Anomaly Detection

Dimensionality reduction: Autoencoders

Paul Irofti
Andrei Pătrașcu
Cristian Rusu

Computer Science Department
Faculty of Mathematics and Computer Science
University of Bucharest
Email: `first.last@fmi.unibuc.ro`

Outline

- Autoencoder architecture
- Convolutional autoencoders (CAE)
- Sparse autoencoder (SAE)
- Variational autoencoders (VAE)
- Connections with other methods

The course main references are Kramer 1991 and Dumoulin and Visin 2016.

Preliminaries

Dimensionality Reduction

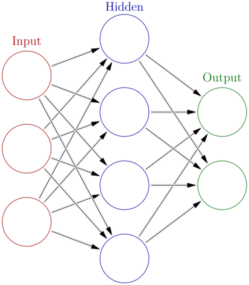
Convolution

- convolution of $f(x)$ and $g(x)$ is
  $(f * g)(x) = \int_{-\infty}^{\infty} f(t)g(x-t)dt$

Convolution

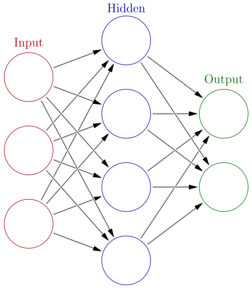
- convolution of $f(x)$ and $g(x)$ is $(f * g)(x) = \int_{-\infty}^{\infty} f(t)g(x-t)dt$
- in discrete case: $(f * g)[n] = \sum_{k=-\infty}^{\infty} f[k]g[n-k]$

Neural Networks



Source: https://en.wikipedia.org/wiki/Neural_network_(machine_learning)

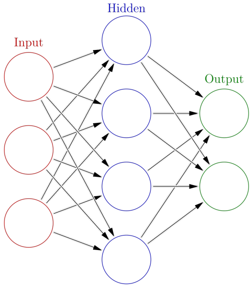
Neural Networks



Source: https://en.wikipedia.org/wiki/Neural_network_(machine_learning)

- each layer has its own set of weights and biases

Neural Networks



Source: https://en.wikipedia.org/wiki/Neural_network_(machine_learning)

- each layer has its own set of weights and biases
- activation function (non-linearity) $f(w^T x + b)$

Backpropagation

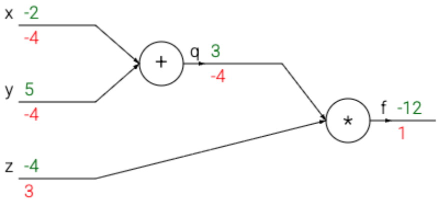
Backpropagation

- uses chain rule to compute gradients and update weights and biases
- derivative definition: $\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$

Backpropagation

- uses chain rule to compute gradients and update weights and biases
- derivative definition: $\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h)-f(x)}{h}$
- chain rule: for $y = f(u)$ and $u = g(x)$ we have $\frac{dy}{dx} = \frac{dy}{du} \frac{du}{dx}$

Backpropagation



The real-valued *"circuit"* on left shows the visual representation of the computation. The forward pass computes values from inputs to output (shown in green). The backward pass then performs backpropagation which starts at the end and recursively applies the chain rule to compute the gradients (shown in red) all the way to the inputs of the circuit. The gradients can be thought of as flowing backwards through the circuit.

Source: `https://cs231n.github.io/optimization-2/`

Autoencoders

Autoencoders

Given dataset $X \in \mathbb{R}^{m \times N}$, an autoencoder is composed of three essential components

Autoencoders

Given dataset $X \in \mathbb{R}^{m \times N}$, an autoencoder is composed of three essential components

- encoder $E : \mathbb{R}^m \to \mathbb{R}^s$ such that $z = E(x) \in \mathbb{R}^s$ where $s \ll m$

Autoencoders

Given dataset $X \in \mathbb{R}^{m \times N}$, an autoencoder is composed of three essential components

- encoder $E : \mathbb{R}^m \to \mathbb{R}^s$ such that $z = E(x) \in \mathbb{R}^s$ where $s \ll m$
- the subspace, latent space, embedding $Z \in \mathbb{R}^s$

Autoencoders

Given dataset $X \in \mathbb{R}^{m \times N}$, an autoencoder is composed of three essential components

- encoder $E : \mathbb{R}^m \to \mathbb{R}^s$ such that $z = E(x) \in \mathbb{R}^s$ where $s \ll m$
- the subspace, latent space, embedding $Z \in \mathbb{R}^s$
- decoder $D : \mathbb{R}^s \to \mathbb{R}^m$ such that $x' = D(z) \in \mathbb{R}^m$

whose aim is to find the true subspace Z for X such that $x \approx D(E(x)),\ \forall x \in X$.

