77
 plot_roc(), **117**, 117, 119
 plot_simulated_particles(), **138**,
 139, 140
 plot_simulated_particles_with
 _positions(), **141**, 141
plotting.py, 5, 6, 9, 12, 14, 16, 29, 52, 53,
 55, 57, 59, 60, 64, 66, 69, 72, 73,
 76, 79
 plot_training(), **291**, 291, 292
 plot_training_metrics(), **219**, 220, 221
 point particle, 255
 PointParticle (deepttrack), 227,
 228, 255
 Poisson (deepttrack), 140, 170, 240
 Poisson noise, 140, 170, 229

policy (reinforcement learning), 558
 pooling layer, 105
 positional_encoding(), **436**, 437, 440, 442, 444, 455, 468, 469
 positional encoding function, 436–438
 positional encodings, **352–353**, 362
 positive pairs (contrastive learning), 252
 precision, 203, 276
 vs. accuracy, 136
 precision_score() (sklearn), 203
 predicting Lorenz system with reservoir computing, 610–619
 prediction error, 560
 predictions-versus-ground-truth plot, 65–67
 prepare_data() (Code Examples 10-1, 10-A and 10-B), **440**, 441, 448, 459
 prepare_data() (Code Example 10-C), **468**, 469
 preprocessing(), **348**, 348
 pretrained embeddings loading (NLP), 316–318
 PretrainedViTModel, **370**, 372
 Prewitt filter, 98
 prior distribution, 188
 process(), **307**, 308
 product() (deeptack, Source method), 213
 PropagateLayer, **493**, 494, 495
 Pygame, 569–572
 Python, xxiv
 documentation, xxiv
 official website, xxiv
Python Crash Course, 3rd edition (Matthes), xxiii
 Python package (creating and importing), 4–5
 PyTorch, xxiv, **100–110**, 287
 convolutional architectures, 108–109
 convolutional layer, 102–104
 dense layers, 109–110
 pooling layers, 105–106
 ReLU activation, 104–105
 upsampling layers, 106–108
 PyTorch tensor (image), 101

Q

Q-function, **558**
 Qiao, Chang, 463
 QAgent, **574**, 576, **576**, **578**, 579, **580**, 581, 583, 584
 Q-learning, 558–561, 598
 QLTetris, **573**, 574, 584, 596
 Q-network (deep Q-learning), 586, 599
 Q-table, 578–579
 quantum dot, 224
 detection, 224–234
 quantum dot image dataset, 224–225
 query-by-committee sampling, 530
 query_random(), **534**, 534, 538, 543, 545
 query strategy, 530
 query_uncertainty(), **536**, 537, 538, 544, 545

R

Rajaraman, Sivaramakrishnan, 111
 RandomCrop (torchvision), 360, 371, 398
 random_ellipse_axes(), **238**, 238
 RandomFlip, **520**, 521
 random graph, 611
 RandomHorizontalFlip (torchvision), 360, 371, 398, 412
 RandomRotation, **520**, 521
 random sampling, 534–536, 543, 548, 550
 random_split() (deeptack, Source method), 196, 213, 285, 308
 random_split() (torch), 114, 134
 RandomVerticalFlip (torchvision), 398, 412
 read_csv() (pandas), 89, 196, 219, 221, 283
 recall, 116, 203, 276
 recall_score() (sklearn), 203
 receiver operating characteristic (ROC), 117
 reconstruction error (autoencoder), 196
 reconstruction term (loss in VAE), 180
 recurrence relations, 279
 RecurrentModel (deeplay), 292, 293, 295, 297

