Estimación de homografías con Deep Learning

Julián Palladino, Damián Silvani

Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Mayo 2019

Introducción

- ¿Qué es una homografía?

Introducción

- ¿Qué es una homografía?
  - Una transformación proyectiva que relaciona dos imágenes.

Introducción

- ¿Qué es una homografía?
  - Una transformación proyectiva que relaciona dos imágenes.
  - Se lo suele representar como una matriz 3×3.

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

Introducción

- ¿Qué es una homografía?
  - Una transformación proyectiva que relaciona dos imágenes.
  - Se lo suele representar como una matriz 3×3.

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

  - Un píxel (x, y) de la primer imagen se corresponde a un pixel (x', y') de la segunda imagen dado:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = H \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Introducción

- ¿Qué es una homografía?
  - Una transformación proyectiva que relaciona dos imágenes.
  - Se lo suele representar como una matriz 3×3.

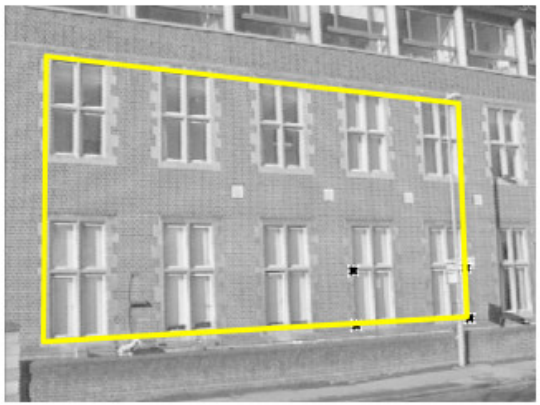
$$H = \begin{pmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{pmatrix}$$

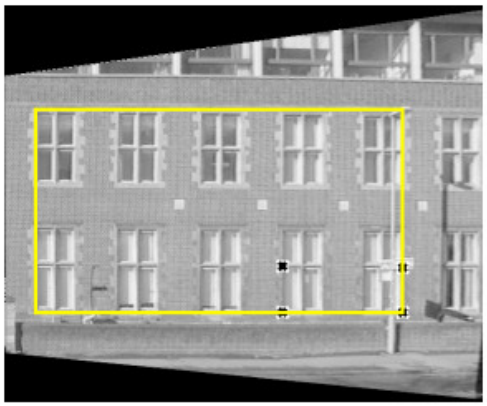
  - Un píxel (x, y) de la primer imagen se corresponde a un pixel (x', y') de la segunda imagen dado:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = H \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Usos y aplicaciones

- ¿Para qué sirve?
  - Registración de imágenes
  - Panorama stitching
  - Calibración de cámaras
  - Reconstrucción 3D
  - etc...





from Hartley & Zisserman

Estimación de homografías

- ¿Cómo se estima *tradicionalmente* entre un par de imágenes?

Estimación de homografías

- ¿Cómo se estima *tradicionalmente* entre un par de imágenes?
  - Se extraen *features* en ambas imágenes mediante algún método como SIFT, SURF, ORB, etc.

Estimación de homografías

- ¿Cómo se estima *tradicionalmente* entre un par de imágenes?
  - Se extraen *features* en ambas imágenes mediante algún método como SIFT, SURF, ORB, etc.
  - Se calculan las correspondencias entre los features de las dos imágenes.

Estimación de homografías

- ¿Cómo se estima *tradicionalmente* entre un par de imágenes?
  - Se extraen *features* en ambas imágenes mediante algún método como SIFT, SURF, ORB, etc.
  - Se calculan las correspondencias entre los features de las dos imágenes.
  - Se descartan correspondencias malas mediante con métodos como RANSAC.

Estimación de homografías

- ¿Cómo se estima *tradicionalmente* entre un par de imágenes?
  - Se extraen *features* en ambas imágenes mediante algún método como SIFT, SURF, ORB, etc.
  - Se calculan las correspondencias entre los features de las dos imágenes.
  - Se descartan correspondencias malas mediante con métodos como RANSAC.
  - Finalmente, se calcula la homografía con las mejores correspondencias, bajo algún criterio.

Deep Learning para estimar homografías

- En el paper proponen utilizar una red convolucional profunda para estimar homografías entre pares de imágenes.

Deep Learning para estimar homografías

- En el paper proponen utilizar una red convolucional profunda para estimar homografías entre pares de imágenes.
- La red convolucional consiste de 10 capas, y toma como entrada dos imágenes en escala de grises.

Deep Learning para estimar homografías

- En el paper proponen utilizar una red convolucional profunda para estimar homografías entre pares de imágenes.
- La red convolucional consiste de 10 capas, y toma como entrada dos imágenes en escala de grises.
- La salida es una homografia, que puede ser utilizada para crear correspondencias entre los pıxeles de la primer imagen a la segunda.

Parametrización de la homografía

- Al aplanar la matriz 3×3 en un vector para usarlo de salida de la red, se *mezclan* los términos de rotación y translación, y dificulta el aprendizaje.

Parametrización de la homografía

- Al aplanar la matriz 3×3 en un vector para usarlo de salida de la red, se *mezclan* los términos de rotación y translación, y dificulta el aprendizaje.
- El paper propone reescribir la homografía usando la parametrización de 4 puntos:

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

donde $\Delta u_i = u'_i - u_i$ y $\Delta v_i = v'_i - v_i$, y u_i y v_i son 4 puntos de las imágenes u y v.

Parametrización de la homografía

- Al aplanar la matriz 3×3 en un vector para usarlo de salida de la red, se *mezclan* los términos de rotación y translación, y dificulta el aprendizaje.
- El paper propone reescribir la homografía usando la parametrización de 4 puntos:

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

  donde $\Delta u_i = u'_i - u_i$ y $\Delta v_i = v'_i - v_i$, y u_i y v_i son 4 puntos de las imágenes u y v.
- Se puede convertir $H_{4puntos}$ a H mediante DLT.

Modelo de regresión y clasificación

- En el paper se evaluan dos modelos.

Modelo de regresión y clasificación

- En el paper se evaluan dos modelos.
  1. **Regresión**: Produce los 8 valores de la matriz $H_{4puntos}$.

Modelo de regresión y clasificación

- En el paper se evaluan dos modelos.
  1. **Regresión**: Produce los 8 valores de la matriz $H_{4puntos}$.
  2. **Clasificación**: Cuantiza la imagen en una grilla de 21 celdas, y devuelve la probabilidad de que cada punto esté en alguna celda de la grilla.

