



Visión por Computador II

CEAI, FIUBA

Profesor: Javier Kreiner, javkrei@gmail.com



Séptima clase:

- Style Transfer
- Reconocimiento de caras
- Generación de imágenes
- Super-Resolution
- Aplicaciones

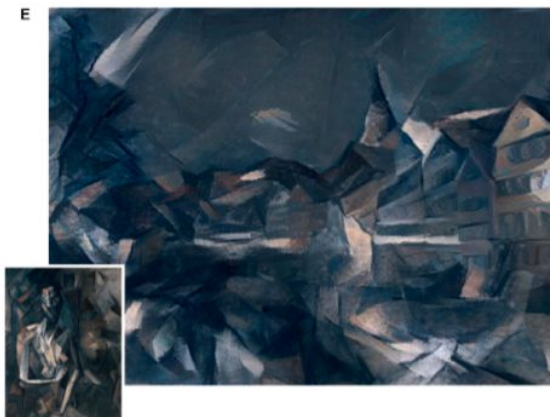
Style transfer

A



B

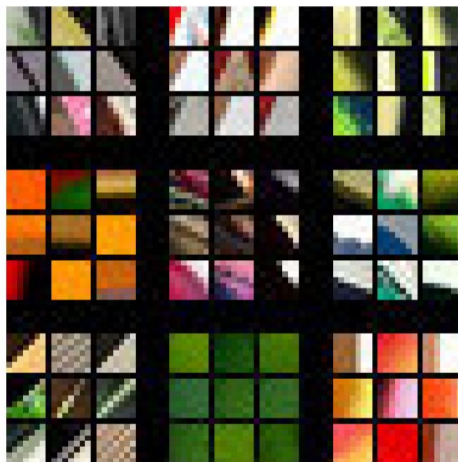




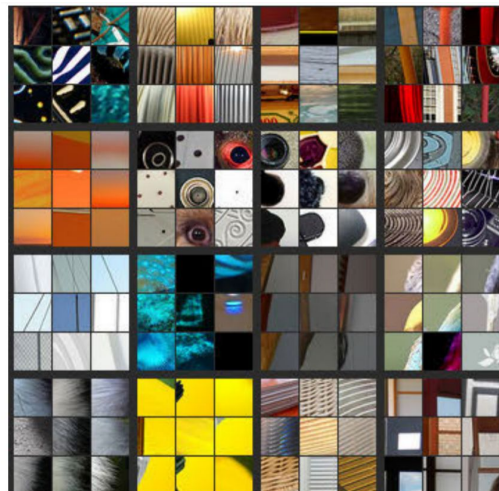
¿Qué aprenden las ConvNets?

- Mirar patches que maximizan activaciones
- ¿Qué pasa en diferentes capas? Se hacen más complejos. Ejemplo de red similar a AlexNet:

