

## **Lecture 06a**

Introducción a los codificadores  
automáticos Autoencoders

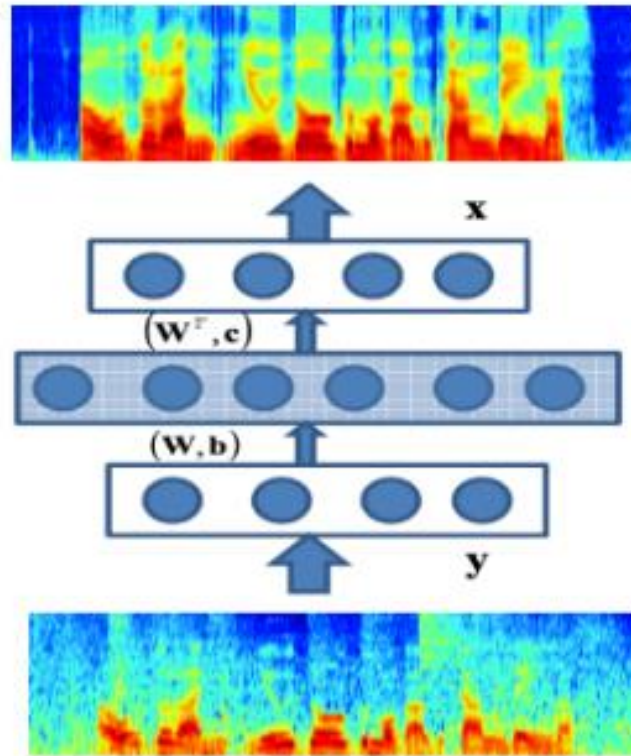
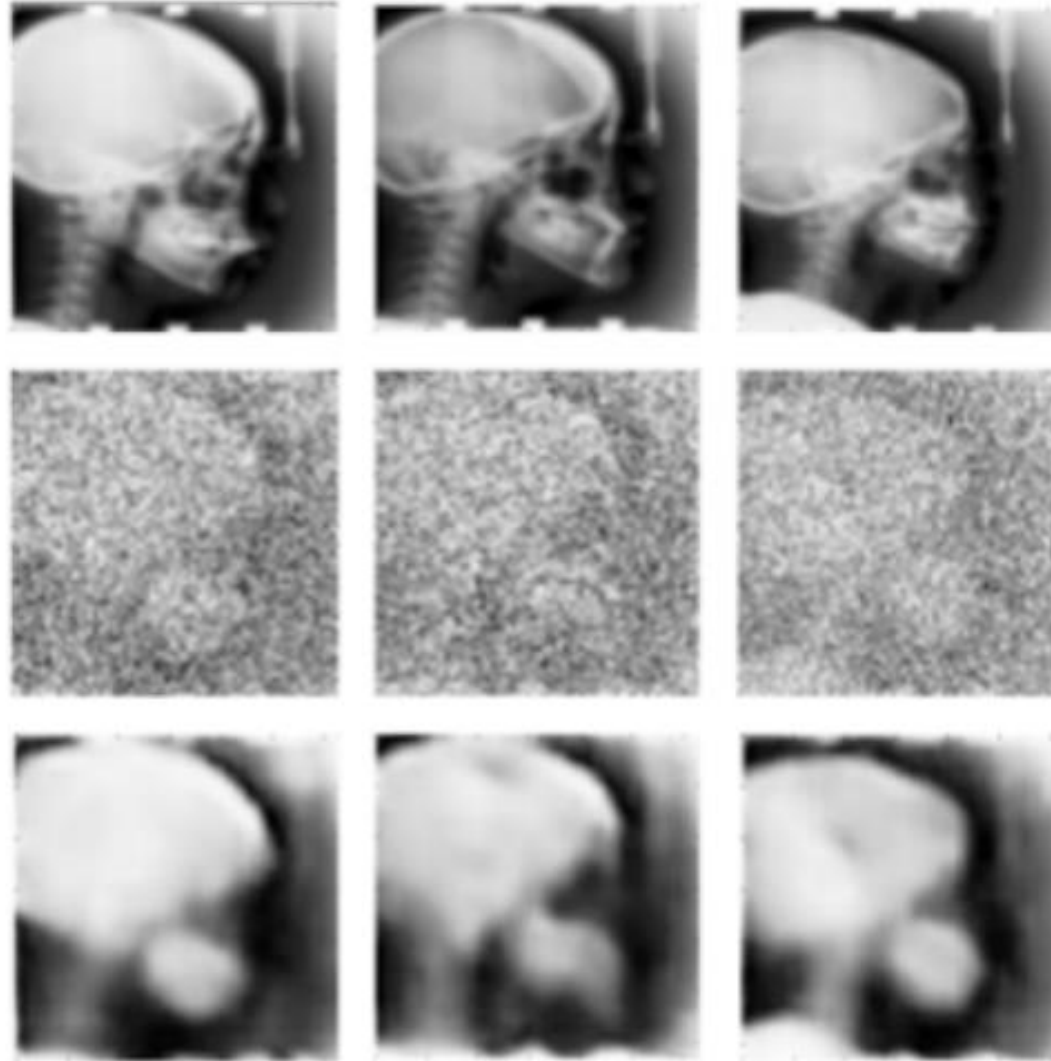


Figure 1: Training neural autoencoder with noisy-clean speech pairs.



Gondara, L. (2016, December). Medical image denoising using convolutional denoising autoencoders. In 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW) (pp. 241-246). IEEE.

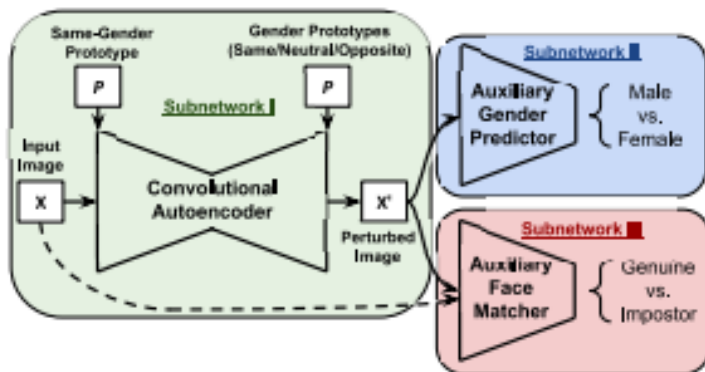
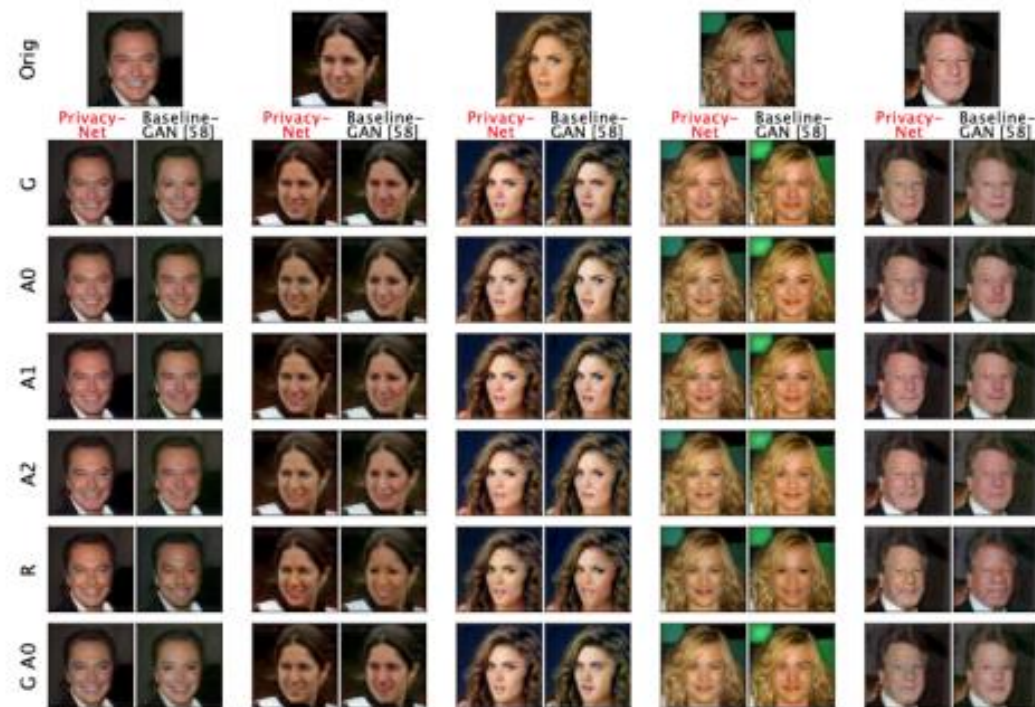


Figure 1. Schematic representation of the semi-adversarial neural network architecture designed to derive perturbations that are able to confound gender classifiers while still allowing biometric matchers to perform well. The overall network consists of three sub-components: a convolutional autoencoder (subnetwork I), an auxiliary gender classifier (subnetwork II), and an auxiliary matcher (subnetwork III).

