Neural Network

A Selective Overview of Deep Learning

Tao Huang

Section 1

Feed-forward neural networks

Model setup

Deep neural networks (DNNs) use composition of a series of simple nonlinear functions to model nonlinearity

$$\boldsymbol{h}^{(L)} = \mathbf{g}^{(L)} \circ \mathbf{g}^{(L-1)} \circ \ldots \circ \mathbf{g}^{(1)}(\boldsymbol{x}), \tag{1}$$

Model setup

Deep neural networks (DNNs) use composition of a series of simple nonlinear functions to model nonlinearity

$$\boldsymbol{h}^{(L)} = \mathbf{g}^{(L)} \circ \mathbf{g}^{(L-1)} \circ \ldots \circ \mathbf{g}^{(1)}(\boldsymbol{x}), \tag{1}$$

for $\ell = 1, \ldots, L$, define

$$\boldsymbol{h}^{(\ell)} = \mathbf{g}^{(l)}\big(\boldsymbol{h}^{(l-1)}\big) \triangleq \boldsymbol{\sigma}\big(\mathbf{W}^{(\ell)}\boldsymbol{h}^{(\ell-1)} + \boldsymbol{b}^{(\ell)}\big), \tag{2}$$

Model setup

Deep neural networks (DNNs) use composition of a series of simple nonlinear functions to model nonlinearity

$$\boldsymbol{h}^{(L)} = \mathbf{g}^{(L)} \circ \mathbf{g}^{(L-1)} \circ \ldots \circ \mathbf{g}^{(1)}(\boldsymbol{x}), \tag{1}$$

for $\ell = 1, \ldots, L$, define

$$\boldsymbol{h}^{(\ell)} = \mathbf{g}^{(l)}\left(\boldsymbol{h}^{(l-1)}\right) \triangleq \boldsymbol{\sigma}\left(\mathbf{W}^{(\ell)} \boldsymbol{h}^{(\ell-1)} + \boldsymbol{b}^{(\ell)}\right), \tag{2}$$

The activation function $\boldsymbol{\sigma}(\cdot)$ is usually applied element-wise, and a popular choice is the ReLU (Rectified Linear Unit) function:

$$[\boldsymbol{\sigma}(\boldsymbol{z})]_j = \max\{z_j, 0\}. \tag{3}$$

Model setup

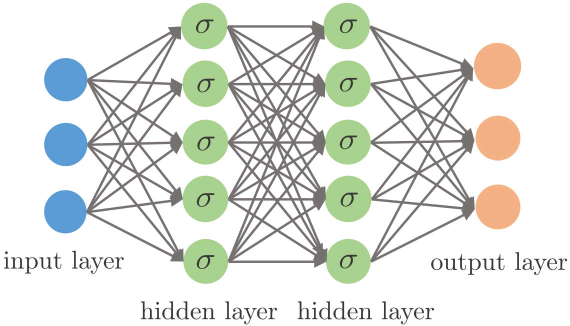


Figure 1: A feed-forward neural network with an input layer, two hidden layers and an output layer. The input layer represents raw features $\{\boldsymbol{x}_i\}_{1\leq i\leq n}$. Both hidden layers compute an affine transform (a.k.s. indices) of the input and then apply an element-wise activation function $\boldsymbol{\sigma}(\cdot)$. Finally, the output returns a linear transform followed by the softmax activation (resp. simply a linear transform) of the hidden layers for the classification (resp. regression) problem.

