Detección de Puntos Característicos en Rostros (Face Keypoints) mediante Reconocimiento Visual con Deep Learning

Mauricio Figueroa Colarte
Fundación Instituto Profesional Duoc UC, Viña del Mar, Chile
maur.figueroac@profesor.duoc.cl

Resumen: Este proyecto aborda la detección de puntos característicos en rostros, una tarea esencial en aplicaciones como la estimación de pose, análisis geométrico del rostro y estimación de estados de ánimo. La problemática radica en la necesidad de identificar con precisión los puntos clave del rostro en imágenes, desafiado por variaciones en expresiones faciales, iluminación y poses. El reconocimiento visual con Deep Learning ha emergido como una solución efectiva para este desafío, permitiendo la creación de modelos altamente precisos y eficientes. Para ello, se diseñó una red convolucional basada en una arquitectura ResNet-18 combinada con un Perceptrón Multicapa (MLP – MultiLayer Perceptron), entrenada para estimar 136 valores correspondientes a 68 keypoints faciales. Utilizando un conjunto de datos de 3462 imágenes para entrenamiento y 770 para validación, el preprocesamiento se realizó con RetinaFace, un detector facial de vanguardia. La implementación del modelo incluyó la función de pérdida MSE (Mean Square Error) y se evaluó con la métrica RMSE (Root Mean Square Error) del conjunto de validación, logrando dicha métrica más baja y estable en un valor de 0.0343 en la época 40. La evaluación final se realizó con 5 muestras de imágenes de prueba incluidas y no incluidas en los conjuntos de entrenamiento y validación, destacando la precisión y robustez del modelo en el reconocimiento facial.

Palabras Clave: Keypoints Faciales, CNN, Deep Learning, Reconocimiento Visual, ResNet-18

1. Introducción

1.1. Contexto y propuesta de solución

La aplicación de un modelo de regresión para la detección de puntos característicos del rostro ha demostrado ser fundamental en diversos campos. Este modelo permite una estimación precisa y normalización de la pose facial, lo que facilita el desarrollo de sistemas de reconocimiento facial más robustos y precisos (Bulat & Tzimiropoulos, 2017).

En el ámbito de la psicología, la detección de keypoints faciales es esencial para la estimación de estados de ánimo y emociones, permitiendo la creación de herramientas avanzadas para el análisis del comportamiento humano y la salud mental (Corneanu et al., 2016). Por ejemplo, el proyecto Affectiva utiliza esta tecnología para analizar las expresiones faciales y evaluar las respuestas emocionales en tiempo real, aplicándose en áreas como la investigación de mercado y la salud mental (Affectiva , 2017).

En la industria del entretenimiento, la detección de puntos faciales se utiliza para la animación facial en películas y videojuegos, mejorando la realidad y expresividad de los personajes animados (Karras et al., 2019). En el ámbito de la seguridad, la tecnología se emplea en sistemas de vigilancia para identificar y rastrear individuos, mejorando la capacidad de respuesta ante incidentes y amenazas. Un ejemplo reciente es el uso de reconocimiento facial en aeropuertos, como el de Dubai, para agilizar el proceso de embarque y aumentar la seguridad (McConvey, 2023).

En la medicina, la detección precisa de puntos faciales es relevante para procedimientos de cirugía reconstructiva y tratamientos estéticos, optimizando los resultados y personalizando los tratamientos (Zhang et al., 2023). Además, en la lucha contra la pandemia de COVID-19, esta tecnología se ha utilizado para monitorear el uso de mascarillas y el distanciamiento social (Vibhuti et al., 2022).
En la educación, se ha aplicado para evaluar el compromiso y las emociones de los estudiantes durante las clases virtuales, proporcionando datos valiosos para mejorar las metodologías de enseñanza (Gupta et al., 2023).

Todos estos trabajos relacionados muestran la versatilidad y la importancia de aplicar modelos de regresión para la detección de puntos faciales en distintas áreas de aplicación. El desafío es identificar con precisión los puntos clave del rostro en imágenes. Esto es complejo debido a variaciones en expresiones faciales, iluminación y poses. La necesidad de una solución eficiente ha impulsado el desarrollo de técnicas avanzadas de Deep Learning.

En base al contexto descrito anteriormente, el principal objetivo de este trabajo es desarrollar un modelo para detectar puntos clave faciales con alta precisión. Se diseñó una red convolucional basada en Resnet-18 combinada con un MLP (Multilayer Perceptron). Esta combinación captura características profundas y realiza tareas de regresión. El modelo estima 136 valores correspondientes a 68 keypoints faciales.

1.2 KeyPoints Faciales

Los KeyPoints faciales (también llamados landmarks) son los pequeños puntos magenta que se muestran en cada una de las caras en la Figura 1 (Kumbhare, 2022). En cada imagen de entrenamiento y prueba, hay una única cara y 68 puntos clave, con coordenadas (x, y), para esa cara. Estos puntos clave marcan áreas importantes del rostro: los ojos, las comisuras de la boca, la nariz y otras zonas. Son relevantes para una variedad de tareas, como filtros faciales, reconocimiento de emociones, reconocimiento de poses, entre otras.

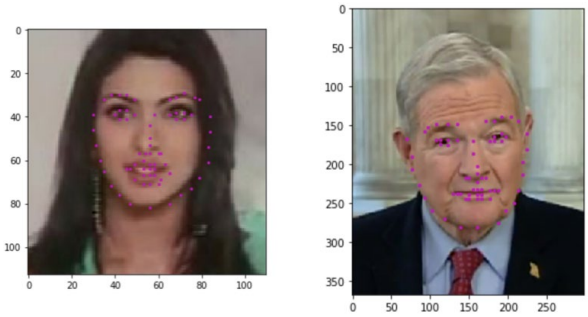

Figura 1: Keypoints faciales como puntos (x,y)

La Figura 2 muestra las zonas enumerados, y se pueden ver que rangos específicos de puntos coinciden con diferentes partes de la cara. A continuación, se presenta la categorización y el rango de números que representa cada sección:

- Contorno del rostro: Puntos: 1-17
- Ceja derecha: Puntos: 18-22
- Ceja izquierda: Puntos: 23-27
- Puente de la nariz: Puntos: 28-31
- Parte inferior de la nariz: Puntos: 32-36
- Ojo derecho: Puntos: 37-42
- Ojo izquierdo: Puntos: 43-48
- Labio exterior: Puntos: 49-60
- Labio interior: Puntos: 61-68

Esta categorización de los landmarks permite segmentar el rostro en diferentes áreas de interés para tareas de análisis facial y procesamiento de imágenes. (Kumbhare, 2022)

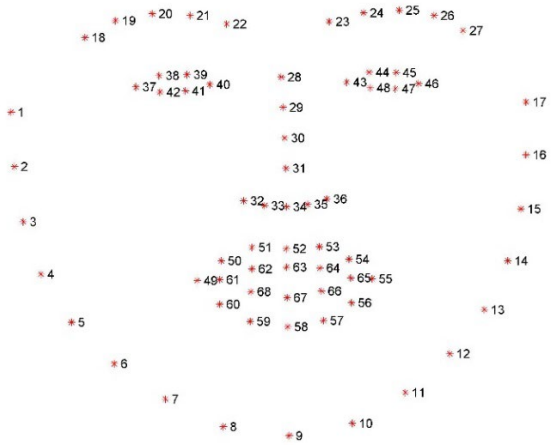

Figura 2: Keypoints numerados y landmarks

1.3 Detección de KeyPoints Faciales

La Figura 3 muestra una imagen de entrada y un gráfico de los puntos estimados por un modelo convolucional.

Para resolver esta tarea se diseñó una red convolucional que se compone de una resnet-18 (He et al., 2015) y una MLP como regresor. La Figura 4 muestra la arquitectura propuesta para resolver este problema. Aquí la cantidad de keypoints es 68, por lo que el modelo (regresor) debe estimar 136 valores, dado que contiene 2 valores (x, y).

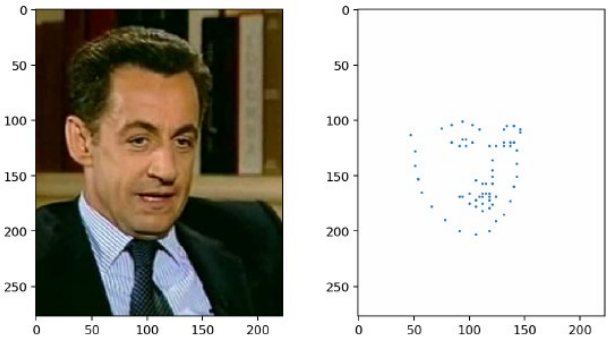

Figura 3: (izquierda) Ejemplo de una imagen de entrada. (derecha) Keypoints de salida.