Vahid Mirjalili, Sebastian Raschka, Anoop Namboodiri, and Arun Ross (2018) Semi-adversarial networks: Convolutional autoencoders for imparting privacy to face images. Proc. of 11th IAPR International Conference on Biometrics (ICB 2018), Gold Coast, Australia



"About half (52%) of U.S. adults said they decided recently not to use a product or service because they were worried about how much personal information would be collected about them."

<https://www.pewresearch.org/fact-tank/2020/04/14/half-of-americans-have-decided-not-to-use-a-product-or-service-because-of-privacy-concerns/>

# Lecture 06a - Introducción a los codificadores automáticos

1. Reducción de dimensionalidad
2. Codificadores automáticos completamente conectados
3. Codificadores automáticos convolucionales
4. Otros tipos de codificadores automáticos

# Extracción de características y reducción de dimensionalidad

- 1. Reducción de dimensionalidad**
2. Codificadores automáticos completamente conectados
3. Codificadores automáticos convolucionales
4. Codificadores automáticos convolucionales en PyTorch
5. Otros tipos de codificadores automáticos

# Aprendizaje sin supervisión

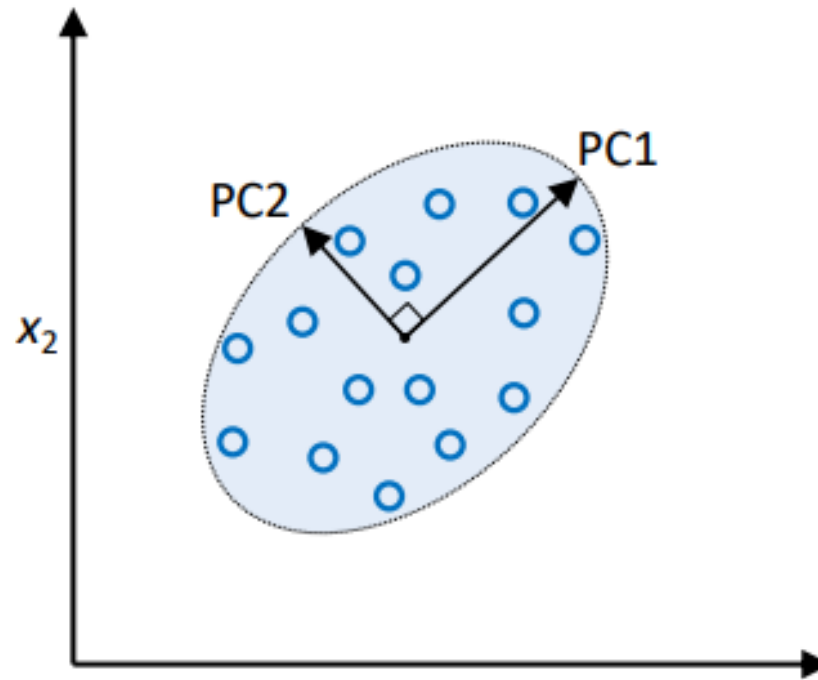
Trabajar con conjuntos de datos sin considerar la variable de destino

Algunas aplicaciones y objetivos:

- Encontrar estructuras ocultas en los datos
- Compresión de datos
- Agrupación
- Recuperar objetos similares
- Análisis exploratorio de datos
- Generando nuevos ejemplos

# Análisis de componentes principales (ACP o PCA por sus siglas en inglés)

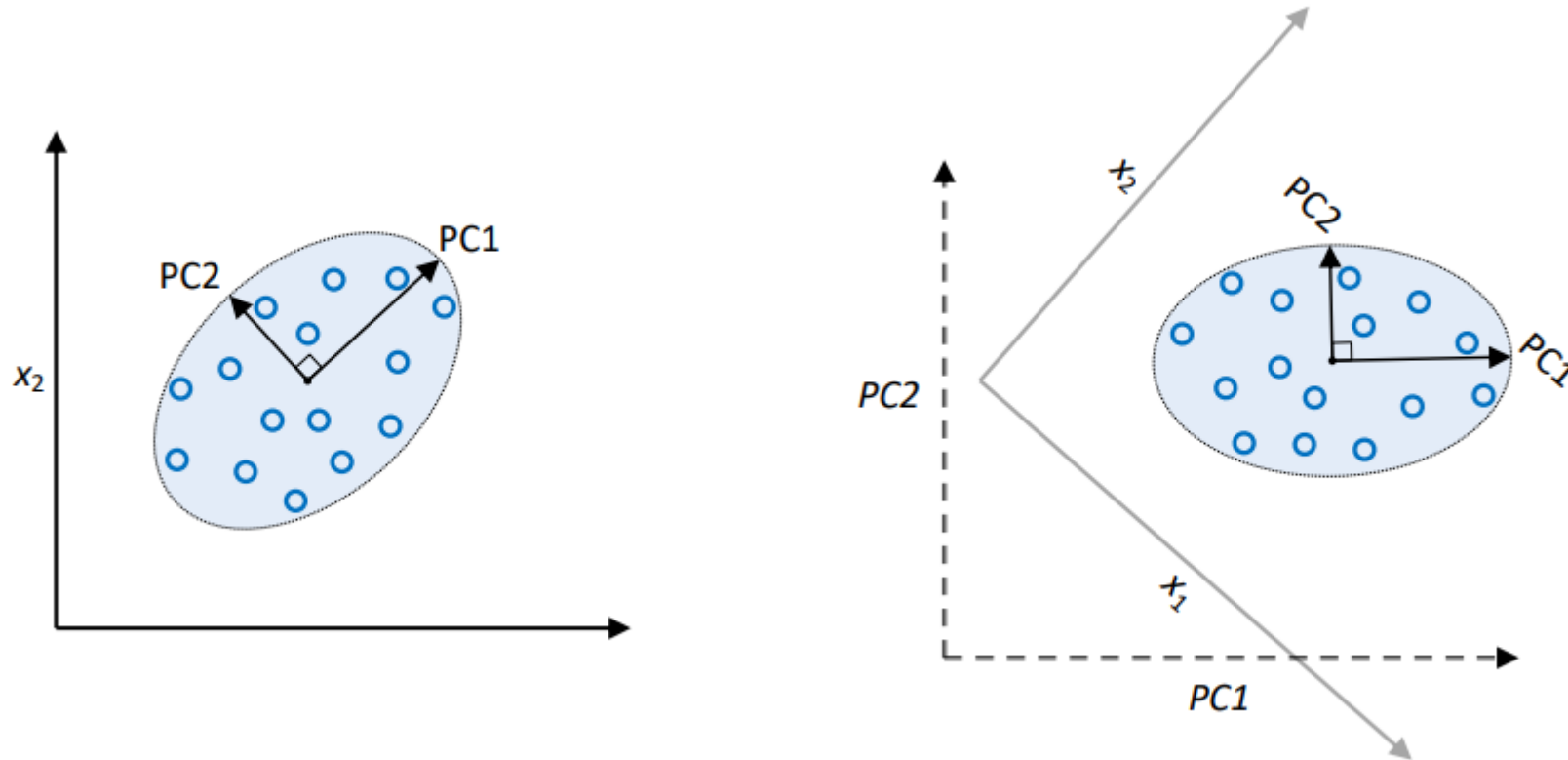
1) Encontrar direcciones de mayor varianza





