

# Deep Learning – Parte 4

## Autoencoders, manifolds e RBMs

### Índice Geral

<b>1</b>	<b>Aprendizado da representação .....</b>	<b>2</b>
<b>2</b>	<b>Hipótese das variedades (manifolds) .....</b>	<b>4</b>
<b>3</b>	<b>Fundamentos de autoencoders.....</b>	<b>14</b>
	3.1 Denoising autoencoders .....	20
	3.2 Pré-treinamento empregando stacked autoencoders.....	23
	3.3 Variational autoencoders .....	27
	3.4 O truque da reparametrização .....	33
	3.5 Disentangled VAE .....	37
<b>4</b>	<b>Restricted Boltzmann machines (RBMs).....</b>	<b>38</b>
	4.1 RBMs × (V)AEs .....	47
<b>5</b>	<b>Alucinação em redes neurais profundas.....</b>	<b>48</b>
	5.1 Neural inpainting.....	51
<b>6</b>	<b>Referências bibliográficas .....</b>	<b>59</b>

# 1 Aprendizado da representação

- O sucesso dos algoritmos de aprendizado de máquina depende fundamentalmente da forma com que os dados são representados (BENGIO et al., 2013).
- Representações diferentes carregam ou escondem diferentes fatores capazes de explicar a distribuição e o comportamento dos dados.
- Embora conhecimento de domínio possa, vez ou outra, estar disponível, é possível também aprender a partir de hipóteses iniciais genéricas (*priors*). Mas o que tem se mostrado mais efetivo é o aprendizado da representação, que vai sintetizar esses *priors* a partir dos dados.
- Classificadores de alto desempenho, por exemplo, requerem a identificação de atributos discriminativos das classes.
- Esta parte do material vai abordar aprendizado de variedades (*manifolds*), *autoencoders* e máquinas de Boltzmann restritas (RBMs, do inglês *Restricted Boltzmann Machines*).

- Se a inteligência artificial envolve conceber um bom modelo de mundo, de modo a maximizar o acerto na tomada de decisão, é possível argumentar que isso só pode ser alcançado ao se aprender como identificar e desembaraçar os fatores explicativos fundamentais que se escondem no baixo nível dos dados observados (BENGIO et al., 2013).
- Com base em um aprendizado de representação competente, tende a ficar mais fácil a extração de informação útil ao se sintetizarem classificadores e outros preditores de alta performance.
- Métodos de *deep learning* são caracterizados pela composição de uma cascata de transformações não-lineares com o propósito de produzir representações dos dados que sejam mais abstratas e mais úteis para a execução da tarefa pretendida.
- Objetivos centrais em aprendizado de representação são: (1) Identificar o que faz uma representação ser melhor que outra; (2) Definir como sintetizar tal representação; (3) Expressar matematicamente os objetivos a serem atingidos.

## 2 Hipótese das variedades (manifolds)

- Hipótese das variedades (*manifolds*): a função densidade de probabilidade que gera os dados se concentra em torno de regiões de baixa dimensão (CAYTON, 2005; NARAYANAN & MITTER, 2010).
- Sendo assim, é muito pouco provável que configurações de variáveis do processo definidas segundo uma distribuição uniforme, ou outras distribuições arbitrárias e definidas a priori, sejam capazes de gerar o tipo de dado que se quer modelar.
- Qual é a probabilidade de se gerar uma imagem que aparente ser real pela escolha independente da intensidade dos pixels?
- Qual é a probabilidade de se gerar um texto sintática e semanticamente válido, numa linguagem, pela escolha independente dos caracteres na sequência?
- De fato, dentre todas as distribuições possíveis de pixels em uma imagem, por exemplo, por mais variação que esteja presente na base de imagens, ainda assim a função densidade de probabilidade que explica as distribuições de pixels em

imagens de interesse prático tende a residir em espaços de baixa dimensão e “volume” reduzido.

- Algo equivalente pode ser dito a respeito de dados de texto, vídeo, fala e música.
- O conceito de variedade é mais rigoroso em teorias matemáticas (baseadas em geometria diferencial) do que em fundamentos de aprendizado de máquina:
  - ✓ Os dados não precisam pertencem estritamente à variedade, podendo apenas estar próximos dela.
  - ✓ A dimensão da variedade não precisa ser a mesma em todo lugar.
  - ✓ A variedade pode residir em espaços contínuos ou discretos.
- O fundamental é mapear onde se concentram as amostras de mais alta probabilidade, e é isso que o aprendizado de variedades busca realizar, seja para distribuições contínuas ou discretas. Logo, o problema é mais de aproximação de uma variedade em um subespaço do que de estimação de densidade de

probabilidade, embora ambas as perspectivas sejam válidas em tarefas de aprendizado de máquina.

- Mais especificamente, o aprendizado da representação, quando implementado a partir de uma rede neural profunda, tem a tarefa implícita de desdobrar a variedade a partir de mapeamentos não-lineares em cascata, para, em seguida, aproximar esta variedade por um hiperplano de dimensão compatível, ou seja, por uma solução de PCA, na última camada.
- No exemplo da Figura 1, a seguir, percebe-se que a variedade se concentra em um volume desprezível de todo o espaço de configurações admissíveis (também chamado de **espaço latente**), indicando que identificar a região em que a variedade se manifesta representa uma informação de alta relevância.

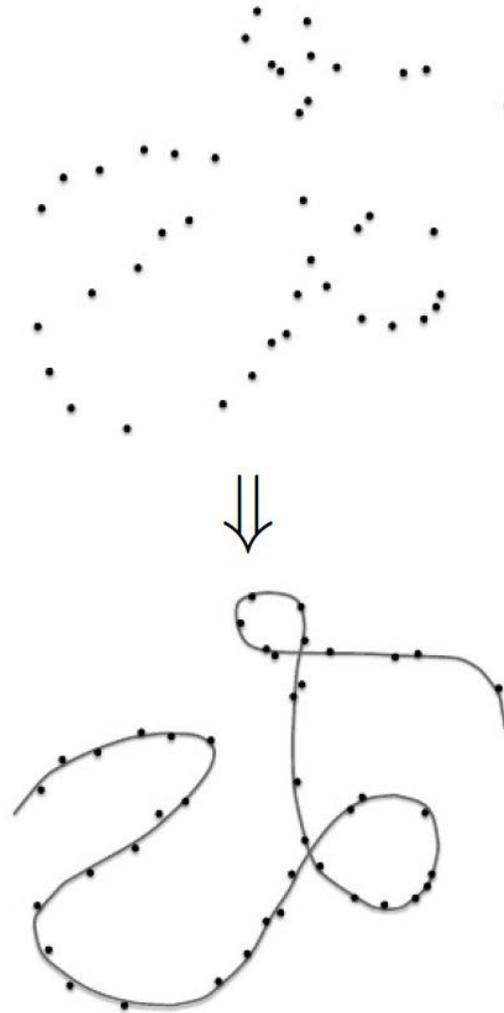


Figura 1 – Dados amostrados de uma distribuição 2D que, de fato, está concentrada em uma variedade unidimensional. Fonte: GOODFELLOW et al. (2016).

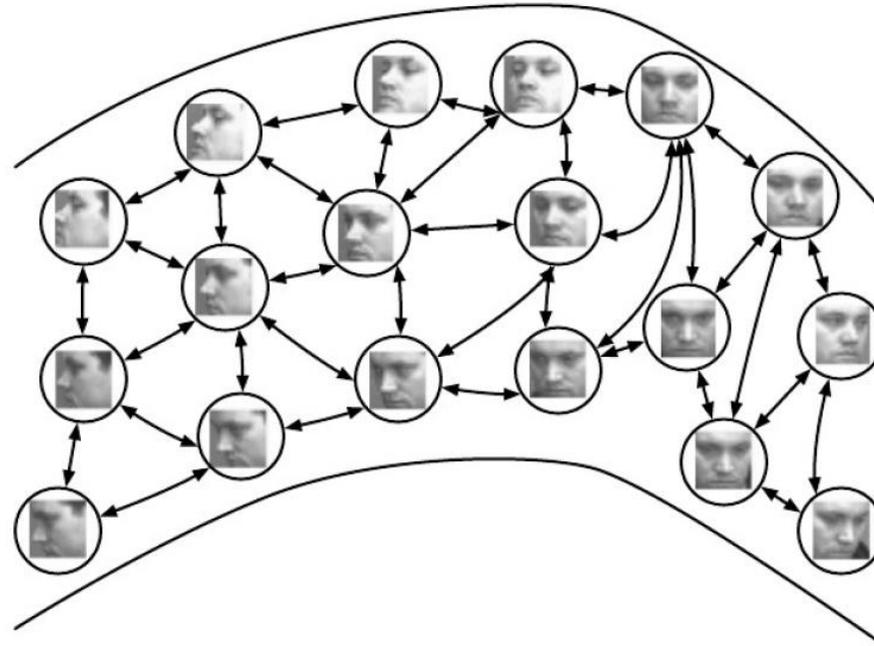


Figure 14.8: Non-parametric manifold learning procedures build a nearest neighbor graph in which nodes represent training examples and directed edges indicate nearest neighbor relationships. Various procedures can thus obtain the tangent plane associated with a neighborhood of the graph as well as a coordinate system that associates each training example with a real-valued vector position, or **embedding**. It is possible to generalize such a representation to new examples by a form of interpolation. So long as the number of examples is large enough to cover the curvature and twists of the manifold, these approaches work well. Images from the QMUL Multiview Face Dataset (Gong *et al.*, 2000).

Figura 2 – Uma vez encontrado o *manifold*, a vizinhança topológica carrega muita informação, abrindo possibilidades para muitas aplicações práticas. Fonte: GOODFELLOW et al. (2016).

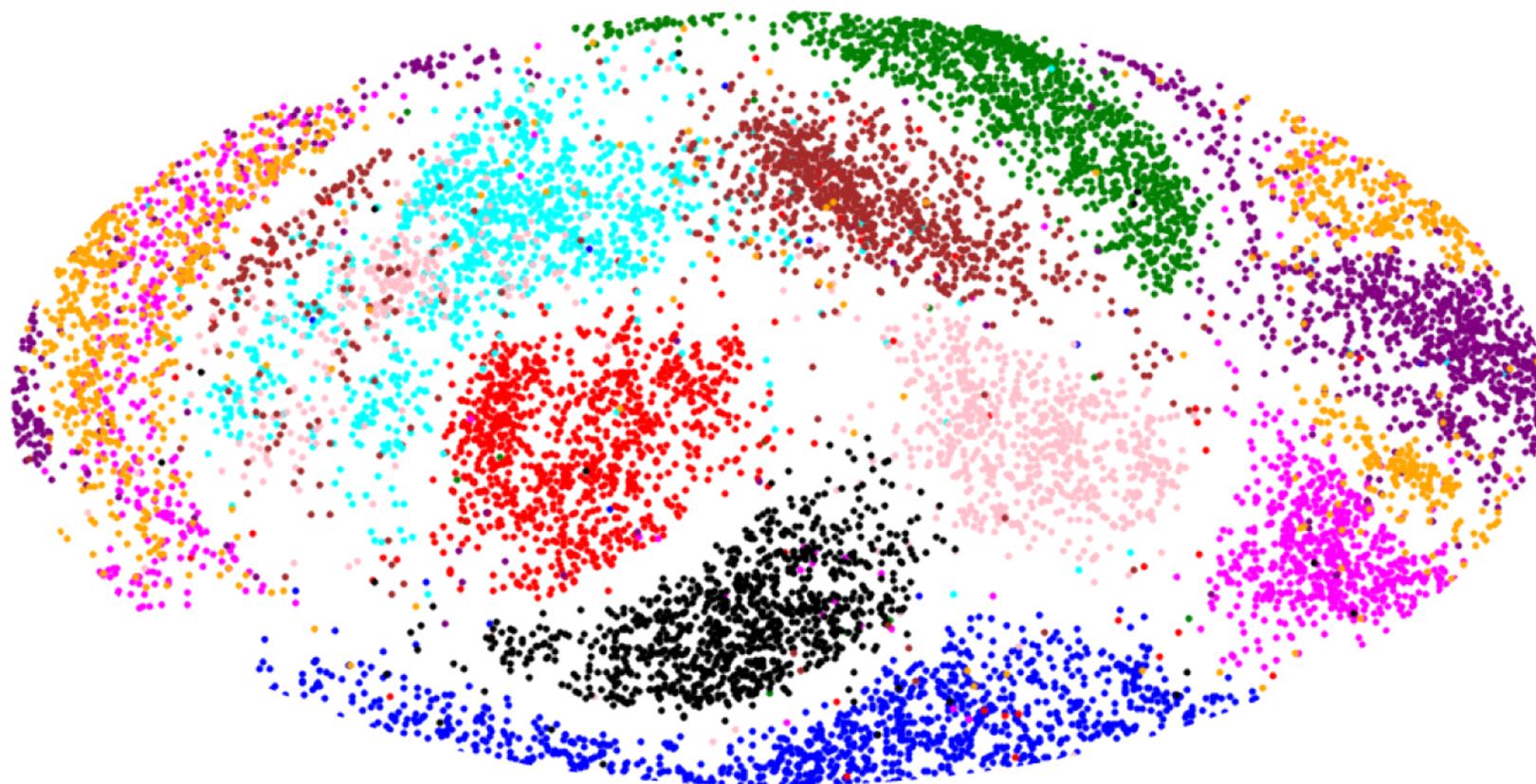


Figura 3 – *Manifold* hiper-esférico para a base MNIST, a partir de um autoencoder variacional.

Fonte: DAVIDSON et al. (2018).

- Identificar adequadamente a variedade permite então responder a perguntas como:
  - ✓ A amostra corresponde a uma configuração provável ou não?
  - ✓ A sentença é sintática e semanticamente plausível ou não?
  - ✓ A imagem tem uma aparência real ou não?
- Responder a essas questões, que admitem respostas binárias, por indicar onde as altas probabilidades se concentram, nos diz muito mais acerca das características, por exemplo, de uma linguagem natural ou imagem, do que a informação adicional associada à atribuição de uma probabilidade precisa a cada sequência possível de caracteres ou subconjunto de pixels.
- É sabido que as variedades de interesse em IA geralmente apresentam muitas oscilações e dobras, o que torna atrativo o emprego de arquiteturas profundas de redes neurais para identificar, desdobrar e suavizar cada variedade.

- Análise de componentes principais (PCA) pode ser interpretada como uma técnica capaz de identificar variedades lineares junto aos dados, conforme ilustrado nas figuras a seguir, extraídas de HASTIE et al. (2009).

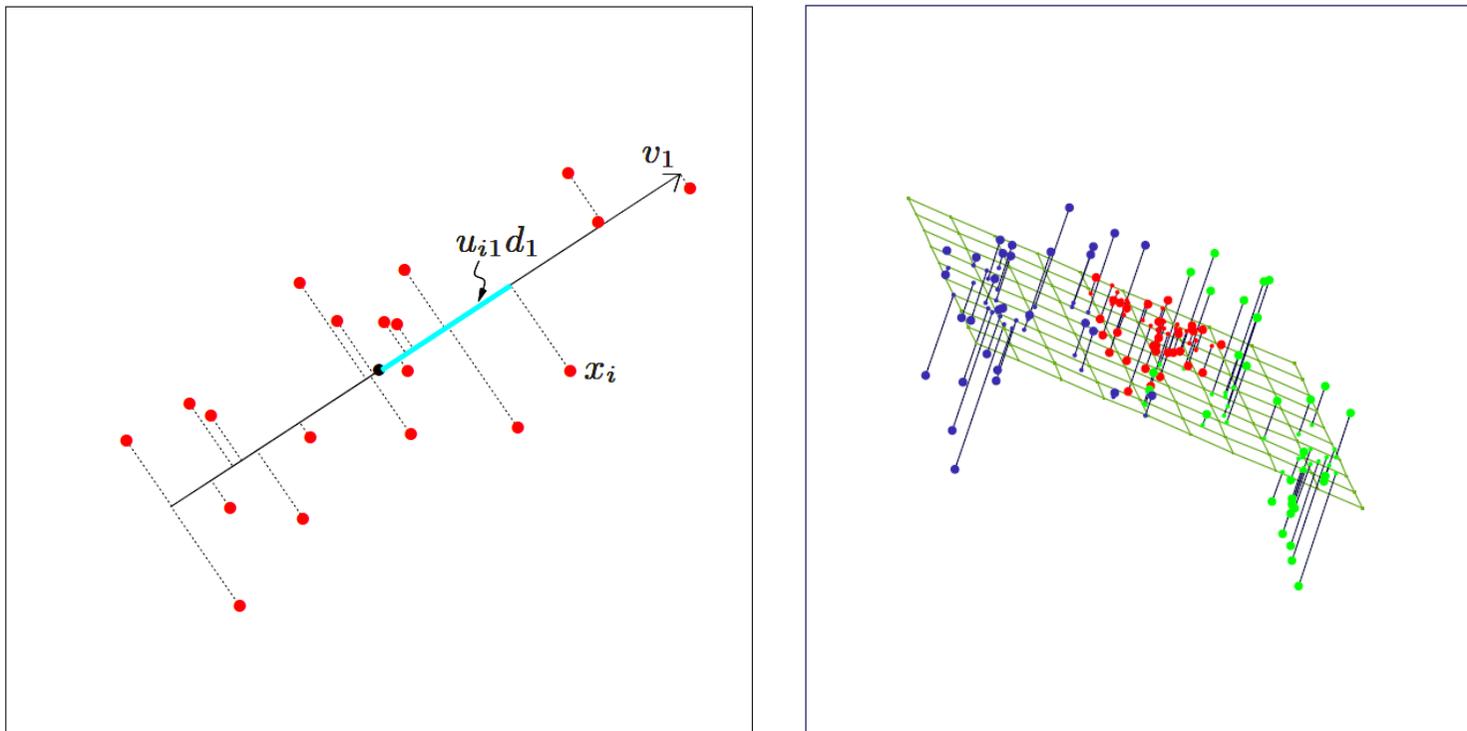


Figura 4 – Variedades lineares obtidas a partir de análise de componentes principais (PCA).