Autoencoders

Given dataset $X \in \mathbb{R}^{m \times N}$, an autoencoder is composed of three essential components

- encoder $E : \mathbb{R}^m \to \mathbb{R}^s$ such that $z = E(x) \in \mathbb{R}^s$ where $s \ll m$
- the subspace, latent space, embedding $Z \in \mathbb{R}^s$
- decoder $D : \mathbb{R}^s \to \mathbb{R}^m$ such that $x' = D(z) \in \mathbb{R}^m$

whose aim is to find the true subspace Z for X such that $x \approx D(E(x)),\ \forall x \in X$.

Remark: this setting can be solved by any dimensionality reduction method (ex. EVD, SVD, PCA, R-PCA etc.).

AE: Optimization Problem

In the context of Neural Networks, we want to optimize the weights ω of the activation functions.

AE: Optimization Problem

In the context of Neural Networks, we want to optimize the weights ω of the activation functions.

Our loss becomes

$$\{\omega_E^\star, \omega_D^\star\} = \arg\min_{\omega_E, \omega_D} \|X - D_{\omega_D}(E_{\omega_E}(X))\| \tag{1}$$

where the ℓ_2 penalty can be replaced by other distance functions.

AE: Optimization Problem

In the context of Neural Networks, we want to optimize the weights ω of the activation functions.

Our loss becomes

$$\{\omega_E^\star, \omega_D^\star\} = \arg\min_{\omega_E, \omega_D} \|X - D_{\omega_D}(E_{\omega_E}(X))\| \tag{1}$$

where the ℓ_2 penalty can be replaced by other distance functions.

Remark: anomalies are the data whose reconstruction error $\|x - x'\|$ is high due to the new subspace.

AE: Architecture

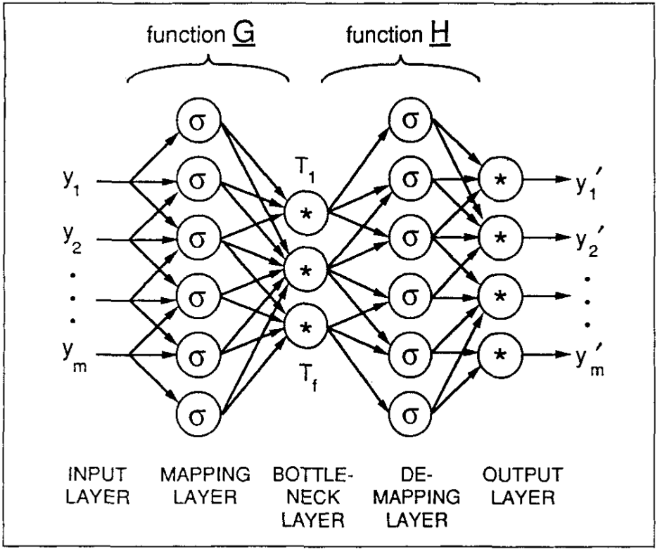


Figure: Autoencoder architecture proposed in (Kramer 1991)

Convolutional Autoencoders

Convolutional Autoencoders (CAE)

Convolutional Autoencoders use convolutional and pool layers in the neural network architecture.

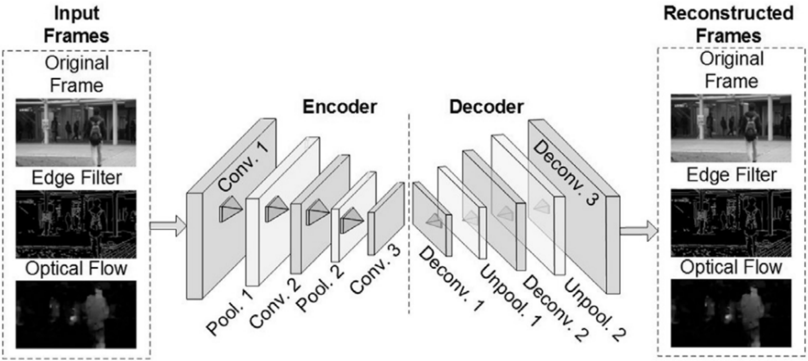


Figure: CAE for anomaly detection in videos (Ribeiro, Lazzaretti, and Lopes 2018)