Model setup

The *soft-max* function:

$$
\begin{aligned}
f_k(\boldsymbol{x};\boldsymbol{\theta}) &\triangleq \frac{\exp(z_k)}{\sum_k \exp(z_k)}, \forall\, k \in [K], \\
\text{where } \boldsymbol{z} &= \mathbf{W}^{(L+1)}\boldsymbol{h}^{(L)} + \boldsymbol{b}^{(L+1)} \in \mathbb{R}^K.
\end{aligned} \tag{4}
$$

Model setup

中山大學 SUN YAT-SEN UNIVERSITY 數學學院 篤行工作室

The *soft-max* function:

$$f_k(\boldsymbol{x};\boldsymbol{\theta}) \triangleq \frac{\exp(z_k)}{\sum_k \exp(z_k)}, \forall\, k \in [K], \quad \text{where } \boldsymbol{z} = \mathbf{W}^{(L+1)}\boldsymbol{h}^{(L)} + \boldsymbol{b}^{(L+1)} \in \mathbb{R}^K. \tag{4}$$

Loss

$$\mathcal{L}(\boldsymbol{f}(\boldsymbol{x};\boldsymbol{\theta}), y) = -\sum_{k=1}^{K} \mathbb{1}\{y = k\} \log p_k, \tag{5}$$

,where $\boldsymbol{\theta} \triangleq \{\mathbf{W}^{(\ell)}, \boldsymbol{b}^{(\ell)} : 1 \leq \ell \leq L+1\}$.

Back-propagation in computational graphs

The key to the above training procedure, namely SGD, is the calculation of the gradient $\nabla \ell_{\mathcal{B}}(\boldsymbol{\theta})$, where

$$\ell_{\mathcal{B}}(\boldsymbol{\theta}) \triangleq |\mathcal{B}|^{-1} \sum_{i \in \mathcal{B}} \mathcal{L}(\boldsymbol{f}(\boldsymbol{x}_i; \boldsymbol{\theta}), y_i). \tag{6}$$

Back-propagation in computational graphs

The key to the above training procedure, namely SGD, is the calculation of the gradient $\nabla \ell_{\mathcal{B}}(\boldsymbol{\theta})$, where

$$\ell_{\mathcal{B}}(\boldsymbol{\theta}) \triangleq |\mathcal{B}|^{-1} \sum_{i \in \mathcal{B}} \mathcal{L}(\boldsymbol{f}(\boldsymbol{x}_i; \boldsymbol{\theta}), y_i). \tag{6}$$

$$\begin{aligned} \frac{\partial \ell_{\mathcal{B}}}{\partial \boldsymbol{h}^{(\ell-1)}} &= \frac{\partial \boldsymbol{h}^{(\ell)}}{\partial \boldsymbol{h}^{(\ell-1)}} \cdot \frac{\partial \ell_{\mathcal{B}}}{\partial \boldsymbol{h}^{(\ell)}} \\ &= (\mathbf{W}^{(\ell)})^\top \mathrm{diag}\left(\mathbb{1}\{\mathbf{W}^{(\ell)} \boldsymbol{h}^{(\ell-1)} + \boldsymbol{b}^{(\ell)} \geq \mathbf{0}\}\right) \frac{\partial \ell_{\mathcal{B}}}{\partial \boldsymbol{h}^{(\ell)}} \end{aligned} \tag{7}$$

Back-propagation in computational graphs

$$\mathbf{W}^{(\ell)} \leftarrow \mathbf{W}^{(\ell)} - \eta \frac{\partial \ell_{\mathcal{B}}}{\partial \mathbf{W}^{(\ell)}}, \text{ where } \frac{\partial \ell_{\mathcal{B}}}{\partial W_{jm}^{(\ell)}} = \frac{\partial \ell_{\mathcal{B}}}{\partial h_j^{(\ell)}} \cdot \sigma' \cdot h_m^{(\ell-1)}, \quad (8)$$

Back-propagation in computational graphs

$$\mathbf{W}^{(\ell)} \leftarrow \mathbf{W}^{(\ell)} - \eta \frac{\partial \ell_{\mathcal{B}}}{\partial \mathbf{W}^{(\ell)}}, \text{ where } \frac{\partial \ell_{\mathcal{B}}}{\partial W_{jm}^{(\ell)}} = \frac{\partial \ell_{\mathcal{B}}}{\partial h_j^{(\ell)}} \cdot \sigma' \cdot h_m^{(\ell-1)}, \quad (8)$$