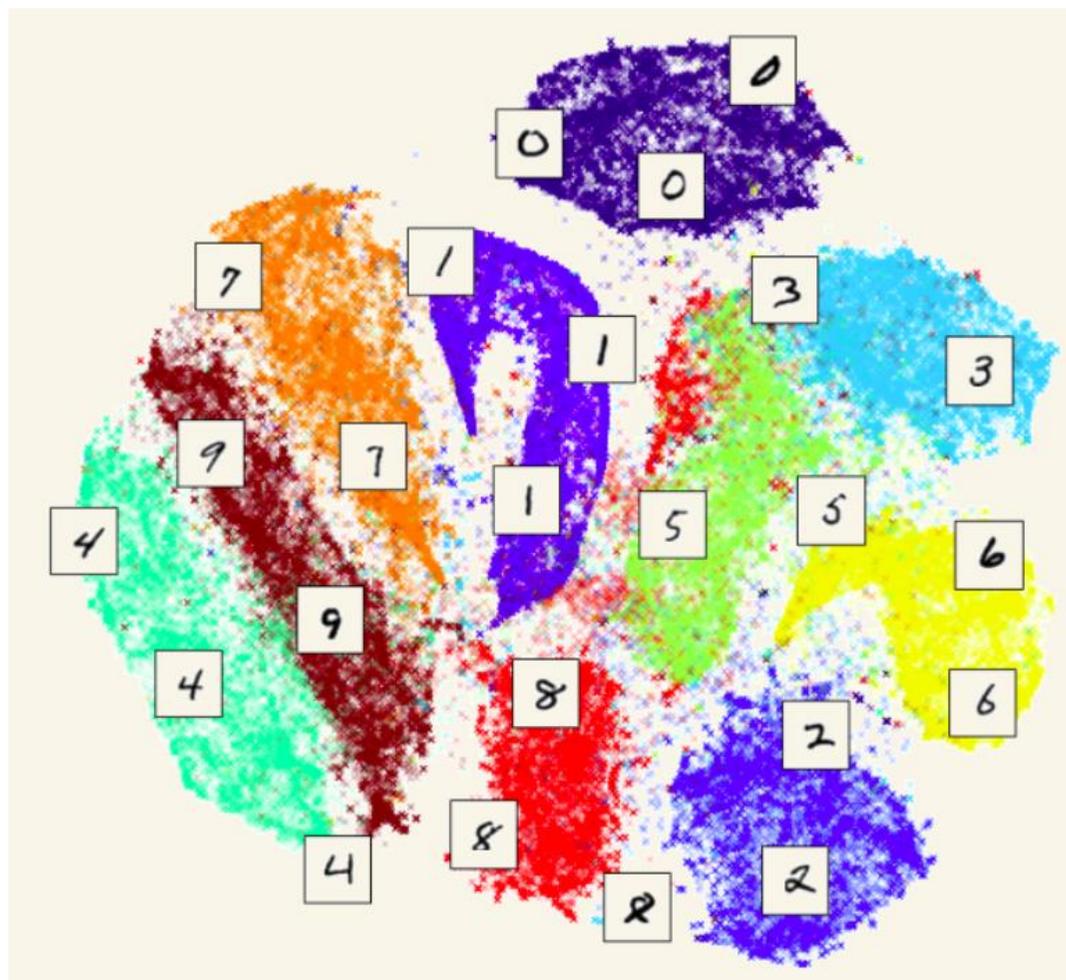


Figura 5 – Projeção não-linear produzida por t-SNE (VAN DER MAATEN & HINTON, 2008).

Fonte: <https://afiodorov.github.io/2019/05/27/tsne/>

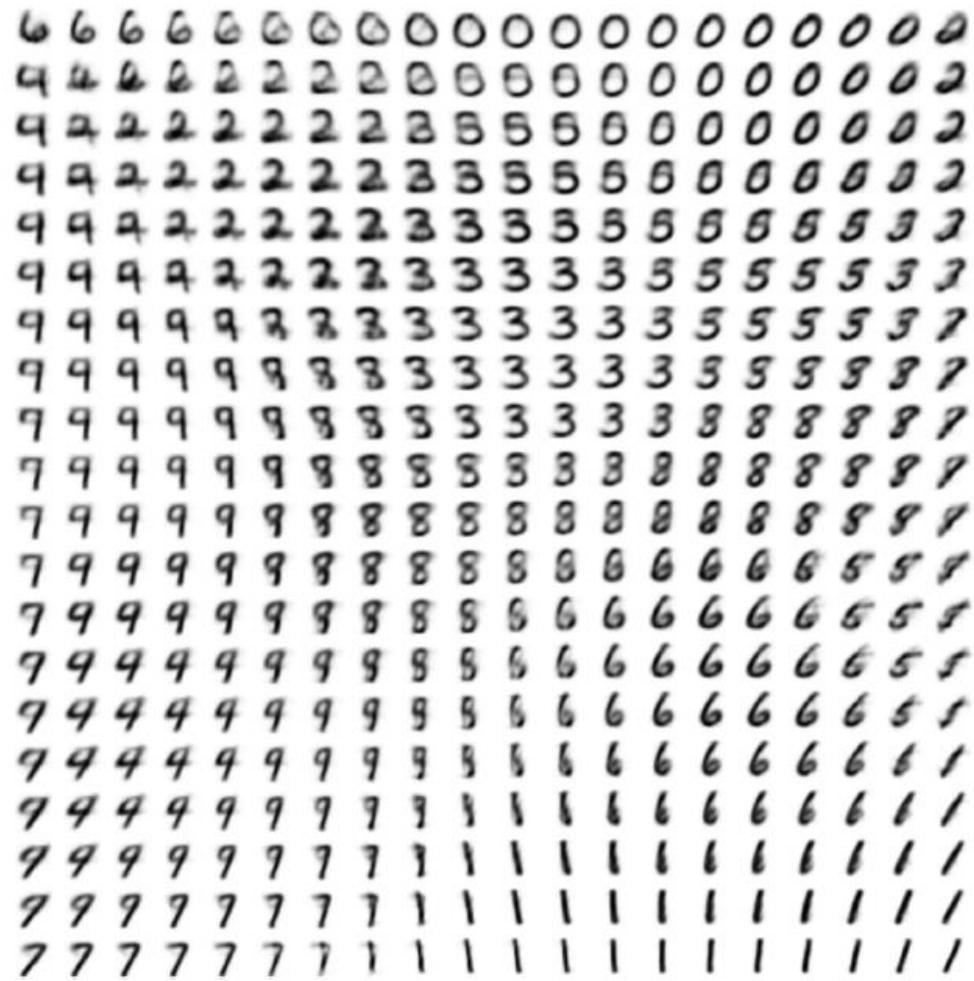


Figura 6 – *Manifold* aprendido por um autoencoder variacional para a base de dados MNIST.

Fonte: GOODFELLOW et al. (2016).

### 3 Fundamentos de autoencoders

- Uma das configurações mais poderosas para *deep learning*, usada, por exemplo, em aprendizado da representação e na identificação de variedades (*manifolds*), é o *autoencoder*, que está associado a treinamento não-supervisionado, mas é guiado por treinamento supervisionado, mais precisamente, auto-supervisionado.
- O objetivo é codificar eficientemente os dados de entrada em um espaço de representação, ou espaço de variáveis latentes, geralmente com redução de dimensão (gerando assim um gargalo, *bottleneck*), e empregar este código para reconstruir a entrada na saída da rede neural.
- Muitas arquiteturas profundas de alto desempenho foram concebidas como uma cascata de camadas de codificadores sintetizados por autoencoders (o código do autoencoder anterior é tomado como a entrada para o treinamento do autoencoder seguinte), processo denominado de *stacked autoencoders*. Autoencoders têm sido

empregados numa grande diversidade de aplicações bem-sucedidas, como em modelos geracionais, eliminação de ruído em imagens e *inpainting* em imagens.

- Em termos de topologia, um autoencoder tradicional pode ser interpretado como duas redes neurais MLP em cascata, uma responsável pelo codificador e a outra pelo decodificador. O emprego de função de ativação linear conduz o codificador do autoencoder a realizar PCA.

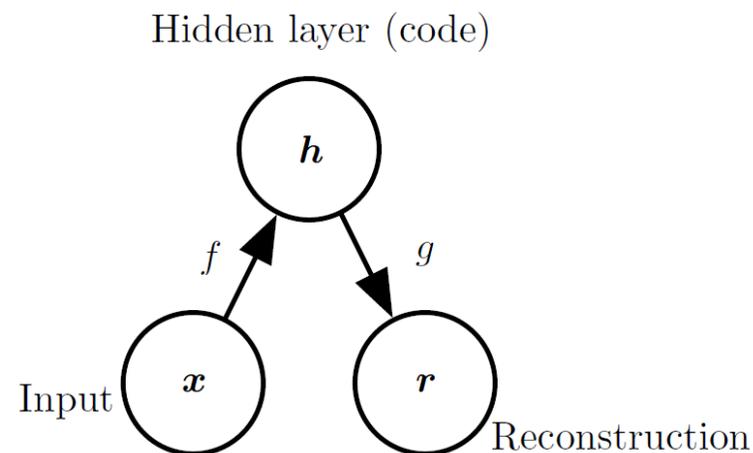


Figura 7 – Sequência de mapeamentos do autoencoder. Fonte: GOODFELLOW et al. (2016).

- Esta sequência de mapeamentos deve ser tal que  $r$  aproxime  $x$ .

- Deve-se evitar que  $f(\cdot)$  – codificador – e  $g(\cdot)$  – decodificador – realizem a função identidade. É até possível que  $h$  tenha uma dimensão maior que  $x$ , mas, para isso, técnicas de regularização são mandatórias.

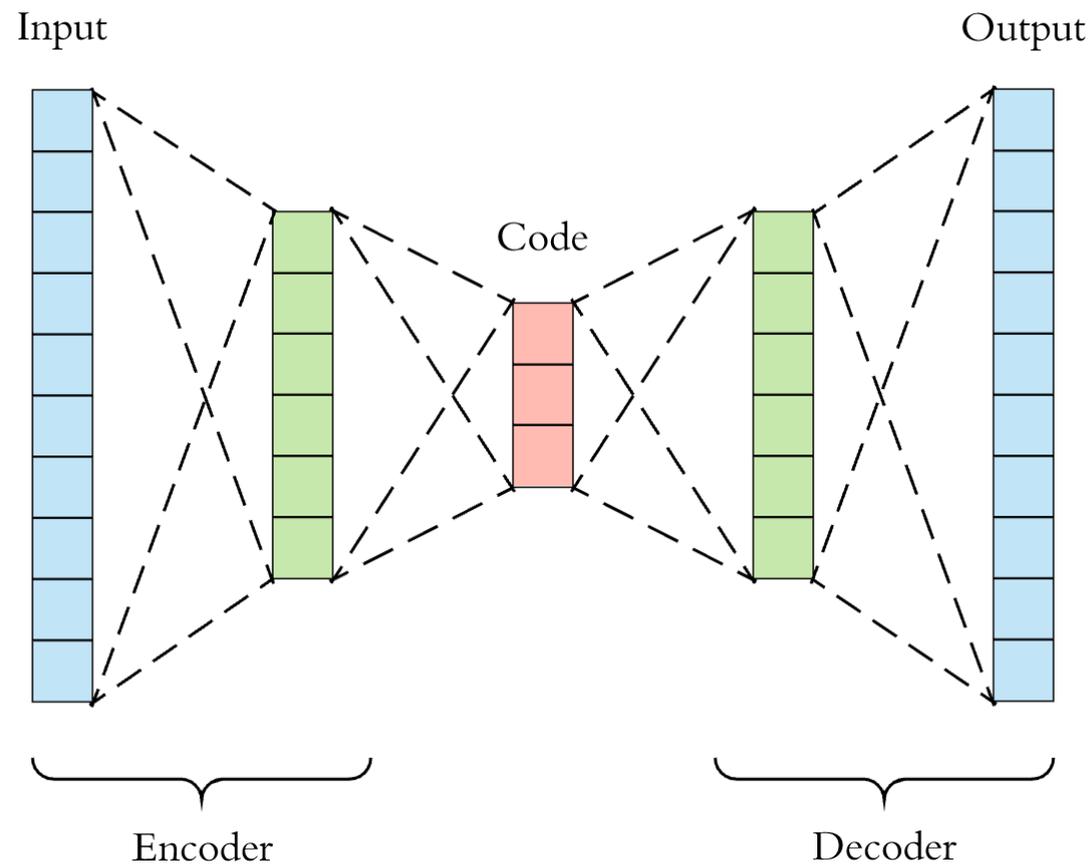


Figura 8 – Configuração tradicional de um autoencoder

- Nada impede o emprego de mais de uma camada intermediária para cada MLP, se for conveniente, ou até a adoção de camadas convolucionais.

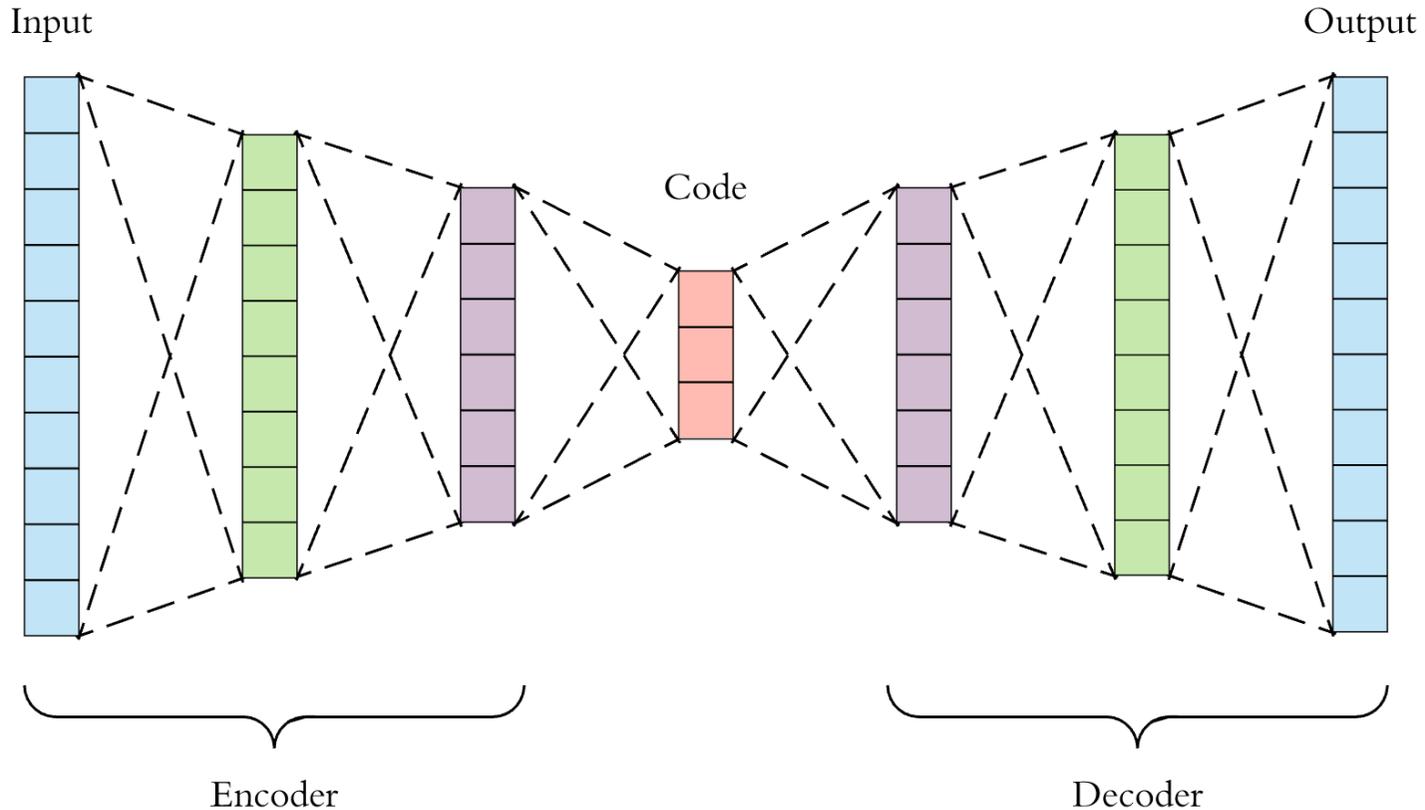


Figura 9 – Autoencoder tradicional, com mais camadas intermediárias

- É fundamental compreender que codificador e decodificador são treinados simultaneamente para reduzir o erro de reconstrução na saída da rede neural.
- Autoencoders convolucionais, como o da Figura 10, podem operar com imagens de resolução maior, pois promovem economia de conexões sinápticas.