Modelo de regresión y clasificación

- En el paper se evaluan dos modelos.
  1. **Regresión**: Produce los 8 valores de la matriz $H_{4puntos}$.
  2. **Clasificación**: Cuantiza la imagen en una grilla de 21 celdas, y devuelve la probabilidad de que cada punto esté en alguna celda de la grilla.
- En este trabajo decidimos sólo trabajar con el modelo de regresión, que tiene mejor precisión.

Generación del dataset

- En el paper se utilizaron las imágenes del dataset MS-COCO 2014. En este trabajo, las del MS-COCO 2017, que tiene mayor cantidad de imágenes.

Generación del dataset

- En el paper se utilizaron las imágenes del dataset MS-COCO 2014. En este trabajo, las del MS-COCO 2017, que tiene mayor cantidad de imágenes.
- Para cada imagen, se realizan los siguientes pasos:

Generación del dataset

- En el paper se utilizaron las imágenes del dataset MS-COCO 2014. En este trabajo, las del MS-COCO 2017, que tiene mayor cantidad de imágenes.
- Para cada imagen, se realizan los siguientes pasos:
  1. Convertir a escala de grises

Generación del dataset

- En el paper se utilizaron las imágenes del dataset MS-COCO 2014. En este trabajo, las del MS-COCO 2017, que tiene mayor cantidad de imágenes.
- Para cada imagen, se realizan los siguientes pasos:
  1. Convertir a escala de grises
  2. Tomar un parche aleatorio en la imagen, de tamaño fijo.

Generación del dataset

- En el paper se utilizaron las imágenes del dataset MS-COCO 2014. En este trabajo, las del MS-COCO 2017, que tiene mayor cantidad de imágenes.
- Para cada imagen, se realizan los siguientes pasos:
  1. Convertir a escala de grises
  2. Tomar un parche aleatorio en la imagen, de tamaño fijo.
  3. Distorsionar las 4 esquinas del parche, a lo sumo ρ píxeles (por ej, 64).

Generación del dataset

- En el paper se utilizaron las imágenes del dataset MS-COCO 2014. En este trabajo, las del MS-COCO 2017, que tiene mayor cantidad de imágenes.
- Para cada imagen, se realizan los siguientes pasos:
  1. Convertir a escala de grises
  2. Tomar un parche aleatorio en la imagen, de tamaño fijo.
  3. Distorsionar las 4 esquinas del parche, a lo sumo ρ píxeles (por ej, 64).
  4. Obtener la homografía a partir del parche distorsionado, y transformar la imagen.

Generación del dataset

- En el paper se utilizaron las imágenes del dataset MS-COCO 2014. En este trabajo, las del MS-COCO 2017, que tiene mayor cantidad de imágenes.
- Para cada imagen, se realizan los siguientes pasos:
  1. Convertir a escala de grises
  2. Tomar un parche aleatorio en la imagen, de tamaño fijo.
  3. Distorsionar las 4 esquinas del parche, a lo sumo ρ píxeles (por ej, 64).
  4. Obtener la homografía a partir del parche distorsionado, y transformar la imagen.
  5. Generar la segunda imagen con el parche original sobre la imagen transformada.

Generación del dataset

Figura: Parche aleatorio sobre imagen.

Figura: Distorsión de las esquinas del parche.

Generación del dataset

Figura: Transformación correspondiente al parche distorsionado.

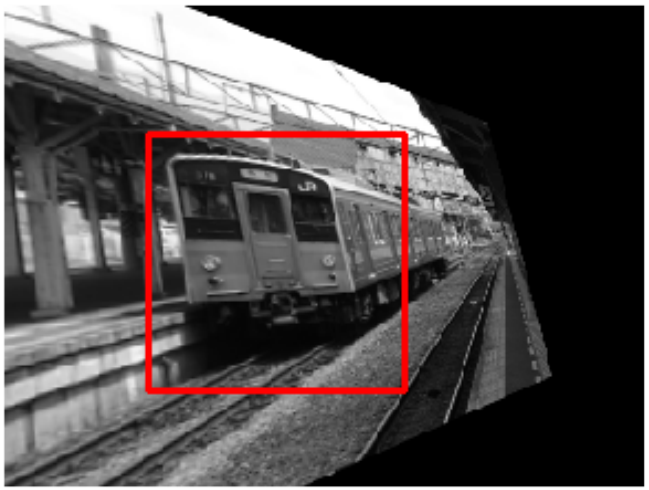


Figura: Captura del parche original sobre imagen transformada.

Generación del dataset

Figura: Par de imágenes para el dataset del modelo de regresión.

Dataset de entrenamiento y evaluación

- Para cada imagen de las 40670 imágenes de COCO, generamos 10 pares.

Dataset de entrenamiento y evaluación

- Para cada imagen de las 40670 imágenes de COCO, generamos 10 pares.
- Total de imágenes para entrenar: **406.700**.

Dataset de entrenamiento y evaluación

- Para cada imagen de las 40670 imágenes de COCO, generamos 10 pares.
- Total de imágenes para entrenar: **406.700**.
- Para la evaluación, se utilizó el dataset de MS COCO 2017 de validación, que consiste de al rededor de **5.000** imágenes.

Modelo de regresión

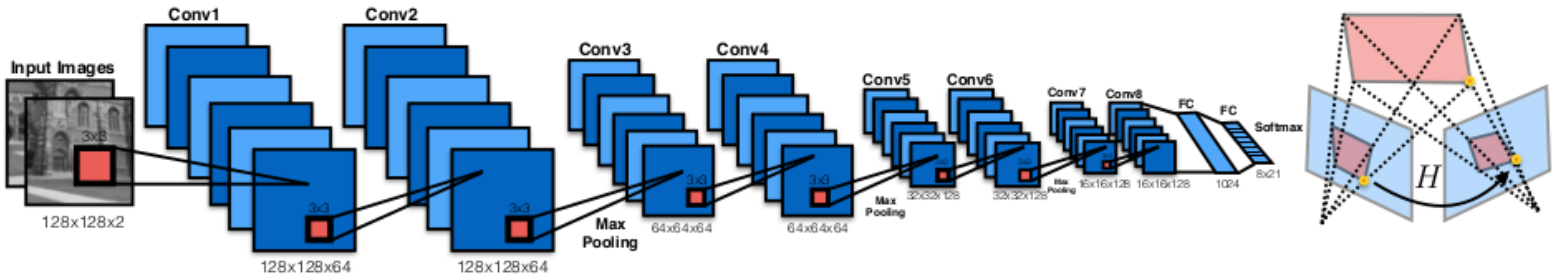


Figura: Modelo de regresión Deep Homography

- La arquitectura propuesta es similar a la arquitectura VGG16.