MLP with a single hidden layer and an ℓ_2 regularization:

$$\ell_{\mathcal{B}}^{\lambda}(\boldsymbol{\theta}) = \ell_{\mathcal{B}}(\boldsymbol{\theta}) + r_{\lambda}(\boldsymbol{\theta}) = \ell_{\mathcal{B}}(\boldsymbol{\theta}) + \lambda \Big(\sum_{j,j'} \big(W_{j,j'}^{(1)}\big)^2 + \sum_{j,j'} \big(W_{j,j'}^{(2)}\big)^2 \Big), \quad (9)$$

Back-propagation in computational graphs

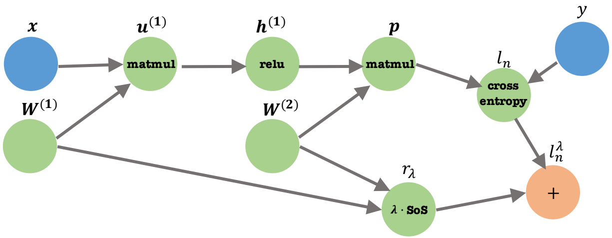


Figure 2: The computational graph illustrates the loss (9). For simplicity, we omit the bias terms. Symbols inside nodes represent functions, and symbols outside nodes represent function outputs (vectors/scalars). `matmul` is matrix multiplication, `relu` is the ReLU activation, `cross entropy` is the cross entropy loss, and `SoS` is the sum of squares.

Numerical experiments

Section 2

Convolutional neural networks

Convolutional layer (CONV)

$$O_{ij}^k = \left\langle [\boldsymbol{X}]_{ij}, \boldsymbol{F}_k \right\rangle = \sum_{i'=1}^{w} \sum_{j'=1}^{w} \sum_{l=1}^{d_3} [\boldsymbol{X}]_{i+i'-1, j+j'-1, l} [\boldsymbol{F}_k]_{i', j', l}. \quad (10)$$

Convolutional layer (CONV)

$$O_{ij}^{k} = \left\langle\, [\boldsymbol{X}]_{ij}\,, \boldsymbol{F}_k \right\rangle = \sum_{i'=1}^{w}\sum_{j'=1}^{w}\sum_{l=1}^{d_3} [\boldsymbol{X}]_{i+i'-1,j+j'-1,l}[\boldsymbol{F}_k]_{i',j',l}. \quad (10)$$

$$\tilde{X}_{ijk} = \sigma(O_{ijk}),\ \forall\, i \in [d_1 - w + 1], j \in [d_2 - w + 1], k \in [\tilde{d}_3]. \quad (11)$$

Convolutional layer

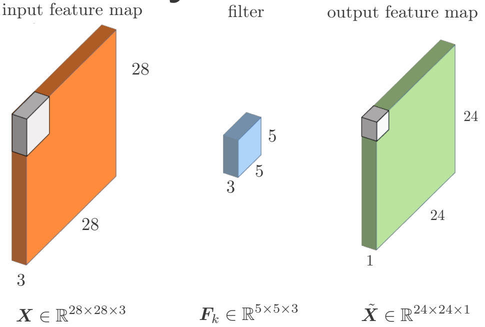


Figure 3: $\boldsymbol{X} \in \mathbb{R}^{28\times28\times3}$ represents the input feature consisting of 28×28 spatial coordinates in a total number of 3 channels / feature maps. $\boldsymbol{F}_k \in \mathbb{R}^{5\times5\times3}$ denotes the k-th filter with size 5×5. The third dimension 3 of the filter automatically matches the number 3 of channels in the previous input. Every 3D patch of $\boldsymbol{X}$ gets convolved with the filter $\boldsymbol{F}_k$ and this as a whole results in a single output feature map $\tilde{X}_{:,:,k}$ with size $24 \times 24 \times 1$. Stacking the outputs of all the filters $\{\boldsymbol{F}_k\}_{1\leq k\leq K}$ will lead to the output feature with size $24 \times 24 \times K$.