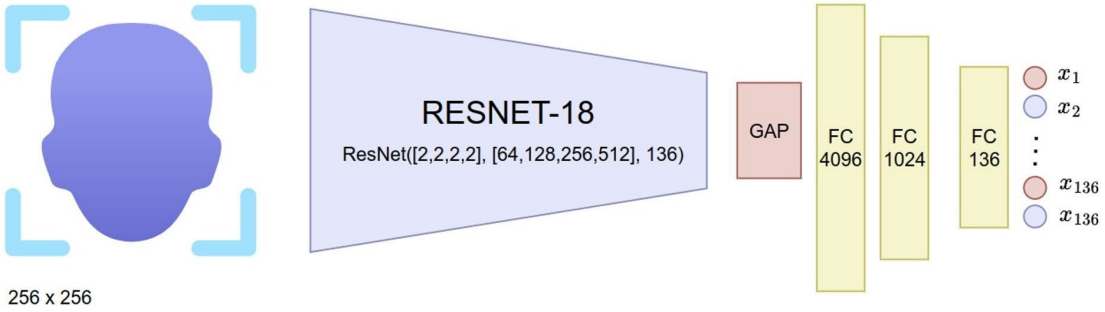

Figura 4: Arquitectura propuesta para la detección de keypoints.

La arquitectura de la Figura 2 presenta un backbone (encoder) ResNet-18, el cual recibe como entrada una imagen de 256 × 256 y devuelve un tensor de 8 × 8 × 512, donde 512 es la cantidad de canales. Esta salida pasa a través de un componente GAP (Global Average Pooling) y tres capas fully-connected (FC). La última de estas capas contiene 136 neuronas que representan la salida del modelo.

La implementación del proyecto se realizó usando la implementación de la Resnet-18 a través del código predefinido según (Saavedra, 2023).

1.4 Metodología de validación del modelo

Para validar el rendimiento del modelo propuesto se propone llevar a cabo mediante dos técnicas de validación, de forma cuantitativa y cualitativa.

1.4.1 Validación cuantitativa

Desde la perspectiva cuantitativa se usarán dos métricas claves para el rendimiento del modelo. Para controlar el proceso de entrenamiento de la red neuronal artificial y su convergencia hacia el objetivo deseado, se utilizará la métrica MSE (Mean Square Error) cuya fórmula se expresa en (1). Para evaluar el desempeño del modelo usando los datos de validación, se hará con el Error Cuadro Medio (RMSE) cuya fórmula se expresa en (2).

$$MSE(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2 \quad (1)$$

Donde, $\boldsymbol{y}$ representa el valor real e $\hat{\boldsymbol{y}}$ representa el valor predicho.

$$RMSE = \sqrt{\frac{1}{N_v}\sum_{i=1}^{i=N_v} ||p - \hat{p}||_2^2} \quad (2)$$

Donde, N_v es el tamaño del conjunto de validación, $\boldsymbol{p}$ es el punto target y $\hat{\boldsymbol{p}}$ es el inferido. Dado que $\boldsymbol{p}$ es un punto 2D, entonces la diferencia entre el punto target y el inferido se estima como la distancia Euclidiana entre ellos.

1.4.2 Validación cualitativa

Para el caso cualitativo, se tomarán 5 casos de rostros con sus respectivas detecciones generadas por el modelo, y se describirá la precisión de la estimación con respecto a los valores reales. Además, se hará la observación de 5 imágenes nuevas, es decir que no hayan sido usadas para el entrenamiento del modelo y sus respectivas predicciones de keypoints.

2 Desarrollo

Para el desarrollo de la solución se puso en práctica la metodología CRISP-DM (Chapman et al., 2000), estructurada para realizar proyectos de minería de datos, pero que ha sido adoptada para proyecto basados en datos en general. Sus fases son las que se muestran en la Figura 6:

1 **Comprensión del Negocio**: Definir objetivos y requerimientos del negocio.
2 **Comprensión de los Datos**: Recopilar y explorar datos iniciales.
3 **Preparación de los Datos**: Limpiar y preparar datos para análisis.
4 **Modelado**: Seleccionar y aplicar técnicas de modelado.
5 **Evaluación**: Evaluar los modelos y verificar que cumplen los objetivos.
6 **Despliegue**: Implementar el modelo en el entorno de producción. Comprensión del Negocio

Figura 5: Fases de la metodología CRISP-DM

2.1 Determinación de los objetivos del negocio

Los objetivos del negocio en el análisis de puntos clave en imágenes están alineados con las estrategias tecnológicas establecidas por el equipo de desarrollo de inteligencia artificial y visión por computadora. La visión del proyecto se puede resumir en la siguiente frase: "Podemos ver rostros, pero debemos entender cada detalle de ellos". Esto hace alusión a que, aunque se ha avanzado significativamente en la captura y procesamiento de imágenes faciales, existe una necesidad de sistematizar y desarrollar aún más el conocimiento y las técnicas asociadas a la identificación precisa y eficiente de puntos clave, abarcando no solo los aspectos técnicos, sino también sus aplicaciones prácticas en seguridad, entretenimiento, y análisis de comportamiento humano.

2.1.1 Objetivo General

Desarrollar un sistema avanzado de análisis de puntos clave en imágenes faciales que permita identificar y procesar con alta precisión y eficiencia los detalles relevantes, optimizando su aplicación en áreas como la seguridad biométrica, el entretenimiento digital y el análisis de comportamiento humano, entre otras, de manera que el modelo obtenido pueda servir para diversas aplicaciones prácticas y efectivas.

2.1.2 Objetivos Específicos

1. Diseñar, entrenar y optimizar un modelo de regresión utilizando la arquitectura ResNet-18 para la identificación precisa de puntos clave en imágenes faciales.
2. Evaluar el rendimiento del modelo de regresión utilizando la métrica RMSE (Root Mean Square Error), asegurando que cumple con los estándares y objetivos del proyecto.
3. Proponer la implementación del modelo en un entorno de producción y desarrollar un plan para futuras mejoras y optimizaciones continuas, garantizando su aplicabilidad en diversas áreas como seguridad biométrica, entretenimiento digital y análisis de comportamiento humano.

2.2 Comprensión de los Datos

2.2.1 Recopilación de datos

Los datos de entrenamiento y validación se encuentran disponibles en formato CSV en (FaceKPoints.Zip, s.f.). Conformado por 3462 imágenes para entrenamiento y 770 para validación.

2.2.2 Descripción de los datos

Tanto los datos de entrenamiento como de validación, corresponden a imágenes de rostros de personas que se estructuran en una carpeta `FaceKeyPoints`, que a su vez contiene dos subcarpetas: `Training` con 3462 imágenes y `Valid` con 2308 imágenes. Estas imágenes se encuentran indexadas, a través de datos estructurados en archivos `training_frames_keypoints.csv` y `valid_frames_keypoints.csv`, respectivamente. La estructura de estos archivos tiene el mismo formato y que se representa según se observa en la Tabla 1.

Tabla 1: Descripción de los datos

Campo	Descripción	Tipo de Datos	Ejemplo de Valor
name	Nombre del archivo de imagen	Texto	Luis_Fonsi_21.jpg
0	Coordenada X del punto clave 1	Número (flotante)	45.0
1	Coordenada Y del punto clave 1	Número (flotante)	98.0
2	Coordenada X del punto clave 2	Número (flotante)	47.0
3	Coordenada Y del punto clave 2	Número (flotante)	106.0
4	Coordenada X del punto clave 3	Número (flotante)	49.0
5	Coordenada Y del punto clave 3	Número (flotante)	110.0
6	Coordenada X del punto clave 4	Número (flotante)	53.0
7	Coordenada Y del punto clave 4	Número (flotante)	119.0
...	...	...	...
134	Coordenada X del punto clave 68	Número (flotante)	77.0
135	Coordenada Y del punto clave 68	Número (flotante)	122.0

Esta tabla describe las columnas de los archivos proporcionados. Los puntos clave (keypoints) se refieren a diferentes posiciones en una imagen y están representados por sus coordenadas X e Y. Cada archivo tiene 136 columnas que representan las coordenadas de 68 puntos clave.

2.2.3 Exploración de los datos

Aunque se asume que el dataset entregado para el proyecto está validado en cuanto a su calidad, se considera conveniente hace un análisis general.

Imágenes

	Name	Format	Size_MB	Width	Height
0	Abdel_Aziz_Al-Hakim_00.jpg	.jpg	0.13	312	270
1	Abdel_Aziz_Al-Hakim_01.jpg	.jpg	0.14	312	270
2	Abdel_Aziz_Al-Hakim_02.jpg	.jpg	0.14	312	270
3	Abdel_Aziz_Al-Hakim_10.jpg	.jpg	0.08	229	238
4	Abdel_Aziz_Al-Hakim_11.jpg	.jpg	0.08	229	238

```
1 df.shape
```

```
(3462, 5)
```

	Size_MB	Width	Height
count	3462.000000	3462.000000	3462.000000
mean	0.055647	195.115251	210.123917
std	0.033490	69.799177	66.375865
min	0.010000	72.000000	87.000000
25%	0.030000	144.000000	160.000000
50%	0.050000	184.500000	201.000000
75%	0.070000	237.000000	246.000000
max	0.230000	494.000000	478.000000

Figura 6: Imágenes FaceKPoints/training

Según se muestra en la Figura 7 están las 3462 imágenes y todas son de formato .jpg y varían en sus dimensiones, por lo que será relevante este punto al momento del preprocesamiento. Por simple inspección, se pudo determinar que en general las imágenes son de buena calidad, sin embargo, hay algunas que difieren en cuanto a nitidez y luminosidad.