- recurrent neural networks (RNNs),
 - 281–282**
 - GRU, 294–296
 - LSTM, 296–298
 - stacked, 293–294
- regionprops() (skimage), 232, 269, 515
- register_forward_hook() (torch), 121, 122, 125, 150, 153, 156
- register_full_backward_hook() (torch), 125
- regression, **51**
- Regressor (deeplay), 87, 135, 173, 200, 217, 231, 244, 292, 293, 295, 297, 490, 508
- regularization
 - for Adam, 363
 - factor
 - for reservoir computing, 613
 - for style transfer, 163
 - term (VAE), 180–181, 183
 - training, 73
 - weight, 18–19
- reinforcement learning, 558
- ReLU (rectified linear unit), **19**, 20, 104–105
- ReLU (torch), 44, 104, 108, 134, 147–148, 351, 353, 481, 487, 492, 494, 501, 522
- ReLU activation function, 43
- remove() (os), 111, 269, 317, 512
- remove() (torch), 122, 125, 150, 154, 156
- removing noise (NLP), 301–302
- reparameterization (Gaussian distribution), 428
- replace() (deeplay), 501
- replace (in dataset), 172
- replay buffer, 587, 591–594
- requires_grad_() (torch), 147, 149
- requires_grad (torch, Tensor attribute), 319, 355, 462
- rescale_intensity() (skimage), 126
- reservoir, 601
- reservoir computing
 - implementation, 610–619
 - introduction, 601–602
- reset gate (GRU), 294, 295
- residual connection, 165, 344, 351, 352, 414. *See also* skip connections
- ResidualMessagePassingNeuralNetwork (deeplay), 501
- resize() (cv2), 128
- resize() (skimage), 126, 369
- Resize (torchvision), 113, 371, 380
- resolve() (deeptack), 138, 141, 142, 171, 172, 174, 175, 177, 178, 230
- reverb, 280
- reverse diffusion
 - implementation, 435–436
 - process, **427**, 430–432
- reward (reinforcement learning), 558
- RGB images, 100
- RMSprop (deeplay, torch), 45, 115, 119
- RMSProp optimizer, **45**
- RNN (torch), 288
- RNNs. *See* recurrent neural networks
- ROC (receiver operating characteristic), 117
- ROC (torchmetrics), 117
- ROC curve, 116–118
- Rombach, Robin, 472
- Ronneberger, Olaf, 210, 248
- Rosenblatt, Frank, 48
- Rumelhart, David E., 49
- Runge-Kutta algorithms, 603

S

- Saharia, Chitwan, 472
- Salakhutdinov, Ruslan R., 207
- SampleToMasks (deeptack), 229, 242
- sampling (active learning)
 - adversarial, 549, 552–553
 - entropy, 545
 - large margin, 545
 - query-by-committee, 530
 - random, 534–536, 543, 548, 550
 - small margin, 545
 - uncertainty, 530, 536–538, 544, 548, 550–552
- sampling process, 428
- SAND (particle simulation) dataset, 497–500
- save() (numpy), 83, 133
- save() (torch), 595
- Scarselli, Franco, 527
- Schmidhuber, Jürgen, 323
- Seaborn, 40