Pooling layer (POOL)

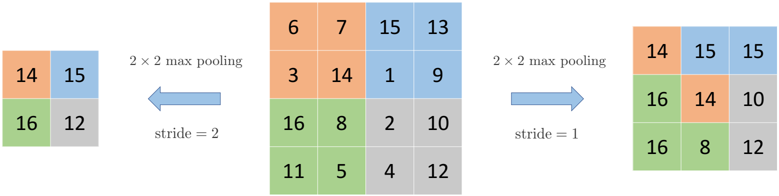


Figure 4: A 2×2 max pooling layer extracts the maximum of 2 by 2 neighboring pixels / features across the spatial dimension.

LeNet

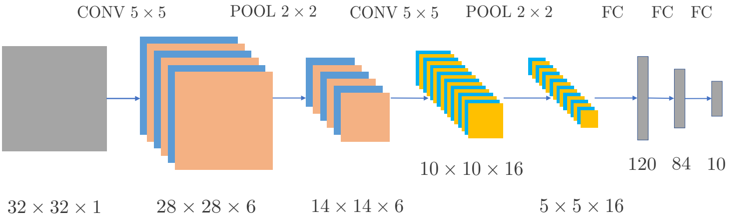


Figure 5: LeNet is composed of an input layer, two convolutional layers, two pooling layers and three fully-connected layers. Both convolutions are valid and use filters with size 5×5. In addition, the two pooling layers use 2×2 average pooling.

Numerical experiments

中山大學 SUN YAT-SEN UNIVERSITY 数學學院 篤行工作室

Section 3

Recurrent neural networks

Vanilla RNNs

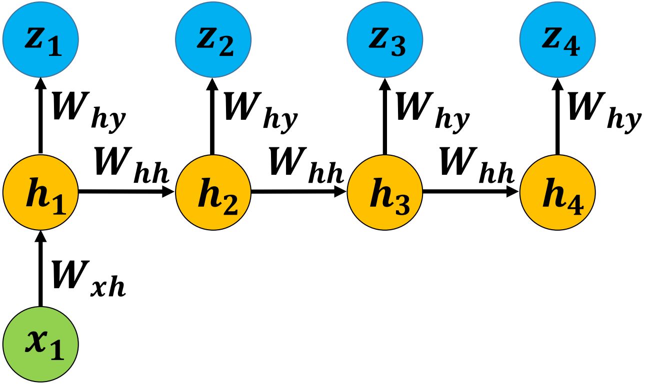


Figure 6: Vanilla RNNs with different inputs/outputs settings. (a) has one input but multiple outputs;

Vanilla RNNs

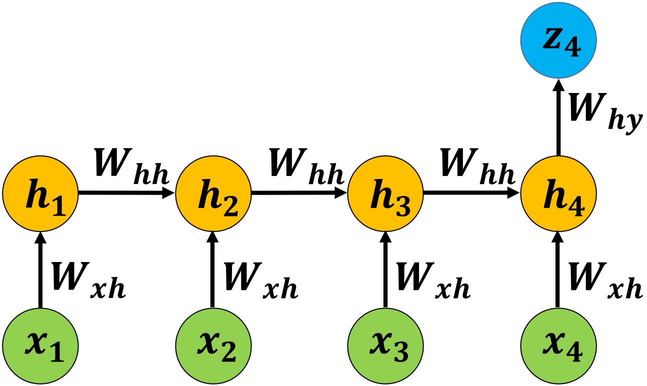


Figure 7: Vanilla RNNs with different inputs/outputs settings.(b) has multiple inputs but one output;

Vanilla RNNs

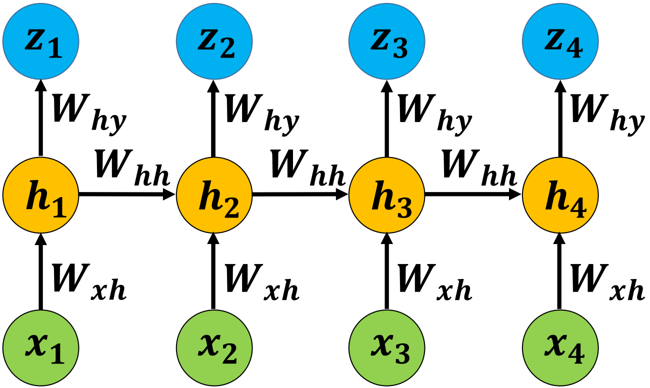


Figure 8: Vanilla RNNs with different inputs/outputs settings. (c) has multiple inputs and outputs. Note that the parameters are shared across time steps.