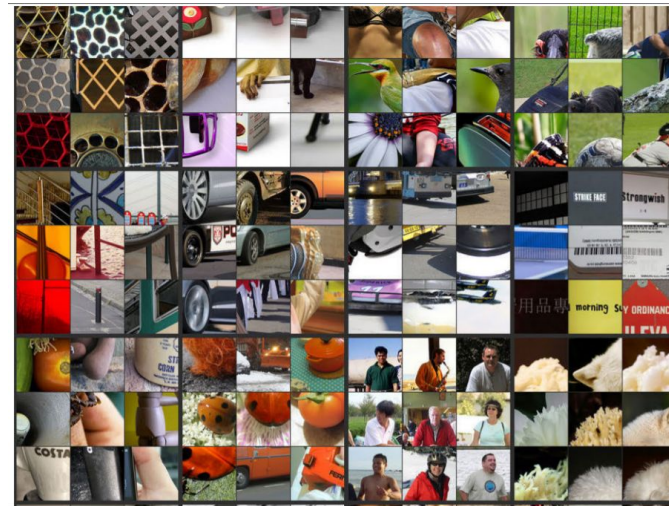
Capa 1



Capa 2

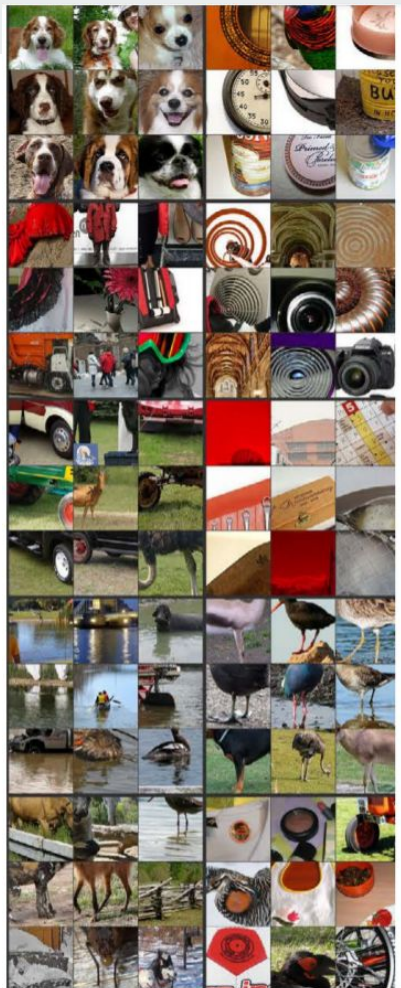


Capa 3



- [Visualizing and understanding Convolutional Networks](#)

Capa 4:



Capa 5:



Cómo definir la función de costo



Contenido: C



Estilo: S



Imágen generada: G

$$J(G) = \alpha J_{\text{content}}(C, G) + \beta J_{\text{style}}(S, G)$$



Procedimiento

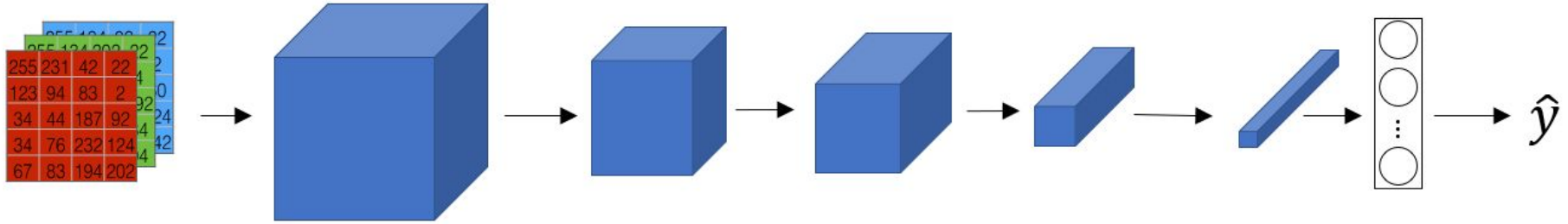
- Inicialización aleatoria de una imagen G
- Descendemos en la dirección del gradiente de $J(G)$
- $G := G - \partial J / \partial G (G)$

Función de costo de contenido



- Consideremos una capa intermedia l de una red de reconocimiento de imágenes
- Usamos una red pre-entrenada VGG, o Resnets, etc.
- Supongamos que $a^{[l]}(C)$ y $a^{[l]}(G)$ son las activaciones correspondientes a C y G
- Si estas dos activaciones son similares entonces las imágenes tienen contenido similar
- $J_{\text{content}}(G) = \|f(a^{[l]}(C)) - f(a^{[l]}(G))\|^2$

Función de costo de estilo



- Imaginemos que queremos usar la activación de la capa l para el estilo. Definimos el estilo como la correlación de activaciones a lo largo de los canales
- Pregunta: ¿cuán correlacionadas están las activaciones a lo largo de canales?
- Usando eso calculamos la 'distancia de estilo'

Imagen estilo

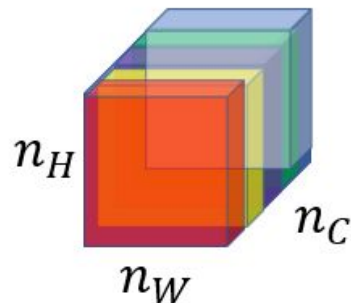
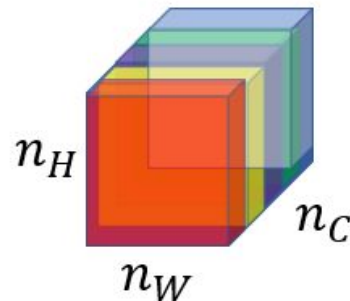