- segmentation, **211**, 358
 - instance, 211
 - panoptic, 211
 - semantic, 207, 211
- segmented tissue images dataset, 213, 216
- select_labels(), **214**, 215
- self-attention, **326**, 332, 342, 343, 362, 373, 374, 439, 455, 456, 458.
 - See also* attention
- self-loop (graph), 481
- self-supervised learning, **252–255**
 - contrastive, **252–253**
 - geometric, **254–255**
 - non-contrastive, **253–254**
- semantic segmentation, 207, **211**
- sensitive dependence on initial conditions, 607. *See also* butterfly effect
- sensitivity, 116
- sentiment analysis, 299
 - with a transformer, 342–357
- Seq2Seq, **315**, 318
- seq2seq (sequence-to-sequence)
 - model, 299
- seq2seq (sequence-to-sequence)
 - transformation, 299
- Seq2SeqDecoder (Code Example 7-1), **310**, 312
- Seq2SeqDecoder (Code Example 8-A), **337**
- Seq2SeqEncoder, **309**, 312
- Seq2SeqModel (Code Example 7-1), **311–314**, 316
- Seq2SeqModel (Code Example 8-A), **338**
- sequence-to-sequence (seq2seq)
 - architecture, 311
 - combine encoder and decoder, 311–315
 - with attention, 338–339
 - define loss, 315
 - implement application, 315–316
 - implement decoder, 310–311
 - with attention, 337–338
 - implement encoder, 308–310
 - load pretrained embeddings, 316–318
 - test, 319–322
 - with attention, 339–340
 - train, 318–319
- sequence-to-sequence (seq2seq)
 - model, 299
- sequence-to-sequence (seq2seq)
 - transformation, 299
- Sequential (deeplay, torch), 104, 105, 107–110, 118, 134, 147–148, 257, 351, 353, 456, 487, 494, 495, 547
- SGD (deeplay, torch), 37, 38
- shot noise, 170, 229
- sigmoid(), **22**, 23, 28, 32, 33, 62, 63, 69, **281**, 282
- sigmoid() (numerically stable), **614**, 614, 616
- sigmoid() (torch), 245
- Sigmoid (torch), 36, 115, 118, 353, 401, 522, 552
- sigmoid function, **19**, 20, 105
 - simple implementation, **22**, 281
 - stable implementation, 614
- signal-to-noise ratio (SNR), 140, 170, 240
- Silver, David, 599
- Simonyan, Karen, 165
- simulate(), **509**, 510
- SimulatedDataset (Code Example 3-B), **142**, 142
- SimulatedDataset (Code Example 4-1), **172**, 172, 177–178
- simulation (with GNN), 497–511
- simulation pipeline
 - cell images, 237–242
 - microscopic particle image, 137–142, 169–171, 176
 - point particle, 255–257
 - quantum dots, 225–231
- single neuron, 2–19, 52–61
 - classification of 1D data with, 3–11
 - classification of 2D data with, 11–17
 - regression of 1D data with, 52–57
 - regression of 2D data with, 57–61
 - training algorithm, 7
 - training code, 8, 54
- skip connections, **209**, **210**
- sklearn, 203–205, 523, 532
- small margin sampling, 545
- Smith, Samuel L., 93
- SNR (signal-to-noise ratio), 140, 170, 240

- Sobel filter, 97, 98, 99
- Softmax (torch), 42, 110, 310, 337
- softmax() (torch), 232, 330, 334, 336, 344, 350
- softmax activation/function, **42**, 110, 211, 311, 329
- Sohl-Dickstein, Jascha, 471
- Source (deepttrack), 196, 213, 235, 269, 285, 308
- spaCy models, 300
- sparse categorical cross-entropy, 212
- special tokens, 305–307, 320, 454
- spectral radius, 611, 612
- Sphere (deepttrack), 169, 176
- splitting the data, 73–79
- squeeze() (numpy), 136, 138, 141, 184, 214, 230, 434
- squeeze() (torch), 47, 101, 151, 190, 192–194, 198, 200, 245, 246, 312–314, 331, 335, 339, 340, 354, 380, 434, 437, 484, 487, 488, 490, 524
- Stable Diffusion, 425, 450
- stacked recurrent neural networks, 293–294
- standard scaled dot-product attention, 333
- state (reinforcement learning), 558
 - in Q-table, 578–579
- steepest-descent training, 65
- step() (torch), 161, 289, 384, 403, 404, 417, 441, 462, 593
- stimulus signals, 2
- stochastic differential equations, 426
- stochastic gradient descent
 - optimizer, 37
- style_transfer(), **160**, **161**, 163
- Subset (torch), 360
- Subtract (deepttrack), 228
- sum() (torch), 125
- super-resolution, 463–471, 472
- super-resolution microscopy, 463
- supervised learning, 1
- Sutskever, Ilya, 323
- Sutton, Richard S., 598, 599
- synapse, 2
- synaptic potential, 2, 3
- synaptic signals, 2
- Szegedy, Christian, 165