Vanilla RNNs

$$\boldsymbol{h}_t = \boldsymbol{f}_\theta(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t). \tag{12}$$

Vanilla RNNs

中山大學 SUN YAT-SEN UNIVERSITY 数学学院 笃行工作室

$$\boldsymbol{h}_t = \boldsymbol{f}_\theta(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t). \tag{12}$$

$$\boldsymbol{h}_t = \mathbf{tanh}\left(\mathbf{W}_{hh}\boldsymbol{h}_{t-1} + \mathbf{W}_{xh}\boldsymbol{x}_t + \boldsymbol{b}_{\boldsymbol{h}}\right), \tag{13}$$

$$\tanh(a) = \frac{e^{2a} - 1}{e^{2a} + 1}, \tag{14}$$

$$\boldsymbol{z}_t = \boldsymbol{\sigma}\left(\mathbf{W}_{hy}\boldsymbol{h}_t + \boldsymbol{b}_{\boldsymbol{z}}\right), \tag{15}$$

Vanilla RNNs

中山大學 SUN YAT-SEN UNIVERSITY 数學學院 篤行工作室

$$\boldsymbol{h}_t = \boldsymbol{f}_{\theta}(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t). \tag{12}$$

$$\boldsymbol{h}_t = \mathbf{tanh}\left(\mathbf{W}_{hh}\boldsymbol{h}_{t-1} + \mathbf{W}_{xh}\boldsymbol{x}_t + \boldsymbol{b}_{\boldsymbol{h}}\right), \tag{13}$$

$$\tanh(a) = \frac{e^{2a}-1}{e^{2a}+1}, \tag{14}$$

$$\boldsymbol{z}_t = \boldsymbol{\sigma}\left(\mathbf{W}_{hy}\boldsymbol{h}_t + \boldsymbol{b}_{\boldsymbol{z}}\right), \tag{15}$$

$$\begin{aligned}\ell_{\mathcal{T}}(\boldsymbol{\theta}) &= \sum_{t\in\mathcal{T}} \mathcal{L}(y_t, \boldsymbol{z}_t) \\ &= -\sum_{t\in\mathcal{T}}\sum_{k=1}^{K} \mathbb{1}\{y_t = k\} \log\left(\frac{\exp([\boldsymbol{z}_t]_k)}{\sum_k \exp([\boldsymbol{z}_t]_k)}\right),\end{aligned} \tag{16}$$

LSTM

$$\begin{pmatrix} \boldsymbol{i}_t \\ \boldsymbol{f}_t \\ \boldsymbol{o}_t \\ \boldsymbol{g}_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \mathbf{W} \begin{pmatrix} \boldsymbol{h}_{t-1} \\ \boldsymbol{x}_t \\ 1 \end{pmatrix}, \tag{17}$$

$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \boldsymbol{g}_t, \tag{18}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t), \tag{19}$$

Multilayer RNNs

$$\boldsymbol{h}_t^\ell = \mathbf{tanh}\left[\mathsf{W}^\ell \begin{pmatrix} \boldsymbol{h}_t^{\ell-1} \\ \boldsymbol{h}_{t-1}^\ell \\ 1 \end{pmatrix}\right], \quad \text{for all } \ell \in [L], \qquad \boldsymbol{h}_t^0 \triangleq \boldsymbol{x}_t. \quad (20)$$