	Name	Format	Size_MB	Width	Height
0	Abdel_Aziz_Al-Hakim_00.jpg	.jpg	0.14	312	270
1	Abdel_Aziz_Al-Hakim_01.jpg	.jpg	0.14	312	270
2	Abdel_Aziz_Al-Hakim_10.jpg	.jpg	0.08	229	238
3	Abdel_Aziz_Al-Hakim_11.jpg	.jpg	0.08	229	238
4	Abdel_Aziz_Al-Hakim_40.jpg	.jpg	0.07	270	314

```
1 df.shape
```

```
(2308, 5)
```

	Size_MB	Width	Height
count	2308.000000	2308.000000	2308.000000
mean	0.055858	195.115251	210.123917
std	0.033542	69.804220	66.380661
min	0.010000	72.000000	87.000000
25%	0.030000	144.000000	160.000000
50%	0.050000	184.500000	201.000000
75%	0.070000	237.000000	246.000000
max	0.230000	494.000000	478.000000

Figura 7: Imágenes FaceKPoints/valid

Según se muestra en la Figura 8 están las 2308 imágenes y todas son de formato .jpg y varían en sus dimensiones, por lo que será relevante este punto al momento del preprocesamiento. Por simple inspección, se pudo determinar que en general las imágenes son de buena calidad, sin embargo, hay algunas que difieren en cuanto a nitidez y luminosidad.

Keypoints

	Unnamed: 0	0	1	2	3	4	5	6	7	8	...	126	127	128	129	130	131	132	133	134	135
0	Luis_Fonsi_21.jpg	45.0	98.0	47.0	106.0	49.0	110.0	53.0	119.0	56.0	...	83.0	119.0	90.0	117.0	83.0	119.0	81.0	122.0	77.0	122.0
1	Lincoln_Chafee_52.jpg	41.0	83.0	43.0	91.0	45.0	100.0	47.0	108.0	51.0	...	85.0	122.0	94.0	120.0	85.0	122.0	83.0	122.0	79.0	122.0
2	Valerie_Harper_30.jpg	56.0	69.0	56.0	77.0	56.0	86.0	56.0	94.0	58.0	...	79.0	105.0	86.0	108.0	77.0	105.0	75.0	105.0	73.0	105.0
3	Angelo_Reyes_22.jpg	61.0	80.0	58.0	95.0	58.0	108.0	58.0	120.0	58.0	...	98.0	136.0	107.0	139.0	95.0	139.0	91.0	139.0	85.0	136.0
4	Kristen_Breitweiser_11.jpg	58.0	94.0	58.0	104.0	60.0	113.0	62.0	121.0	67.0	...	92.0	117.0	103.0	118.0	92.0	120.0	88.0	122.0	84.0	122.0

5 rows × 137 columns

```
1 df.shape
```

```
(3462, 137)
```

Figura 8: FaceKPoints de entrenamiento

Según se muestra en la Figura 9 están las 3462 imágenes indexadas son sus respectivos Keypoints, que, además, según muestra la Figura 10, los datos están completos , y según la Figura 11 están en su propia escala.

	Unnamed: 0	0	1	2	3	4	5	6	7	8	...	126	127	128	129	130	131	132	133	134	135
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3457	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
3458	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
3459	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
3460	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
3461	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False

```
1 #cuenta las filas y las columnas del dataset
2 null_count = df.isnull().sum().sum()
3 print('Number of null values:', null_count)
```

```
Number of null values: 0
```

Figura 9: FaceKPoints de entrenamiento sin missing values

	0	1	2	3	4	5	6	7	8	9	...	126
count	3462.000000	3462.000000	3462.000000	3462.000000	3462.000000	3462.000000	3462.000000	3462.000000	3462.000000	3462.000000	...	3462.000000
mean	56.967938	91.170422	57.829001	101.813692	59.875794	111.523397	62.229636	120.947718	65.965627	131.238013	...	103.046216
std	24.778216	29.267132	24.717583	32.760806	25.268444	36.063468	25.972395	39.287462	27.021073	42.915450	...	38.867226
min	-17.000000	23.000000	-23.000000	31.000000	-17.000000	38.000000	-17.000000	44.000000	-17.000000	51.000000	...	32.000000
25%	40.000000	70.000000	41.000000	78.000000	43.000000	85.000000	44.000000	92.000000	47.000000	99.000000	...	75.000000
50%	53.000000	87.000000	54.000000	97.000000	56.000000	106.000000	58.000000	115.000000	61.000000	125.000000	...	97.000000
75%	70.000000	108.000000	71.000000	120.000000	73.000000	133.000000	76.000000	144.000000	81.000000	157.750000	...	124.000000
max	224.000000	210.000000	224.000000	232.000000	224.000000	253.000000	228.000000	269.000000	228.000000	292.000000	...	295.000000

8 rows × 136 columns

Figura 10: FaceKPoints de entrenamiento escala propia

	Unnamed: 0	0	1	2	3	4	5	6	7	8	...	126	127	128	129	130	131	132	133	134	135
0	James_Wolfensohn_00.jpg	25.0	44.0	25.0	49.0	25.0	54.0	25.0	59.0	26.0	...	42.0	65.0	46.0	65.0	42.0	65.0	39.0	65.0	38.0	65.0
1	Valerie_Harper_30.jpg	55.0	62.0	52.0	72.0	52.0	83.0	55.0	90.0	55.0	...	78.0	104.0	88.0	106.0	78.0	106.0	75.0	104.0	70.0	104.0
2	Kristen_Breitweiser_11.jpg	60.0	88.0	60.0	98.0	63.0	109.0	65.0	116.0	68.0	...	96.0	114.0	106.0	117.0	96.0	117.0	94.0	117.0	89.0	117.0
3	Elizabeth_Dole_31.jpg	41.0	96.0	43.0	104.0	47.0	113.0	49.0	120.0	56.0	...	98.0	118.0	103.0	118.0	98.0	120.0	96.0	122.0	94.0	122.0
4	Kit_Bond_20.jpg	35.0	67.0	37.0	74.0	41.0	81.0	44.0	87.0	48.0	...	76.0	96.0	83.0	93.0	76.0	98.0	73.0	98.0	69.0	98.0

5 rows × 137 columns

```
df.shape
```

```
(770, 137)
```

Figura 11: FaceKPoints de validación

Según se muestra en la Figura 12 están sólo 770 imágenes indexadas son sus respectivos Keypoints, que, además, según muestra la Figura 13, los datos están completos, y según la Figura 14 están en su propia escala.

	Unnamed: 0	0	1	2	3	4	5	6	7	8	...	126	127	128	129	130	131	132	133	134	135
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
765	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
766	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
767	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
768	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
769	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False

```
#cuenta las filas y las columnas del dataset
null_count = df.isnull().sum().sum()
print('Number of null values:', null_count)
```

```
Number of null values: 0
```

Figura 12: FaceKPoints de validación sin missing values

	0	1	2	3	4	5	6	7	8	9	...	126	12
count	770.000000	770.000000	770.000000	770.000000	770.000000	770.000000	770.000000	770.000000	770.000000	770.000000	...	770.000000	770.000000
mean	56.990909	90.677922	57.741558	101.392208	59.825974	111.153247	62.171429	120.628571	65.872727	130.932468	...	103.236364	125.74545
std	23.162257	29.027097	23.042042	32.811824	23.506762	36.120588	24.246978	39.524379	25.349149	43.208065	...	37.672585	40.91308
min	2.000000	33.000000	2.000000	40.000000	4.000000	45.000000	6.000000	49.000000	10.000000	53.000000	...	38.000000	48.000000
25%	42.000000	69.000000	42.000000	77.250000	44.000000	85.000000	46.000000	91.000000	48.000000	100.000000	...	76.000000	95.250000
50%	53.000000	87.500000	54.000000	97.000000	56.000000	106.000000	58.000000	115.000000	61.000000	125.000000	...	96.000000	118.000000
75%	70.000000	107.000000	71.000000	120.000000	74.000000	131.750000	76.000000	142.750000	80.000000	154.000000	...	123.000000	150.000000
max	181.000000	205.000000	175.000000	222.000000	175.000000	236.000000	181.000000	254.000000	181.000000	275.000000	...	245.000000	250.000000

8 rows × 136 columns

Figura 13: FaceKPoints de entrenamiento escala propia

2.2.4 Calidad de los datos

Según el análisis preliminar y de alto nivel, realizado anteriormente, se puede apreciar que la calidad de los datos es generalmente buena, con algunas áreas de mejora necesarias. Todas las imágenes, tanto en el conjunto de entrenamiento (3462 imágenes) como en el de validación (2308 imágenes), están en formato .jpg y varían en sus dimensiones, lo que requerirá atención en el preprocesamiento. Visualmente, la mayoría de las imágenes son de buena calidad, aunque algunas muestran variaciones en nitidez y luminosidad.

