



Deep Image Homography Estimation



Estructura de esta presentación

- ◀ **Introducción**
- ◀ **Implementación**
 - ◀ Arquitectura de la Red
 - ◀ Parametrización de 4 puntos
 - ◀ Armado de Dataset
 - ◀ Diferencias de Implementación
- ◀ **Resultados**
 - ◀ Números
 - ◀ Reconstrucción
- ◀ **Conclusiones**

Introducción





Introducción

- ▶ Implementamos la Red Neuronal descrita en el paper *Deep Image Homography Estimation*, de DeTone et al. (HomographyNet)
- ▶ Sólo la versión de Regresión, no la de Clasificación.
- ▶ Obtuvimos buenos resultados con menos datos! (con leves diferencias de implementación)

Problema: Estimación de Homografías

A partir de dos imágenes,
calcular su matriz de
homografía



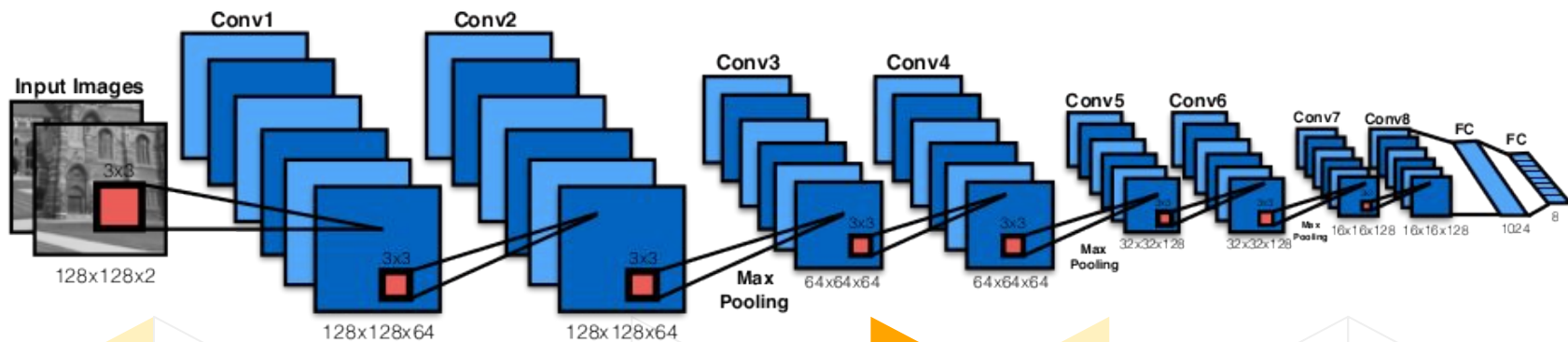
$$H = \begin{bmatrix} 0.44 & -0.38 & 128.84 \\ -0.10 & 0.39 & 73.09 \\ -0.01 & -0.01 & 1.00 \end{bmatrix}$$

Implementación



Arquitectura de HomographyNet

Versión de Regresión





Arquitectura

- ▶ Entrada: imagen en grayscale de dos canales de $128 \times 128 \times 2$
- ▶ 8 capas convolucionales
 - ▶ Primeras 4: 64 filtros, últimas: 128
- ▶ Capa de max-pooling intercalada luego de cada 2 de las convolucionales
- ▶ 2 capas totalmente conectadas al final
 - ▶ Primera de 1024
 - ▶ Última de 8



Arquitectura

- ▶ Dropout con probabilidad 0.5 antes y después de la primer capa totalmente conectada
- ▶ Batch Normalization luego de cada capa convolucional
- ▶ Función de activación ReLu
- ▶ Función de pérdida es la distancia euclidiana
- ▶ SGD con momento de 0.9 con un learning rate base de 0.005, decrementando en un factor de 10 cada 30000 iteraciones.

La parametrización de 4 puntos de la Homografía





Motivación

- ▶ Tradicionalmente, las Homografías se parametrizan como una matriz de 3x3

$$H = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix}$$

- ▶ Sin embargo, si expandimos esto a un vector de 9 parámetros, se mezclan los términos de rotación y traslación...



Motivación

- ◀ La submatriz $[H_{11}, H_{12}, H_{21}, H_{22}]$ representa la parte rotacional de la transformación y $[H_{13}, H_{23}]$ la parte traslacional.
- ◀ Balancear esto en un problema de optimización, como es el entrenamiento de una Red Neuronal, es difícil...