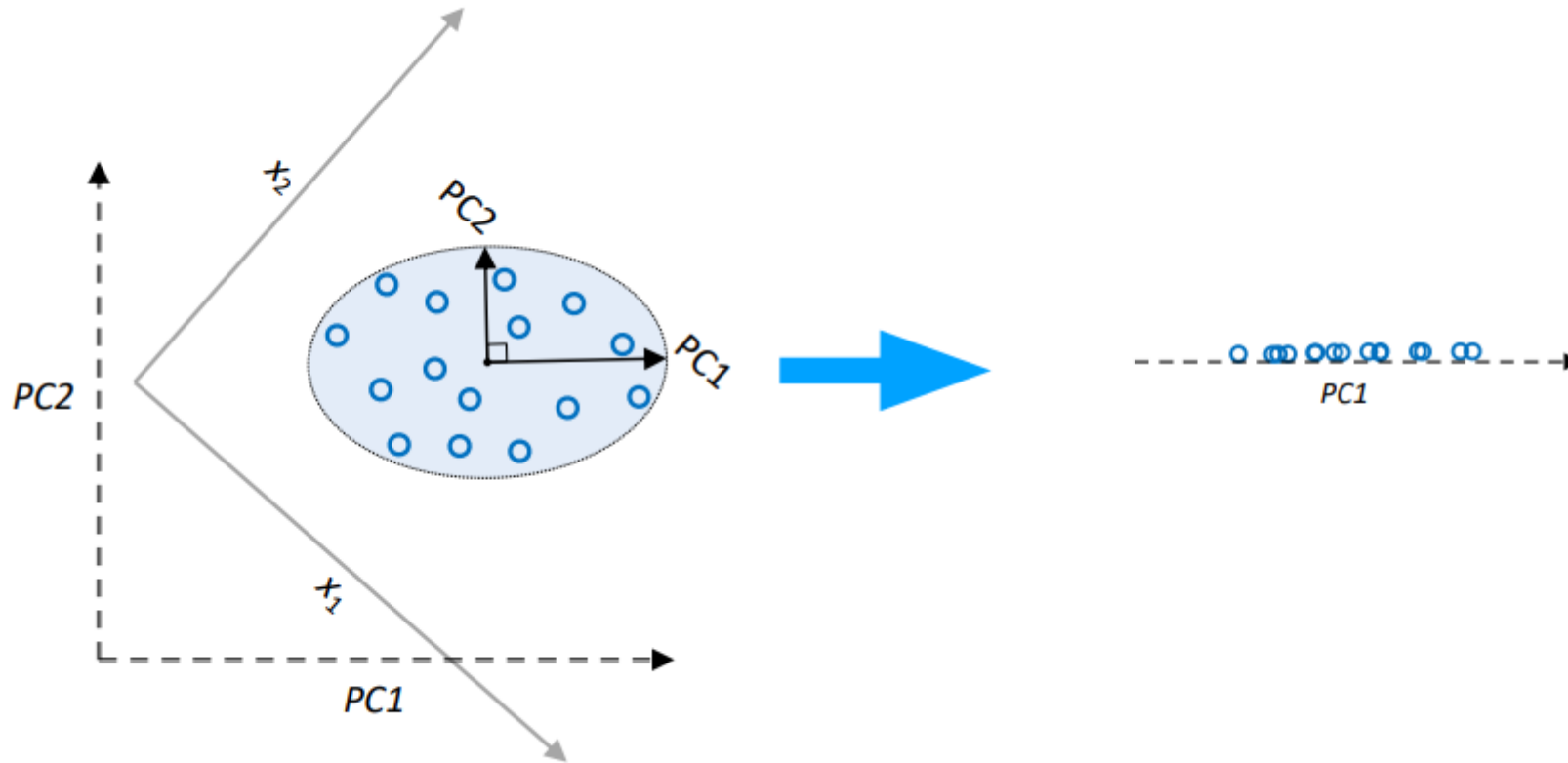
# Análisis de componentes principales (PCA)

2) Transformar características en direcciones de máxima variación.



# Análisis de componentes principales (PCA)

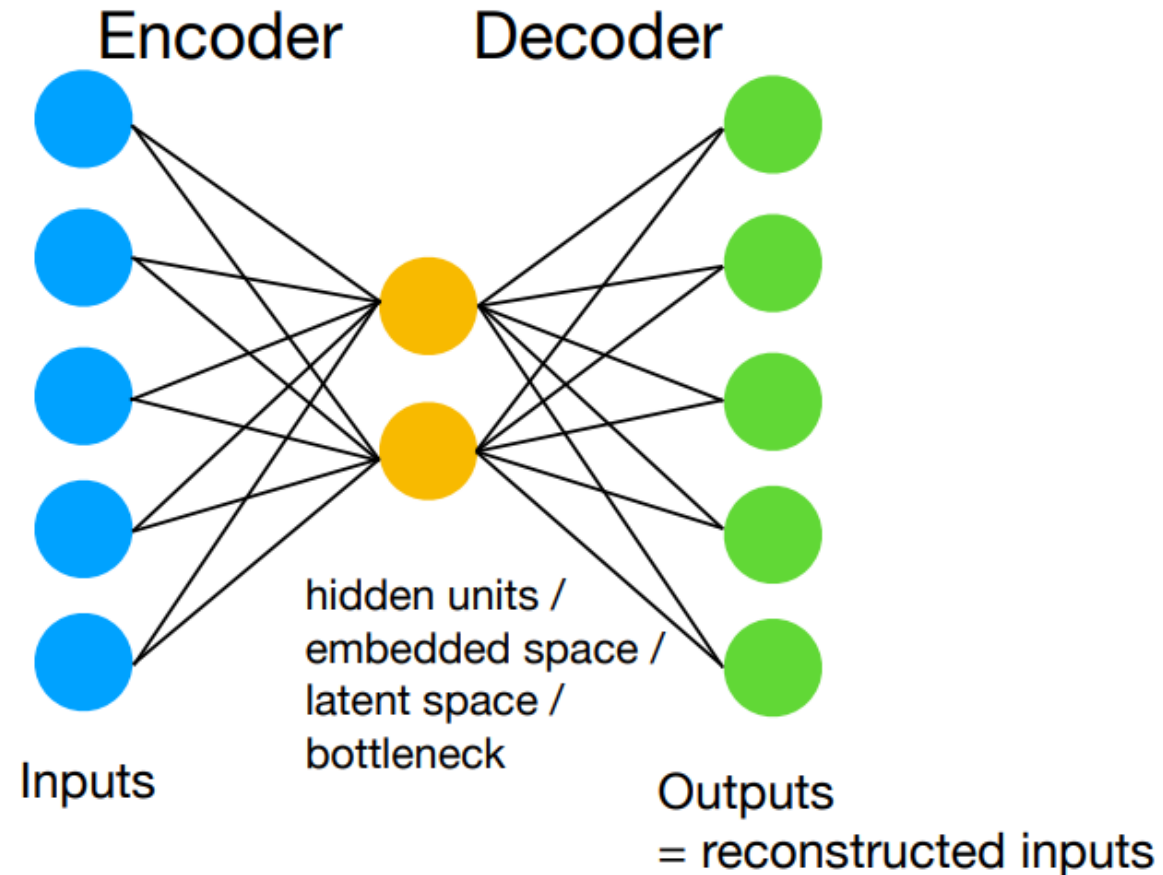
- 3) Por lo general, considere el subconjunto de vectores propios de mayor varianza (reducción de dimensionalidad)