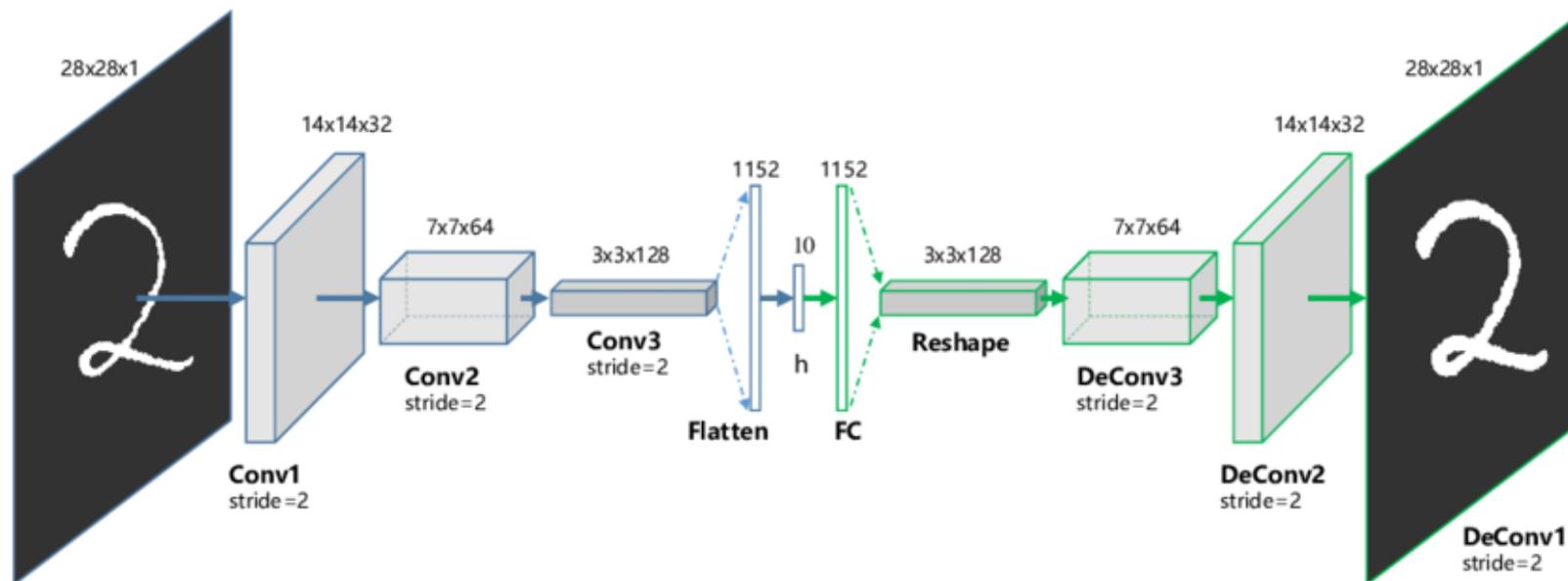


Figura 10 – Exemplo de um autoenconder convolucional. Fonte: GUO et al. (2017).

- As aplicações podem ser bastante variadas, como detecção de outliers e segmentação de imagens, como a seguir.

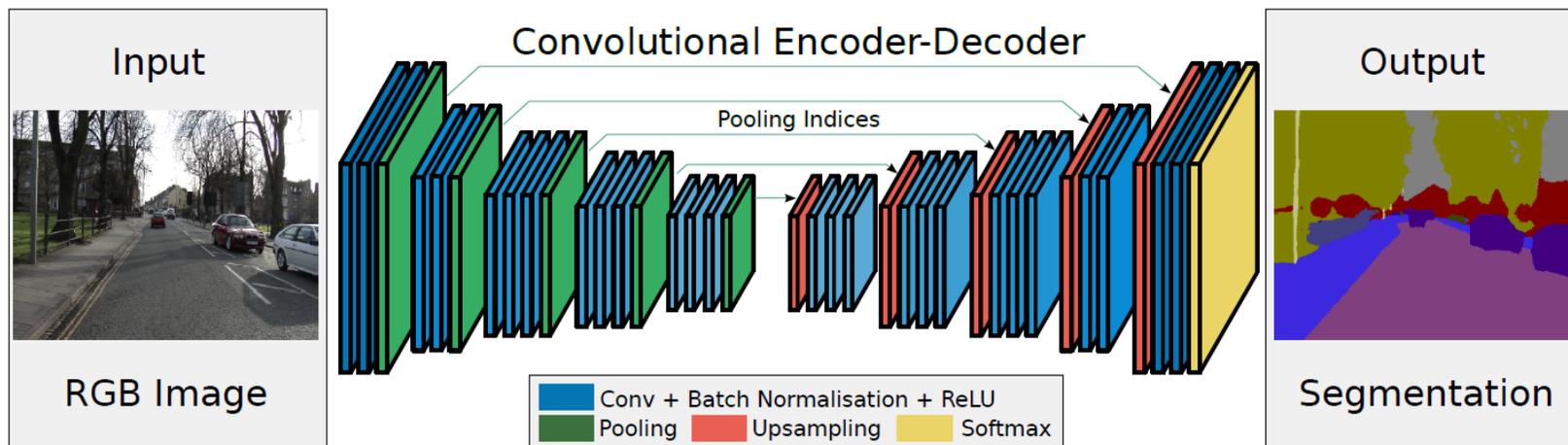


Figura 11 – Princípio de operação da SegNet, para segmentação de imagens. Fonte:

BADRINARAYANAN et al. (2016).

- Quando o foco é no gargalo, é comum a imposição de restrições ao menos junto ao decodificador, visando a obtenção de codificadores ainda mais expressivos.

### 3.1 Denoising autoencoders

- Objetivo: gerar uma representação latente mais robusta, que captura a essência da variação nos dados de entrada.

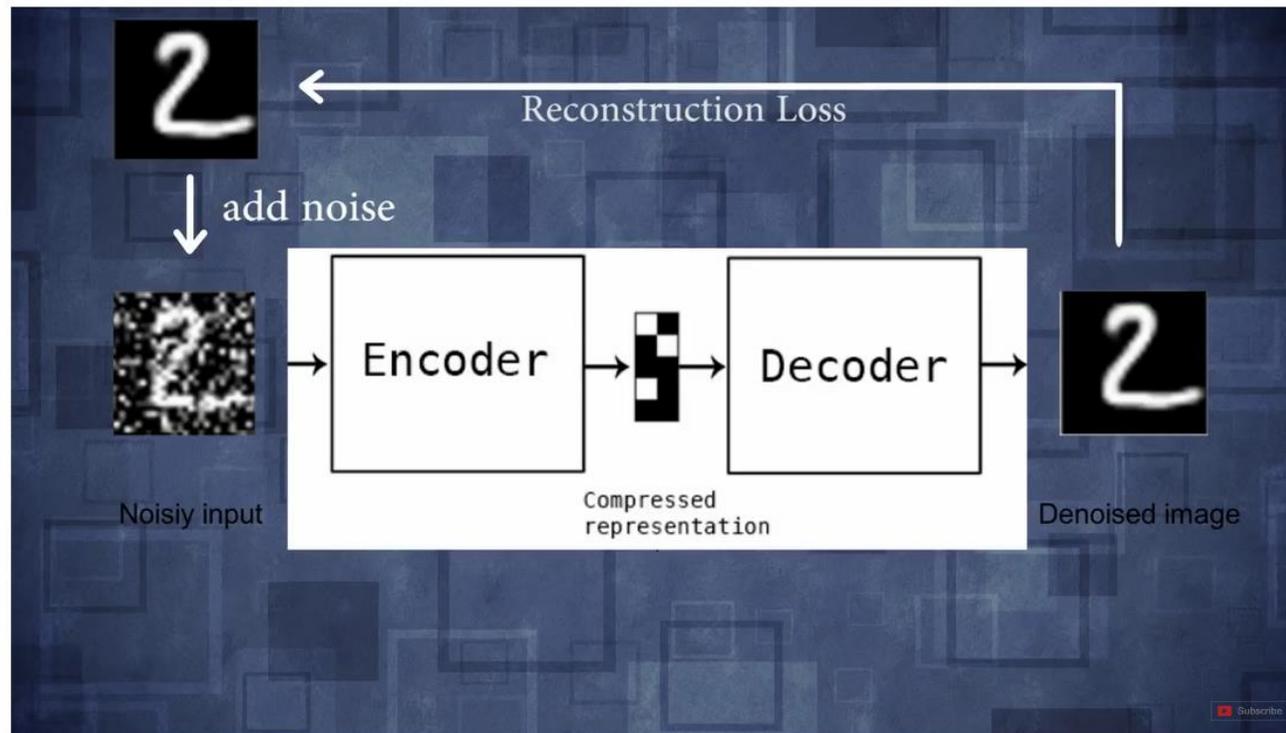


Figura 12 – Princípio de operação do *denoising autoencoder*.

Fonte: <https://www.youtube.com/watch?v=9zKuYvjFFS8>

# Denoising Autoencoder

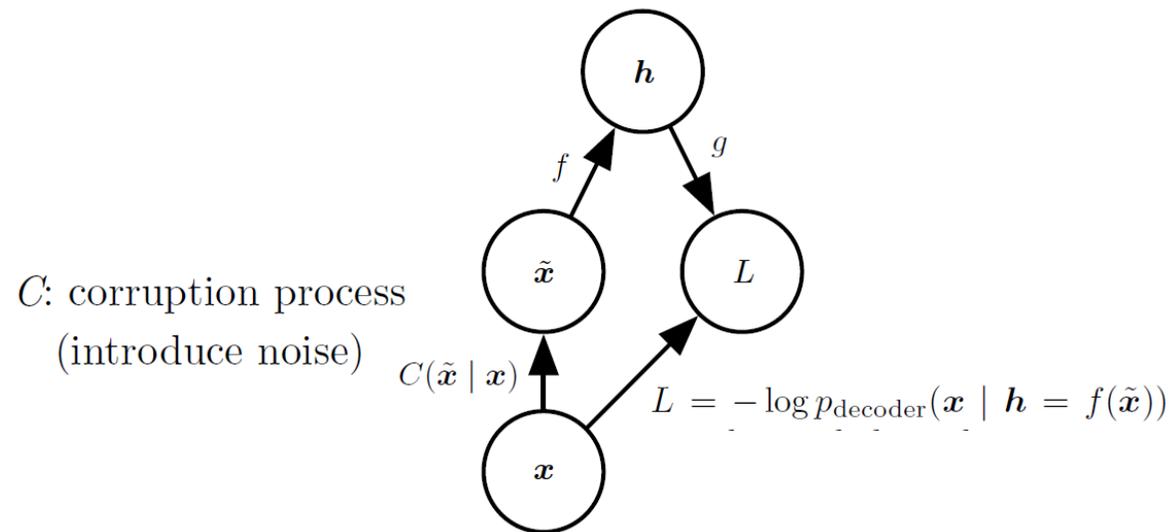


Figura 13 – Princípio de operação do *denoising autoencoder* com ênfase nos mapeamentos e na função de perda. Fonte: GOODFELLOW et al. (2016).

- São variados os tipos de ruído que podem ser introduzidos, dependendo da aplicação pretendida. Exemplos: gaussiano, sal e pimenta, oclusão, dropout, etc.
- A norma  $L_1$  tende a produzir melhores resultados que a norma  $L_2$ , pois esta última acaba gerando imagens um pouco embaçadas, como aquelas da Figura 14.



Figura 14 – Resultado da operação de um *denoising autoencoder* treinado para a base MNIST. À esquerda, dados originais; no centro, entrada ruidosa; à direita, saída do autoencoder. Fonte:

<https://towardsdatascience.com/denoising-autoencoders-explained-dbb82467fc2>

- Para mais detalhes, consultar VINCENT et al. (2010).
- Outras formas de regularização:
  - ✓ [*Sparse autoencoders*] e [*autoencoders with dropout at the hidden layer*];
  - ✓ *Weight sharing*;
  - ✓ Emprego de um decodificador linear (limitação de flexibilidade).

### 3.2 Pré-treinamento empregando stacked autoencoders

- Apesar do seu elevado potencial de representação, é desafiador treinar uma rede neural profunda a partir de pesos iniciais aleatórios, particularmente pela grande dimensão do problema de otimização associado ao treinamento, pela existência de muitos mínimos locais ruins e pontos de sela, e pelo desafio associado à regularização do mapeamento produzido pela rede neural profunda, dada a reduzida quantidade de dados de entrada-saída, quando comparada com a quantidade de pesos sinápticos ajustáveis (flexibilidade do modelo de aproximação).
- Sendo assim, técnicas que permitam partir de pesos sinápticos de alguma forma mais próximos daqueles que seriam obtidos após um treinamento bem-sucedido são indicadas, sendo denominadas de pré-treinamento.
- Dentre as mais competentes técnicas de pré-treinamento, se encontram aquelas baseadas na máquina de Boltzmann restrita (RBM) (HINTON et al., 2006) e nos autoencoders (BENGIO et al., 2006).

## Unsupervised Pretraining

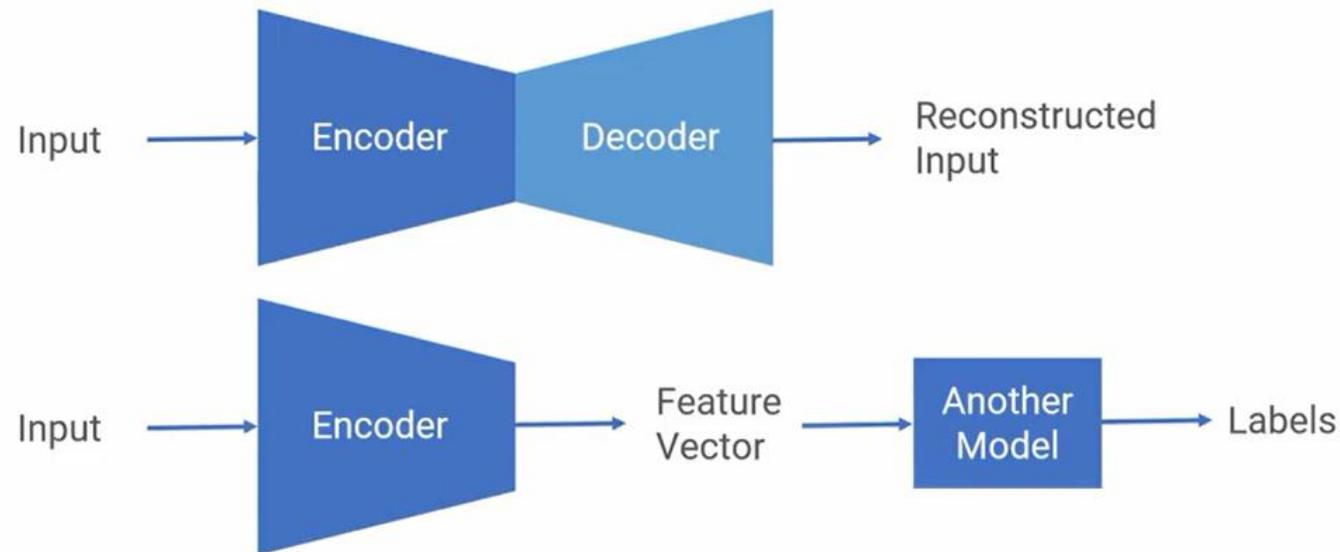


Figura 15 – Pré-treinamento considerando uma camada de codificação. Fonte:

[https://www.youtube.com/watch?v=P8\\_W5Wc4zeg](https://www.youtube.com/watch?v=P8_W5Wc4zeg)

- No caso de autoencoders, cada uma das camadas da rede neural profunda que se entende serem destinadas ao aprendizado da representação devem corresponder ao codificador de um autoencoder treinado para reproduzir a entrada daquela respectiva camada.

- Sendo assim, a saída do codificador que forma a camada anterior da rede neural é tomada como entrada para o treinamento do autoencoder seguinte, e assim por diante, formando uma cascata de codificadores sintetizados por autoencoders.
- Após montar essas camadas pré-treinadas, promove-se um refinamento pelo treinamento conjunto de toda a rede neural profunda, agora já bem menos suscetível a mínimos locais ruins, por exemplo.
- Dicas práticas de como implementar o pré-treinamento podem ser obtidas em:  
<https://machinelearningmastery.com/greedy-layer-wise-pretraining-tutorial/>
- No tratamento de imagens, o pré-treinamento camada a camada não é mais comum, tendo sido substituído por técnicas de *transfer learning*, que tomam um modelo extrator de características de alto desempenho, treinado em bases de imagens contendo uma quantidade gigantesca de imagens, sendo que **apenas a camada de saída é reprojeta para o contexto específico de aplicação.**

- Mais recentemente, com *batch normalization* e *FIXUP initialization* (ver Tópico 8 – Parte 1, Seção 6), os desafios de se chegar a uma boa inicialização e de conseguir treinar redes neurais muito profundas parecem superados, o que reduz a demanda por pré-treinamento.