Imagen generada



- Queremos que las correlaciones entre canales sean parecidas en ambas imágenes
- ¿Cómo logramos esto?
-

Sea $a_{i,j,k}^{[l]}$ = activación en (i, j, k) . $G^{[l]}$ es $n_c^{[l]} \times n_c^{[l]}$

Gram matrix:

$$G_{kk'}^{[l]} = \sum_{i=1}^{n_H^{[l]}} \sum_{j=1}^{n_W^{[l]}} a_{ijk}^{[l]} a_{ijk'}^{[l]}$$

$$G^{[l]} = A^{[l]} (A^{[l]})^T$$

Usamos la norma de Frobenius entre matrices como distancia:

$$J_S^{[l]}(G^{[l](S)}, G^{[l](G)}) = \frac{1}{4(n_W^{[l]} n_H^{[l]})^2} \left\| G^{[l](S)} - G^{[l](G)} \right\|_{\mathcal{F}}^2 \quad \|G\|_{\mathcal{F}} = \sqrt{\sum_{ij} (g_{ij})^2}$$

Hacemos esto con varias capas:

$$J_S(\mathbf{x}, \mathbf{y}) = \sum_{l=0}^L \lambda_l J_S^{[l]}(G^{[l](S)}, G^{[l](G)})$$



Ejemplo de programación

- colab:
https://colab.research.google.com/drive/1sB8g1EPm8lB72_hsyHa3MazC-HRWJ0oq?usp=sharing

Reconocimiento de caras



Dos problemas relacionados:

1. Face verification: dado una imagen de entrada + un ID/nombre, decir si la imagen corresponde a esa persona
2. Face recognition: dada una base de datos con K personas y una imagen de entrada -> encontrar la persona a la que corresponde la imagen o decir que no es ninguna de esas personas



One-shot Learning, Zero-shot Learning

- One-shot learning: en este problema tenemos un solo ejemplo de lo que queremos aprender, por ejemplo cuando tenemos una sola foto de una persona en la base de datos
- Zero-shot learning: en este problema no hay ningún ejemplo del que aprender, esto significa que al momento de test voy a tener ejemplos que pertenecen a clases que no vi en el training set



Naive Face Verification 1

- Calcular la distancia euclídea entre las imágenes que nos presentan y la que tenemos en la base de datos
- No funciona demasiado bien, hay diferencias de iluminación, de ángulo de sombras



Naive approach 2

- construir una red que tenga $K+1$ salidas
- qué pasa cuando ingresa un nuevo empleado?
- tenemos pocos ejemplos, cómo entrenamos?

Función de similitud entre imágenes



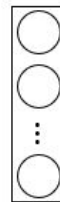
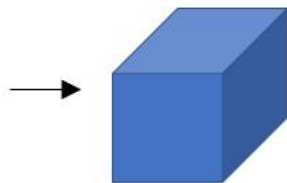
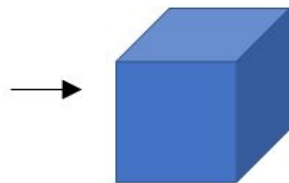
$d(\text{img1}, \text{img2})$ = grado de similitud de las imágenes

If $d(\text{img1}, \text{img2}) \leq \tau$
 $> \tau$

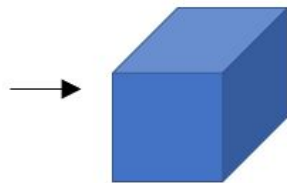
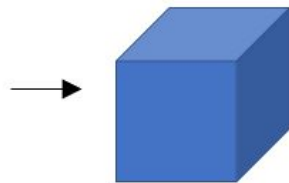
Red siamesa