# Un perceptrón multicapa en forma de reloj de arena

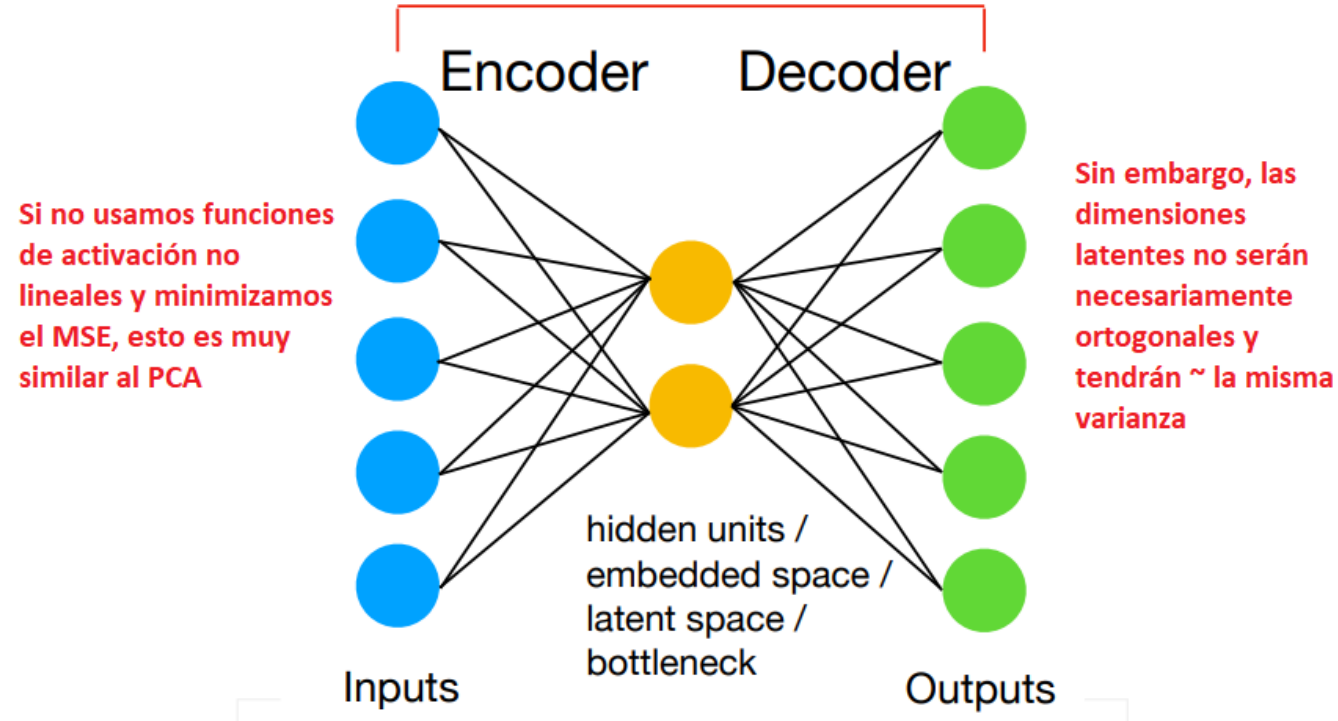
1. Reducción de dimensionalidad
2. **Codificadores automáticos completamente conectados**
3. Codificadores automáticos convolucionales
4. Codificadores automáticos convolucionales en PyTorch
5. Otros tipos de codificadores automáticos

# Un codificador automático básico completamente conectado (perceptrón multicapa)

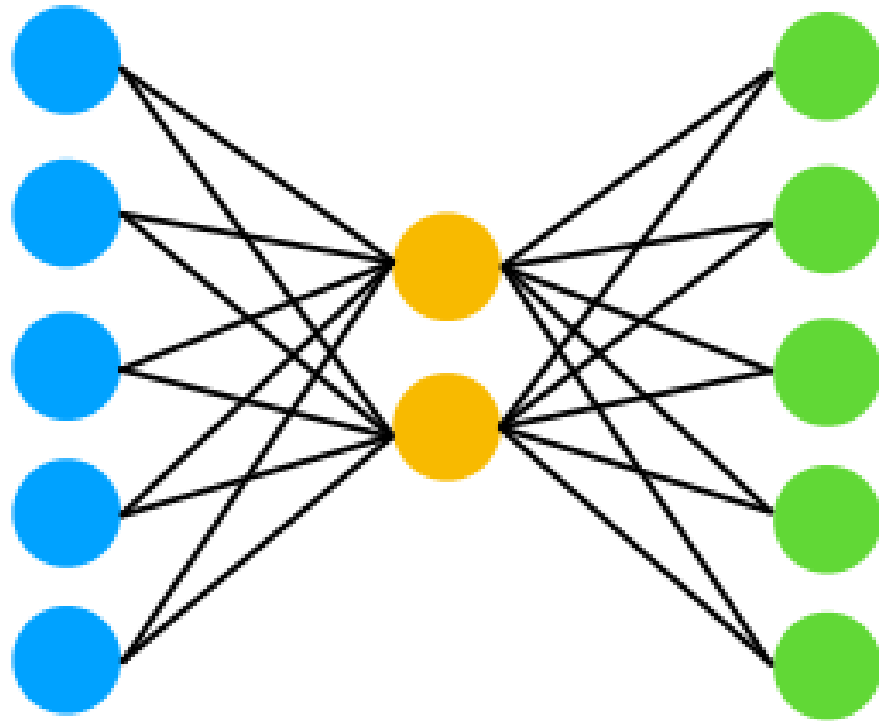


# Un codificador automático básico completamente conectado (perceptrón multicapa)

$$\mathcal{L}(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_2^2 = \sum_i (x_i - x'_i)^2$$



# Un codificador automático básico completamente conectado (perceptrón multicapa)

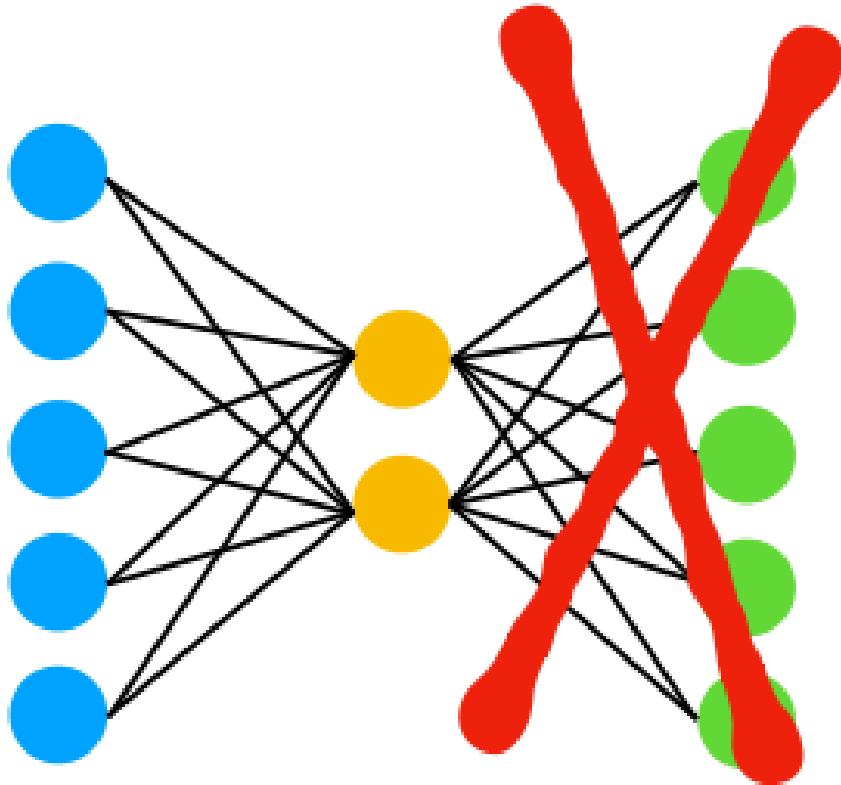


## Pregunta:

Si podemos lograr lo mismo con PCA, que es esencialmente un tipo de factorización matricial que es más eficiente que Backprop + SGD, ¿por qué molestarse con los codificadores automáticos?