Modelo de regresión

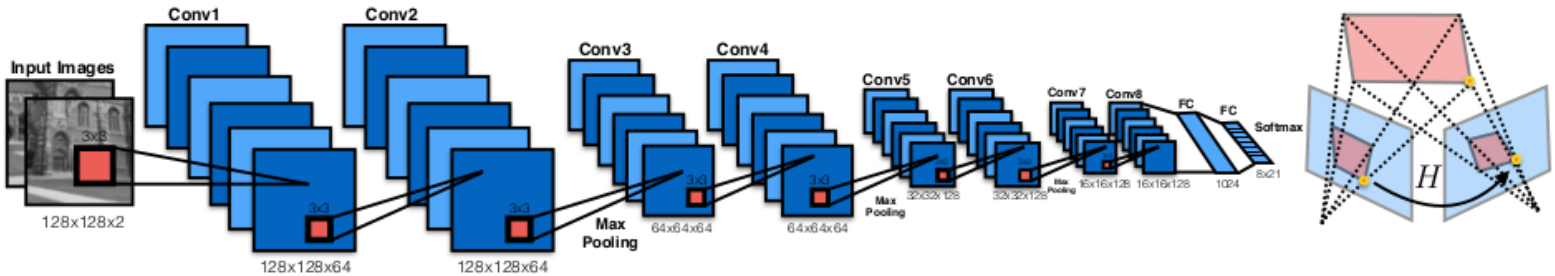


Figura: Modelo de regresión Deep Homography

- La arquitectura propuesta es similar a la arquitectura VGG16.
- Consta de 10 capas:
  1. 8 capas convolucionales

Modelo de regresión

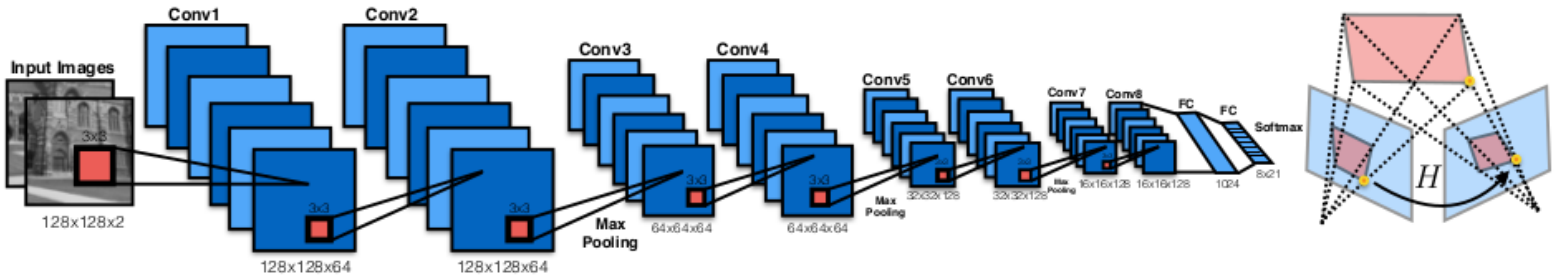


Figura: Modelo de regresión Deep Homography

- La arquitectura propuesta es similar a la arquitectura VGG16.
- Consta de 10 capas:
  1. 8 capas convolucionales
  2. 2 capas densas o completamente conectadas (*fully-connected*, *FC*)

Modelo de regresión

- Las capas convolucionales son capas cuyas unidades están conectadas de tal manera que puedan aprender los pesos correspondientes al kernel de una operación convolucional.

Modelo de regresión

- Las capas convolucionales son capas cuyas unidades están conectadas de tal manera que puedan aprender los pesos correspondientes al kernel de una operación convolucional.
- Las capas densas son capas cuyas unidades estan conectadas entre sí por completo.

Modelo de regresión

- Las capas convolucionales son capas cuyas unidades están conectadas de tal manera que puedan aprender los pesos correspondientes al kernel de una operación convolucional.
- Las capas densas son capas cuyas unidades estan conectadas entre sí por completo.
- La función de error (*loss function*) utilizada es la distancia L2 euclideana.

Modelo de regresión

- Las capas convolucionales son capas cuyas unidades están conectadas de tal manera que puedan aprender los pesos correspondientes al kernel de una operación convolucional.
- Las capas densas son capas cuyas unidades estan conectadas entre sí por completo.
- La función de error (*loss function*) utilizada es la distancia L2 euclideana.
- La última capa densa no posee función de activación, dado que es un modelo de regresión.

Parámetros de Entrenamiento

- Se entrenó por medio de Descenso por Gradiente Estocástico (SGD), con un *learning rate* (LR) de 0.005

Parámetros de Entrenamiento

- Se entrenó por medio de Descenso por Gradiente Estocástico (SGD), con un *learning rate* (LR) de 0.005
- El LR se decrementa de a poco a medida que se sigue entrenando (cada 10 *epochs*)

Parámetros de Entrenamiento

- Se entrenó por medio de Descenso por Gradiente Estocástico (SGD), con un *learning rate* (LR) de 0.005
- El LR se decrementa de a poco a medida que se sigue entrenando (cada 10 *epochs*)
- Regularizadores:
  - Hay capas de *Batch Normalization* entre cada par de capas convolucionales, para normalizar los valores entre capa y capa.

Parámetros de Entrenamiento

- Se entrenó por medio de Descenso por Gradiente Estocástico (SGD), con un *learning rate* (LR) de 0.005
- El LR se decrementa de a poco a medida que se sigue entrenando (cada 10 *epochs*)
- Regularizadores:
  - Hay capas de *Batch Normalization* entre cada par de capas convolucionales, para normalizar los valores entre capa y capa.
  - Se agregaron capas de *Dropout* antes de las capas densas, para evitar sobreajuste.

Parámetros de Entrenamiento

- Se entrenó por medio de Descenso por Gradiente Estocástico (SGD), con un *learning rate* (LR) de 0.005
- El LR se decrementa de a poco a medida que se sigue entrenando (cada 10 *epochs*)
- Regularizadores:
  - Hay capas de *Batch Normalization* entre cada par de capas convolucionales, para normalizar los valores entre capa y capa.
  - Se agregaron capas de *Dropout* antes de las capas densas, para evitar sobreajuste.
- Se entrenó con 64 imágenes por lote (*batch size*).

Métrica de evaluación

- La métrica utilizada al evaluar es el Mean Average Corner Error (MACE).

Métrica de evaluación

- La métrica utilizada al evaluar es el Mean Average Corner Error (MACE).
- Se calcula como la distancia L2 entre las posiciones de las esquinas según las etiquetas y las predicciones.