Multilayer RNNs

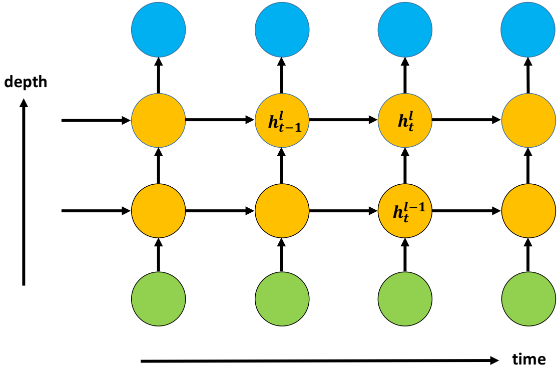
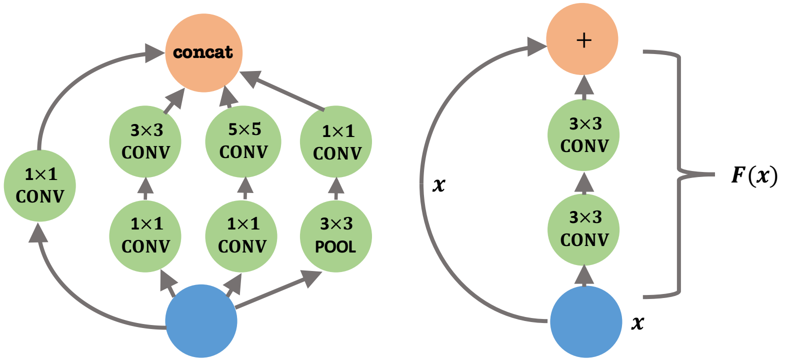


Figure 9: A vanilla RNN with two hidden layers. Higher-level hidden states $\boldsymbol{h}_t^\ell$ are determined by the old states $\boldsymbol{h}_{t-1}^\ell$ and lower-level hidden states $\boldsymbol{h}_t^{\ell-1}$. Multilayer RNNs generalize both feed-forward neural nets and one-hidden-layer RNNs.

Section 4

Modules

Skip connections



(a) "Inception" module

(b) Skip connections

Figure 10: (a) The "Inception" module from GoogleNet. `Concat` means combining all features maps into a tensor. (b) Skip connections are added every two layers in ResNets.

Skip connections

$$\boldsymbol{x} \longmapsto \sigma(\boldsymbol{x} + \mathsf{F}(\boldsymbol{x})) = \sigma(\boldsymbol{x} + \mathsf{W}'\sigma(\mathsf{W}\boldsymbol{x} + \boldsymbol{b}) + \boldsymbol{b}'), \tag{21}$$

Section 5

Autoencoders

Autoencoders

中山大學 SUN YAT-SEN UNIVERSITY 数学学院 笃行工作室

$$\boldsymbol{f}\left(\boldsymbol{x}\right) = \boldsymbol{W}_f \boldsymbol{x} \triangleq \boldsymbol{h} \tag{22}$$

$$\boldsymbol{g}\left(\boldsymbol{h}\right) = \boldsymbol{W}_g \boldsymbol{h}, \tag{23}$$

$$\boldsymbol{W}_f \in \mathbb{R}^{k \times d} \text{and} \boldsymbol{W}_g \in \mathbb{R}^{d \times k}. \tag{24}$$

Autoencoders

中山大學 SUN YAT-SEN UNIVERSITY 数学学院 笃行工作室

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{W}_f \boldsymbol{x} \triangleq \boldsymbol{h} \tag{22}$$

$$\boldsymbol{g}(\boldsymbol{h}) = \boldsymbol{W}_g \boldsymbol{h}, \tag{23}$$

$$\boldsymbol{W}_f \in \mathbb{R}^{k \times d} \text{and} \boldsymbol{W}_g \in \mathbb{R}^{d \times k}. \tag{24}$$

$$\min_{\boldsymbol{f},\boldsymbol{g}} \quad \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(\boldsymbol{x}_i, \boldsymbol{g}(\boldsymbol{h}_i)) \text{ with } \boldsymbol{h}_i = \boldsymbol{f}(\boldsymbol{x}_i), \forall i \in [n]. \tag{25}$$