# Posibles aplicaciones del codificador automático

Después del entrenamiento, ignore esta parte

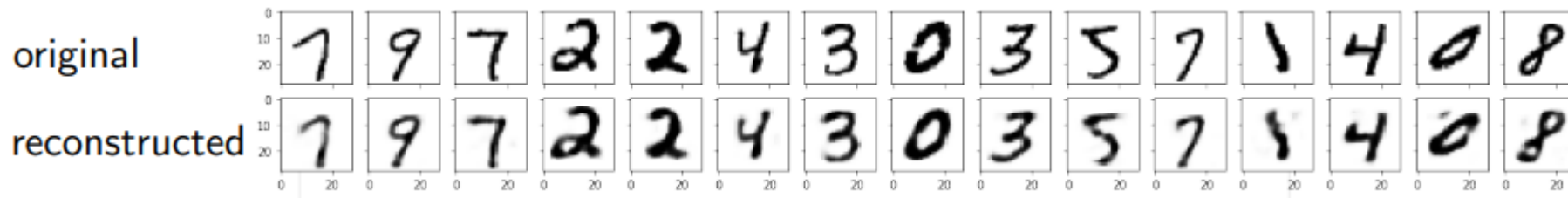
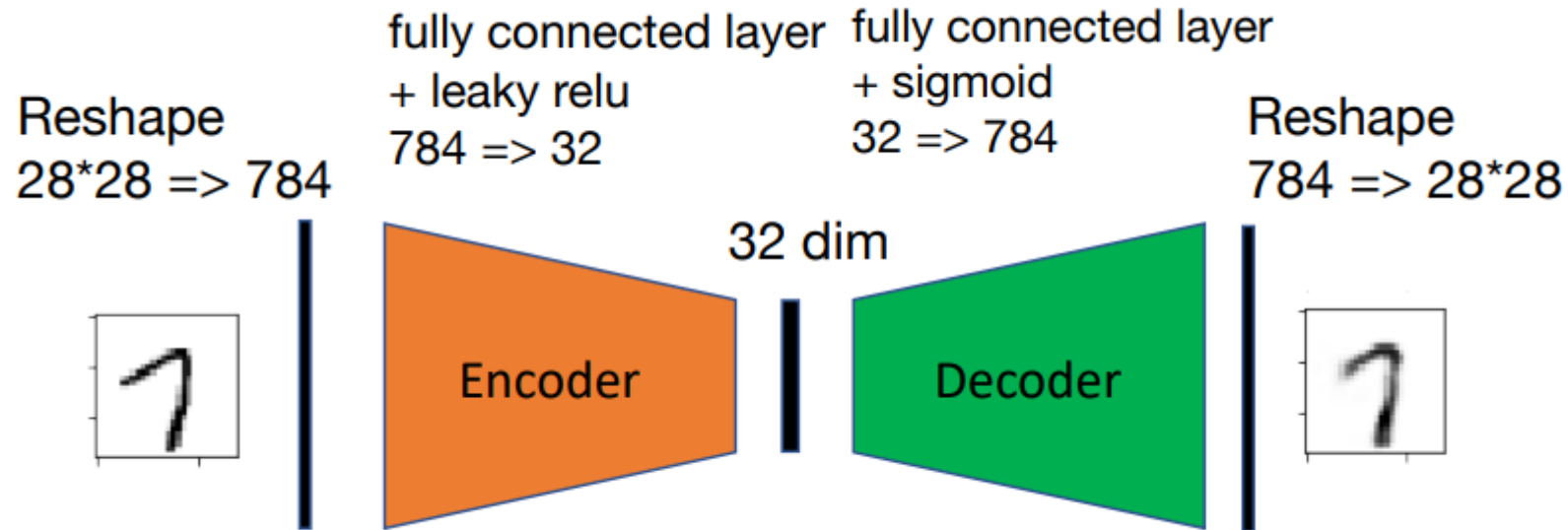


Utilice los embeddings (espacio latente) como entrada a los métodos clásicos de aprendizaje automático (SVM, KNN, Random Forest, ...)

O, de forma similar a transfer learning, entrene el codificador automático en un conjunto de datos de imagen grande, luego ajuste la parte del codificador con un conjunto de datos más pequeño o entrene su propia capa de salida (clasificación)

El espacio latente también se puede usar para la visualización (EDA, agrupamiento), pero existen mejores métodos para eso

# Un codificador automático simple

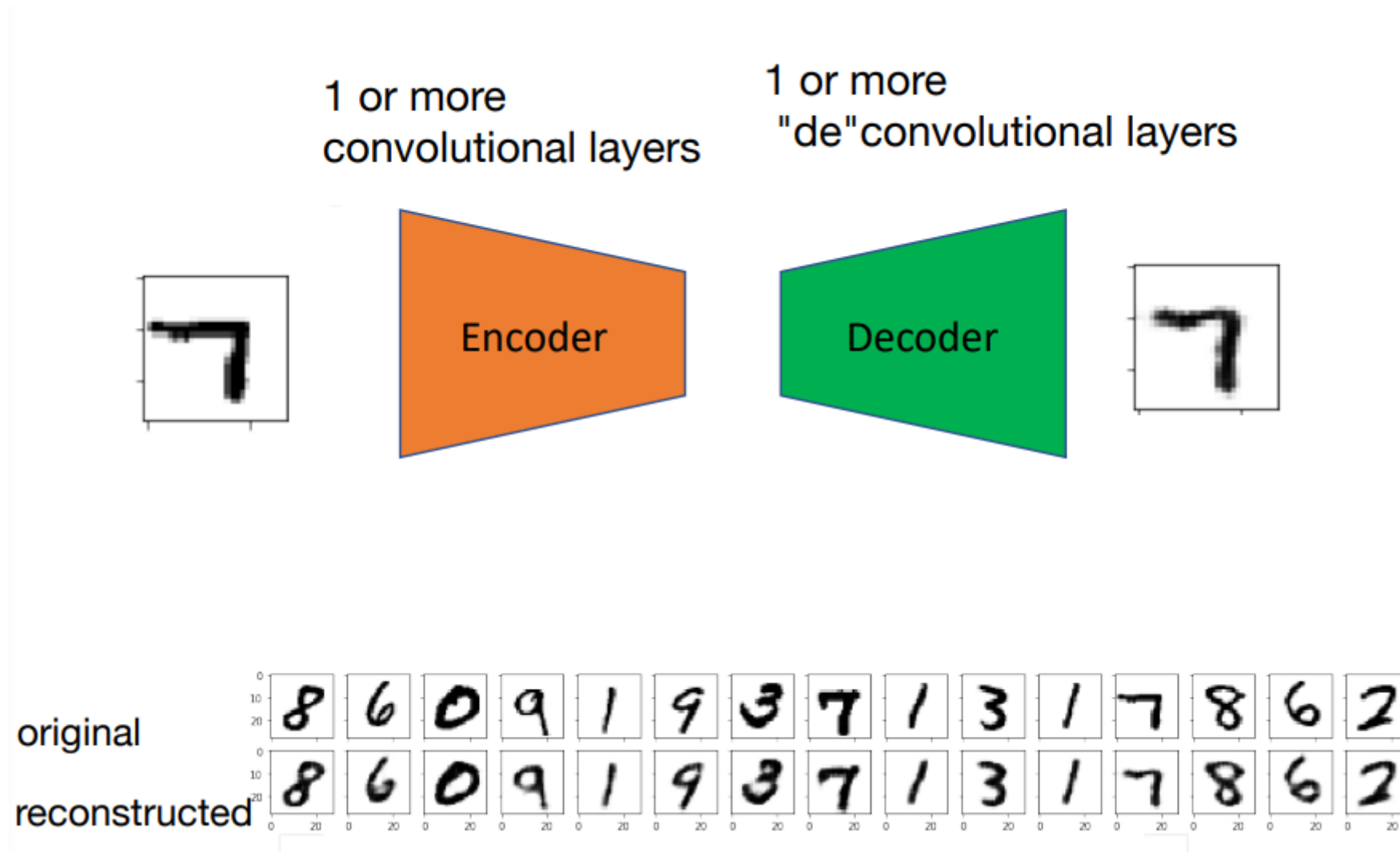




# Un perceptrón multicapa en forma de reloj de arena

1. Reducción de dimensionalidad
2. Codificadores automáticos completamente conectados
- 3. Codificadores automáticos convolucionales**
4. Codificadores automáticos convolucionales en PyTorch
5. Otros tipos de codificadores automáticos