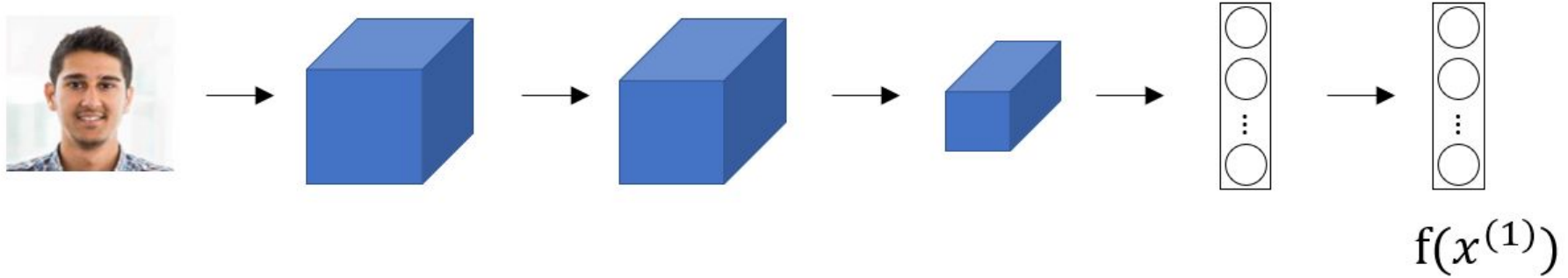
$x^{(1)}$



$x^{(2)}$



Objetivo del entrenamiento:



Aprender una red tal que:

Si $x^{(i)}, x^{(j)}$ son la misma persona, $\|f(x^{(i)}) - f(x^{(j)})\|^2$ es pequeño

Si $x^{(i)}, x^{(j)}$ son dos personas diferentes, $\|f(x^{(i)}) - f(x^{(j)})\|^2$ es grande

Triplet loss



Anchor



Positivo



Anchor



Negativo



Anchor



⋮



Positivo



⋮



Negativo



⋮



Función de costo



$$\|f(A)-f(P)\|^2 + \alpha \leq \|f(A)-f(N)\|^2$$

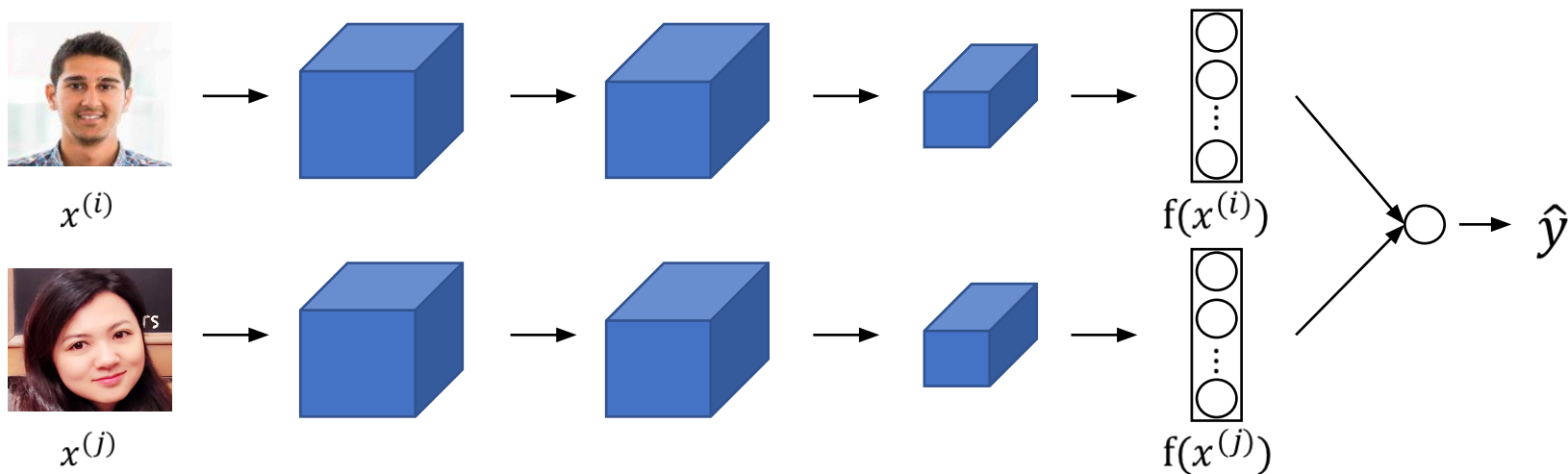
$$\|f(A)-f(P)\|^2 - \|f(A)-f(N)\|^2 + \alpha \leq 0$$

$$\mathcal{J}(A,P,N) = \max(\|f(A)-f(P)\|^2 - \|f(A)-f(N)\|^2 + \alpha, 0)$$

$$J = \sum \mathcal{J}(A,P,N)$$









¿Cómo elegir las tuplas? Elegir los ejemplos difíciles.