En cuanto a los Keypoints, las imágenes de entrenamiento están correctamente indexadas y los datos están completos y en su propia escala, lo que facilita su uso directo. Sin embargo, en el conjunto de validación, solo 770 imágenes están indexadas con sus respectivos Keypoints. A pesar de estas diferencias, los datos no presentan valores perdidos y están bien estructurados.

En resumen, aunque los datos son de alta calidad, se debe considerar la variabilidad en las dimensiones de las imágenes y las diferencias en la cantidad de imágenes indexadas con Keypoints entre los conjuntos de entrenamiento y validación.

2.3 Preparación de los Datos

Como se indicó en la sección anterior, la fuente de datos ya se encuentra consolidada en archivos separados por coma (Keypoints), y las imágenes en sus respectivas carpetas. Como el conjunto de datos descrito es clave para obtener buenos resultados, se realizó un proceso de preprocesamiento de las imágenes en conjunto con sus keypoints.

Los datos de entrenamiento y validación se encuentran disponibles en formato CSV en (FaceKPoints.Zip, s.f.). Conformado por 3462 imágenes para entrenamiento y 770 para validación.

Es importante señalar que, cada imagen pasó por un preprocesamiento utilizando RetinaFace (Deng, et al., 2019). RetinaFace es un detector facial de vanguardia basado en deep learning. Detecta rostros con alta precisión, incluso en multitudes. La Figura 5 muestra un ejemplo de una imagen con y sin el preprocesamiento de RetinaFace (Serengil, 2024).

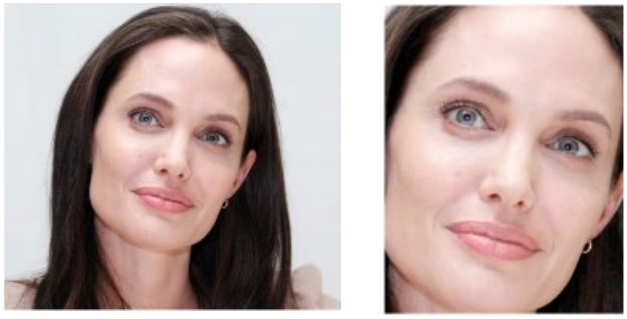

Figura 14: (izquierda) Ejemplo de una imagen de entrada. (derecha) imagen de salida recortada.

Para el entrenamiento, se utilizó un conjunto de datos compuesto por 3462 imágenes. Estas imágenes estaban anotadas con los keypoints faciales. Adicionalmente, se usaron 770 imágenes para la validación del modelo. A continuación, se describe los esencial del proceso de preparación de los datos y los resultados obtenidos, los detalles de la implementación se encuentran en el código desarrollado (Figueroa, 2024) . Cabe señalar que la ejecución de este proceso requirió de aproximadamente 5 horas de procesamiento en CPU.

1.- Definición de la clase FaceKeypointsDataset: Se definió una clase para manejar el dataset de puntos clave faciales, cargando la metadata de las imágenes y sus correspondientes puntos desde archivos CSV según se muestra en la Figura 15.

```python
class FaceKeypointsDataset:
    def __init__(self, csv_file, root_dir):
        self.keypoints_frame = pd.read_csv(csv_file)
        self.root_dir = root_dir

    def __len__(self):
        return len(self.keypoints_frame)

    def __getitem__(self, idx):
        img_name = os.path.join(self.root_dir, self.keypoints_frame.iloc[idx, 0])
        image = cv2.imread(img_name)
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        keypoints = self.keypoints_frame.iloc[idx, 1:].values.astype('float').reshape(-1, 2)
        return {'image': image, 'keypoints': keypoints}
```

Figura 15: Clase FaceKeypointsDataset

2.- Conversión del dataset a tensores: Función *dataset_to_tensor* que convierte, a partir del dataset, las imágenes extraídas desde sus carpetas, se almacenan en arrays numpy con sus respectivos puntos clave según se muestra en la Figura 16.

```python
def dataset_to_tensor(dataset):
    images = []
    keypoints = []

    for i in range(len(dataset)):
        sample = dataset[i]
        images.append(sample['image'])
        keypoints.append(sample['keypoints'])

    images = np.array(images)
    keypoints = np.array(keypoints).reshape(len(keypoints), -1)

    return images, keypoints
```

Figura 16: Función dataset_to_tensor

3.- Ajuste de puntos clave: Función ajustar_keypoints que permite redimensionar los puntos clave consistentemnte con el redimensionamiento de las imágenes, lo cual es necesario para prepararlas como entrada para el entrenamiento (Figura 17).

```python
def ajustar_keypoints(keypoints, old_shape, new_shape):
    old_h, old_w = old_shape
    new_h, new_w = new_shape
    scale_x = new_w / old_w
    scale_y = new_h / old_h
    adjusted_keypoints = keypoints * np.array([scale_x, scale_y])
    return adjusted_keypoints
```

Figura 17: Función ajustar_keypoints

4.- Preprocesamiento de imágenes y puntos clave: Función preprocess_image , que con el soporte de la biblioteca RetinaFace, permite detectar los rostros desde las imágenes, privilegiando aquella cara que esté en primer plano (cuando hay más de una), para posteriormente recortarla, redimensionarla y ajustar los puntos clave a la nueva imagen (Figura 18).

```python
def preprocess_image(image, keypoints):
    faces = RetinaFace.detect_faces(image)

    if isinstance(faces, dict) and len(faces) > 0:
        # Encontrar la cara más grande
        largest_face = None
        largest_area = 0
        for key in faces.keys():
            facial_area = faces[key]['facial_area']
            x1, y1, x2, y2 = facial_area
            area = (x2 - x1) * (y2 - y1)
            if area > largest_area:
                largest_area = area
                largest_face = faces[key]

        if largest_face:
            facial_area = largest_face['facial_area']
            x1, y1, x2, y2 = facial_area

            # Recortar la imagen de acuerdo con la detección de la cara
            cropped_image = image[y1:y2, x1:x2]

            # Redimensionar la imagen recortada a 256x256
            resized_image = cv2.resize(cropped_image, (256, 256))

            # Ajustar los puntos clave originales para que coincidan con la nueva imagen redimensionada
            keypoints_adjusted = keypoints - np.array([x1, y1])
            keypoints_adjusted = ajustar_keypoints(keypoints_adjusted, cropped_image.shape[:2], (256, 256))

            # Normalizar la imagen y los puntos clave
            resized_image = resized_image / 255.0
            keypoints_adjusted = keypoints_adjusted / 256.0

            return resized_image, keypoints_adjusted
    else:
        print("No se detectó ninguna cara en la imagen o el retorno no es un diccionario.")
        return None, None
```

Figura 18: Función preprocess_image

5.- Preprocesamiento del dataset completo: Teniendo como base la ejecución de la función preprocess_image, la función preprocess_dataset, permite el preprocesamiento iterativo de un conjunto completo de imágenes y sus puntos clave (Figura 19). En la ejecución de este proceso fue necesario poner un control para los casos en que RetinaFace no fue capaz de captar un rostro (se detectaron sólo 6 casos), también para que en los casos en que había más de una cara, priorizara la que se encuentra en primer plano (el rostro más grande detectado).

```python
def preprocess_dataset(images, keypoints):
    preprocessed_images = []
    preprocessed_keypoints = []

    for i in range(len(images)):
        preprocessed_image, preprocessed_keypoint = preprocess_image(images[i], keypoints[i].reshape(-1, 2))
        if preprocessed_image is not None:
            preprocessed_images.append(preprocessed_image)
            preprocessed_keypoints.append(preprocessed_keypoint)

    return np.array(preprocessed_images), np.array(preprocessed_keypoints).reshape(-1, 68 * 2)
```

Figura 19: Función preprocess_dataset

Al ejecutar el preprocesamiento del dataset completo, se pueden visualizar los resultados esperados, tomando una imagen de entrenamiento como muestra (Figura 20), y otra imagen de muestra extraída del conjunto de validación (Figura 21).

```python
def preprocess_dataset(images, keypoints):
    preprocessed_images = []
    preprocessed_keypoints = []

    for i in range(len(images)):
        preprocessed_image, preprocessed_keypoint = preprocess_image(images[i], keypoints[i].reshape(-1, 2))
        if preprocessed_image is not None:
            preprocessed_images.append(preprocessed_image)
            preprocessed_keypoints.append(preprocessed_keypoint)

    return np.array(preprocessed_images), np.array(preprocessed_keypoints).reshape(-1, 68 * 2)
```

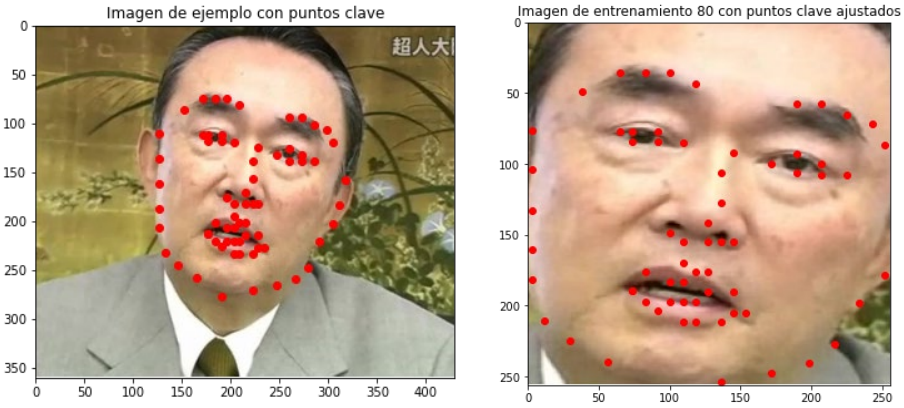

Figura 20: (izquierda) Ejemplo de una imagen de entrenamiento sin procesar con sus keypoints. (derecha) imagen de salida recortada, redimensionada y keypoints ajustados.