Solución

- ▶ Parametrización de 4 puntos.
- ▶ Se basa en guardar los offsets entre las 4 esquinas del par de imágenes.

$$H_{4point} = \begin{bmatrix} \Delta u_1 & \Delta v_1 \\ \Delta u_2 & \Delta v_2 \\ \Delta u_3 & \Delta v_3 \\ \Delta u_4 & \Delta v_4 \end{bmatrix}$$

- ▶ Donde $\Delta u_i = u'_i - u_i$ y $\Delta v_i = v'_i - v_i$, siendo (u_i, v_i) la esquina i de la imagen original y (u_i, v'_i) la esquina i de la imagen perturbada, con $i \in \{1 \dots 4\}$



Solución

- ◀ La red toma como labels de entrenamiento el arreglo de 8 posiciones correspondiente y también devuelve como predicción un arreglo con las mismas características.
- ◀ Es fácil pasar de este arreglo a una matriz de homografía con DLT o funciones como **getPerspectiveTransform()** de OpenCV

Armado del Dataset





Armado del Dataset

- ◀ Dataset propio generado a partir del set de entrenamiento **MS-COCO**.
- ◀ Las imágenes son escaladas a 320x240 y transformadas a grayscale.
- ◀ Luego, se generan 500000 datos de entrenamiento a partir de estas.



Generación de los datos

- ▶ Eligen un recorte de tamaño 128x128 a partir de una imagen más grande en una posición p elegida al azar, evitando los bordes para prevenir defectos.
- ▶ Las cuatro esquinas son perturbadas al azar en el rango $[-32, 32]$
- ▶ Las cuatro correspondencias definen una homografía \mathbf{H}_{AB}

Generación de los datos

- ▶ Aplican la inversa de esta homografía a la imagen grande para producir una segunda imagen grande \mathbf{I}' .
- ▶ Se realiza otro recorte \mathbf{I}'_p en la misma posición p .
- ▶ \mathbf{I}_p e \mathbf{I}'_p se combinan en una imagen de dos canales que será pasada como input a la red.
- ▶ La parametrización de 4 puntos de \mathbf{H}_{AB} se usa como label para esta imagen en la fase de entrenamiento.

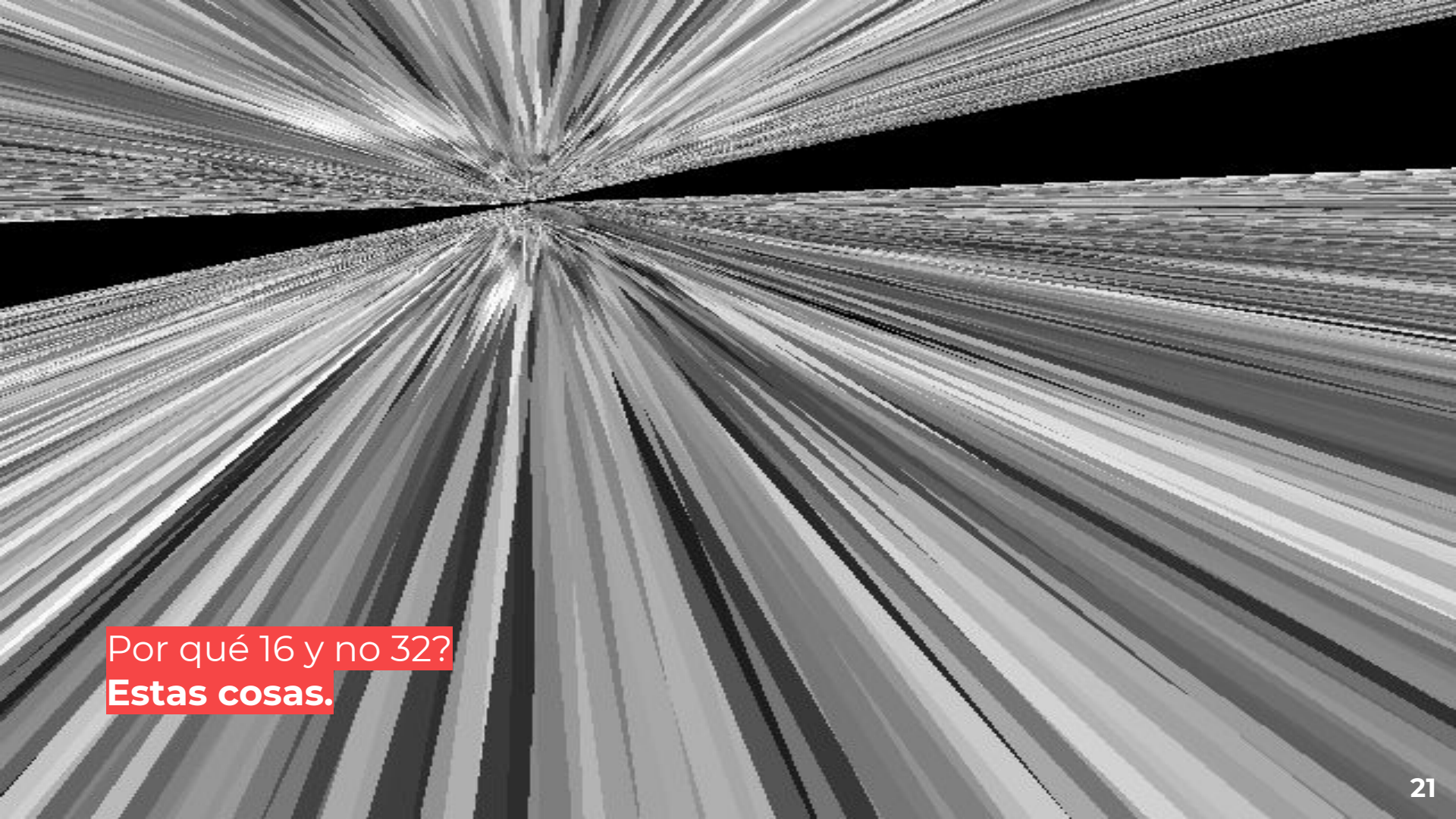
Diferencias de Implementación con el Paper