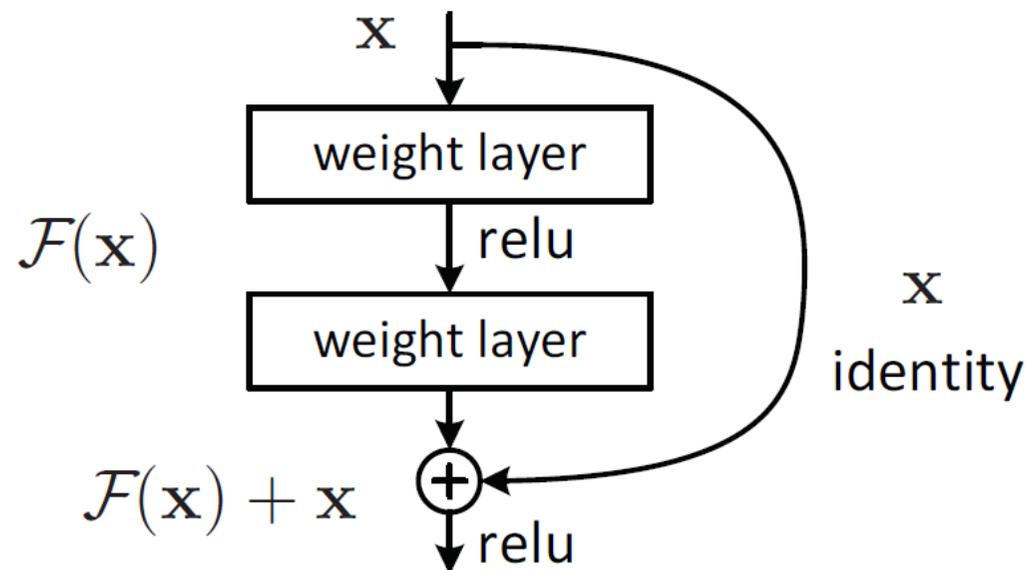
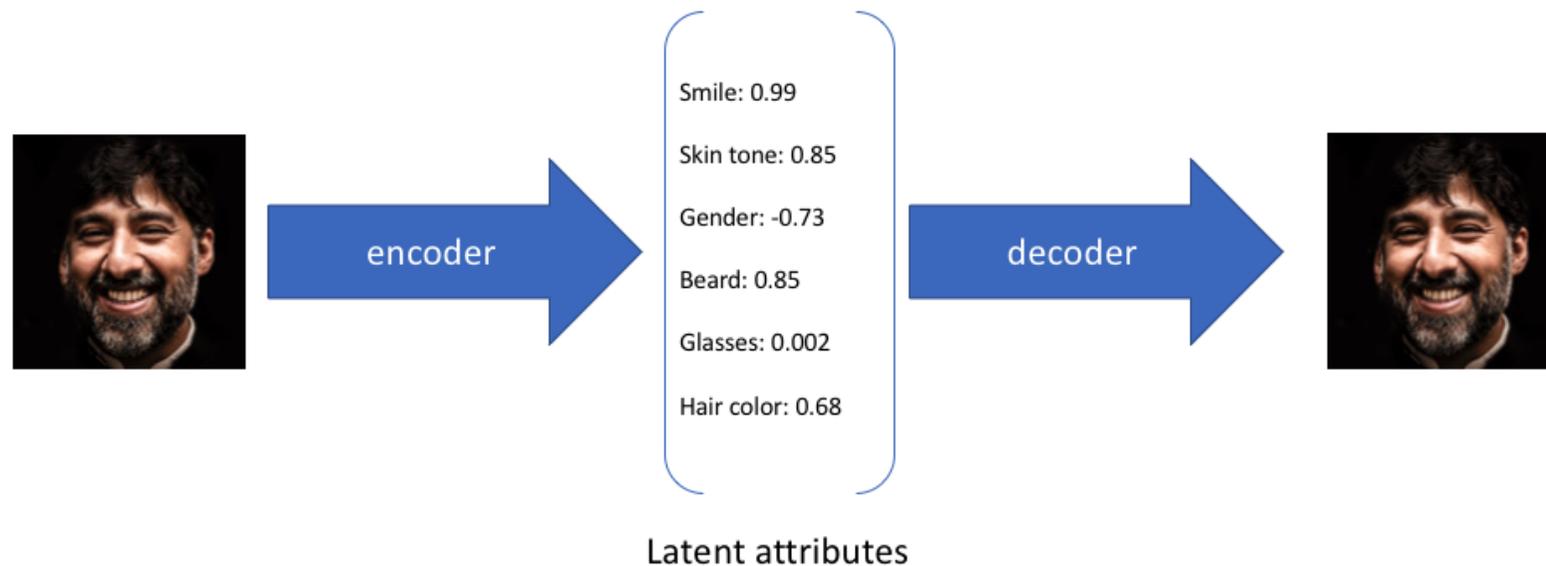


Figura 16 – Bloco residual da ResNet, que atingiu um erro de 3,57% no conjunto de teste da ImageNet em 2015. Fonte: HE et al. (2015).

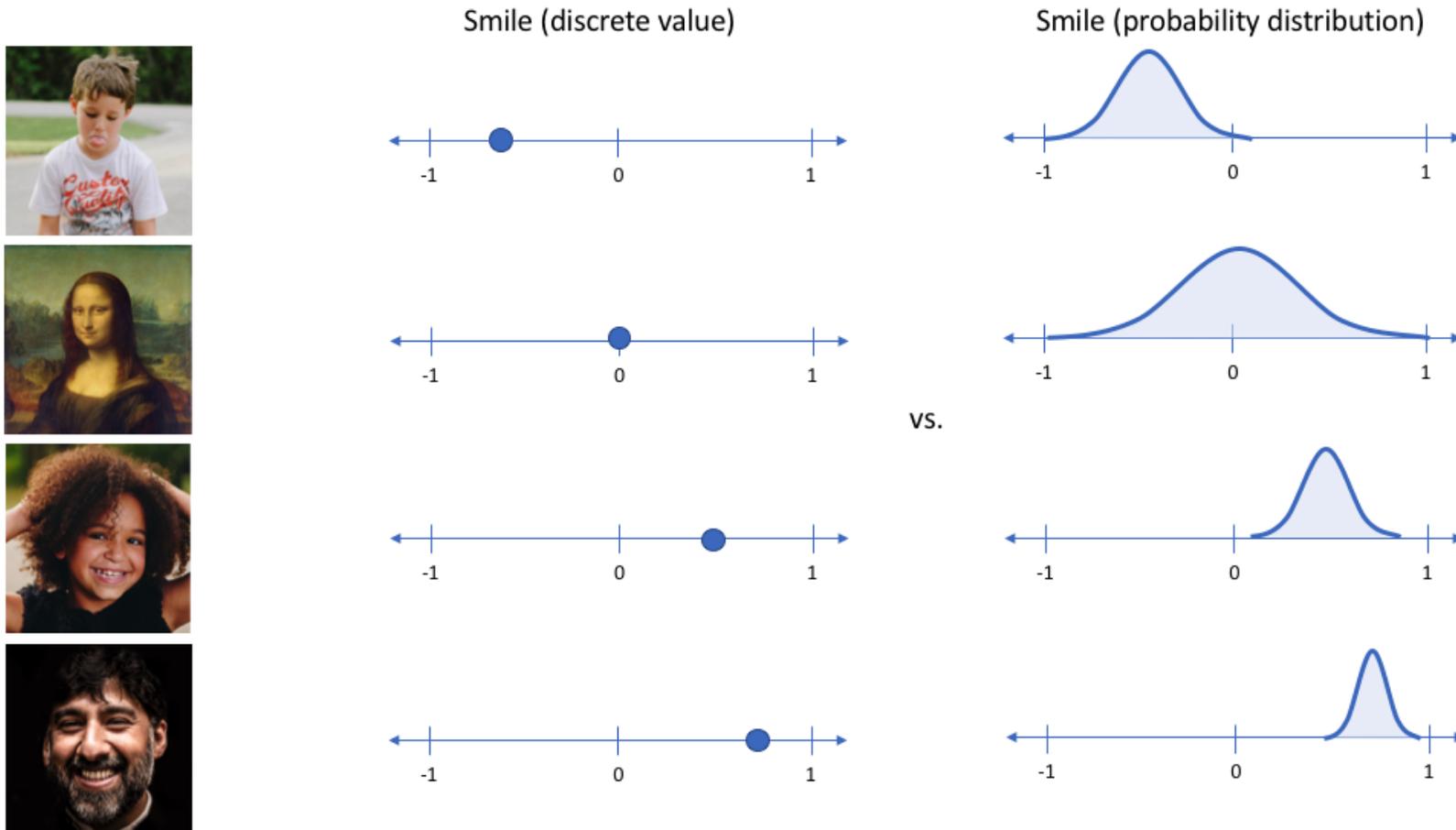
### 3.3 Variational autoencoders

- Em lugar de mapear a entrada diretamente numa variável latente, o que se busca é mapear a entrada numa distribuição normal para a variável latente, com média e variância.
- As figuras desta seção têm como fonte:

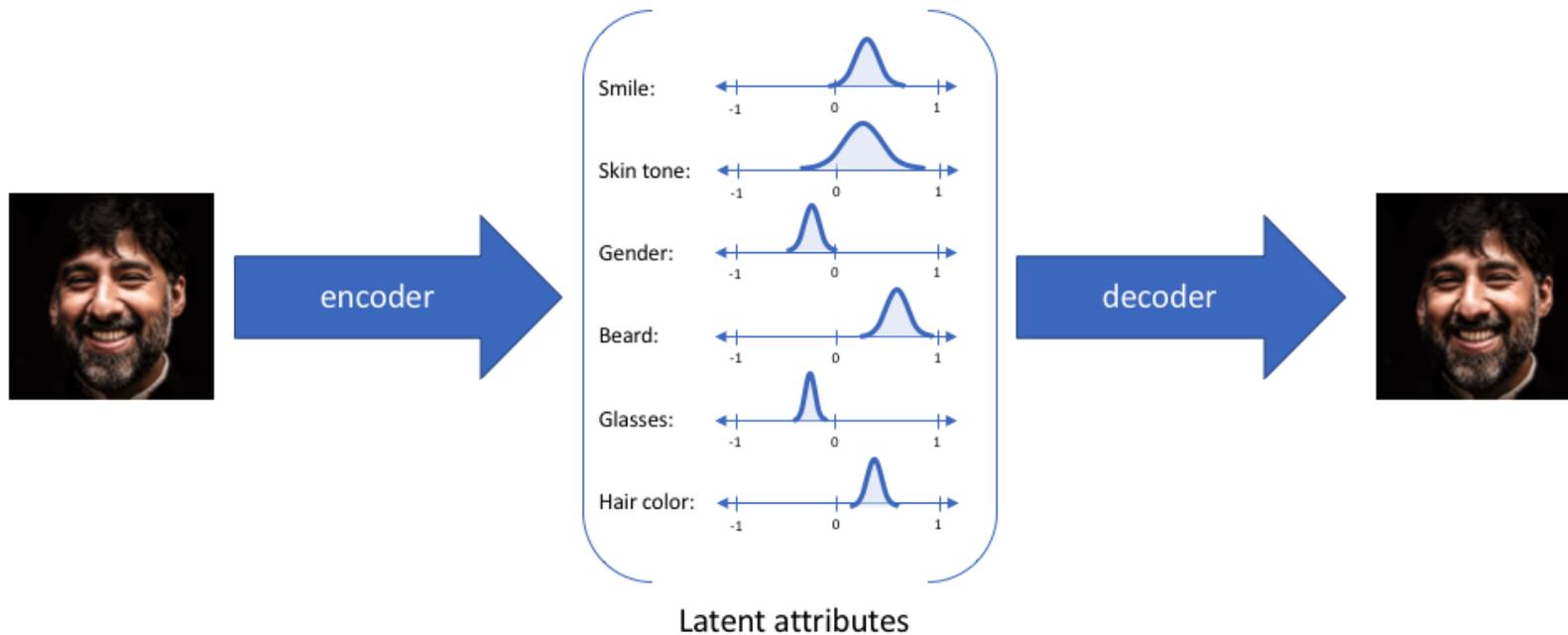
<https://www.jeremyjordan.me/variational-autoencoders/>



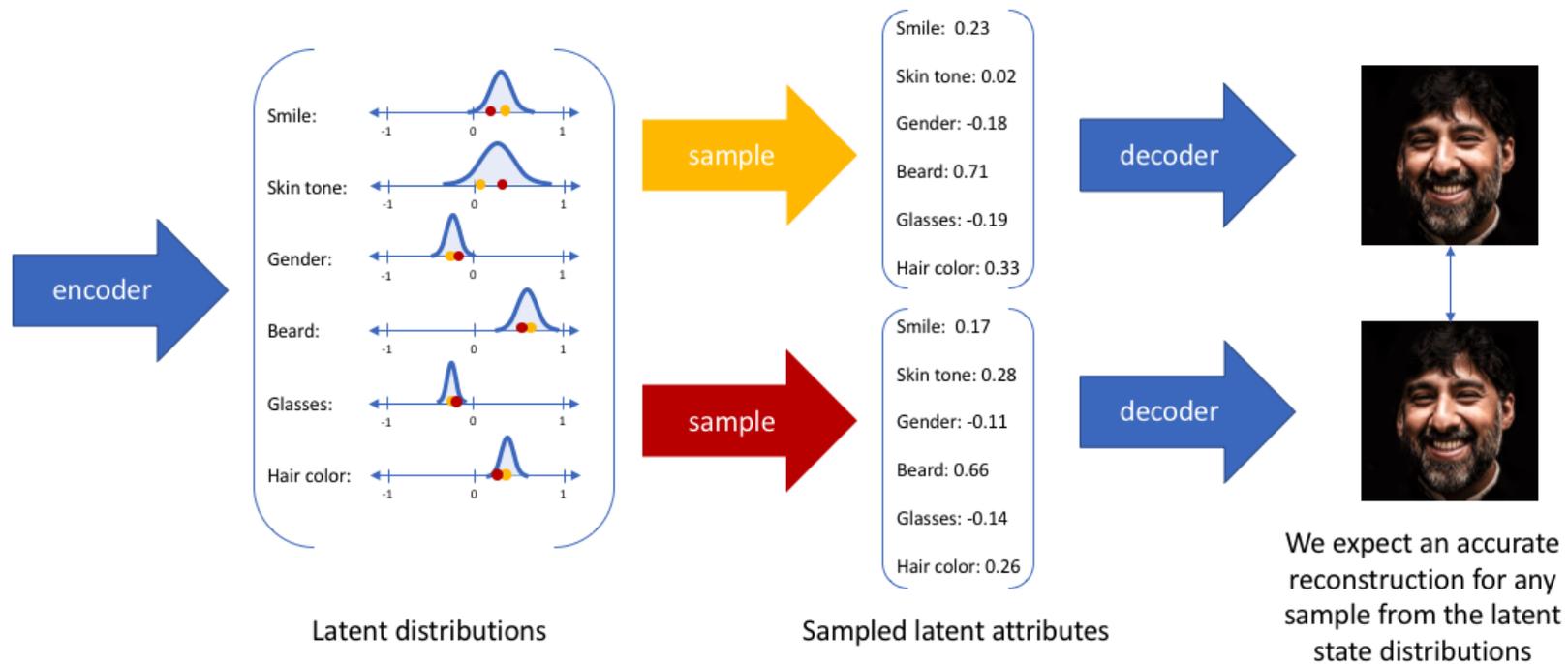
Caso com variáveis latentes determinísticas: autoencoder tradicional



- É possível, assim, definir a distribuição de probabilidade de características fundamentais existentes nos dados de treinamento do autoencoder.



- As variáveis latentes agora são funções densidade de probabilidade gaussianas.



- Como o decodificador é treinado já considerando a etapa de amostragem, as reconstruções tendem a ser acuradas mesmo para amostragens distintas (amarela e vermelha na figura acima) das distribuições de variáveis latentes.

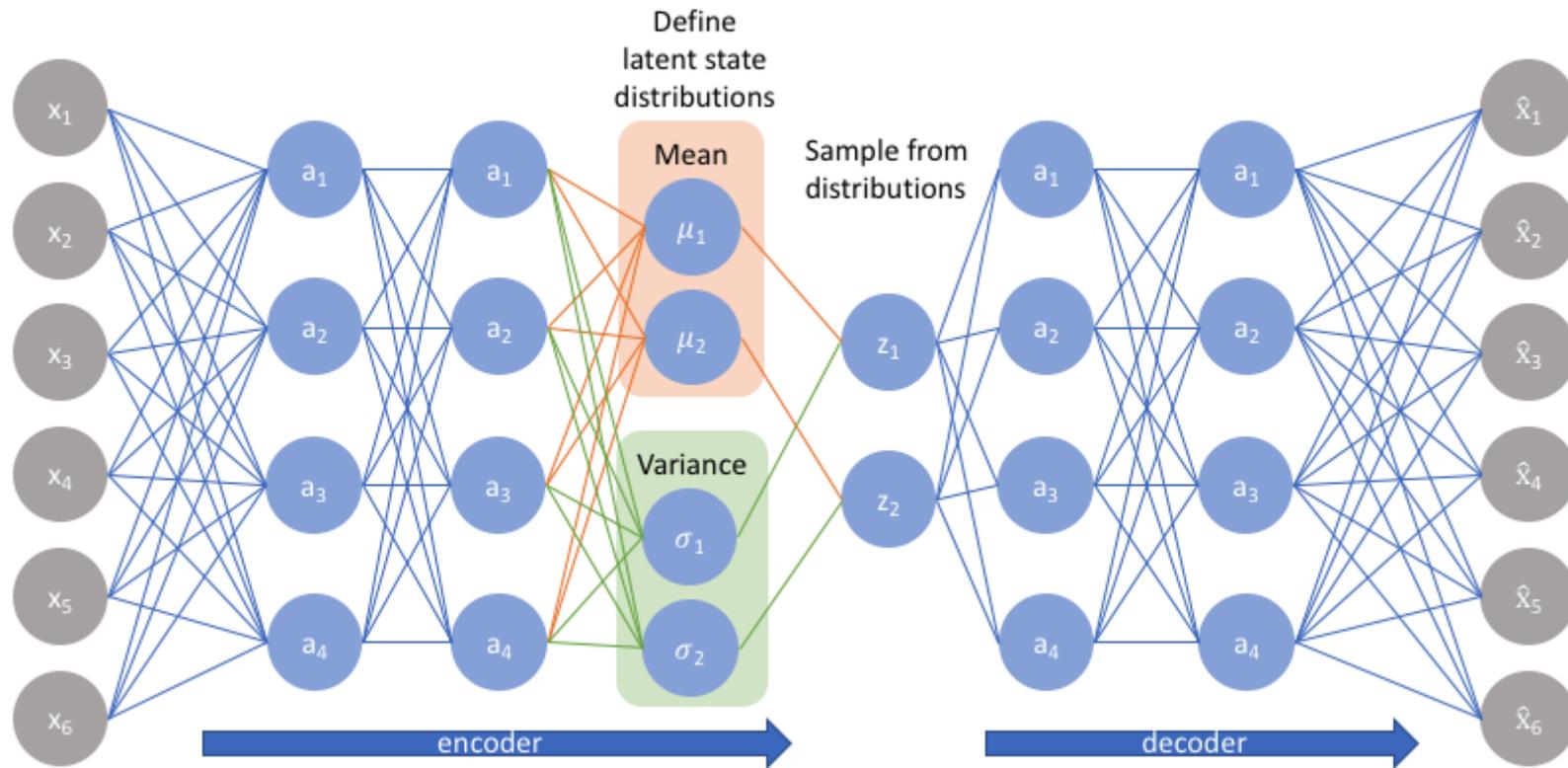


Figura 17 – Configuração do autoencoder variacional considerando duas variáveis latentes.