Plantear un problema de clasificación supervisado



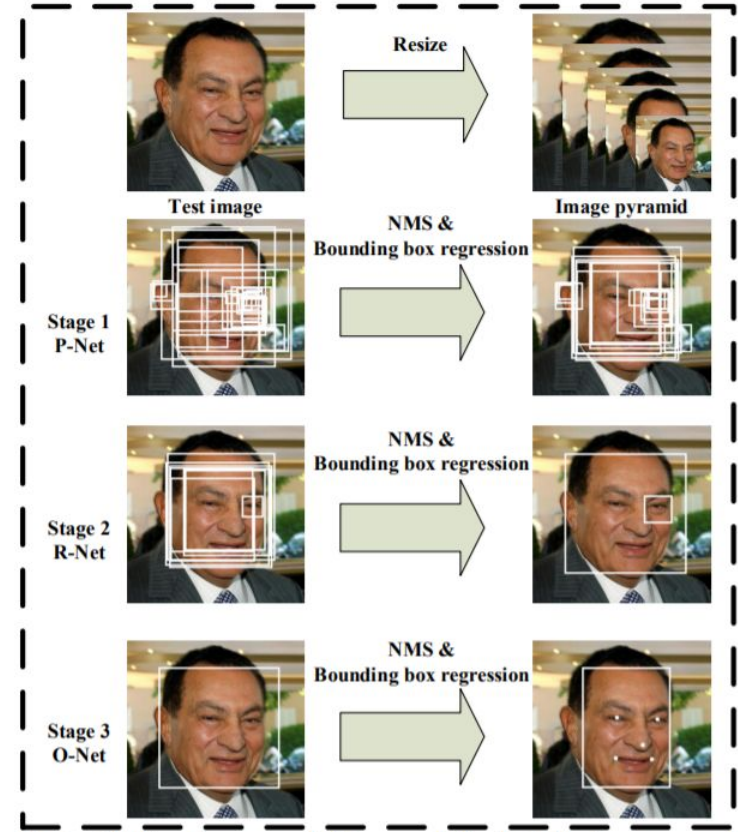
¿Cómo armar el dataset?



x		y
		1
		0
		0
		1

¿Cómo extraer la cara de la imagen?

- [Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks](#)
- Primero se hace una 'pirámide' de imágenes, re-escalando la imagen en varios tamaños
- Luego hay 3 etapas de redes convolucionales:
 - P-Net (proposal network): usan una red convolucional para generar propuestas de bounding boxes, se usa non-max suppression para fusionar ventanas superpuestas
 - R-Net (refine network): usan otra red convolucional para rechazar muchas de las propuestas
 - O-Net (output network): similar a la anterior, hace output de la caja y landmarks