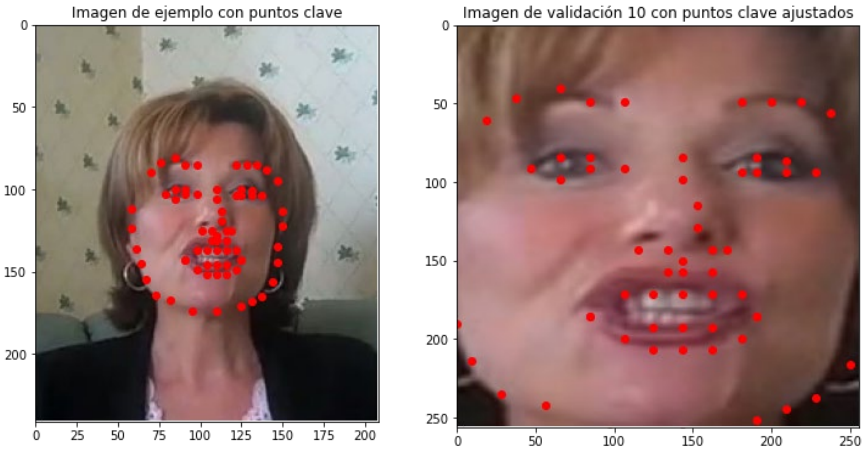

Figura 21: (izquierda) Ejemplo de una imagen de validación sin procesar con sus keypoints. (derecha) imagen de salida recortada, redimensionada y keypoints ajustados.

2.4 Modelado

En esta etapa, se selecciona la arquitectura indicada en la Figura 4 para la construcción y evaluación del modelo a través de experimentos. La idea central de estos experimentos es evaluar la efectividad de la arquitectura ResNet-18 en la predicción de keypoints en imágenes faciales.

En una investigación realizada muy someramente, respecto a los modelos de este tipo, si bien podrían existir otros mejores, para este trabajo se usó ResNet-18 como la arquitectura elegida. Por ello, se aplicará exclusivamente esta configuración de Redes Neuronales Convolucionales para la realización de los experimentos de evaluación.

Los experimentos desarrollados se basaron en sensibilizar el número de épocas de entrenamiento para la validación de la arquitectura. Esto implicó que se hicieron experimentos con 5, 10, 15, 20, 30, 35 hasta 40 épocas. Sin embargo, muchos de estos experimentos fueron fallidos debido a los malos resultados obtenidos, cuya causa fue el mal preprocesamiento de las imágenes. A lo anterior, se suma que se dedicaron muchas horas de procesamiento de CPU para poder evaluar. Por ejemplo, para 40 épocas el proceso demoraba aproximadamente 9 horas, para 35 épocas 8 horas, para 30 épocas 7 horas aproximadamente, si se suman, por lo menos 3 días de prueba y error.

Una vez determinadas las causas de las fallas, se procedió a aplicar la correcta normalización (valores en rango [0,1]), escalamiento de las imágenes y sus respectivos keypoints. En base a los variados experimentos, no todos documentados en este informe, se determinó usar un modelo entrenado en 40 épocas, ya que, a partir de dicha época, convergía mejor y se hacía estable según las mediciones de MSE y RMSE.

A continuación, se describen los aspectos esenciales de la implementación del modelo utilizado para el entrenamiento y validación definitivo (los detalles se ven en el código entregado con este informe).

1.- Carga de datos: En la Figura 22, se muestra la forma en que se cargan las imágenes y los puntos clave preprocesados desde archivos numpy (preprocessed_train_images.npy, preprocessed_train_keypoints.npy,preprocessed_valid_images.npy,preprocessed_valid_keypoints.npy) .

```
train_images=np.load('preprocessed_train_images.npy')
train_keypoints =np.load('preprocessed_train_keypoints.npy')
valid_images=np.load('preprocessed_valid_images.npy')
valid_keypoints =np.load('preprocessed_valid_keypoints.npy')
```

Figura 22: Carga de imágenes y keypoints preprocesados

2.- Definición del modelo: La Figura 23 muestra cómo se define la clase KeypointModel, que implementa la arquitectura ResNet-18 como backbone, a partir del código entregado (Saavedra, 2023). Además, se agregan las capas fully-connected para completar la arquitectura presentada en la Figura 4: Arquitectura propuesta para la detección de keypoints.

```
class KeypointModel(tf.keras.Model):
    def __init__(self):
        super(KeypointModel, self).__init__()
        self.resnet_backbone = ResNetBackbone(block_sizes=[2, 2, 2, 2], filters=[64, 128, 256, 512])

        self.fc = tf.keras.Sequential([
            tf.keras.layers.GlobalAveragePooling2D(), #Capa GAP
            tf.keras.layers.Dense(4096, activation='relu', kernel_initializer='he_normal'),  # Capa FC con 4096 neuronas
            tf.keras.layers.Dropout(0.4),
            tf.keras.layers.Dense(1024, activation='relu', kernel_initializer='he_normal'),  # Capa FC con 1024 neuronas
            tf.keras.layers.Dropout(0.4),
            tf.keras.layers.Dense(136, kernel_initializer='he_normal')  # 68 puntos clave * 2 (x, y)
        ])

    def call(self, inputs, training=False):
        x = self.resnet_backbone(inputs, training=training)
        x = self.fc(x)
        return x
```

Figura 23: Clase KeypointModel que implementa la arquitectura ResNet-18

3.- Construcción y compilación del modelo: La Figura 24 muestra cómo se inicializa el modelo, se define el optimizador Adam y se compila el modelo utilizando MSE (Mean Square Error) como función de pérdida y RMSE (Root Mean Square Error) como métrica.

```
model = KeypointModel()
opt = tf.keras.optimizers.Adam(learning_rate=0.001)
model.compile(optimizer=opt,loss='mse',metrics=[tf.keras.metrics.RootMeanSquaredError()])
```

Figura 24: Construcción y compilación del modelo

4.- Entrenamiento del modelo: La Figura 25 muestra el código que permite realizar el entrenamiento del modelo, utilizando los datos de entrenamiento/validación, basado en la métrica de validación RMSE, y con early stopping para evitar el sobreajuste.

```
early_stopping = EarlyStopping(monitor='val_root_mean_squared_error', patience=8, restore_best_weights=True)
history = model.fit(train_images, train_keypoints, batch_size=64, epochs=40, validation_data=(valid_images,
  valid_keypoints),callbacks=[early_stopping])
```

Figura 25: Entrenamiento del Modelo

5.- Guardado del modelo y el historial de entrenamiento: La Figura 26 muestra cómo se guarda el modelo entrenado en el formato SavedModel de Tensorflow, y el historial de entrenamiento como un archivo CSV, para su posterior análisis.

```
history_df = pd.DataFrame(history.history)
history_df.to_csv('training_history_epoch40-Arq2-RetinaFace_early_preprocess.csv', index=False)
model.save('model-resnet-epoch40-Arq2-RetinaFace_early_preprocess_2', save_format='tf')
```

Figura 26: Código para guardar modelo e historial de métricas

3 Resultados Experimentales y Discusión

En esta sección, se presentarán los resultados del modelo ResNet-18 de manera detallada, comenzando con una descripción de las métricas de evaluación empleadas para medir su precisión y robustez. Se analizarán las curvas de aprendizaje, que ofrecen una visión sobre el comportamiento del modelo durante el entrenamiento, así como las tasas de error y los valores de loss obtenidos en los conjuntos de datos de validación y prueba. Además, se incluirán visualizaciones de los keypoints predichos sobre imágenes de prueba para ilustrar la eficacia del modelo en condiciones reales.

3.1 Evaluación Cuantitativa

Desde la perspectiva cuantitativa se analizarán y discutirán las dos métricas claves para el rendimiento del modelo. Es decir, aquellas métricas que permiten controlar el proceso de entrenamiento de la red neuronal artificial y su convergencia hacia el objetivo deseado, como son MSE (Mean Square Error) y RMSE (Root Mean Square Error). Como el historial de entrenamiento fue almacenado en un archivo CSV, la generación de las gráficas se obtiene como muestra la Figura 27.