Autoencoders

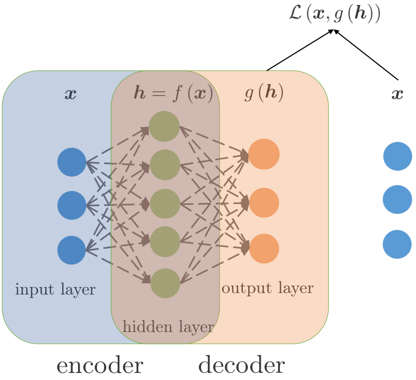


Figure 11: First an input $\boldsymbol{x}$ goes through the decoder $\boldsymbol{f}(\cdot)$, and we obtain its hidden representation $\boldsymbol{h} = \boldsymbol{f}(\boldsymbol{x})$. Then, we use the decoder $\boldsymbol{g}(\cdot)$ to get $\boldsymbol{g}(\boldsymbol{h})$ as a reconstruction of $\boldsymbol{x}$. Finally, the loss is determined from the difference between the original input $\boldsymbol{x}$ and its reconstruction $\boldsymbol{g}(\boldsymbol{f}(\boldsymbol{x}))$.

Section 6

Generative adversarial networks

GAN

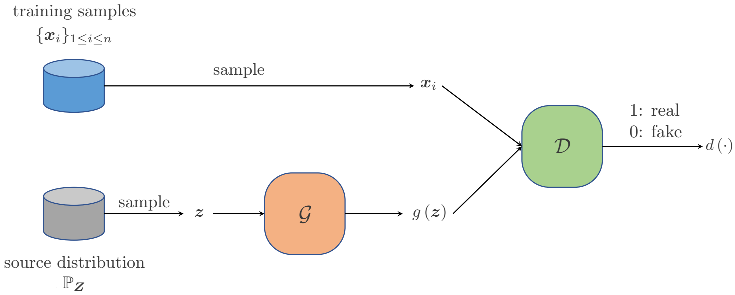


Figure 12: GANs consist of two components, a generator $\mathcal{G}$ which generates fake samples and a discriminator $\mathcal{D}$ which differentiate the true ones from the fake ones.

GAN

$$\min_{\boldsymbol{\theta}_{\mathcal{G}}} \max_{\boldsymbol{\theta}_{\mathcal{D}}} \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_{\boldsymbol{X}}} \left[\log\left(d\left(\boldsymbol{x}\right)\right)\right] + \mathbb{E}_{\boldsymbol{z} \sim \mathbb{P}_{\boldsymbol{Z}}} \left[\log\left(1 - d\left(\mathbf{g}\left(\boldsymbol{z}\right)\right)\right)\right]. \quad (26)$$

GAN

$$\min_{\boldsymbol{\theta}_{\mathcal{G}}} \max_{\boldsymbol{\theta}_{\mathcal{D}}} \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_{\boldsymbol{X}}} \left[\log \left(d \left(\boldsymbol{x} \right) \right) \right] + \mathbb{E}_{\boldsymbol{z} \sim \mathbb{P}_{\boldsymbol{Z}}} \left[\log \left(1 - d \left(\mathbf{g} \left(\boldsymbol{z} \right) \right) \right) \right]. \tag{26}$$

$$\min_{\mathbb{P}_{\mathcal{G}}} \max_{d(\cdot)} \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_{\boldsymbol{X}}} \left[\log \left(d \left(\boldsymbol{x} \right) \right) \right] + \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_{\mathcal{G}}} \left[\log \left(1 - d \left(\boldsymbol{x} \right) \right) \right]. \tag{27}$$