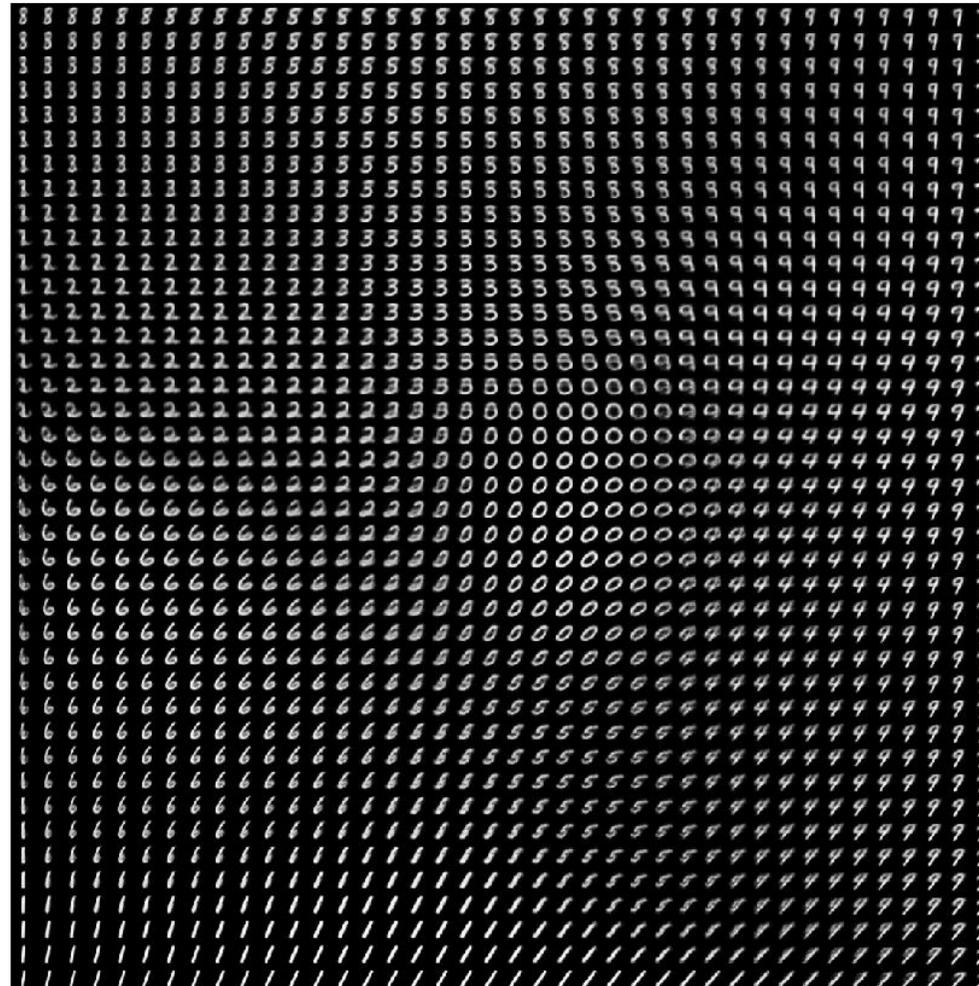


Figura 18 – Visualização das imagens geradas pelo decodificador de um autoencoder variacional com duas variáveis latentes, treinado para os dados MNIST. Foi usada uma variação em grade para as duas variáveis latentes. Cenário equivalente ao da Figura 6.

- A função-objetivo a ser minimizada no treinamento de um autoencoder variacional é a seguinte:

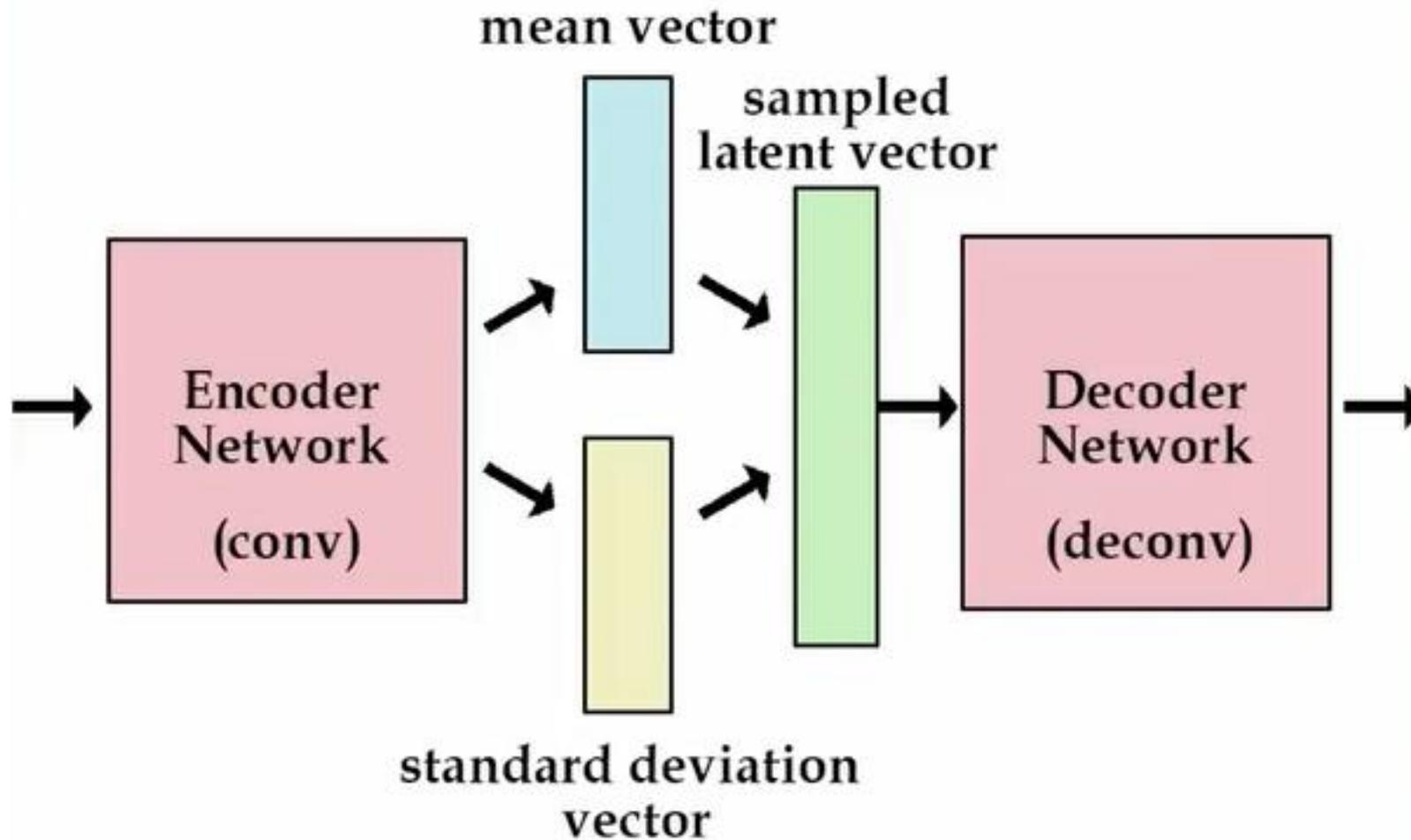
$$L(\theta, \phi; \mathbf{x}, \mathbf{z}) = E_{q_\phi(\mathbf{z}|\mathbf{x})} \left\{ \log [p_\theta(\mathbf{x}|\mathbf{z})] \right\} - D_{KL} [q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})]$$

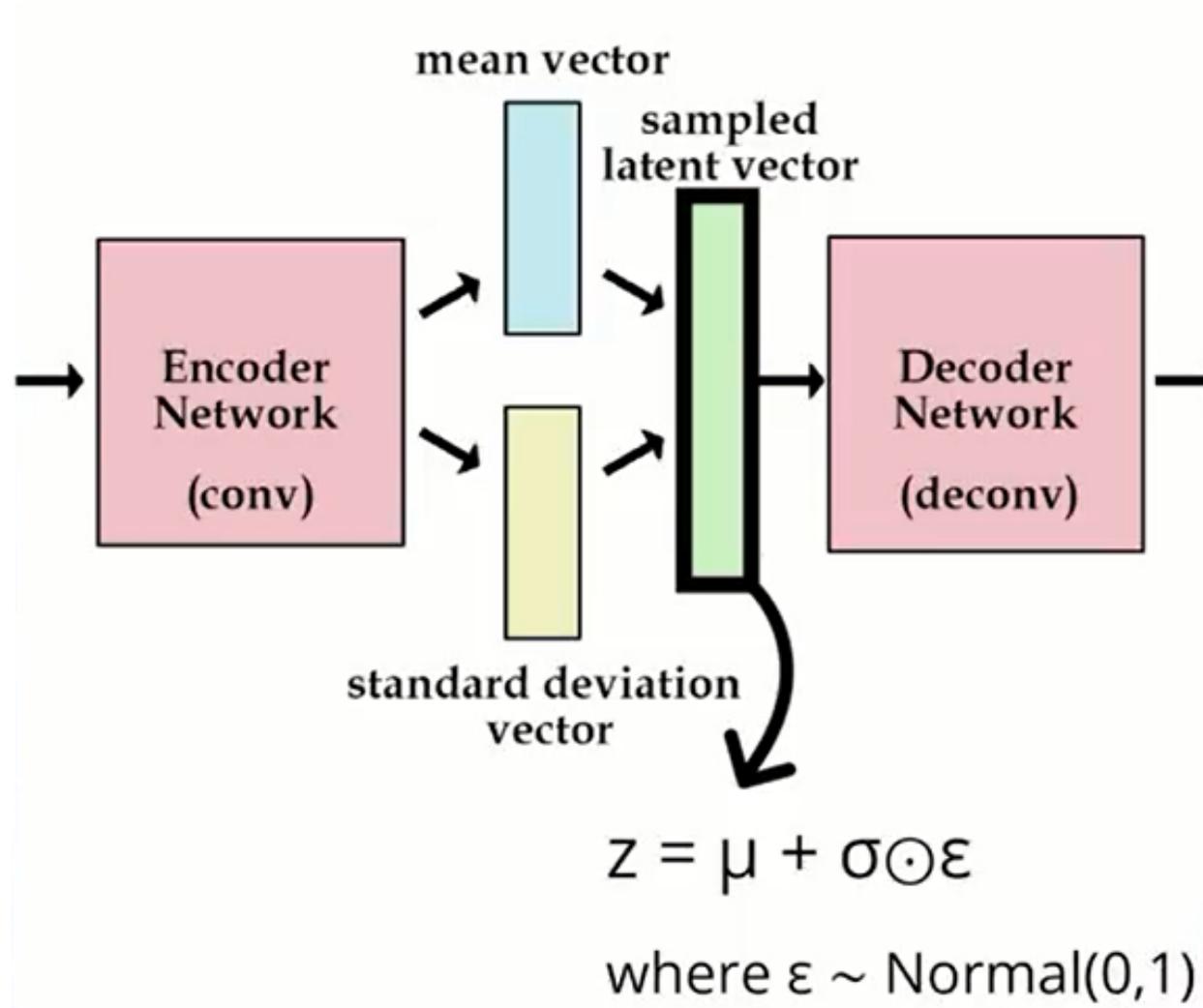
- O primeiro termo penaliza o erro de reconstrução, sendo uma expectativa matemática pelo fato de estarmos operando com distribuições estatísticas agora.
- O segundo termo, que corresponde ao divergente de Kullback-Leibler, penaliza a diferença entre a distribuição resultante e uma distribuição normal com desvio padrão unitário.

### 3.4 O truque da reparametrização

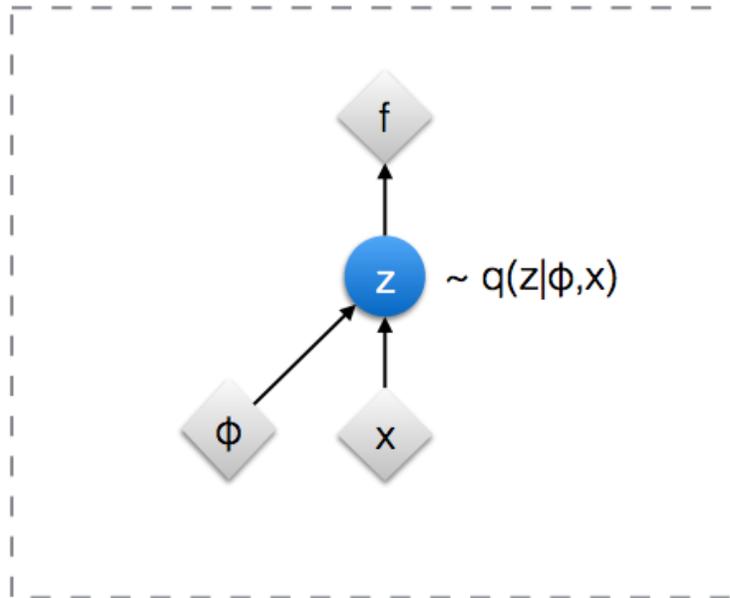
- Como retropropagar o erro através de uma operação de amostragem?
- Para maiores detalhes, consultar:
  - ✓ <https://stats.stackexchange.com/questions/199605/how-does-the-reparameterization-trick-for-vaes-work-and-why-is-it-important>

✓ <http://blog.shakirm.com/2015/10/machine-learning-trick-of-the-day-4-reparameterisation-tricks/>

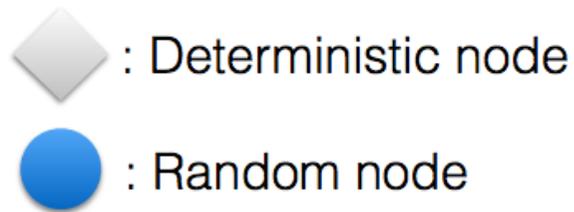
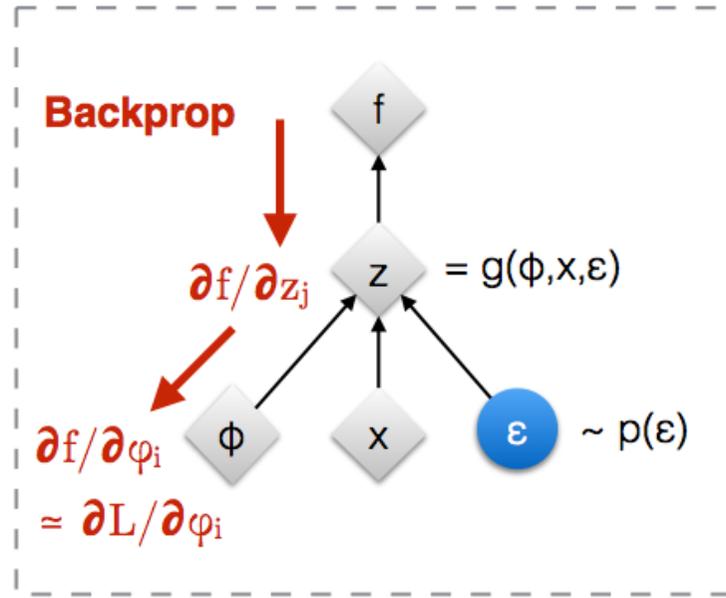




## Original form



## Reparameterised form

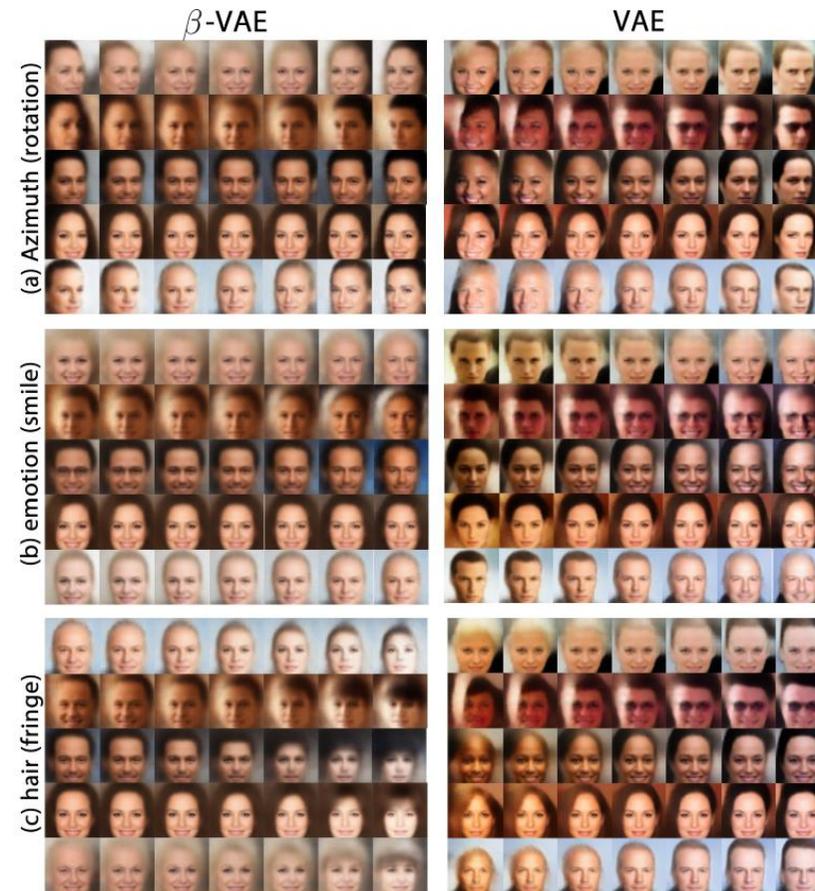


[Kingma, 2013]  
 [Bengio, 2013]  
 [Kingma and Welling 2014]  
 [Rezende et al 2014]