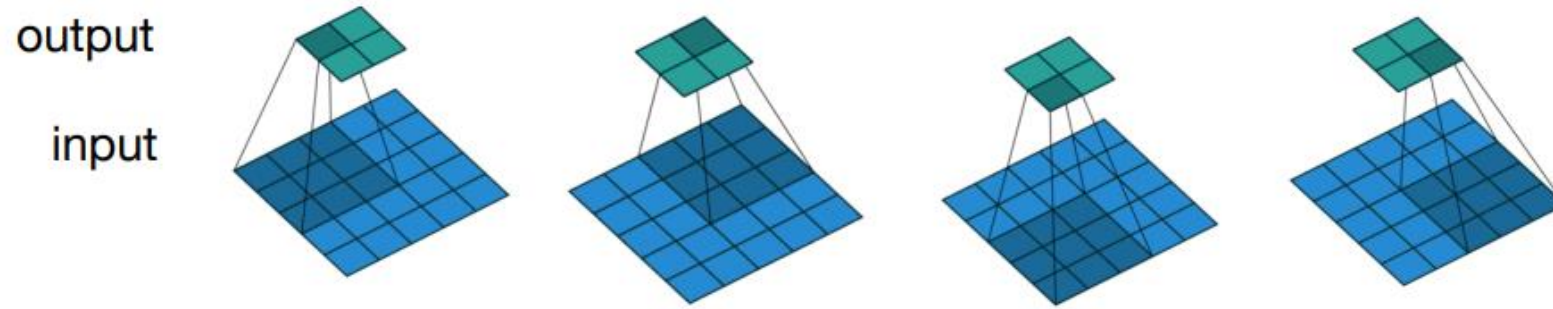
# Un codificador automático convolucional



# Convolución transpuesta

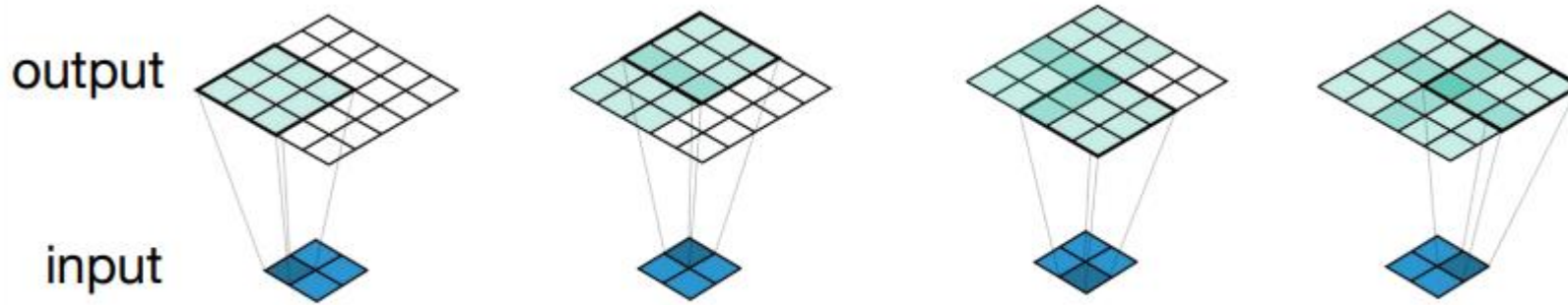
- Nos permite aumentar el tamaño del mapa de características de salida en comparación con el mapa de características de entrada
- Sinónimos:
  - A menudo también (incorrectamente) llamada "deconvolución" temáticamente, la deconvolución se define como la inversa de la convolución, que es diferente de las convoluciones transpuestas)
  - El término "unconv" a veces también se usa
  - La convolución fraccionada es otro (¿mejor?) término para eso.

# Convolución regular:



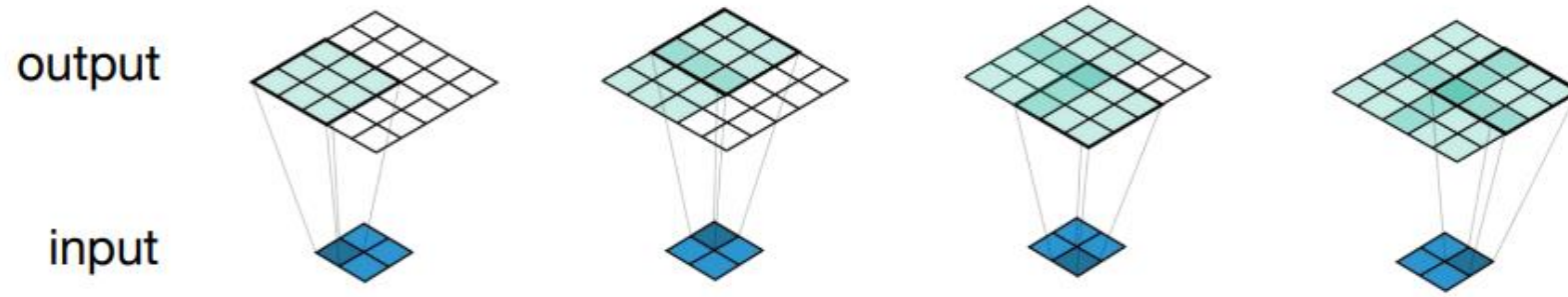
Dumoulin, Vincent, and Francesco Visin. "A guide to convolution arithmetic for deep learning." arXiv preprint arXiv:1603.07285 (2016).  
<https://arxiv.org/abs/1603.07285>

# Convolución transpuesta (paso = 2):



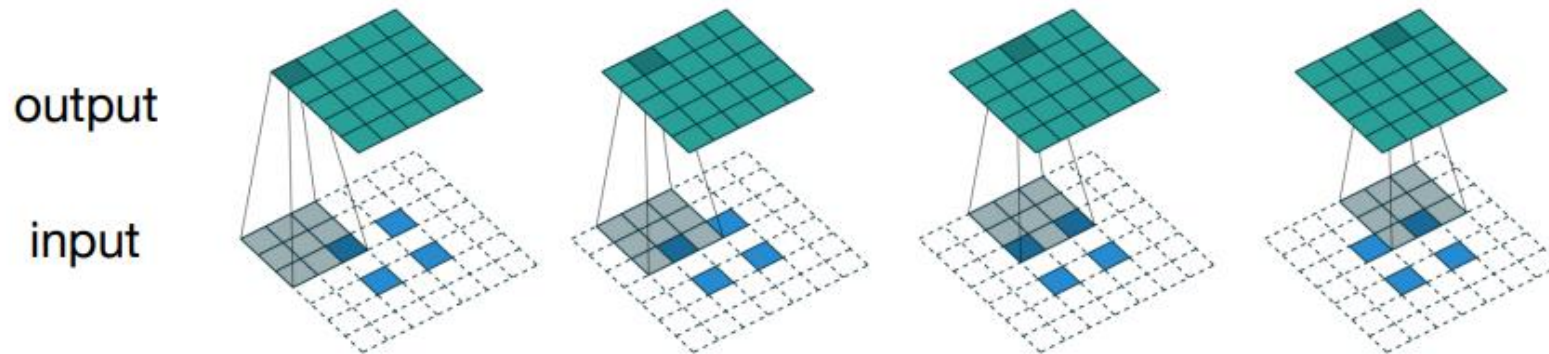
A Conv2DTranspose with 3x3 kernel and stride of 2x2 applied to a 2x2 input to give a 5x5 output.  
<https://medium.com/apache-mxnet/transposed-convolutions-explained-with-ms-excel-52d13030c7e8>

# Convolución transpuesta (kernel 3x3, paso = 2):



A Conv2DTranspose with 3x3 kernel and stride of 2x2 applied to a 2x2 input to give a 5x5 output.  
(<https://medium.com/apache-mxnet/transposed-convolutions-explained-with-ms-excel-52d13030c7e8>)

# Convolución transpuesta (emulada con convolución directa)



Dumoulin, Vincent, and Francesco Visin. "A guide to convolution arithmetic for deep learning." arXiv preprint arXiv:1603.07285 (2016).  
<https://arxiv.org/abs/1603.07285>