GAN

$$\min_{\boldsymbol{\theta}_{\mathcal{G}}} \max_{\boldsymbol{\theta}_{\mathcal{D}}} \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_{\boldsymbol{X}}} \left[\log \left(d \left(\boldsymbol{x} \right) \right) \right] + \mathbb{E}_{\boldsymbol{z} \sim \mathbb{P}_{\boldsymbol{Z}}} \left[\log \left(1 - d \left(\mathbf{g} \left(\boldsymbol{z} \right) \right) \right) \right]. \tag{26}$$

$$\min_{\mathbb{P}_{\mathcal{G}}} \max_{d(\cdot)} \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_{\boldsymbol{X}}} \left[\log \left(d \left(\boldsymbol{x} \right) \right) \right] + \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_{\mathcal{G}}} \left[\log \left(1 - d \left(\boldsymbol{x} \right) \right) \right]. \tag{27}$$

$$d^{*} \left(\boldsymbol{x} \right) = \frac{\mathbb{P}_{\boldsymbol{X}} \left(\boldsymbol{x} \right)}{\mathbb{P}_{\boldsymbol{X}} \left(\boldsymbol{x} \right) + \mathbb{P}_{\mathcal{G}} \left(\boldsymbol{x} \right)}. \tag{28}$$

GAN

$$\min_{\boldsymbol{\theta}_{\mathcal{G}}} \max_{\boldsymbol{\theta}_{\mathcal{D}}} \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_{\boldsymbol{X}}} \left[\log\left(d\left(\boldsymbol{x}\right)\right)\right] + \mathbb{E}_{\boldsymbol{z} \sim \mathbb{P}_{\boldsymbol{Z}}} \left[\log\left(1 - d\left(\mathbf{g}\left(\boldsymbol{z}\right)\right)\right)\right]. \tag{26}$$

$$\min_{\mathbb{P}_{\mathcal{G}}} \max_{d(\cdot)} \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_{\boldsymbol{X}}} \left[\log\left(d\left(\boldsymbol{x}\right)\right)\right] + \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_{\mathcal{G}}} \left[\log\left(1 - d\left(\boldsymbol{x}\right)\right)\right]. \tag{27}$$

$$d^*\left(\boldsymbol{x}\right) = \frac{\mathbb{P}_{\boldsymbol{X}}\left(\boldsymbol{x}\right)}{\mathbb{P}_{\boldsymbol{X}}\left(\boldsymbol{x}\right) + \mathbb{P}_{\mathcal{G}}\left(\boldsymbol{x}\right)}. \tag{28}$$

$$\min_{\mathbb{P}_{\mathcal{G}}} \quad \mathrm{JS}\left(\mathbb{P}_{\boldsymbol{X}} \parallel \mathbb{P}_{\mathcal{G}}\right), \tag{29}$$

where $\mathrm{JS}(\cdot\|\cdot)$ denotes the Jensen–Shannon divergence between two distributions

$$\mathrm{JS}\left(\mathbb{P}_{\boldsymbol{X}}\|\mathbb{P}_{\mathcal{G}}\right) = \frac{1}{2}\mathrm{KL}\Big(\mathbb{P}_{\boldsymbol{X}} \parallel \tfrac{\mathbb{P}_{\boldsymbol{X}}+\mathbb{P}_{\mathcal{G}}}{2}\Big) + \frac{1}{2}\mathrm{KL}\Big(\mathbb{P}_{\mathcal{G}} \parallel \tfrac{\mathbb{P}_{\boldsymbol{X}}+\mathbb{P}_{\mathcal{G}}}{2}\Big). \tag{30}$$

it minimizes the Wasserstein distance between $\mathbb{P}_{\boldsymbol{X}}$ and $\mathbb{P}_{\mathcal{G}}$:

$$\min_{\boldsymbol{\theta}_{\mathcal{G}}} \quad \mathrm{WS}\left(\mathbb{P}_{\boldsymbol{X}}\|\mathbb{P}_{\mathcal{G}}\right) = \min_{\boldsymbol{\theta}_{\mathcal{G}}} \sup_{f:f \text{ 1-Lipschitz}} \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_{\boldsymbol{X}}}\left[f\left(\boldsymbol{x}\right)\right] - \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_{\mathcal{G}}}\left[f\left(\boldsymbol{x}\right)\right],$$

Questions