- Para essas referências, favor consultar KINGMA et al. (2015). Recomenda-se assistir ao vídeo em: <https://www.youtube.com/watch?v=9zKuYvjFFS8&t=467s>

### 3.5 Disentangled VAE

$$L(\theta, \phi; \mathbf{x}, \mathbf{z}, \beta) = E_{q_\phi(\mathbf{z}|\mathbf{x})} \left\{ \log [p_\theta(\mathbf{x}|\mathbf{z})] \right\} - \beta D_{KL} [q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})]$$



Fonte: HIGGINS et al. (2017)

## 4 Restricted Boltzmann machines (RBMs)

- Restricted Boltzmann Machines (RBMs) correspondem a grafos probabilísticos não-direcionados contendo uma camada de variáveis observáveis e uma camada de variáveis latentes, também chamadas de extratores de características.
- RBMs estão entre os mais comuns blocos construtivos de modelos probabilísticos profundos, sendo capaz de identificar automaticamente padrões existentes nos dados de entrada (HINTON et al., 2006). A composição de várias camadas de RBMs é denominada de *deep belief network*.
- As RBMs têm este nome por representarem uma versão restrita da máquina de Boltzmann, apresentada na Figura 19.
- Não iremos abordar aqui a modelagem e o treinamento de RBMs. Apresentaremos apenas a derivação que leva à distribuição de probabilidade conjunta entre as unidades visíveis e escondidas. Para mais detalhes sobre o treinamento, favor recorrer a FISCHER & IGEL (2014).

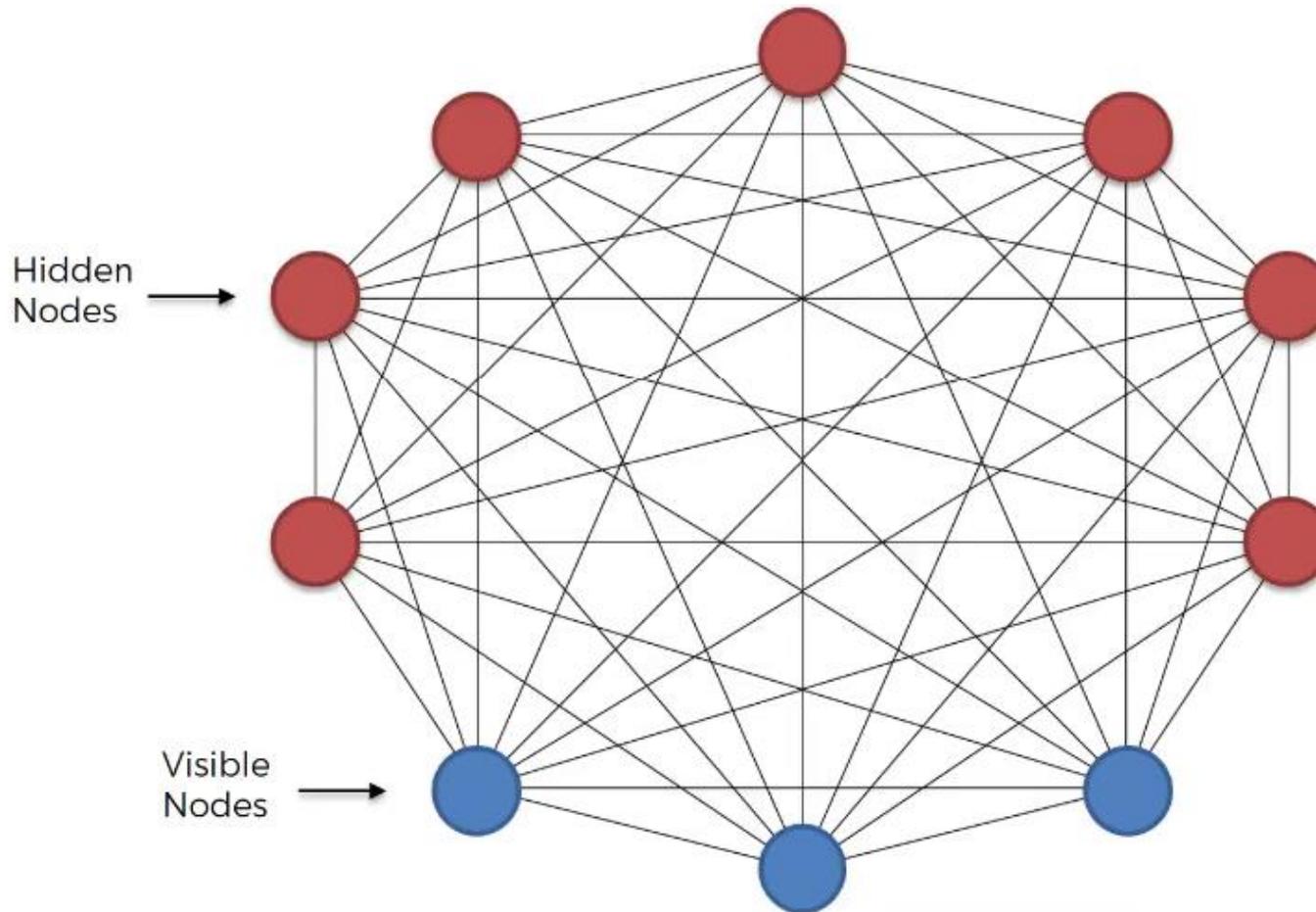


Figura 19 – Máquina de Boltzmann, composta por unidades visíveis e unidades escondidas, sendo que todas as unidades se conectam entre si.

Fonte: <https://medium.com/datadriveninvestor/an-intuitive-introduction-of-boltzmann-machine-8ec54980d789>

- O conjunto de figuras a seguir foi obtido de:

<https://www.youtube.com/watch?v=puux7KZQfsE&t=178s>

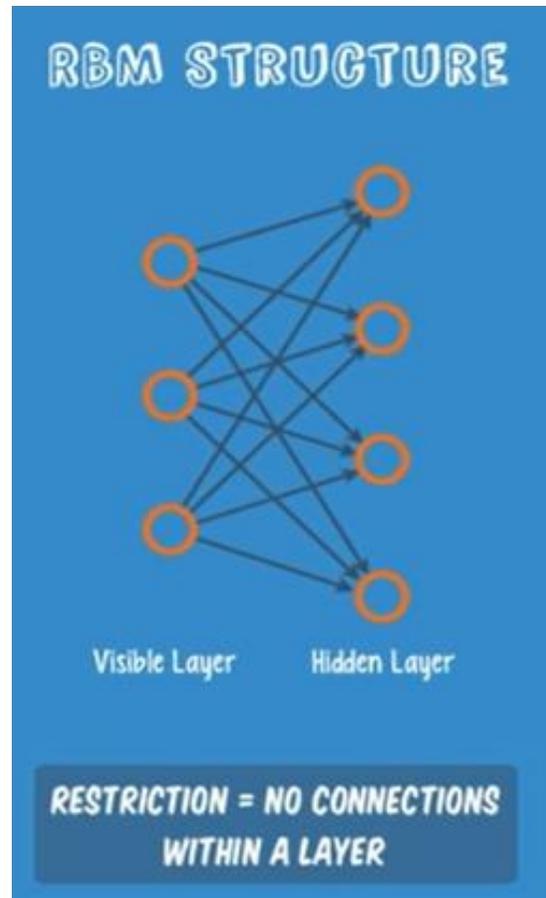
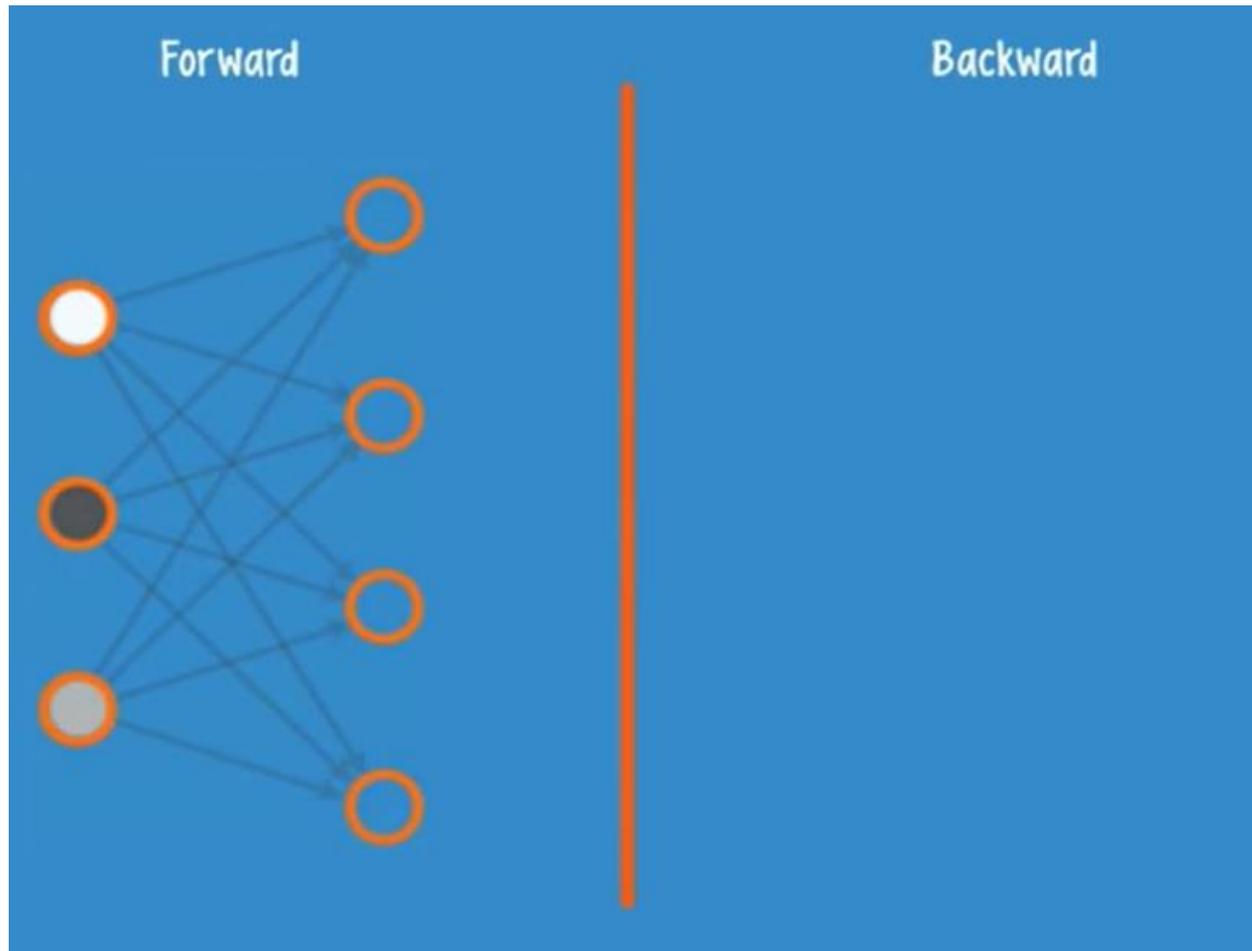
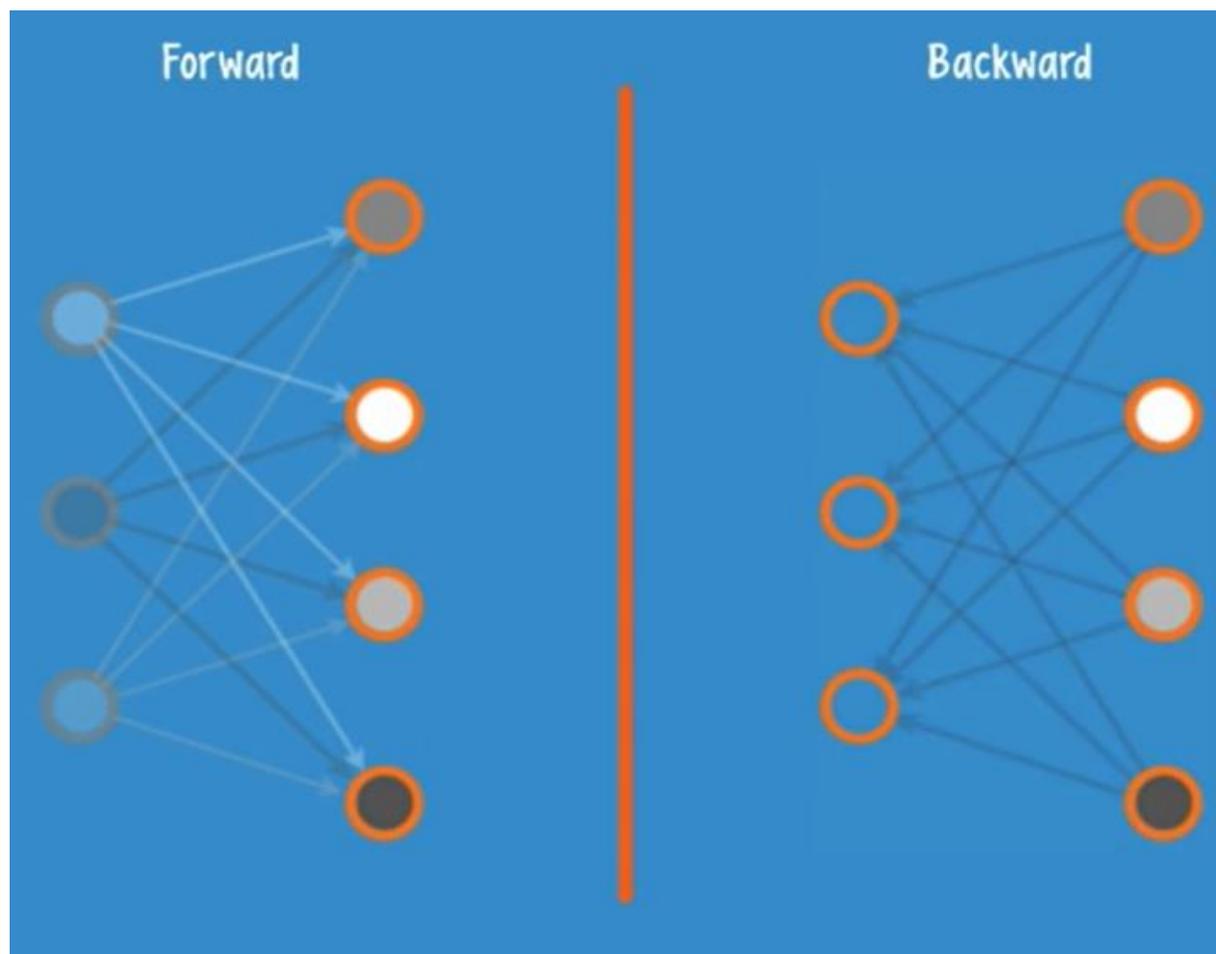


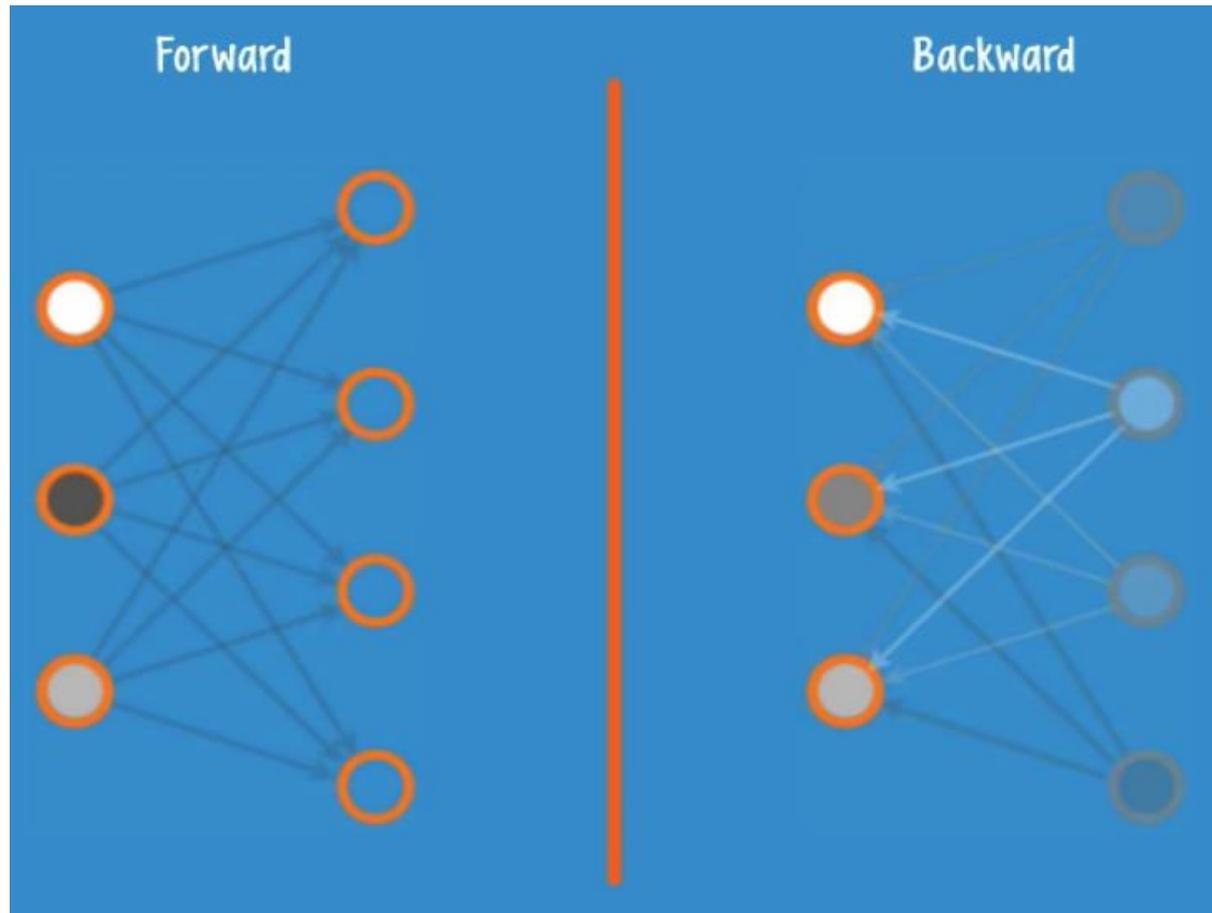
Figura 20 – Máquina de Boltzmann restrita, com conexões apenas entre unidades visíveis e escondidas.