Convolution Padding

Replication Padding

5	5	0	8	7	8	1	1
5	**5**	**0**	**8**	**7**	**8**	**1**	1
1	**1**	**9**	**5**	**0**	**7**	**7**	7
6	**6**	**0**	**2**	**4**	**6**	**6**	6
9	**9**	**7**	**6**	**6**	**8**	**4**	4
8	**8**	**3**	**8**	**5**	**1**	**3**	3
7	**7**	**2**	**7**	**0**	**1**	**0**	0
7	7	2	7	0	1	0	0

Reflection Padding

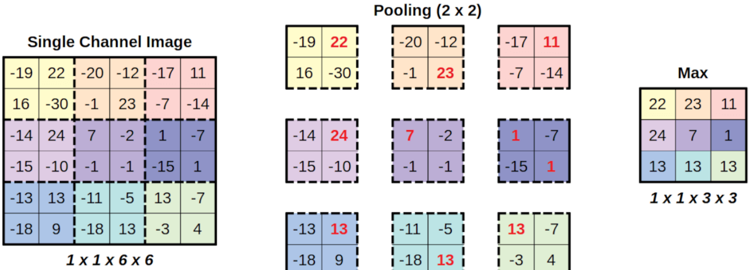
9	1	9	5	0	7	7	7
0	**5**	**0**	**8**	**7**	**8**	**1**	8
9	**1**	**9**	**5**	**0**	**7**	**7**	7
0	**6**	**0**	**2**	**4**	**6**	**6**	6
7	**9**	**7**	**6**	**6**	**8**	**4**	8
3	**8**	**3**	**8**	**5**	**1**	**3**	1
2	**7**	**2**	**7**	**0**	**1**	**0**	1
3	8	3	8	5	1	3	1

Circular Padding

0	7	2	7	0	1	0	7
1	**5**	**0**	**8**	**7**	**8**	**1**	5
7	**1**	**9**	**5**	**0**	**7**	**7**	1
6	**6**	**0**	**2**	**4**	**6**	**6**	6
4	**9**	**7**	**6**	**6**	**8**	**4**	9
3	**8**	**3**	**8**	**5**	**1**	**3**	8
0	**7**	**2**	**7**	**0**	**1**	**0**	7
1	5	0	8	7	8	1	5

Source: https://en.wikipedia.org/wiki/Pooling_layer

Convolution Max Pooling



Source: https://en.wikipedia.org/wiki/Pooling_layer

CAE: Average Pooling

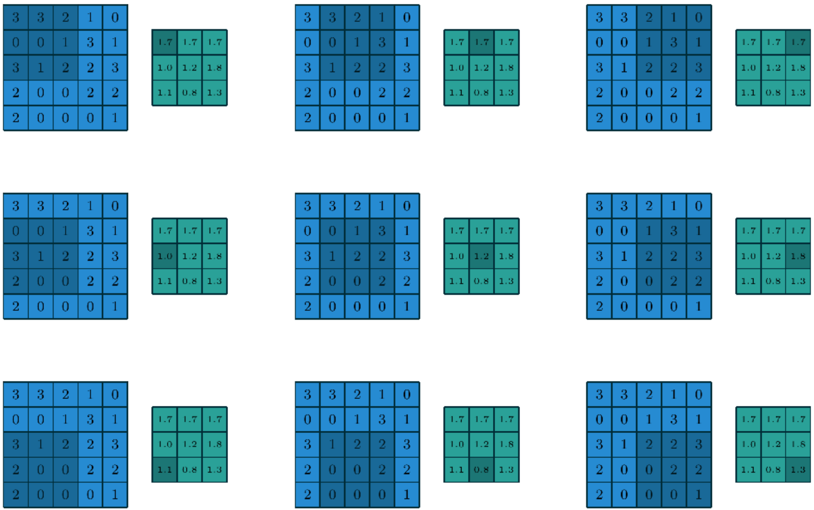


Figure: CAE with average pooling (Dumoulin and Visin 2016)