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Cargar el archivo CSV
file_path = 'training_history_epoch40-Arq2-RetinaFace_early_preprocess.csv'
history = pd.read_csv(file_path)

# Definir un umbral para considerar los valores "cercanos"
threshold = 0.1

# Encontrar la primera época en que la diferencia es menor al umbral para loss y val_loss
start_epoch_loss = np.where(np.abs(history['loss'] - history['val_loss']) < threshold)[0][0]

# Encontrar la primera época en que la diferencia es menor al umbral para rmse y val_rmse
start_epoch_rmse = np.where(np.abs(history['root_mean_squared_error'] - history['val_root_mean_squared_error']) <
threshold)[0][0]

# Seleccionar la época de inicio más grande para ambas métricas
start_epoch = max(start_epoch_loss, start_epoch_rmse)

# Graficar loss vs val_loss desde la época determinada
plt.figure(figsize=(12, 6))
plt.plot(history['loss'][start_epoch:], label='Training Loss', linewidth=2)
plt.plot(history['val_loss'][start_epoch:], label='Validation Loss', linewidth=2)
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Training Loss vs Validation Loss (Desde la época %d)' % start_epoch)
plt.legend()
plt.grid(True)
plt.show()
```

Figura 27: Código para generar gráfica Training Loss vs Validation Loss

Al ejecutar el análisis de las métricas se pueden interpretar los siguientes resultados cuantitativos:

- La Figura 28 muestra la evolución de la pérdida durante el entrenamiento y la validación desde la época en que ambas métricas comienzan a converger.
- Se observa que la pérdida de entrenamiento (Training Loss) disminuye constantemente a medida que avanzan las épocas, lo cual indica que el modelo está aprendiendo correctamente a partir de los datos de entrenamiento.
- La pérdida de validación (Validation Loss) también disminuye, aunque con algunas fluctuaciones, lo que es común debido a la variabilidad inherente de los datos de validación.
- La convergencia de ambas curvas sugiere que el modelo no está sobreajustando (overfitting) significativamente, ya que la pérdida de validación no se dispara, sino que sigue una tendencia similar a la de la pérdida de entrenamiento.

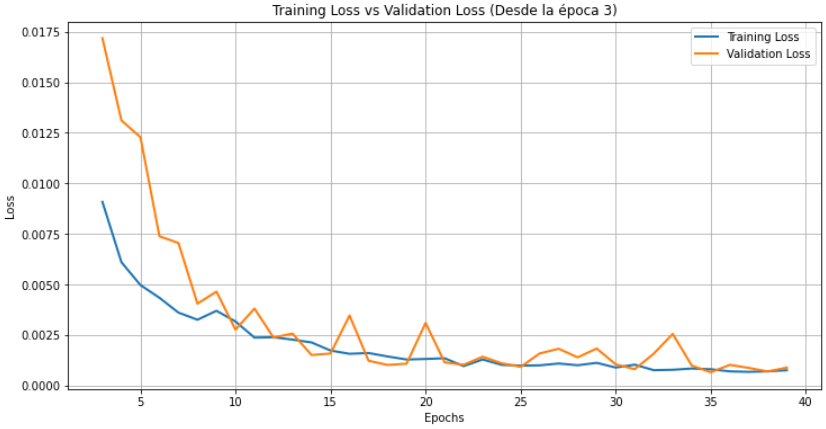

Figura 28: Curva Training Loss vs Validation Loss

- El código de la Figura 29 genera la gráfica de la Figura 30, que muestra la evolución del error cuadrático medio (RMSE) durante el entrenamiento y la validación desde la época en que ambas métricas comienzan a converger.
- Al igual que con la pérdida, el RMSE de entrenamiento (Training RMSE) muestra una tendencia decreciente, indicando que el modelo está reduciendo el error en las predicciones sobre los datos de entrenamiento.
- El RMSE de validación (Validation RMSE) también disminuye, aunque con algunas variaciones. Este comportamiento es esperado y refleja cómo el modelo generaliza a datos no vistos.
- La proximidad entre las curvas de RMSE de entrenamiento y validación sugiere que el modelo tiene un buen equilibrio entre sesgo y varianza, lo que es indicativo de un buen rendimiento general del modelo.

```python
plt.figure(figsize=(12, 6))
plt.plot(history['root_mean_squared_error'][start_epoch:], label='Training RMSE', linewidth=2)
plt.plot(history['val_root_mean_squared_error'][start_epoch:], label='Validation RMSE', linewidth=2)
plt.xlabel('Epochs')
plt.ylabel('RMSE')
plt.title('Training RMSE vs Validation RMSE (Desde la época %d)' % start_epoch)
plt.legend()
plt.grid(True)
plt.show()
```

Figura 29: Código para generar gráfica RMSE vs Validation RMSE

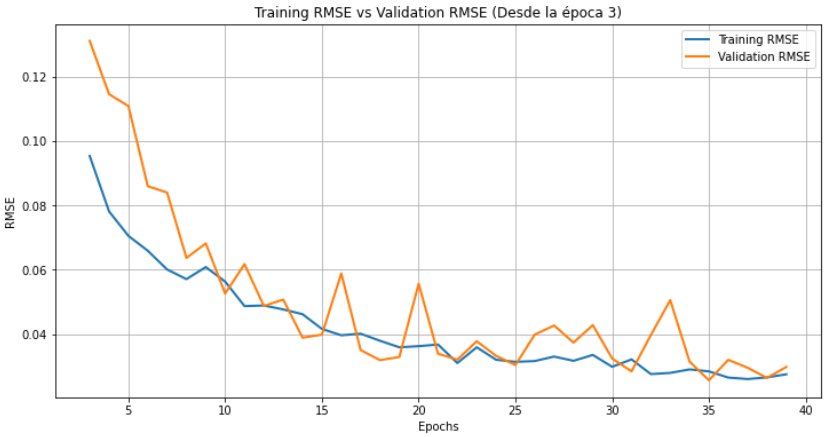

Figura 30: Curva Training RMSE vs Validation RMSE

Como discusión final del análisis cuantitativo, basado en las gráficas y la interpretación de las métricas de pérdida y RMSE, se puede concluir que el modelo entrenado con la arquitectura ResNet-18 para la detección de rostros muestra una buena capacidad de aprendizaje y generalización. La convergencia de las métricas de entrenamiento y validación sugiere que el modelo no está sufriendo de sobreajuste severo y es efectivo en la tarea para la cual fue diseñado. Sin embargo, siempre es recomendable realizar más validaciones y pruebas con diferentes conjuntos de datos para confirmar estos hallazgos. Para complementar la conclusión cuantitativa, se realizó una evaluación cualitativa para determinar si desde esa perspectiva es visualmente preciso el modelo de regresión obtenido.

3.2 Evaluación Cualitativa

Para el caso cualitativo, se tomaron cinco (5) casos de rostros con sus respectivas detecciones generadas por el modelo, y se describirá la precisión de la estimación con respecto a los valores reales (Experimento Cualitativo 1). Además, se hará la observación de cinco (5) imágenes nuevas, es decir que no hayan sido usadas para el entrenamiento del modelo y sus respectivas predicciones de keypoints (Experimento Cualitativo 2).

Experimento Cualitativo 1: Este experimento consisten en tomar las 5 primeras imágenes del conjunto de datos de validación, y realizar la predicción con el modelo entrenado, para posteriormente desplegar las imágenes originales preprocesadas, los keypoints reales y los keypoints predichos.

```python
def visualize_predictions(model, images, keypoints, num_images=5):
    predictions = model.predict(images[:num_images])
    for i in range(num_images):
        image = images[i]
        true_keypoints = keypoints[i].reshape(-1, 2)
        predicted_keypoints = predictions[i].reshape(-1, 2)

        fig, axes = plt.subplots(1, 3, figsize=(15, 5))

        # Mostrar la imagen
        axes[0].imshow(image)
        axes[0].axis('on')
        axes[0].set_title('Imagen')

        # Mostrar los puntos clave reales sin la imagen de fondo
        axes[1].scatter(true_keypoints[:, 0] * image.shape[1], true_keypoints[:, 1] * image.shape[0], c='r', marker='o')
        axes[1].set_title('Puntos Reales')
        axes[1].axis('equal')
        axes[1].invert_yaxis()
        axes[1].axis('on')