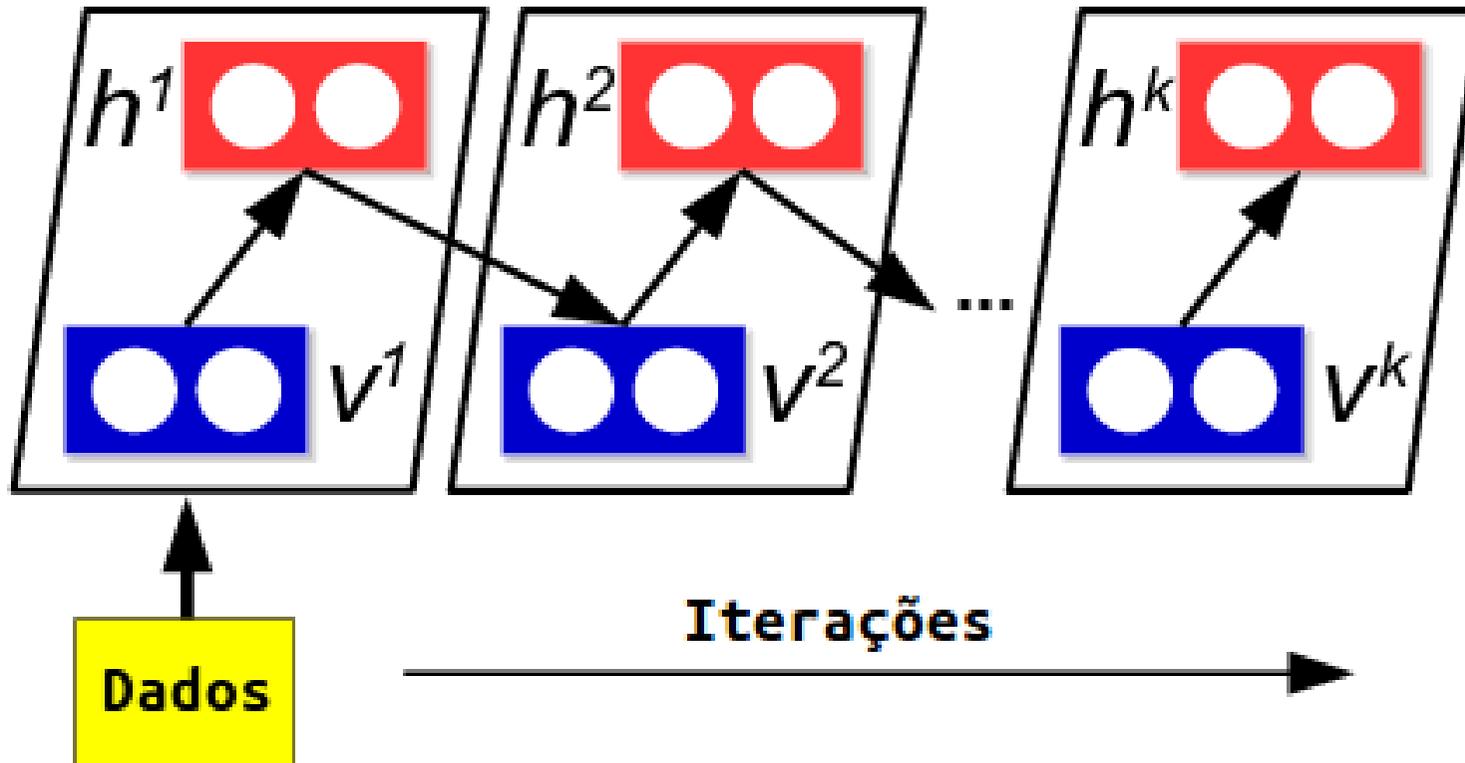
Passo 1: Para cada padrão de entrada, deve-se realizar o passo *forward* e produzir o valor das variáveis latentes.



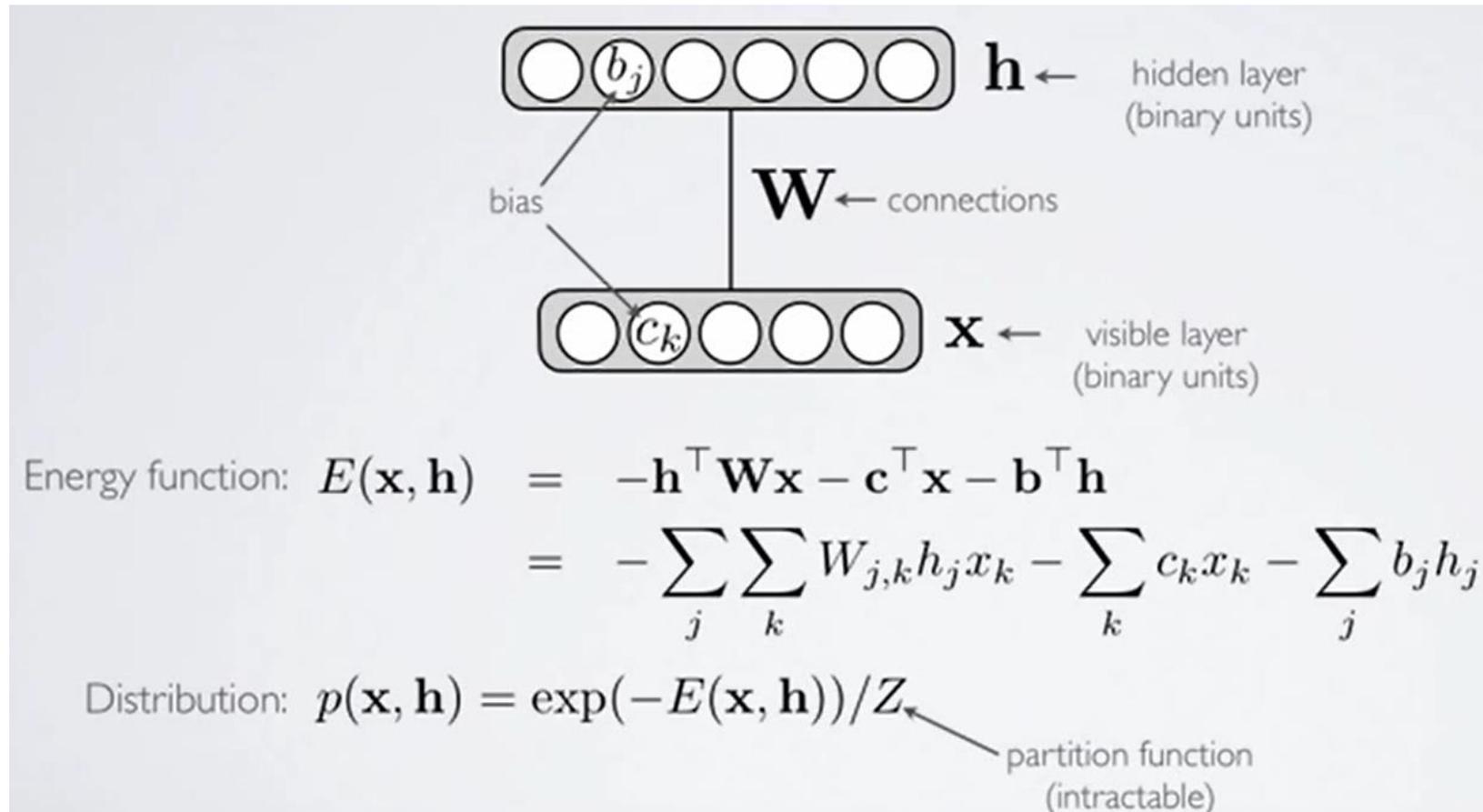
Passo 2: O valor das variáveis latentes do passo *forward* é o ponto de partida para o passo *backward*.



Passo 3: A diferença entre os valores de partida das entradas visíveis e os valores reconstruídos pelo passo backward é usada no ajuste dos pesos sinápticos.



Fonte: <https://matheusfacure.github.io/2017/07/30/RBM/>



Fonte: [https://www.youtube.com/watch?v=lekCh\\_i32iE](https://www.youtube.com/watch?v=lekCh_i32iE)

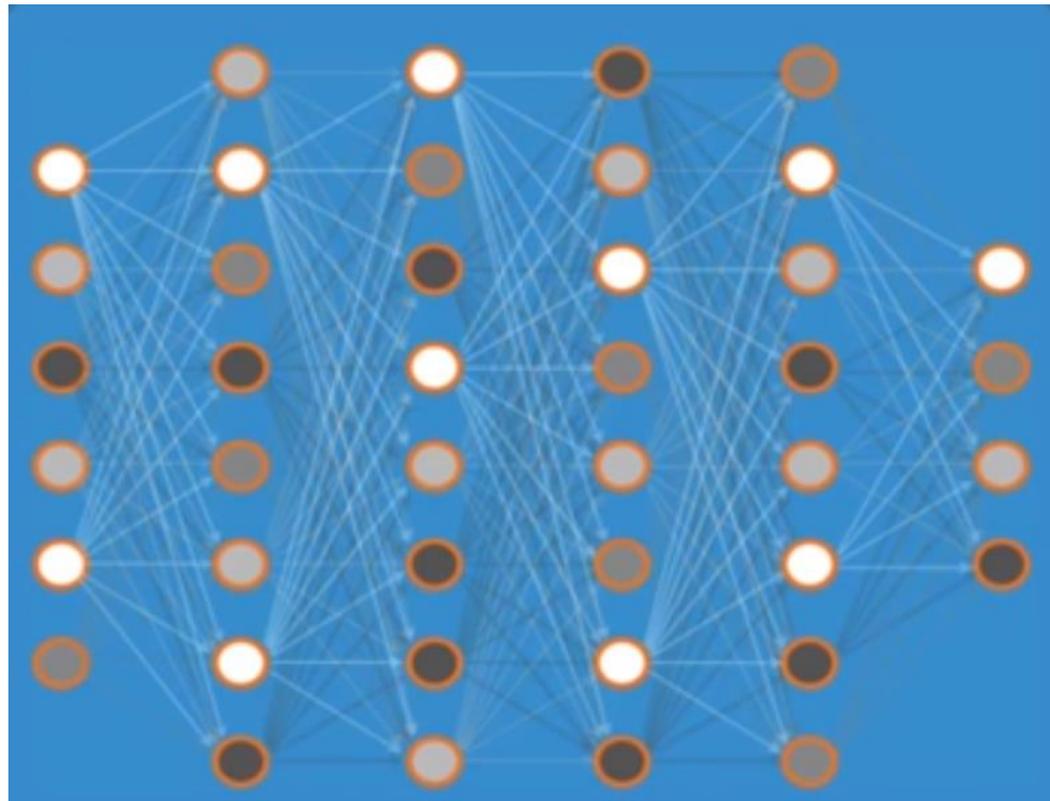
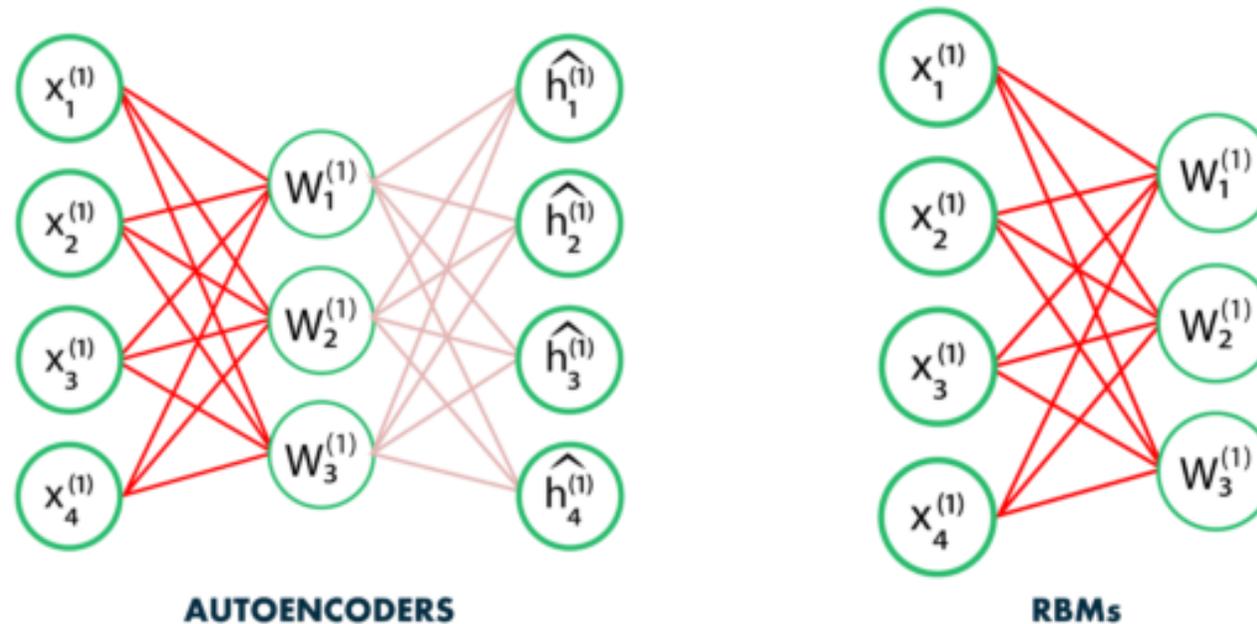


Figura 21 – Cascata de RBMs, formando uma *Deep belief network*, que nada mais é que uma MLP pré-treinada e que pode sofrer um refinamento para ajustes finos.

Fonte: [https://www.youtube.com/watch?v=E2Mt\\_7qked0](https://www.youtube.com/watch?v=E2Mt_7qked0)

## 4.1 RBMs x (V)AEs



Fonte: <https://www.edureka.co/blog/restricted-boltzmann-machine-tutorial/>

- RBMs têm uma fundamentação estatística mais elaborada, que influencia também na técnica de regularização, mas ambas as máquinas buscam o mesmo propósito: codificar as instâncias de entrada de modo que o código gerado possa ser usado para reconstruir a referida instância com erro baixo.

## 5 Alucinação em redes neurais profundas

- As alucinações são caracterizadas quando a rede neural já treinada (não restrita a autoencoders) tem alguns de seus componentes internos salientados, permitindo revelar o seu papel na representação da imagem de entrada. É predominantemente voltada para o tratamento de imagens, as quais são chamadas de alucinações, por envolver estímulos internos da rede neural.

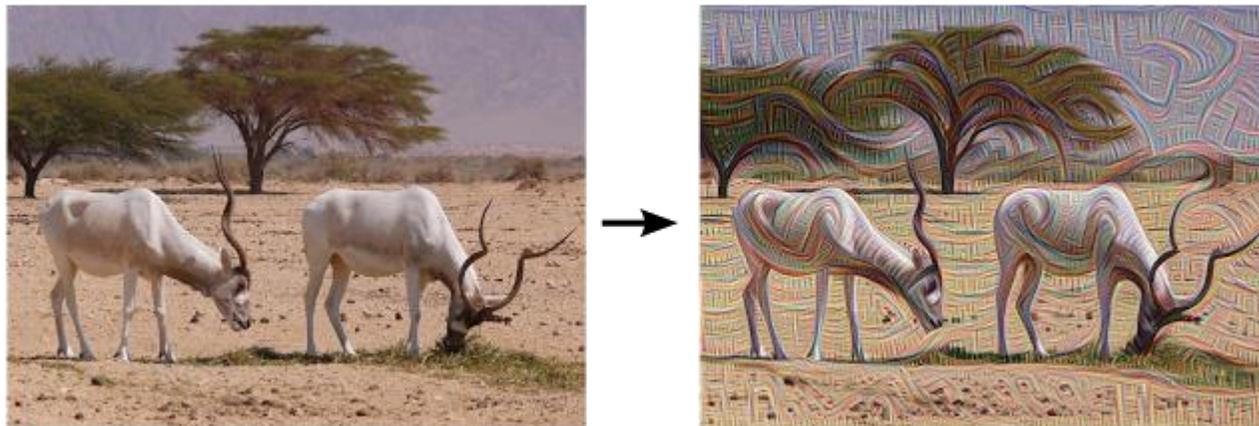


Figura 22 – Efeito de alucinação quando camadas mais próximas da entrada são salientadas.