T

- Tanh (torch), 400–401
- target network
 - deep Q-learning, 586
 - non-contrastive learning, 253
- teacher forcing, 313
- teacher-student learning paradigm, 586
- temperature prediction with an RNN, 282–298
- temporal difference error, 586
- temporal difference learning, 598
- tensor processing units (TPUs), xxv
- tensor_to_image(), **151**, 152, 153, 161
- test() (lightning), 39, 42, 44, 45, 116, 119, 135, 143, 144, 222, 356, 364, 369, 372, 491, 496, 509, 548, 549
- test/testing dataset, 39, **74**
- Tetriminos, 561
- Tetris, 561
- Tetris, **562–568**, 568, 573, 576–577, 589
- text embeddings, 450
- text encoder, 450
- TextEncoder, **455**, 457
- text-to-image transformation, 450–463
- text translation
 - with attention, 336–342
 - with an RNN, 299–322
- textual prompts, 450
- threshold (classifier), 116
- thresholding (image), 233
- tic-tac-toe*, 560
- tiffiffle() (tiffiffle), 464
- time evolution, 603
- to() (torch), 91, 259, 264, 288–290, 311–314, 320, 322, 339, 381, 382, 384, 386, 391, 392, 400–402, 405, 406, 414, 415, 420, 433–436, 439, 440, 444, 448, 456–462, 467, 468, 482, 490, 509
- to_dense_adj() (torch_geometric), 484, 490
- to_jshtml() (matplotlib), 499, 510, 525
- tokenization (NLP), 300
- tokenize(), **300**, 300, **301**, 301, 302, 307, 320, 348, 356, 453, 454
- tokenizer, 311
- Tolstikhin, Ilya, 207
- to_numpy_array() (networkx), 612

ToTensor (deepttrack), 170, 175, 176, 181, 190, 198, 215, 230, 235, 242, 256, 272, 286, 308
 ToTensor (torchvision), 113, 149, 360, 361, 371, 380, 398, 412, 432, 465, 547
 TPR (true-positive rate), 116
 TPUs (tensor processing units), xxv
 tqdm() (tqdm), 276, 395–396, 435, 447, 452, 467
 train() (torch), 386, 405, 419, 442, 459, 462, 469. *See also* eval()
 TrainableAttention, **334**
 train_active_learning(), **533**, 534, 537, 538, **541**, 543–545
 train_disc(), **403**, 406
 trainer, **38**
 Trainer (deeplay, lightning), 38, 42, 44, 45, 89, 116, 119, 135, 143, 173, 178, 183, 191, 200, 218, 221, 232, 244, 260, 264, 267, 273, 292, 293, 319, 356, 363, 368, 372, 490, 496, 508, 522, 548, 549
 train_gen(), **404**, 406
 training dataset, **73**
 training a neuron
 algorithm, 7
 implementation, 8, 54
 training an RNN
 implementation, 289
 monitoring training loss, 290
 monitoring validation loss, 290–291
 training_step() (deeplay, Application method), 259, 367
 train_model() (Code Example 9-C), **416**, 420
 train_model() (Code Examples 12-1 and 12-2), **532**, 533, 542
 train_preprocess() (deeplay, Application method), 316
 transformer, 325, **344–345**
 decoder-only, 345
 encoder-only, 345
 transformer encoder, 345, 351–355, 361, 362
 TransformerEncoderLayer, **351**, 353
 TransformerEncoderModel, **353**, 355
 TransformLayer, **492**, 494, 495
 translate() (Code Example 7-A), **320**, 320, 321
 translate() (Code Example 8-A), **340**, 340, 341
 translate() (kornia), 258
 translation, 257
 equivariance, 257
 translation (NLP), 299
 Trencadís Lizard, 158
 trilinear upsampling, 108
 true-positive rate (TPR), 116
 .txt files, 81, 83, 84, 299, 454, 518, 581, 583