Métrica de evaluación

- La métrica utilizada al evaluar es el Mean Average Corner Error (MACE).
- Se calcula como la distancia L2 entre las posiciones de las esquinas según las etiquetas y las predicciones.
- El error se promedia sobre los 4 puntos, y luego se promedia sobre todo el conjunto de imágenes.

Evaluación

- Se evaluó la performance del modelo contra otros métodos *baselines*:

Evaluación

- Se evaluó la performance del modelo contra otros métodos *baselines*:
  1. **Homografía identidad**, es decir, aquella que no aplica ninguna transformación. Es una buena referencia, facil de implementar, de una muy mala estimación de homografía.

Evaluación

- Se evaluó la performance del modelo contra otros métodos *baselines*:
  1. **Homografía identidad**, es decir, aquella que no aplica ninguna transformación. Es una buena referencia, facil de implementar, de una muy mala estimación de homografía.
  2. **Homografía ORB+*findHomography***, implementación tradicional utilizando OpenCV. Referencia de una homografía estimada con métodos clásicos, muy usados en la práctica.

Evaluación

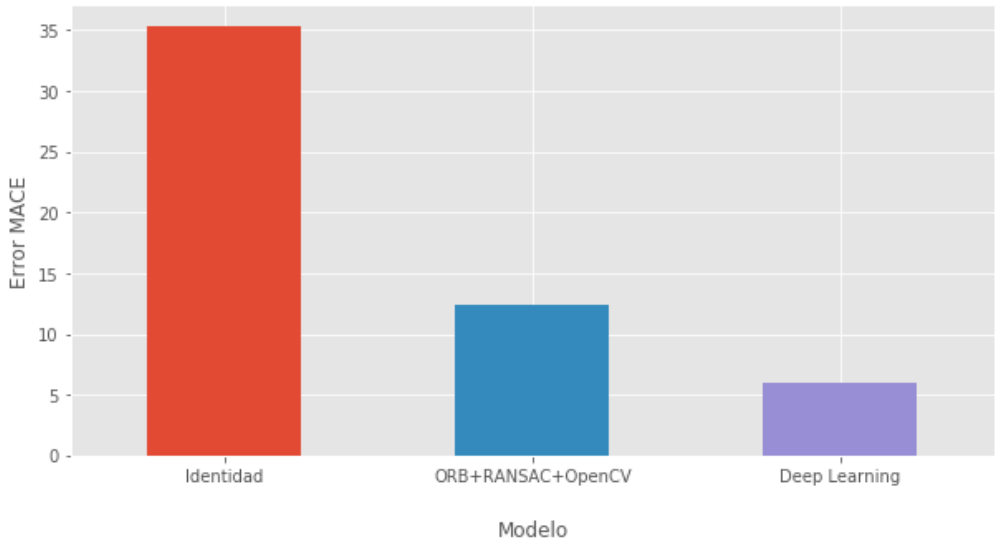


Figura: Comparación del modelo vs. otras alternativas

Variación de ρ

- Es parámetro ρ queda fijo al momento de crear el dataset.

Variación de ρ

- Es parámetro ρ queda fijo al momento de crear el dataset.
- *Determina la dificultad del problema a resolver*: Es el valor máximo que puede tomar la distorsión de cada esquina del parche tomado.

Variación de ρ

- Es parámetro ρ queda fijo al momento de crear el dataset.
- *Determina la dificultad del problema a resolver*: Es el valor máximo que puede tomar la distorsión de cada esquina del parche tomado.
- Valores chicos: poca distorsión; Valores grandes: gran distorsión, y mayor dificultad del modelo para aprender la homografía.