Fonte: <https://medium.com/hashworks/deep-dreaming-with-deep-learning-487835ebf315>

- As figuras a seguir já envolvem salientar unidades mais distantes da entrada de uma rede profunda já treinada para a extração de atributos de imagens, produzindo alucinações mais complexas e abstratas, envolvendo uma mistura de padrões já aprendidos.



Fonte: <https://medium.com/hashworks/deep-dreaming-with-deep-learning-487835ebf315>



Fonte: <https://medium.com/hashworks/deep-dreaming-with-deep-learning-487835ebf315>

- Essas alucinações permitem constatar como uma rede neural profunda consolidou o aprendizado da representação de imagens durante o treinamento e como esses modelos de classificação podem ser ludibriados.
- Um vídeo contendo resultados recursivos de alucinações se encontra em:

<https://www.youtube.com/watch?v=sh-MQboWJug>

## 5.1 Neural inpainting

- “*Inpainting refers to the art of restoring lost parts of an image and reconstructing them based on the background information.*”

Fonte: <https://medium.com/jamieai/image-inpainting-with-deep-learning-dd8555e56a32>

- De acordo com a definição, portanto, *inpainting* requer que haja um uso intenso do contexto, sendo bastante natural para a mente humana, mas não tão óbvio para uma máquina.
- É evidente que existem muitas outras formas de explorar esse contexto, sem necessariamente recorrer a redes neurais profundas, como autoencoders. Em processamento de imagens, elas se dividem em métodos baseados em difusão, em fragmentação, em código esparsa, em modelos-exemplo, dentre outros. Não vamos entrar em detalhes e certamente cada método tem seus pontos fortes e fracos.
- Vamos nos ater ao tratamento via redes neurais profundas.

- Com redes neurais profundas, existem duas abordagens principais: algum tipo de *denoising autoencoder* e alucinação.
- As propostas via *denoising autoencoder* são mais simples de compreender. Para mais detalhes, consulte XIE et al. (2012). Geralmente são buscadas várias camadas de codificadores em cascata, onde cada um deles é treinado para minimizar a diferença entre o código gerado pela imagem completa e aquele gerado pela imagem com informação faltante. Assim, ao final desta cascata de minimizações, um decodificador pode ser usado para reconstruir uma aproximação da imagem completa, a partir de um código da imagem incompleta já bem mais próximo de um código de uma imagem completa.
- Já no caso de alucinação, a técnica de preenchimento de partes grandes de imagens corrompidas envolve a exploração e regularização da última camada de uma rede neural profunda já treinada, por exemplo, para operar como um

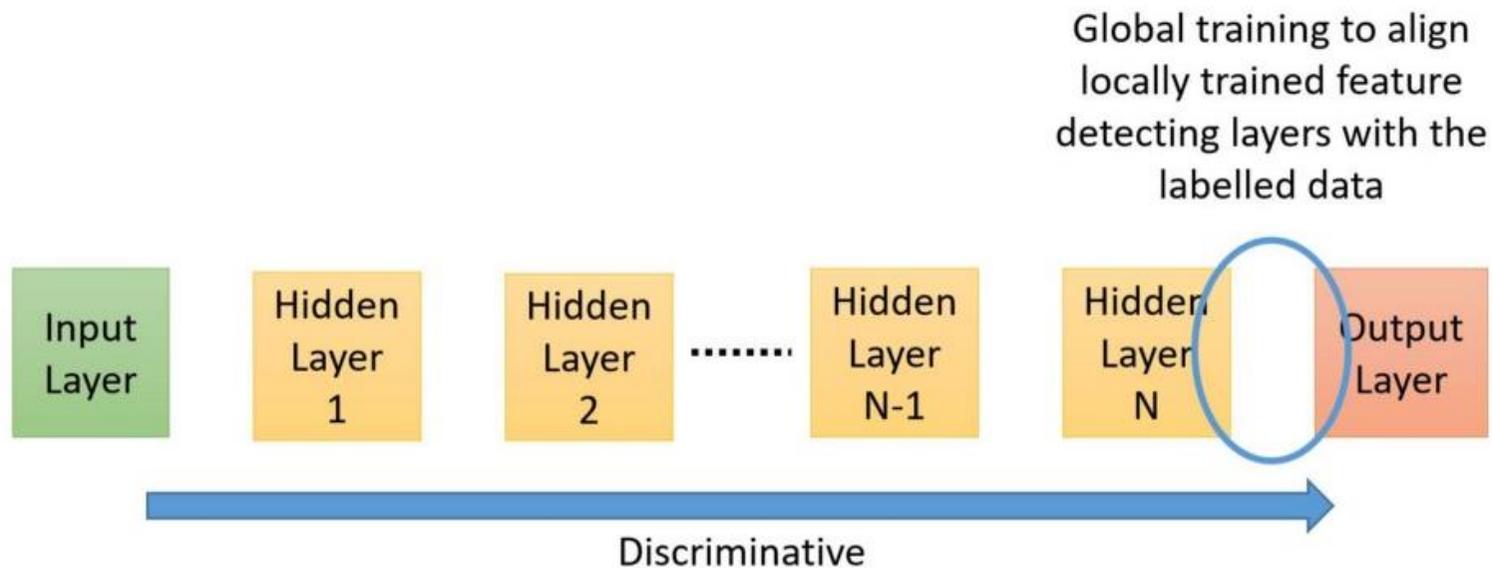
classificador. Esta última camada já deve trabalhar com detalhes e estruturas complexas da imagem de entrada (expressa o ‘manifold já desdobrado’).



Fonte: <https://www.sciencealert.com/nvidia-neural-network-editing-program-for-irregular-inpainting>

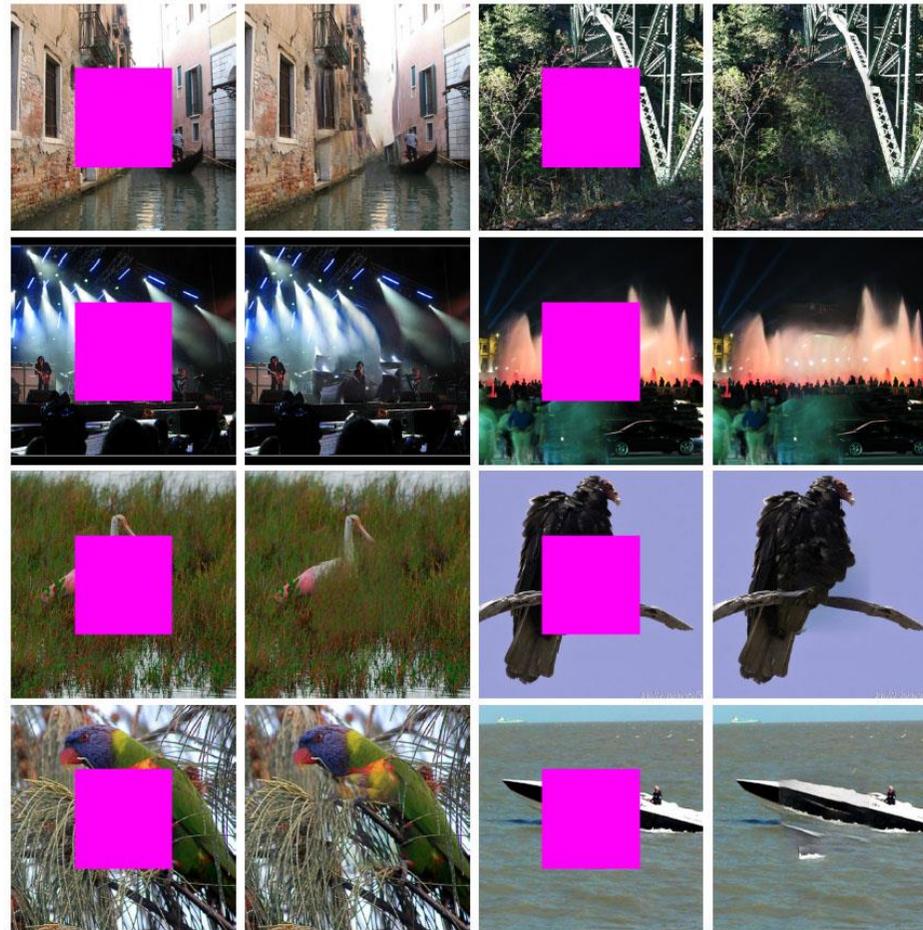
- A reconstrução das partes faltantes da imagem deve recorrer ao conhecimento adquirido pelo classificador, que potencialmente foi treinado a partir de milhões ou até bilhões de imagens pertencentes ao conjunto de treinamento.

- Este conhecimento está condensado nas camadas que aprenderam a representação de imagens. Regularização e uma função de perda adequada devem ser buscadas.

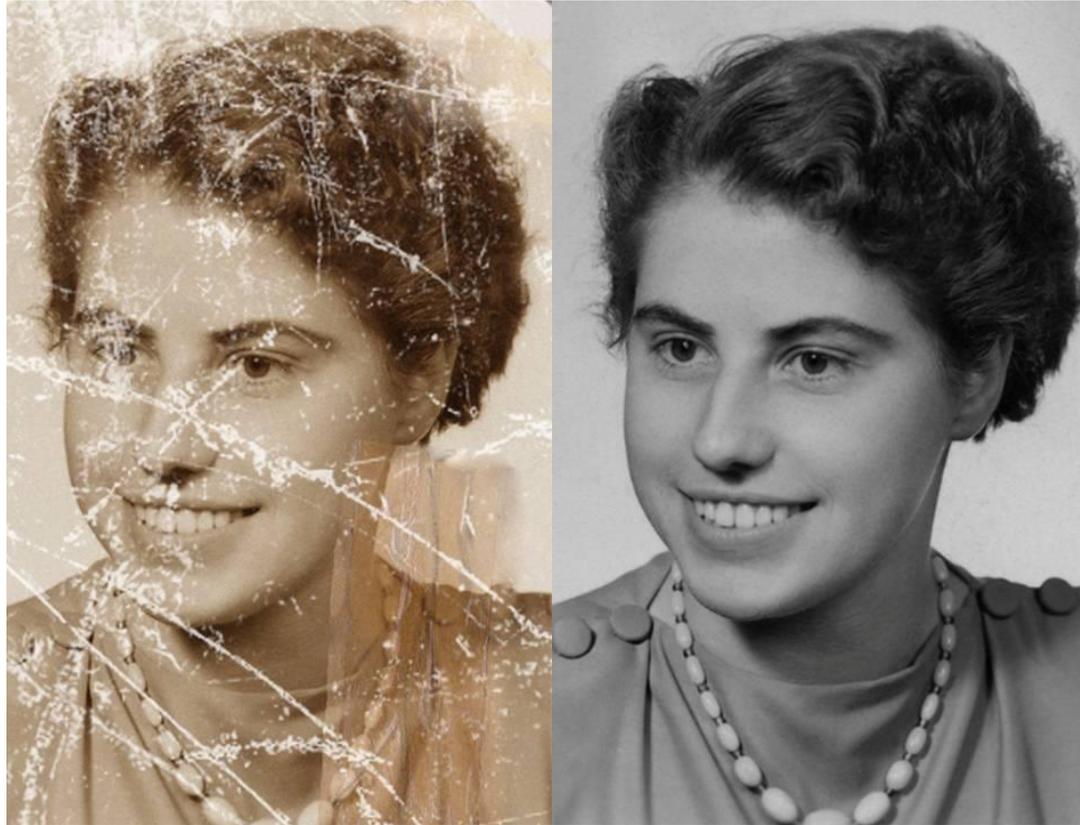


Fonte: <https://medium.com/jamieai/image-inpainting-with-deep-learning-dd8555e56a32>

- Uma vez definida esta função de perda regularizada, por exemplo, enfatizando a preservação de bordas na imagem, técnicas baseadas em gradiente podem ser aplicadas para minimizar a perda, atuando sobre a imagem de entrada.



Fonte: [harryyang.org/inpainting/](http://harryyang.org/inpainting/)



Fonte: <https://medium.com/jamieai/image-inpainting-with-deep-learning-dd8555e56a32>

- Há técnicas alternativas para *inpainting*, incluindo GANs (*Generative Adversarial Networks*), as quais serão ainda abordadas neste curso.

- Tende a ser fácil para o ser humano reconstruir uma seção perdida de uma imagem, pois comparamos o contexto com o conhecimento que temos do mundo. Isso nos permite interpolar e/ou extrapolar as partes ausentes. É isso que se quer implementar com as técnicas de alucinação em redes neurais profundas.



Fonte: <https://medium.com/jamieai/image-inpainting-with-deep-learning-dd8555e56a32>

- O potencial de aplicação dessas técnicas de *inpainting* é elevado:
  - ✓ Restauração de imagens, como nos exemplos já apresentados.
  - ✓ Edição de vídeos, incluindo a possibilidade de retirada de objetos de uma cena ou tratamento de conteúdo sensível.
  - ✓ Eliminação de ruídos de diversas naturezas.
  - ✓ Compressão de conteúdo para desafogar sistemas de *streaming*, reconstruindo o conteúdo completo no destino.
  - ✓ Aplicações com interesse estético e de entretenimento.
- Além disso, essa técnica não se restringe ao tratamento de imagens, podendo ser estendida a textos e sequências sonoras, por exemplo.
- O diferencial adquirido com arquiteturas profundas está na capacidade de lidar com aspectos semânticos mais abstratos, como completar as partes faltantes do rosto de uma pessoa, coerente com o contexto oferecido pela informação disponível.

## 6 Referências bibliográficas

- BADRINARAYANAN, V.; KENDALL, A.; CIPOLLA, R. “SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation”, arXiv:1511.00561v3, 2016.
- BENGIO, Y.; COURVILLE, A.; VINCENT, P. “Representation Learning: A Review and New Perspectives”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 8, pp. 1798-1828, 2013.
- BENGIO, Y.; LAMBLIN, P.; POPOVICI, D.; LAROCHELLE, H. “Greedy Layer-Wise Training of a Deep Network”, Advances in Neural Information Processing Systems 19 (NIPS), 2006.
- BISHOP, C.M. “Pattern Recognition and Machine Learning”, Springer, 2006.
- CAYTON, L. “Algorithms for manifold learning”, Technical Report CS2008-0923, UCSD, 2005.
- CHEN, R.T.Q.; LI, X.; GROSSE, R.; DUVENAUD, D. “Isolating Sources of Disentanglement in VAEs”, arXiv:1802.04942v5, 2019.
- DAVIDSON, T.R.; FALORSI, L.; DE CAO, N.; KIPF, T.; TOMCZAK, J.M. “Hyperspherical Variational Auto-Encoders”, arXiv:1804.00891v2, 2018.

- FISCHER, A.; IGEL, C. “Training Restricted Boltzmann Machines: An Introduction”, *Pattern Recognition*, vol. 47, pp. 25-39, 2014.
- GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. “Deep Learning”, MIT Press, 2016.
- GUO, X.; LIU, X.; ZHU, E.; YIN, J. “Deep Clustering with Convolutional Autoencoders”, *ICONIP*, pp. 373-382, 2017.
- HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J.H. “The Elements of Statistical Learning: Data Mining, Inference, and Prediction”, Springer, 2009.
- HE, K.; ZHANG, X.; REN, S.; SUN, J. “Deep Residual Learning for Image Recognition”, arXiv:1512.03385v1, 2015.
- HIGGINS, I.; MATTHEY, L.; PAL, A.; BURGESS, C.; GLOROT, X.; BOTVINICK, M.; MOHAMED, S.; LERCHNER, A. “ $\beta$ -VAE: Learning Basic Visual Concepts with a Constrained Variational Framework”, *ICLR*, 2017.
- HINTON, G.E. “A Practical Guide to Training Restricted Boltzmann Machines”, *UTML TR* 2010-003, 2010.
- HINTON, G.E.; OSINDERO, S.; TEH, Y.W. “A fast learning algorithm for deep belief nets”, *Neural Computation*, vol. 18, no. 7, pp. 1527-1554, 2006.

KINGMA, D.P.; SALIMANS, T.; WELLING, M. “Variational Dropout and Local Reparameterization Trick”, Advances in Neural Information Processing Systems 28 (NIPS), pp. 2575-2583, 2015.

NARAYANAN, H.; MITTER, S. “Sample complexity of testing the manifold hypothesis”, Advances in Neural Information Processing Systems 23 (NIPS), 2010.

VAN DER MAATEN, L.; HINTON, G. “Visualizing Data using t-SNE”, Journal of Machine Learning Research, vol. 9, pp. 2579-2605, 2008.

VINCENT, P.; LAROCHELLE, H.; LAJOIE, I.; BENGIO, Y. & MANZAGOL, P.-A. “Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion”, Journal of Machine Learning Research, vol. 11, pp. 3371-3408, 2010.

XIE, J.; XU, L.; CHEN, E. “Image Denoising and Inpainting with Deep Neural Networks”, Advances in Neural Information Processing Systems 25 (NIPS), 2012.

Dicas para a implementação de autoencoders:

<https://blog.keras.io/building-autoencoders-in-keras.html>