CAE: Max Pooling

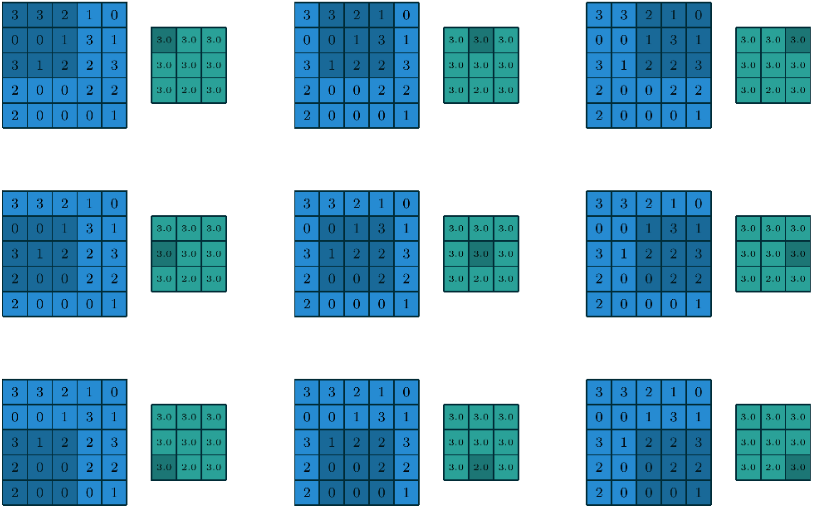


Figure: CAE with max pooling (Dumoulin and Visin 2016)

CAE: Deconvolution is Transposed Convolution

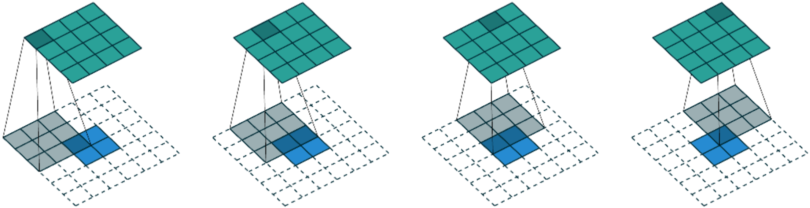


Figure: CAE with transposed convolution (Dumoulin and Visin 2016)

Remark: this includes a convolution and upscaling operation that are sometimes termed “deconv” and “unpool”.

CAE: Transposed Convolution with Stride

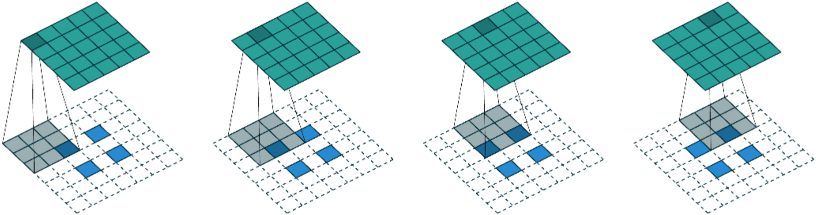


Figure: CAE transposed convolution with stride (Dumoulin and Visin 2016)

Sparse Autoencoders

Sparse Autoencoders (SAE)

Sparse autoencoders enforce sparsity in the mid-layer representations thus allowing subspaces of size $\tilde{m} > m$ but with at most $k < m < \tilde{m}$ non-zeros.

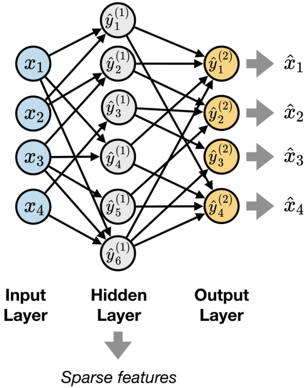


Figure: SAE with overcomplete layer (Dutta et al. 2018)

SAE: Hard-Thresholding with ℓ_0

k-Spare Autoencoders Makhzani and Frey 2013 compute the activation function in each node and then select the k largest values

$$x^{(\ell)} = \mathrm{max}_\mathrm{k}\, \varphi_{\omega_\ell}(x^{(\ell-1)}) \tag{2}$$

where max_k is the function selecting the top-k values in x whose associated support is supp_k.

SAE: Regularization with ℓ_1

Sparse regularized autoencoders add regularization to the AE problem

$$\arg\min_{\omega_E,\omega_D} \|X - D_{\omega_D}(E_{\omega_E}(X))\| + \lambda\phi(X) \tag{3}$$

where $\phi(X)$ is a sparse inducing penalty usually applied at each layer.

SAE: Regularization with ℓ_1

Sparse regularized autoencoders add regularization to the AE problem

$$\arg\min_{\omega_E,\omega_D} \|X - D_{\omega_D}(E_{\omega_E}(X))\| + \lambda\phi(X) \tag{3}$$

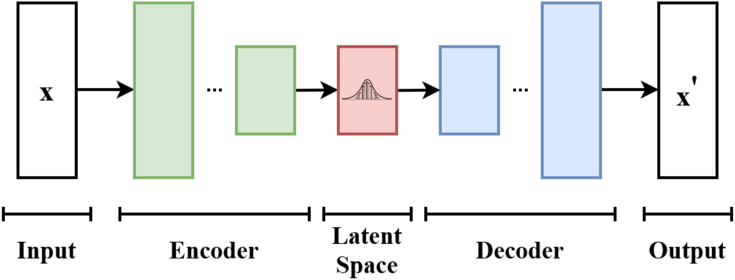
where $\phi(X)$ is a sparse inducing penalty usually applied at each layer.