        # Mostrar los puntos clave predichos sin la imagen de fondo
        axes[2].scatter(predicted_keypoints[:, 0] * image.shape[1], predicted_keypoints[:, 1] * image.shape[0], c='b', marker='o')
        axes[2].set_title('Puntos Predichos')
        axes[2].axis('equal')
        axes[2].invert_yaxis()
        axes[2].axis('on')

        plt.show()
visualize_predictions(model, train_images, train_keypoints, num_images=5)
```

Figura 31: Código para desplegar la comparación de las 5 imágenes

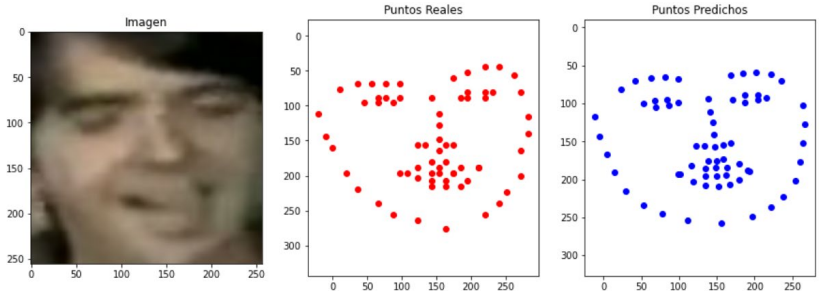

Figura 32: (izquierda) imagen de validación 1. (centro) keypoints reales. (derecha) keypoints predichos.

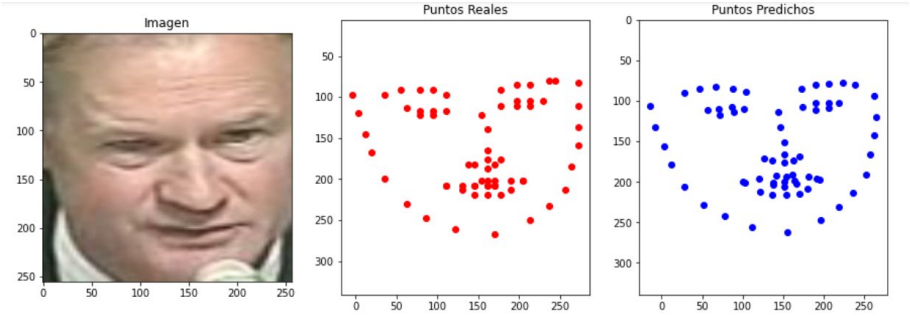

Figura 33: (izquierda) imagen de validación 2. (centro) keypoints reales. (derecha) keypoints predichos.

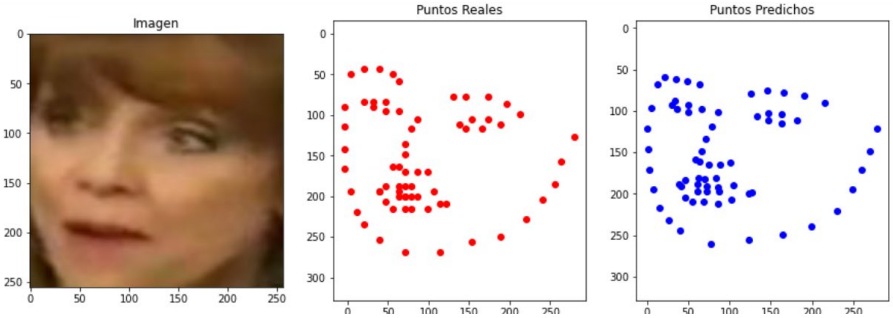

Figura 34: (izquierda) imagen de validación 3. (centro) keypoints reales. (derecha) keypoints predichos.

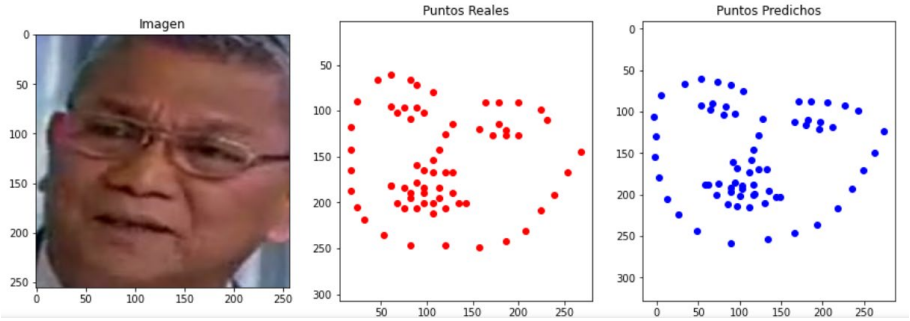

Figura 35: (izquierda) imagen de validación 4. (centro) keypoints reales. (derecha) keypoints predichos.

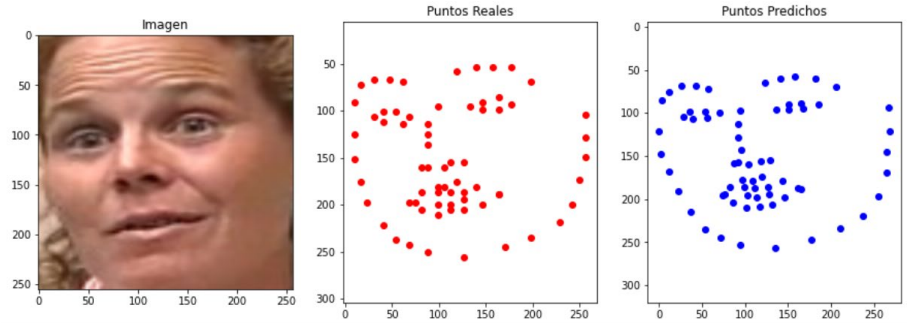

Figura 36: (izquierda) imagen de validación 5. (centro) keypoints reales. (derecha) keypoints predichos.

De las Figura 32, Figura 33, Figura 34, Figura 35 y Figura 36, se puede observar que cualitativamente el modelo de regresión es un muy buen predictor, ya que los puntos azules de predicción son casi idénticos. Por lo mismo, se podría pensar que este resultado tan cercano a la perfección podría ser producto de un sobreajuste. Sin embargo, para dilucidar dicha sospecha, el experimento se amplió a predecir KeyPoints de 5 rostros que nunca participaron como parte del entrenamiento.

La Figura 37, muestra el código ejecutado para ejecutar y visualizar las predicciones de los Keypoints a nuevas imágenes, denominadas imágenes de test.

```python
def visualize_predictions(model, images, num_images=5):
    predictions = model.predict(images[:num_images])
    for i in range(num_images):
        image = images[i]
        predicted_keypoints = predictions[i].reshape(-1, 2)

        fig, axes = plt.subplots(1, 2, figsize=(10, 5))

        # Mostrar la imagen
        axes[0].imshow(image)
        axes[0].axis('on')
        axes[0].set_title('Imagen')

        # Mostrar los puntos clave predichos sin la imagen de fondo
        axes[1].scatter(predicted_keypoints[:, 0] * image.shape[1], predicted_keypoints[:, 1] * image.shape[0],
  c='b', marker='o')
        axes[1].set_title('Puntos Predichos')
        axes[1].axis('equal')
        axes[1].invert_yaxis()
        axes[1].axis('on')