Ejemplo de programación

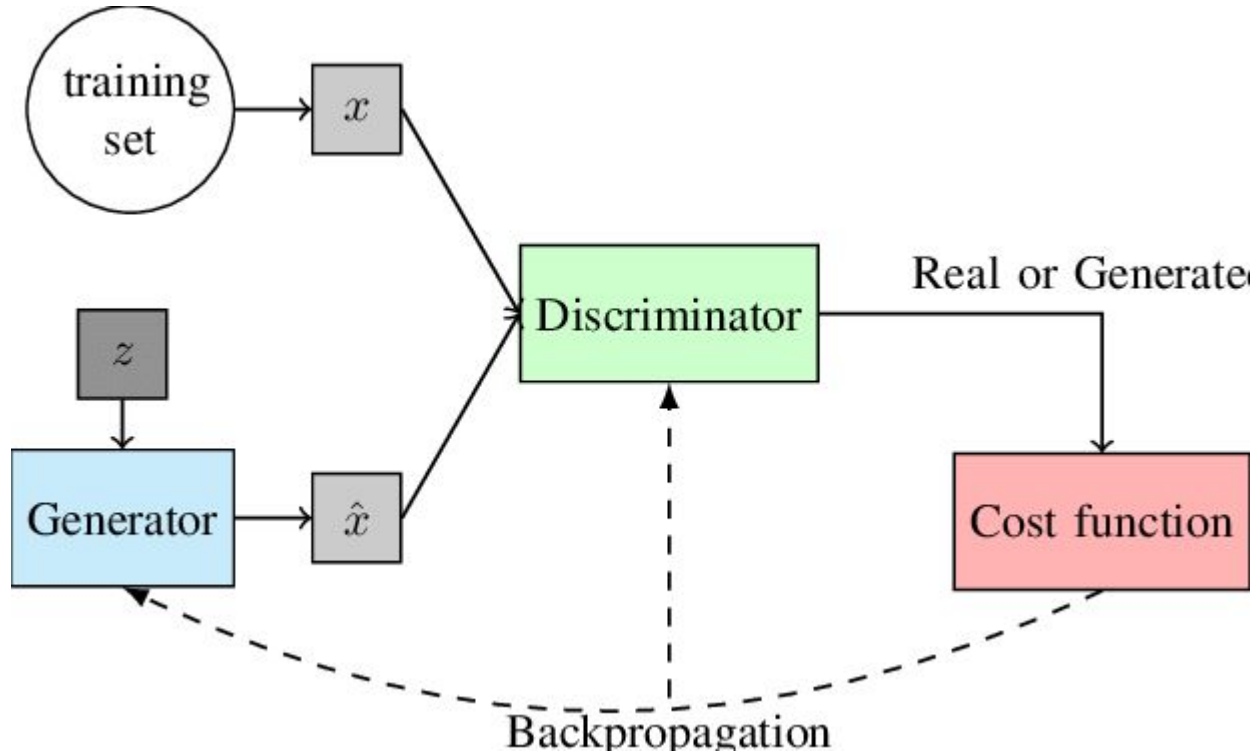
- Ejemplo de reconocer imágenes de celebrities, colab:
<https://colab.research.google.com/drive/1DPLe93k0XOuSjVFsfCYuiRnr1wATEHat?usp=sharing>



Generación de imágenes

- Generación de imágenes parecidas a las de un dataset

Generative Adversarial Networks





Ejemplo

- colab:
<https://colab.research.google.com/drive/16ZXWRFa3wT8liLxwLrkwXoJLtKx50Awm?usp=sharing>



(Image Super) Resolution

- El problema de Image Super Resolution, o super resolución de imágenes intenta reconstruir una imagen de alta resolución de una imagen de baja resolución
- En particular tenemos Single Image Super Resolution (SISR), que intenta hacerlo de una sola imagen
- En general la relación entre I^{LR} y I^{HR} depende de la situación
- Muchos estudios asumen que I^{LR} es una versión sub-sampleada de manera bicúbica de I^{HR} , pero hay otras transformaciones posibles: blur, ruido, etc.
- Papers: [Enhanced Deep Residual Networks for Single Image Super-Resolution](#), [Wide Activation for Efficient and Accurate Image Super-Resolution](#), [Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network](#)



Ejemplo de programación

- <https://github.com/krasserm/super-resolution>

Ejemplos de aplicaciones en la industria



- Otra aplicaciones:
 - Real-time multi-person pose estimation: <https://youtu.be/pW6nZXeWlGM>
 - Real-time analysis of behaviour: <https://youtu.be/xhp47v5OBXQ>
- Aplicaciones en la industria:
 - Autos autónomos
 - Evaluación de estado de edificios con drones para aseguradoras
 - Procesamiento de imágenes en medicina: e.g. detección de cáncer
 - Colorización automática: <https://youtu.be/ys5nMO4Q0iY>
 - DeepFakes: https://youtu.be/MVBe6_o4cMI
 - Aplicación práctica de super resolution: renderización 3D (ejemplo, juegos)

nvidia Deep Learning Super Sampling (DLSS)



- <https://www.nvidia.com/en-us/geforce/news/nvidia-dlss-2-0-a-big-leap-in-ai-rendering/>
- mejora la imagen y a su vez la performance en más de 50%+ y a veces más de 70%
- <https://www.youtube.com/watch?v=ccPUj5cCs4c&feature=youtu.be>
- <https://youtu.be/KwDs6LrocR4>