The ℓ_1 convex norm is a popular candidate for $\phi(\cdot)$ such that at each layer we have

$$\phi(X) = \sum_{\ell} \omega_\ell \left\|\varphi(X^{\ell-1})\right\|_1 \tag{4}$$

Variational Autoencoder (VAE)

Variational autoencoders (Kingma 2013) model input vector x as a mixture of (Gaussian) distributions.



Source: https://en.wikipedia.org/wiki/Variational_autoencoder

AE: Subspace Connections

The resulting autoencoder subspace representations z or final representations x' are coupled with existing methods to enhance anomaly detection performance.

DeepSVDD: AE and SVDD

Given dataset X and autoencoder (E, D) with weights ω, DeepSVDD (Ruff et al. 2018) formulates the soft-boundary as

$$\min_{R,\omega} R^2 + \frac{1}{\nu N} \sum_{i=1}^{N} \max\{0, \|D(E(X)) - c\|^2 - R^2\} + \frac{\lambda}{2} \sum_{\ell} \left\|\omega^{\ell}\right\|_F^2 \quad (5)$$

DeepSVDD: AE and SVDD

Given dataset X and autoencoder (E, D) with weights ω, DeepSVDD (Ruff et al. 2018) formulates the soft-boundary as

$$\min_{R,\omega} R^2 + \frac{1}{\nu N} \sum_{i=1}^{N} \max\{0, \|D(E(X)) - c\|^2 - R^2\} + \frac{\lambda}{2} \sum_{\ell} \left\|\omega^{\ell}\right\|_F^2 \quad (5)$$

For unbalanced datasets, where we have more normal data than anomalies, the authors propose a simplified one-class objective

$$\min_{\omega} \frac{1}{N} \sum_{i=1}^{N} \|D(E(X)) - c\|^2 + \frac{\lambda}{2} \sum_{\ell} \left\|\omega^{\ell}\right\|_F^2 \quad (6)$$

DeepOCSVM: AE and OC-SVM

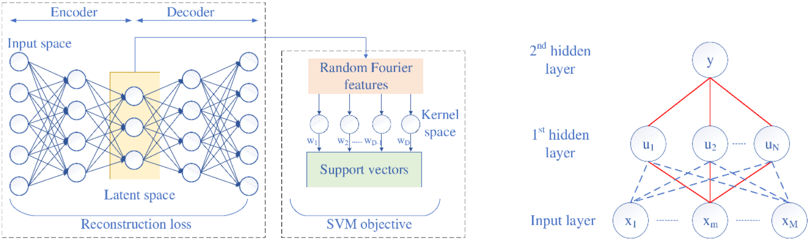


Figure: DeepOCSVM architecture with Fourier Features (Nguyen and Vien 2019)

The DeepOCSVM loss is

$$\alpha \|X - D(E(X))\| + \frac{1}{2}\|w\|^2 - \rho + \frac{1}{\nu N}\sum_{i=1}^{N}\max\{0, \rho - w^\top z(x_i)\} \quad (7)$$

where $z(\cdot)$ is the Random Fourier Features (RFF) function.

References

Dumoulin, Vincent and Francesco Visin (2016). "A guide to convolution arithmetic for deep learning". In: *arXiv preprint arXiv:1603.07285*.

Dutta, Sanghamitra et al. (2018). "A unified coded deep neural network training strategy based on generalized polydot codes". In: *2018 IEEE International Symposium on Information Theory (ISIT)*. IEEE, pp. 1585–1589.

Kingma, Diederik P (2013). "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114*.

Kramer, Mark A (1991). "Nonlinear principal component analysis using autoassociative neural networks". In: *AIChE journal* 37.2, pp. 233–243.

Makhzani, Alireza and Brendan Frey (2013). "K-sparse autoencoders". In: *arXiv preprint arXiv:1312.5663*.

Nguyen, Minh-Nghia and Ngo Anh Vien (2019). "Scalable and interpretable one-class svms with deep learning and random fourier features". In: *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2018, Dublin, Ireland, September 10–14, 2018, Proceedings, Part I 18*. Springer, pp. 157–172.

Ribeiro, Manassés, André Eugênio Lazzaretti, and Heitor Silvério Lopes