        plt.show()
test_images=np.load('test_images.npy')
model = tf.keras.models.load_model('model-resnet-epoch40-Arq2-RetinaFace_early_preprocess')
```

Figura 37: Código para desplegar las predicciones de las 5 imágenes de test

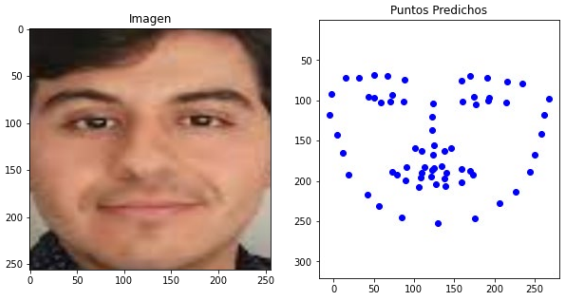

Figura 38: (izquierda) imagen de test 1. (derecha) keypoints predichos.

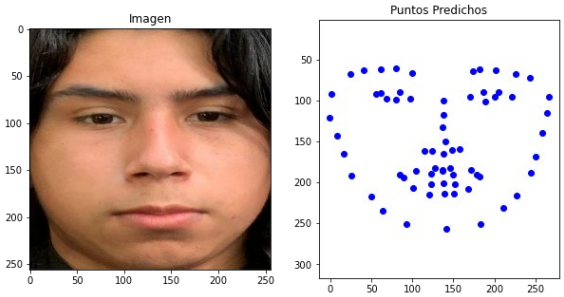

Figura 39: (izquierda) imagen de test 2. (derecha) keypoints predichos.

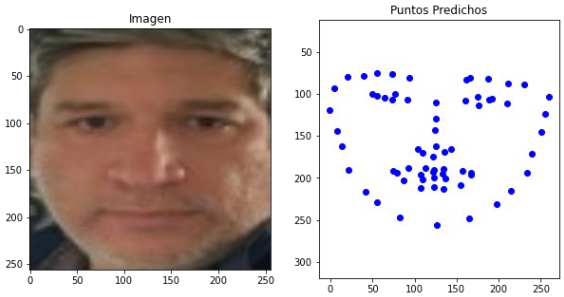

Figura 40: (izquierda) imagen de test 3.(derecha) keypoints predichos.

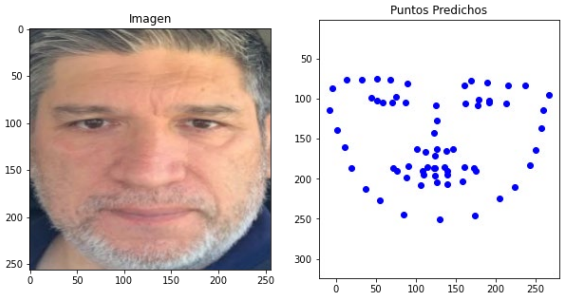

Figura 41: (izquierda) imagen de test 4.(derecha) keypoints predichos.

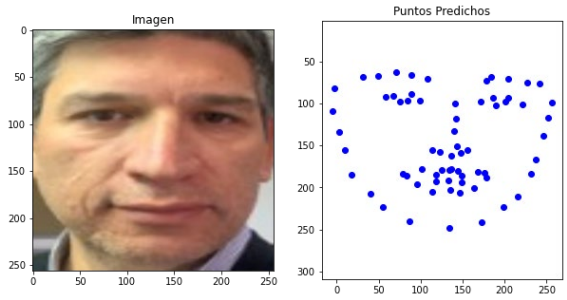

Figura 42: (izquierda) imagen de test 5.(derecha) keypoints predichos.

A simple vista, con una mirada cualitativa, el modelo predice de manera muy precisa y armónica los keypoints faciales, logrando una notable correspondencia con las posiciones reales en las imágenes. Esta precisión es aún más destacable considerando que los rostros utilizados para las pruebas no fueron vistos durante el entrenamiento del modelo.

Esto demuestra la capacidad del ResNet-18 para generalizar efectivamente y realizar predicciones precisas incluso en nuevas imágenes, lo cual resalta la robustez y el potencial del modelo para aplicaciones prácticas en el campo de la visión por computadora.

4 Conclusiones

El principal resultado de este trabajo fue la validación de un modelo de regresión capaz de predecir con alta precisión los puntos clave faciales (keypoints), superando las expectativas iniciales respecto a la capacidad de una red neuronal convolucional como ResNet-18 combinada con un Perceptrón Multicapa (MLP) para esta tarea específica. Los resultados obtenidos demostraron que, mediante un entrenamiento riguroso y un adecuado preprocesamiento de datos, es posible alcanzar precisión destacable en la estimación de 136 valores correspondientes a 68 keypoints faciales.

Es importante recalcar que este éxito no está exento de un arduo trabajo, especialmente en la fase de preparación de datos y entrenamiento, que resultó ser la tarea más exigente pero crítica para el logro de un aprendizaje efectivo del modelo. El análisis y preprocesamiento de un conjunto de datos compuesto por 3462 imágenes para entrenamiento y 770 para validación implicó un considerable esfuerzo computacional, con tiempos de procesamiento que variaron entre 6 y 20 horas, resaltando la necesidad de contar con recursos computacionales robustos para optimizar el proceso.

Los esfuerzos rinden sus frutos puesto que la superación de las expectativas de este trabajo no solo permitió validar el modelo, sino que también dio paso a un proyecto fundacional tecnológicamente innovador en el ámbito del reconocimiento facial. Esto orienta a los desarrolladores a tener una herramienta precisa y eficiente para la identificación de puntos clave del rostro, con aplicaciones prácticas en seguridad biométrica, entretenimiento digital, análisis de comportamiento humano y campos médicos como la cirugía reconstructiva y tratamientos estéticos.

La precisión y robustez del modelo ResNet-18, demostrada en este trabajo, subraya su potencial como herramienta de preferencia para analizar y procesar imágenes faciales de manera automática, tarea que sería poco factible realizar manualmente con la misma exactitud y eficiencia. Esta capacidad de predicción se ha demostrado tanto en imágenes incluidas en el conjunto de entrenamiento como en aquellas no vistas anteriormente por el modelo, lo que evidencia su capacidad de generalización.

Para el desarrollo de este proyecto se aplicó la metodología CRISP-DM (Cross Industry Standard Process for Data Mining), la cual estructuró el proceso en seis fases: comprensión del negocio, comprensión de los datos, preparación de los datos, modelado, evaluación y despliegue. Esta metodología fue fundamental para asegurar un enfoque sistemático y riguroso, facilitando la obtención de resultados consistentes y reproducibles.

Cada fase del proceso CRISP-DM aportó al éxito del proyecto, desde la identificación de objetivos y requisitos del negocio hasta la implementación y evaluación del modelo de regresión.

Con todas las características favorables que ya se han mencionado respecto de este modelo inteligente para la detección de keypoints faciales, y cuyo alcance satisface la primera necesidad del proyecto, es importante recalcar que se presentan muchas oportunidades para formular proyectos sucesivos e ir entregando mayor valor agregado en los siguientes aspectos:

1. Completar la automatización de la solución base para que el preprocesamiento de imágenes y la predicción de keypoints puedan programarse como un servicio de ejecución autónoma sin intervención humana.
2. Mejorar el nivel de aprendizaje y la precisión del modelo mediante la incorporación de nuevas técnicas de regularización y optimización basadas en la retroalimentación continua del uso del modelo en diversas aplicaciones.
3. Evolucionar el modelo para que sea capaz de identificar keypoints en condiciones aún más variadas y desafiantes, como diferentes expresiones faciales, condiciones de iluminación y poses extremas, lo que permitiría su aplicación en contextos aún más diversos y exigentes.
4. Ampliarse a otras áreas de aplicación que requieran la identificación precisa de puntos clave faciales, tales como la evaluación de emociones en tiempo real, la animación digital avanzada y el monitoreo de cumplimiento de medidas sanitarias en contextos de salud pública.

El dinamismo y la rápida evolución de las aplicaciones en el ámbito del reconocimiento facial demandan que las soluciones tecnológicas estén a la altura del ritmo exigido, asegurando decisiones oportunas y basadas en datos precisos. Es aquí donde el aporte de la Inteligencia Artificial, y en particular de modelos como el desarrollado en este proyecto, demuestra su relevancia y potencial para transformar diversas industrias mediante la automatización y precisión en la detección de keypoints faciales.

5 Referencias

Affectiva . (2017). Retrieved from Affectiva and emotion AI: https://www.affectiva.com/emotion-ai/

Bulat, A., & Tzimiropoulos, G. (2017). How far are we from solving the 2D & 3D face alignment problem? (and a dataset of 230,000 3D facial landmarks). 2017 IEEE International Conference on Computer Vision (ICCV). IEEE.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & & Wirth, R. (2000). CRISP-DM 1.0: Step-by-step data mining guide. SPSS inc.

Corneanu, C. A., Simon, M. O., Cohn, J. F., & Guerrero, S. E. (2016). Survey on RGB, 3D, thermal, and multimodal approaches for facial expression recognition: History, trends, and affect-related applications. IEEE Trans. Pattern Anal. Mach. Intell., 38, 1548–1568.

Deng, J., Guo, J., Zhou, Y., Yu, J., Kotsia, I., & Zafeiriou, S. (2019). RetinaFace: Single-stage Dense Face Localisation in the Wild. arXiv [cs.CV]. Obtenido de http://arxiv.org/abs/1905.00641https://arxiv.org/abs/1905.00641

FaceKPoints.Zip. (s.f.). Recuperado el 16 de 07 de 2024, de Dropbox: https://www.dropbox.com/scl/fi/o14y37or4mvpqgcywv8pb/FaceKPoints.zip?rlkey=0guvi6x5so8pmwup7eb5qulnu&dl=0

Figueroa, M. (2024). Tarea1_RV_DCC_UChile: Código Tarea 1 Reconocimiento Visual. Obtenido de https://github.com/mfiguer/Tarea1_RV_DCC_UChile

Gupta, S., Kumar, P., & Tekchandani, R. K. (2023). Facial emotion recognition based real-time learner engagement detection system in online learning context using deep learning models. Multimed. Tools Appl., 82, 11365–11394.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. doi:https://doi.org/10.48550/arXiv.1512.03385

Karras, T., Laine, S., & Aila, T. (2019). A style-based generator architecture for generative adversarial networks. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE.

McConvey, J. R. (2023, September). Singapore, Dubai, Aruba airports embrace biometrics for passport-free traveler clearance. Singapore, Dubai, Aruba airports embrace biometrics for passport-free traveler clearance.

Saavedra, J. M. (28 de 08 de 2023). Convnet/resnet.Py at main · jmsaavedrar/machine_learning. Obtenido de https://github.com/jmsaavedrar/machine_learning/blob/main/convnet/resnet.py

Serengil, S. I. (2024). retinaface: RetinaFace: Deep Face Detection Library for Python. Obtenido de https://github.com/serengil/retinaface

Vibhuti, Jindal, N., Singh, H., & Rana, P. S. (2022). Face mask detection in COVID-19: a strategic review. Multimed. Tools Appl., 81, 40013–40042.

Zhang, M. M., Di, W. J., Song, T., Yin, N. B., & Wang, Y. Q. (2023). Exploring artificial intelligence from a clinical perspective: A comparison and application analysis of two facial age predictors trained on a large-scale Chinese cosmetic patient database. Skin Res. Technol., 29.