Variación de ρ: Resultados

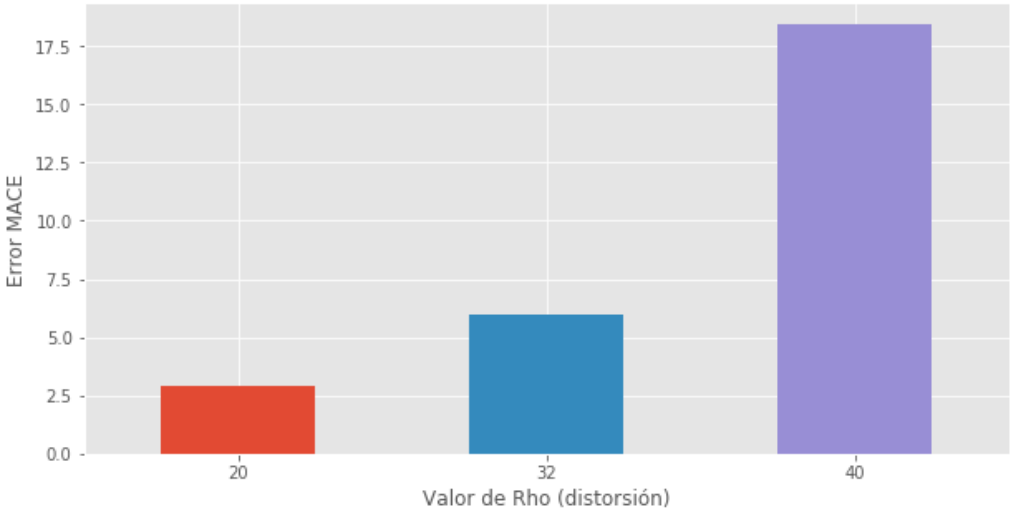


Figura: Comparación de MACE variando ρ en 20, 32, 40

Ejemplo 1 - $\rho = 20$, MACE = 1,449342


Figura: Par de imágenes


Figura: Verdad vs. Predicción

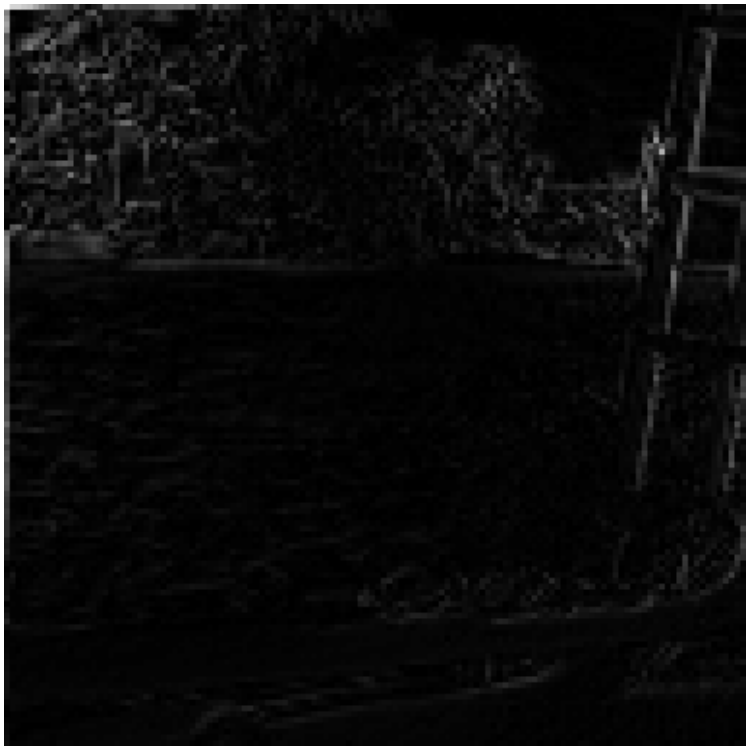


Figura: Diferencia entre verdad y predicción

Ejemplo 2 - $\rho = 20$, MACE = 2,8664753


Figura: Par de imágenes


Figura: Verdad vs. Predicción

Figura: Diferencia entre verdad y predicción

Ejemplo 3 - $\rho = 20$, MACE = 1,1404595


Figura: Par de imágenes


Figura: Verdad vs. Predicción

Figura: Diferencia entre verdad y predicción

Ejemplo 1 - $\rho = 32$, MACE = 11,794792


Figura: Par de imágenes


Figura: Verdad vs. Predicción

Figura: Diferencia entre verdad y predicción

Ejemplo 2 - $\rho = 32$, MACE = 8,77356


Figura: Par de imágenes


Figura: Verdad vs. Predicción

Figura: Diferencia entre verdad y predicción

Ejemplo 3 - $\rho = 32$, MACE = 10,546589


Figura: Par de imágenes


Figura: Verdad vs. Predicción

Figura: Diferencia entre verdad y predicción

Ejemplo 1 - $\rho = 40$, MACE = 22,574299


Figura: Par de imágenes


Figura: Verdad vs. Predicción

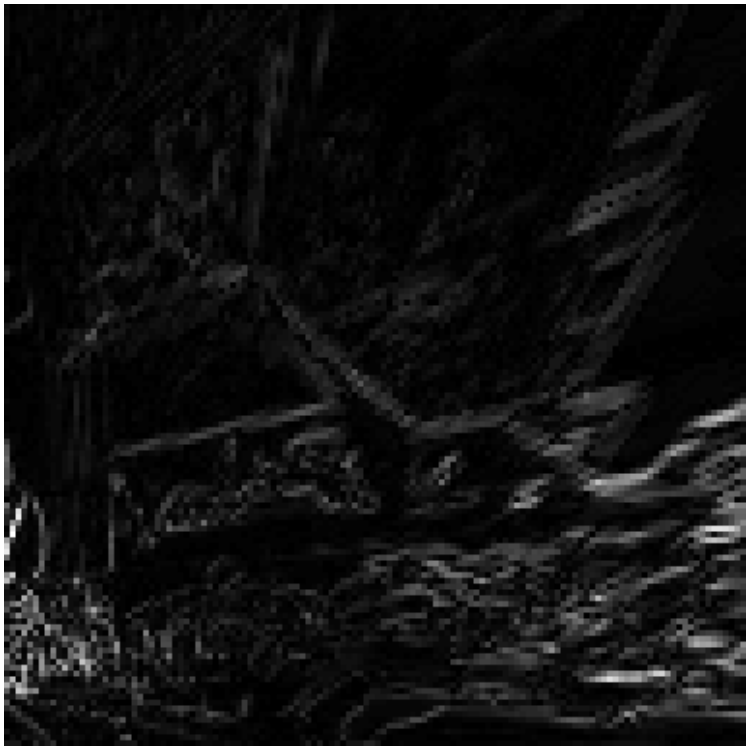


Figura: Diferencia entre verdad y predicción

Ejemplo 2 - $\rho = 40$, MACE = 22,364542


Figura: Par de imágenes


Figura: Verdad vs. Predicción

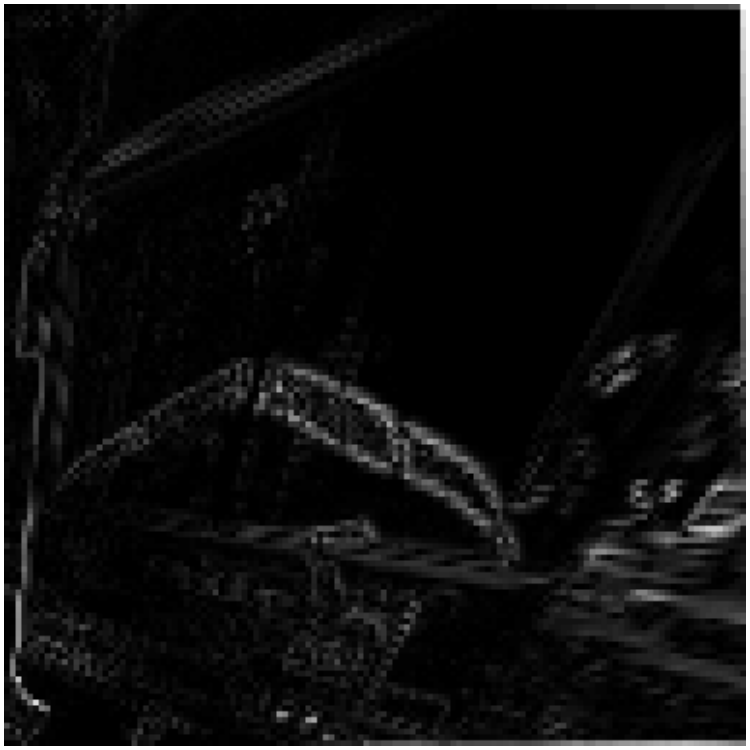


Figura: Diferencia entre verdad y predicción

Ejemplo 3 - $\rho = 40$, MACE = 25,489574

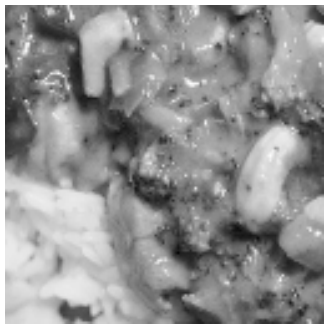
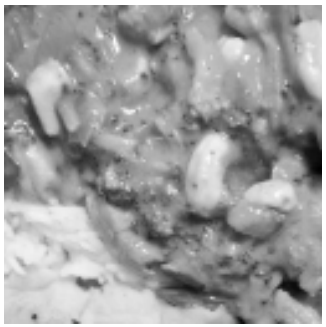