Diferencias de implementación

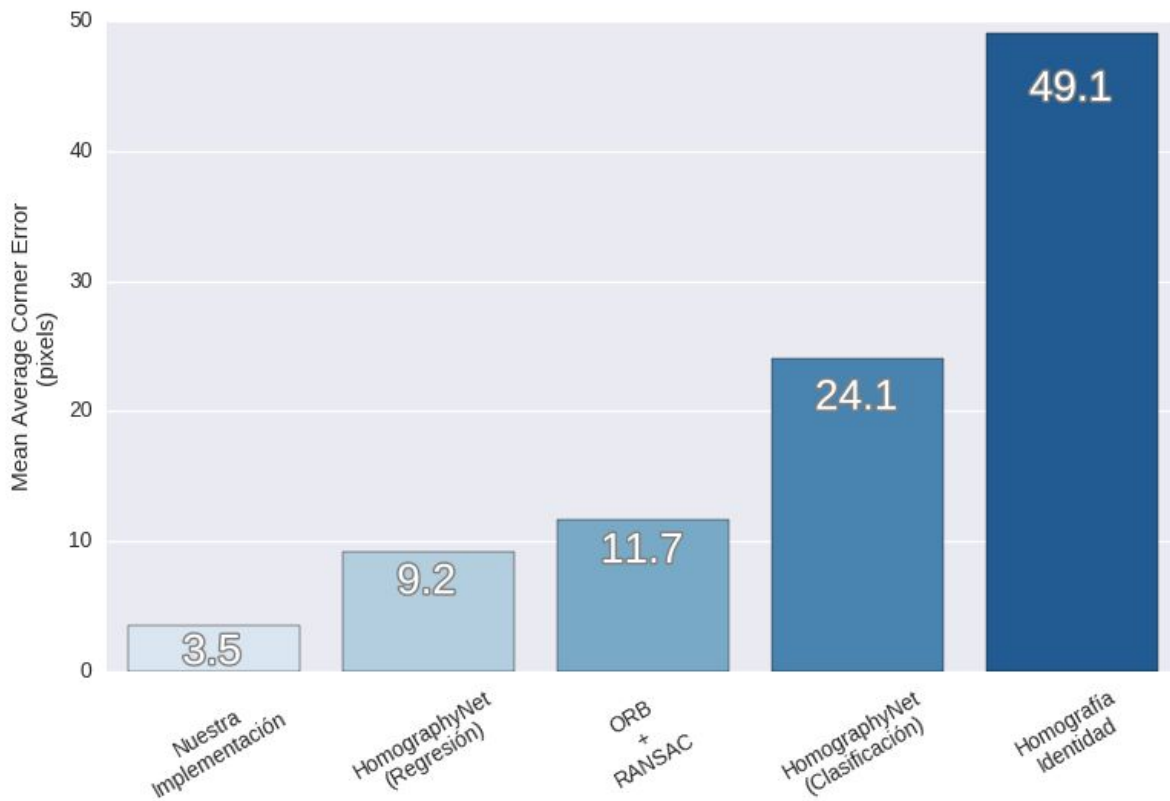
- ▶ **Rango de perturbación de 16 píxeles en vez de 32 en la generación del dataset.**
- ▶ Reducción en el volumen de datos que utilizamos:
 - ▶ De 500000 a ~83000
- ▶ Metodología de testeo:
 - ▶ Separamos un 10 % del dataset para testeo
- ▶ Mantuvimos el learning rate base de 0.005
- ▶ En vez de Caffe, decidimos optar por el módulo de Keras de TensorFlow.