## Convolución regular: (paso = 1):

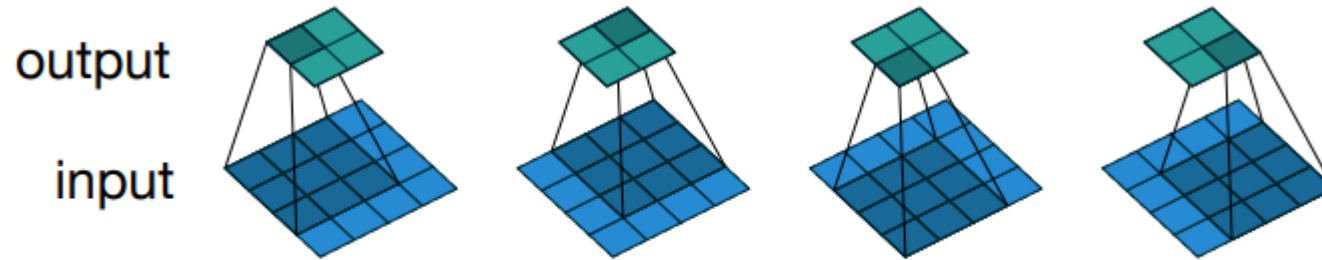
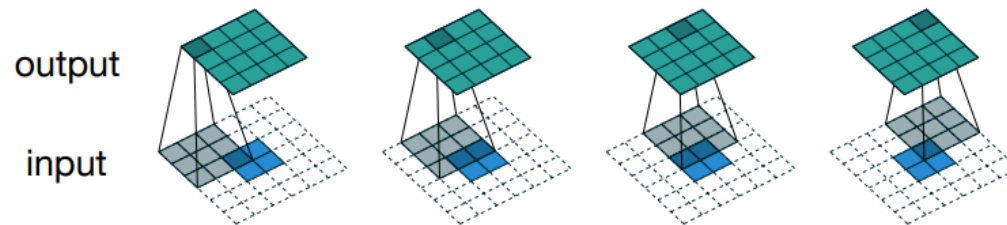


Figure 2.1: (No padding, unit strides) Convolving a  $3 \times 3$  kernel over a  $4 \times 4$  input using unit strides (i.e.,  $i = 4$ ,  $k = 3$ ,  $s = 1$  and  $p = 0$ ).

## Convolución transpuesta (emulada con convolución directa): (paso = 1)



# Convolución transpuesta

```
1 import torch

2 torch.manual_seed(123)
a = torch.rand(4).view(1, 1, 2, 2)

conv_t = torch.nn.ConvTranspose2d(in_channels=1,
                                   out_channels=1,
                                   kernel_size=(3, 3),
                                   padding=0,
                                   stride=1)

# output = s(n-1)+k-2p = 1*(2-1)+3-2*0 = 4
conv_t(a)

3 tensor([[[[-0.2863, -0.2766, -0.1478, -0.3274],
            [-0.3522, -0.5356, -0.1591, -0.2911],
            [-0.3054, -0.4644, -0.3286, -0.2444],
            [-0.2332, -0.2557, -0.1876, -0.3970]]]],
        grad_fn=<ThnnConvTranspose2DBackward>)
```

```
1 torch.manual_seed(123)
a = torch.rand(16).view(1, 1, 4, 4)

conv_t = torch.nn.ConvTranspose2d(in_channels=1,
                                   out_channels=1,
                                   kernel_size=(3, 3),
                                   padding=0,
                                   stride=1)

# output = s(n-1)+k-2p = 1*(4-1)+3-2*0 = 6
conv_t(a).size()

2 torch.Size([1, 1, 6, 6])
```

$$\text{output} = s(n - 1) + k - 2p$$

```
1 torch.manual_seed(123)
a = torch.rand(64).view(1, 1, 8, 8)

conv_t = torch.nn.ConvTranspose2d(in_channels=1,
                                   out_channels=1,
                                   kernel_size=(3, 3),
                                   padding=0,
                                   stride=1)

# output = s(n-1)+k-2p = 1*(8-1)+3-2*0 = 10
conv_t(a).size()

2 torch.Size([1, 1, 10, 10])
```

# Deconvolution and Checkerboard Artifacts

AUGUSTUS ODENA  
Google Brain

VINCENT DUMOULIN  
Université de Montréal

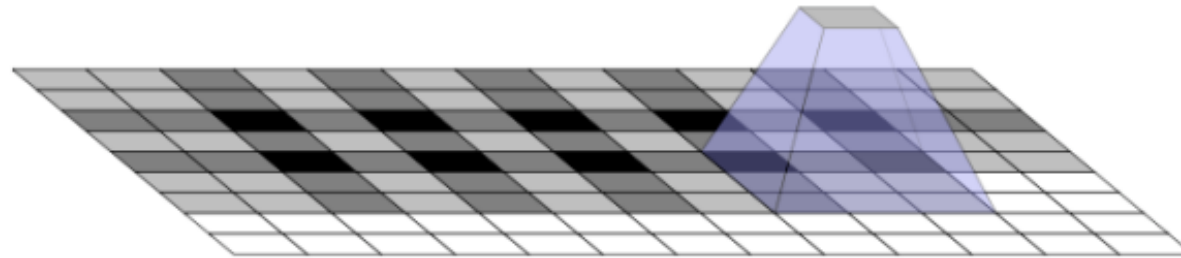
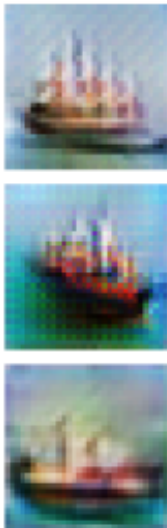
CHRIS OLAH  
Google Brain

Oct. 17  
2016

Citation:  
Odena, et al., 2016

<https://distill.pub/2016/deconv-checkerboard/>

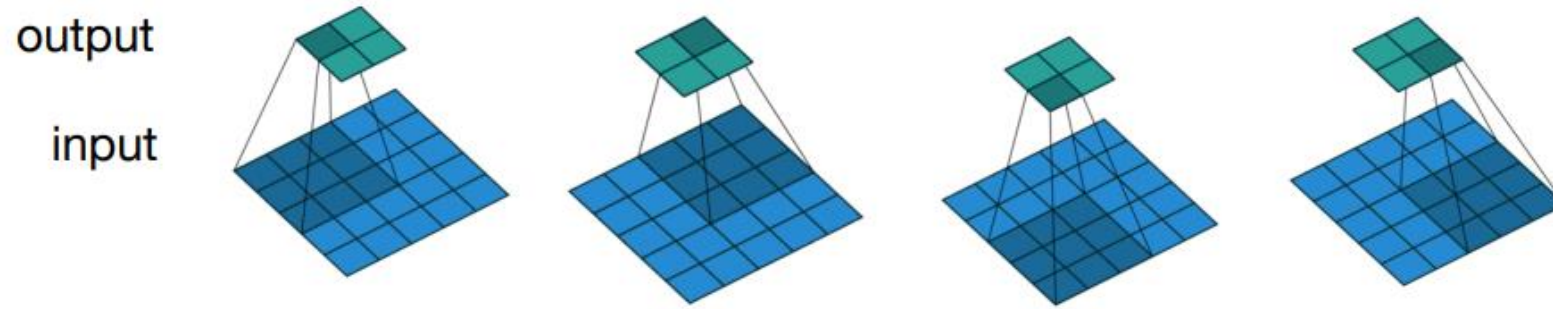
Un buen artículo interactivo que destaca los peligros de la convolución transpuesta.



En resumen, recomienda reemplazar la convolución transpuesta por muestreo ascendente (interpolación) seguido de convolución regular

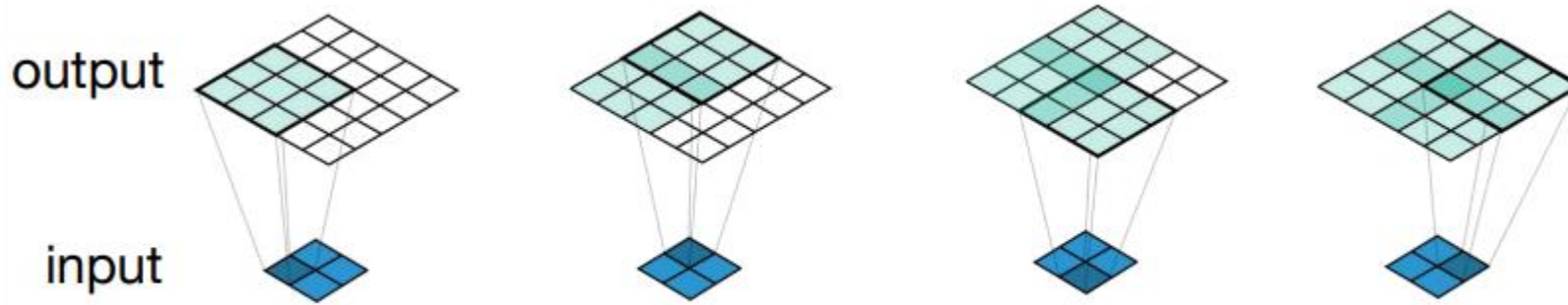


# Convolución regular:



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

# Convolución transpuesta (paso = 2):



A Conv2DTranspose with 3x3 kernel and stride of 2x2 applied to a 2x2 input to give a 5x5 output.  
(<https://medium.com/apache-mxnet/transposed-convolutions-explained-with-ms-excel-52d13030c7e8>)