Figura: Par de imágenes

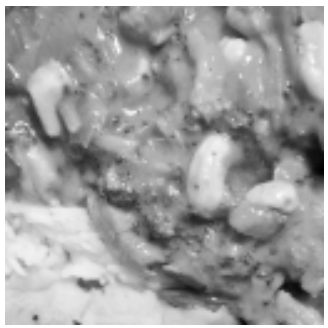
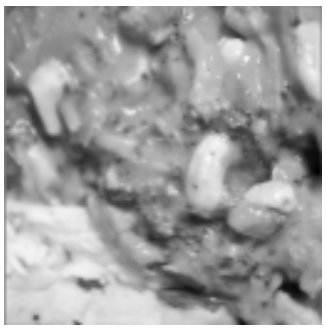



Figura: Verdad vs. Predicción

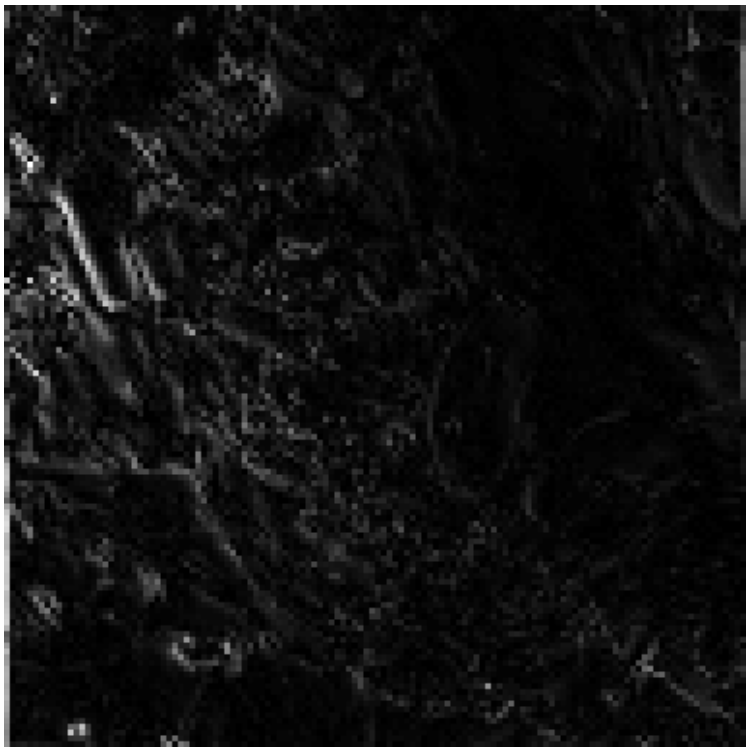


Figura: Diferencia entre verdad y predicción

Sin regularizadores

- Se evaluó también comparar la performance sin las capas de Dropout y Batch normalization.

Sin regularizadores

- Se evaluó también comparar la performance sin las capas de Dropout y Batch normalization.
- **Dropout** es un regularizador que combate el sobreajuste. Durante entrenamiento, en cada etapa de backpropagation, desactiva un número de unidades de la capa, elegidas aleatoriamente.

Sin regularizadores

- Se evaluó también comparar la performance sin las capas de Dropout y Batch normalization.
- **Dropout** es un regularizador que combate el sobreajuste. Durante entrenamiento, en cada etapa de backpropagation, desactiva un número de unidades de la capa, elegidas aleatoriamente.
- **Batch normalization** se asegura que los pesos estén escalados. Normaliza la salida de la activación previa, restándole el promedio del batch y dividiendo por el desvıo estandard del batch.

Sin regularizadores: Resultados

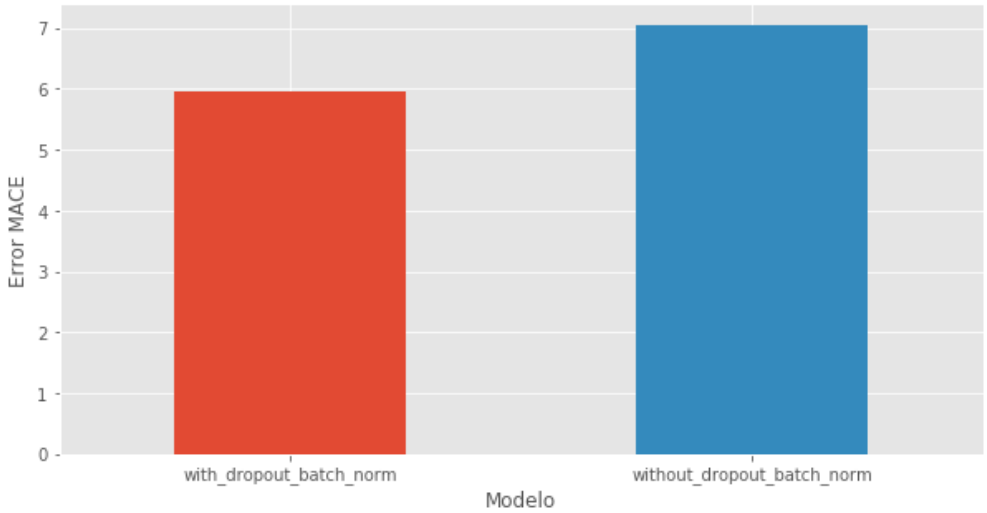


Figura: Experimento removiendo capas de dropout y batch norm

Transfer Learning

- Transfer learning es una técnica muy utilizada en redes convolucional en la práctica.

Transfer Learning

- Transfer learning es una técnica muy utilizada en redes convolucional en la práctica.
- Es muy util cuando se tienen pocos datos para entrenar sobre una red muy profunda.

Transfer Learning

- Transfer learning es una técnica muy utilizada en redes convolucional en la práctica.
- Es muy util cuando se tienen pocos datos para entrenar sobre una red muy profunda.
- Idea:
  1. Tomar un modelo de una arquitectura ya entrenada con un dataset genérico.