U

uncertainty (reinforcement learning), 560
 uncertainty sampling, 530, 536–538, 544, 548, 550–552, 556
 UncertaintyStrategy (deeplay), 551
 underfitting, **77**, 290, 386
 undirected graph, 477
 U-Net, **210–211**, 248
 UNet2d (deeplay), 217, 231, 244, 400
 Unicode (encoding standard), 302
 UniformStrategy (deeplay), 550
 universal approximators (neural network as), 49
 unlabeled data, 252
 unlabeled pool, 530
 unpooling layer, 106
 unprocess(), **319**, 320, 322, 339
 Unsqueeze (deepttrack), 198
 unsqueeze() (torch), 102, 104, 105, 107, 108, 110, 113, 121, 122, 125, 149, 174, 175, 177, 178, 184, 192, 194, 200, 223, 232, 273, 275, 276, 313, 320, 335, 350, 435, 440, 455, 468, 493
 unsupervised clustering with VAE, 186–187
 unsupervised learning, 167
 update() (deepttrack), 138, 141, 142, 171, 172, 174, 175, 177, 178, 227, 230, 238
 update gate (GRU), 294, 295
 UpdateLayer, **494**, 494, 495
 Upsample (torch), 107
 upsampling layer, 106

V

VAE (variational autoencoder),
179–188, 188, 207

validation dataset, 39, **73**

Value (deeptack), 186, 190, 198, 272,
286, 308

value function (reinforcement
learning), 558

vanishing gradient problem, 19, 43, 49,
165, 211, 293, 323, 380

variance schedule, 428

VariationalAutoEncoder (deeplay), 182

variational autoencoder (VAE),
179–188, 188, 207

Vaswani, Ashish, 373

Velickovic, Petar, 528

vertex (graph), 476

VGG16, 147–149, 159, 402

vgg16() (torchvision), 147

VGG16_Weights (torchvision), 147

VideoCapture (cv2), 128

view() (torch), 160, 204, 205, 263, 315,
344, 366, 590

VirtualStainingDataset, **395–397**,
398, 399

virtual staining with GAN, 394–409

vision transformer (ViT), **361–363**, 374

Visual Studio (VS) Code, xxiv

ViT (deeplay), 362, 368

ViT architecture, **361**

ViTImageProcessor (transformers), 371

ViTModel (transformers), 370–371

Vocab, **303**, 304, 305, 348, 454

vocabulary, 303, 348

vocabulary building (NLP), 303–306,
453–455

Volpe, Giovanni, 472

VS Code (Visual Studio Code), xxiv

W

WassersteinAutoEncoder (deeplay), 191

Wasserstein autoencoder (WAE),
188–195, 207

Wasserstein distance, 188, 189

Watkins, Christopher J.C.H., 598

weight_decay (parameter Adam),
363, 368

weighted cross-entropy loss, **224**

weighted loss function, 231

weight normalization/regularization,
18–19

weights (neuron), 2, 3

Welling, Max, 207, 527

Werbos, Paul John, 49

where() (numpy), 533, 541, 614

where() (torch), 263

Widrow, Bernard, 48

with (Python), **4**. *See also* context
manager

- with Fwd_Hook, 154
- with Fwd_Hooks, 157, 160, 162
- with numpy.load(), 498
- with open(), 3, 12, 81, 84, 85, 302,
307, 317, 451, 453, 498
- with pandas.option_context(),
346, 357
- with torch.no_grad(), 290, 314,
338, 339, 405, 435, 444, 509,
590, 592, 593

Z

Zalando, 189

Zemel, Richard S., 207

zero_() (torch), 150

zero_grad() (torch), 161, 289, 384, 403,
404, 416–417, 441, 462, 593

Zhu, Jun-Yan, 424

ZINC (molecular properties) dataset,
483–486

Zisserman, Andrew, 165