# Un perceptrón multicapa en forma de reloj de arena

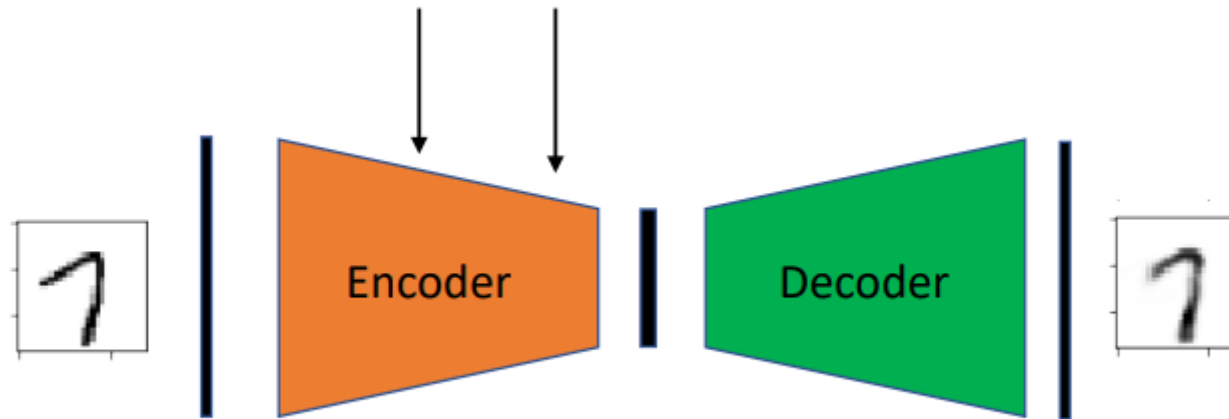
1. Reducción de dimensionalidad
2. Codificadores automáticos completamente conectados
3. Codificadores automáticos convolucionales
4. Otros tipos de codificadores automáticos

# Un perceptrón multicapa en forma de reloj de arena

1. Reducción de dimensionalidad
2. Codificadores automáticos completamente conectados
3. Codificadores automáticos convolucionales
4. **Otros tipos de codificadores automáticos**

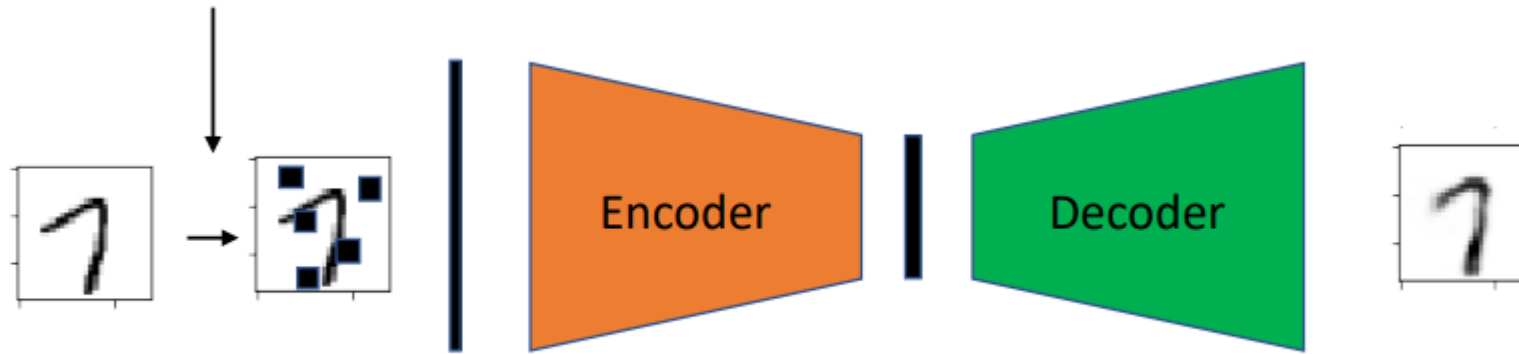
# Convolución transpuesta

Agregue capas de dropout para obligar a las redes a aprender características redundantes



# Codificadores automáticos de reducción de ruido

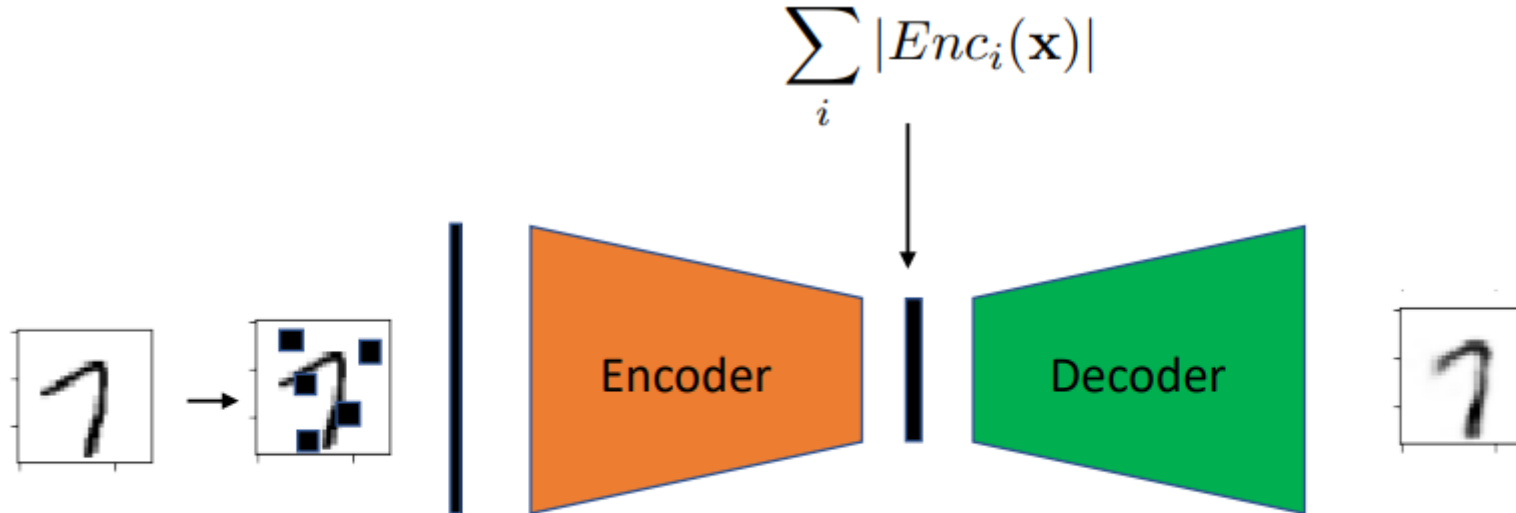
Agregue dropout después de la entrada o agregue ruido a la entrada para aprender a eliminar el ruido de las imágenes



Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. A. (2008, July). Extracting and composing robust features with denoising autoencoders. In Proceedings of the 25th International Conference on Machine Learning (pp. 1096-1103). ACM. <http://www.cs.toronto.edu/~larocheh/publications/icml-2008-denoising-autoencoders.pdf>

# Codificador automático disperso

Agregue una penalización L1 a la pérdida para aprender representaciones de características dispersas



$$\mathcal{L} = ||\mathbf{x} - Dec(Enc(\mathbf{x}))||_2^2 + \sum_i |Enc_i(\mathbf{x})|$$

# Codificador automático variacional

$$L^{[i]} = -\mathbb{E}_{z \sim q_w(z|x^{[i]})} [\log p_w(x^{[i]}|z)] + \text{KL} (q_w(z|x^{[i]}) \| p(z))$$

Expected neg. log likelihood  
term; wrt to encoder distribution

Kullback-Leibler divergence term  
where  $p(z) = \mathcal{N}(\mu = 0, \sigma^2 = 1)$