Transfer Learning

- Transfer learning es una técnica muy utilizada en redes convolucional en la práctica.
- Es muy util cuando se tienen pocos datos para entrenar sobre una red muy profunda.
- Idea:
  1. Tomar un modelo de una arquitectura ya entrenada con un dataset genérico.
  2. Reemplazar la última capa densa por una que se adecue a nuestra salida

Transfer Learning

- Transfer learning es una técnica muy utilizada en redes convolucional en la práctica.
- Es muy util cuando se tienen pocos datos para entrenar sobre una red muy profunda.
- Idea:
  1. Tomar un modelo de una arquitectura ya entrenada con un dataset genérico.
  2. Reemplazar la última capa densa por una que se adecue a nuestra salida
  3. Congelar todas las capas salvo la(s) última(s) (dependiendo de qué tanto queremos que reaprenda nuevos patrones)

Transfer Learning

- Transfer learning es una técnica muy utilizada en redes convolucional en la práctica.
- Es muy util cuando se tienen pocos datos para entrenar sobre una red muy profunda.
- Idea:
  1. Tomar un modelo de una arquitectura ya entrenada con un dataset genérico.
  2. Reemplazar la última capa densa por una que se adecue a nuestra salida
  3. Congelar todas las capas salvo la(s) última(s) (dependiendo de qué tanto queremos que reaprenda nuevos patrones)
  4. Reentrenar sobre nuestro dataset.

¿Por qué funciona?

- Las primeras capas aprenden atributos básicos comunes en toda imagen: bordes, esquinas, features.

¿Por qué funciona?

- Las primeras capas aprenden atributos básicos comunes en toda imagen: bordes, esquinas, features.
- Las capas cada vez van aprendiendo atributos más genéricos y específicos de los objetos en las imágenes.

¿Por qué funciona?

- Las primeras capas aprenden atributos básicos comunes en toda imagen: bordes, esquinas, features.
- Las capas cada vez van aprendiendo atributos más genéricos y específicos de los objetos en las imágenes.
- Luego, las primeras capas suelen ser reutilizables dado que suelen ser útiles en las imagenes de un dataset nuevo.

TL para homografías

- **Idea**: Aplicar TL sobre alguna red entrenada con millones de imágenes para estimar homografías.

TL para homografías

- **Idea**: Aplicar TL sobre alguna red entrenada con millones de imágenes para estimar homografías.
- Pero...

TL para homografías

- **Idea**: Aplicar TL sobre alguna red entrenada con millones de imágenes para estimar homografías.
- Pero... no existen redes para estimar homografías.

TL para homografías

- **Idea**: Aplicar TL sobre alguna red entrenada con millones de imágenes para estimar homografías.
- Pero... no existen redes para estimar homografías.
- Luego, intentamos utilizar una red para clasificar objetos y readaptarla utilizando las ideas de Deep Homography.

TL para homografías

- **Idea**: Aplicar TL sobre alguna red entrenada con millones de imágenes para estimar homografías.
- Pero... no existen redes para estimar homografías.
- Luego, intentamos utilizar una red para clasificar objetos y readaptarla utilizando las ideas de Deep Homography.
- La red utilizada es **MobileNetV2**, que tiene buenos resultados y requiere poco cómputo.

TL para homografías

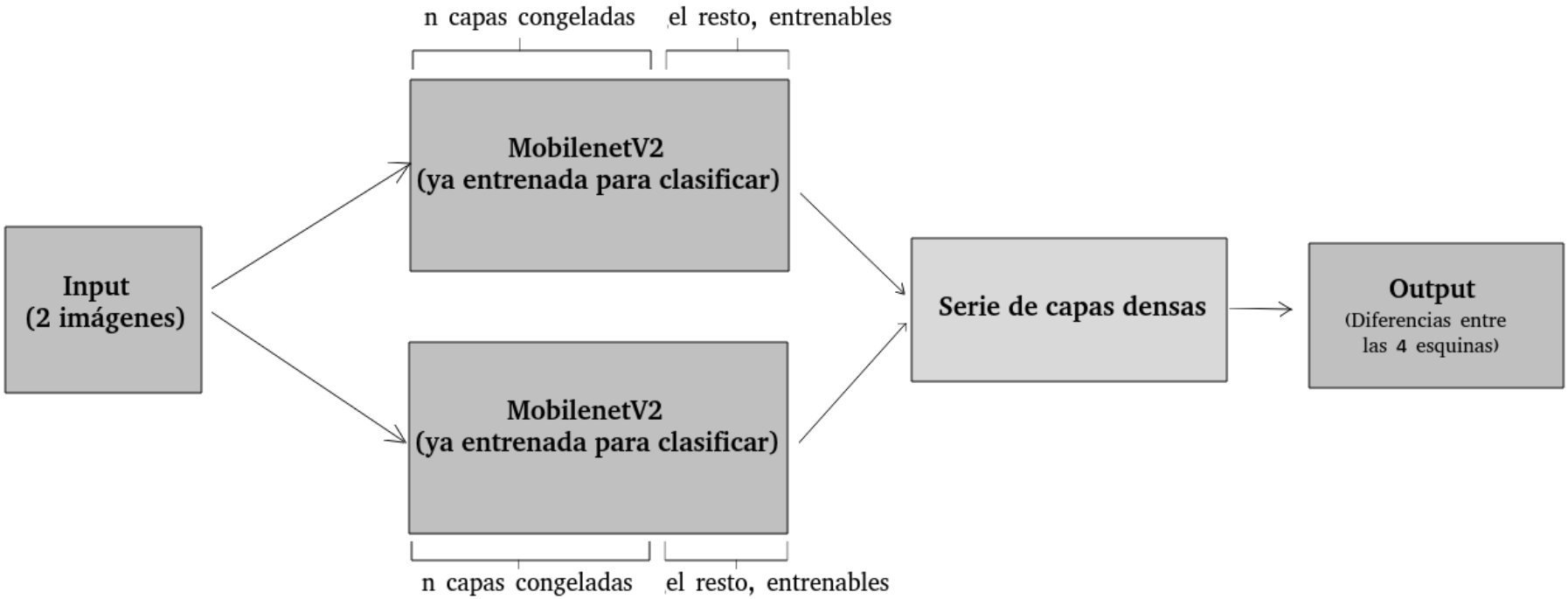


Figura: Modelo para estimar homografías usando MobileNetV2

TL para homografías: Resultados

- La red no logró aprender lo necesario para alcanzar la performance de Deep Homography.

TL para homografías: Resultados

- La red no logró aprender lo necesario para alcanzar la performance de Deep Homography.
- Los mejores resultados se lograron con:
  - Las primeras 130 capas de las redes MobileNetV2 congeladas (al entrenar).

TL para homografías: Resultados

- La red no logró aprender lo necesario para alcanzar la performance de Deep Homography.
- Los mejores resultados se lograron con:
  - Las primeras 130 capas de las redes MobileNetV2 congeladas (al entrenar).
  - La serie de capas densas consistiendo en una de 1024, y luego una final de 8.