The background of the slide is an abstract, high-contrast black and white image. It features a dense pattern of thin, parallel lines that radiate from a central point at the top, creating a strong sense of perspective and depth, similar to a tunnel or a starburst effect. The lines vary in length and orientation, creating a complex, textured appearance.

Por qué 16 y no 32?
Estas cosas.

Resultados





MACE: 3,5

El promedio, sobre todo el set de imagenes, del error medio (en distancia euclidiana) de las cuatro esquinas de cada imagen.





Pero...

- ◀ Hay que tener en cuenta que aplicamos modificaciones sobre el algoritmo original de armado de dataset.
 - ◀ Perturbación que utilizamos fue de 16, no 32.
- ◀ Sin embargo, uno intuitivamente esperaría que el error se duplique en el caso de una perturbación de 32.
 - ◀ En ese caso, nuestra implementación superaría los resultados obtenidos en el paper.



Números crudos

	MSE	MACE
Train	0,047	3,906
Test	0,043	3,526

Reconstrucción de una imagen

A partir de la homografía estimada



A modo ilustrativo...

- ▶ Reconstrucción de la transformación de una imagen a partir de la homografía estimada con la red.



Recorte



Imagen
perturbada
(ground
truth)



Reconstrucción
a partir de la
homografía
obtenida por
la red

Notas

- Es necesario guardar la posición del recorte para poder obtener la homografía en forma matricial.



Recorte

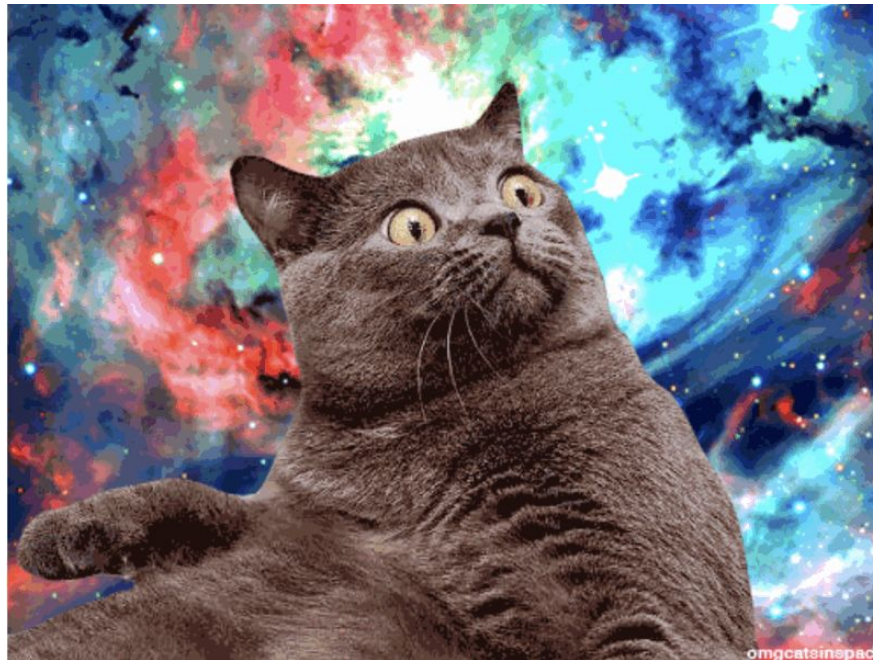


Imagen perturbada
(ground truth)



Reconstrucción
a partir de la
homografía
obtenida por
la red

La imagen original es esta



Conclusiones





Conclusiones

- ◀ Queda pendiente pruebas sin modificaciones significativas, por ejemplo perturbación de 16 vs. 32.
 - ◀ Sin embargo, creemos que es una aproximación fiel debido a que obtuvimos buenos resultados en cuanto a métricas y pruebas empíricas.
- ◀ Usar redes neuronales para estimar homografías parece ser efectivo y superador, inclusive con datasets de tamaños sustancialmente menores a los del paper.



Gracias!

Preguntas?



Créditos y referencias

- ▶ Trabajo y presentación por Emanuel Lamela y Andreas Sturmer
- ▶ Basada en el paper **Deep Image Homography Estimation** por Daniel DeTone, Tomasz Malisiewicz, y Andrew Rabinovich (2016)
- ▶ Plantilla por [SlidesCarnival](#)