TL para homografías: Resultados

- La red no logró aprender lo necesario para alcanzar la performance de Deep Homography.
- Los mejores resultados se lograron con:
  - Las primeras 130 capas de las redes MobileNetV2 congeladas (al entrenar).
  - La serie de capas densas consistiendo en una de 1024, y luego una final de 8.
  - Regularización L2 de 0.01.

TL para homografías: Resultados

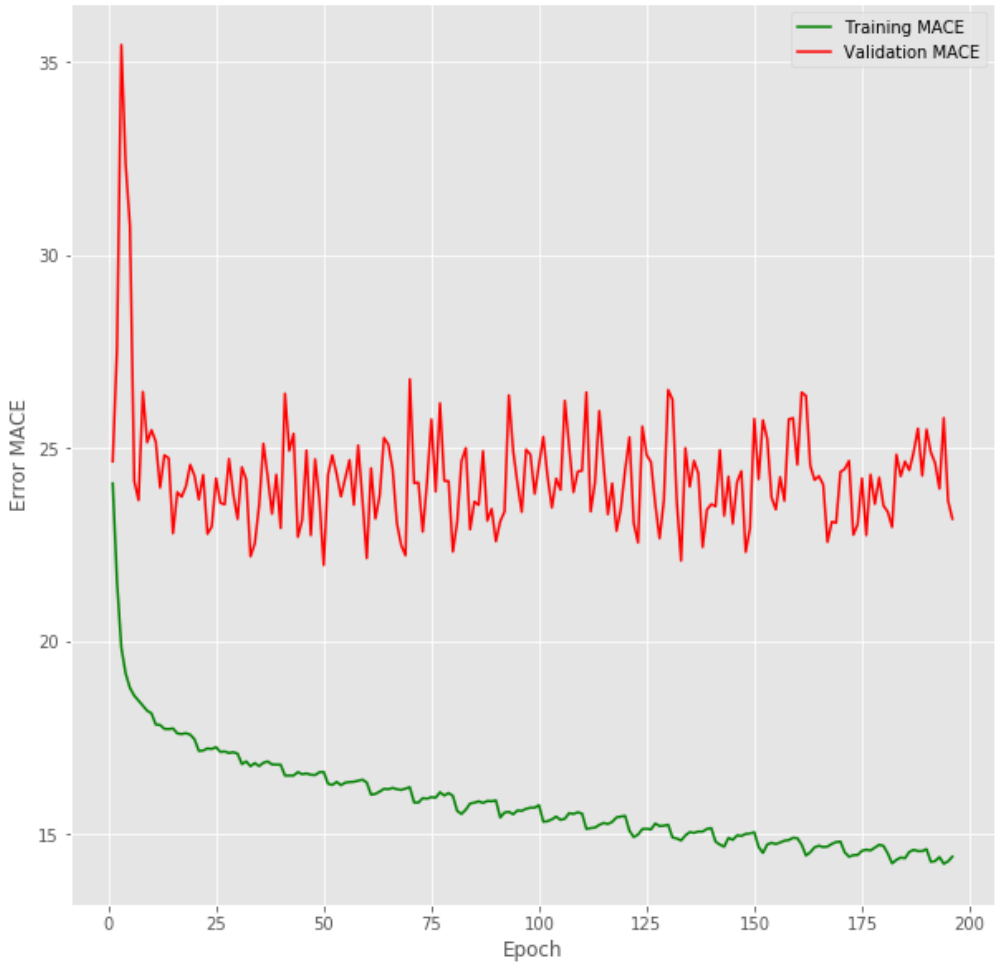


Figura: Entrenamiento del modelo de transfer learning

Conclusiones

- Se logró implementar el modelo y corroborar los resultados del paper, y en ciertos casos mejorarlos, quizás debido a un mejor dataset o mejores parámetros de entrenamiento.

Conclusiones

- Se logró implementar el modelo y corroborar los resultados del paper, y en ciertos casos mejorarlos, quizás debido a un mejor dataset o mejores parámetros de entrenamiento.
- No sólo fue desafiante para nosotros armar y entrenar el modelo, si no también la generación del dataset, dado que tuvimos que recurrir a los conocimientos aprendidos en la materia.

Conclusiones

- Se logró implementar el modelo y corroborar los resultados del paper, y en ciertos casos mejorarlos, quizás debido a un mejor dataset o mejores parámetros de entrenamiento.
- No sólo fue desafiante para nosotros armar y entrenar el modelo, si no también la generación del dataset, dado que tuvimos que recurrir a los conocimientos aprendidos en la materia.
- Aprendimos a mantener orden y claridad al hacer los experimentos, dado que alto costo temporal de correrlos.

Conclusiones

- Se logró implementar el modelo y corroborar los resultados del paper, y en ciertos casos mejorarlos, quizás debido a un mejor dataset o mejores parámetros de entrenamiento.
- No sólo fue desafiante para nosotros armar y entrenar el modelo, si no también la generación del dataset, dado que tuvimos que recurrir a los conocimientos aprendidos en la materia.
- Aprendimos a mantener orden y claridad al hacer los experimentos, dado que alto costo temporal de correrlos.
- El uso de la biblioteca Keras para Python nos facilitó muchísimo en la implementación de la red, abstrayéndonos de conceptos que no dominamos.

Trabajo futuro

- Intentar mejorar la performance del modelo de TL: Probar capas convolucionales en vez de densas; usar más imágenes; data augmentation más agresivo.

Trabajo futuro

- Intentar mejorar la performance del modelo de TL: Probar capas convolucionales en vez de densas; usar más imágenes; data augmentation más agresivo.
- Trabajar con imágenes a color. El modelo trabaja ocn imágenes en escala de grises, pero quizás la información de color aporte al encontrar buenos descriptores.

Trabajo futuro

- Intentar mejorar la performance del modelo de TL: Probar capas convolucionales en vez de densas; usar más imágenes; data augmentation más agresivo.
- Trabajar con imágenes a color. El modelo trabaja ocn imágenes en escala de grises, pero quizás la información de color aporte al encontrar buenos descriptores.
- Comparación de tiempo en la predicción sobre método tradicional.

Trabajo futuro

- Intentar mejorar la performance del modelo de TL: Probar capas convolucionales en vez de densas; usar más imágenes; data augmentation más agresivo.
- Trabajar con imágenes a color. El modelo trabaja ocn imágenes en escala de grises, pero quizás la información de color aporte al encontrar buenos descriptores.
- Comparación de tiempo en la predicción sobre método tradicional.
- Evaluar sobre un dataset de homografías real. El utilizado para entrenar es un dataset artificial.

¿Preguntas?

¿Preguntas?