UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA MECÁNICA

DETECCIÓN AUTOMÁTICA DE GRIETAS BASADA EN IMÁGENES DE PUENTES DE CONCRETO A TRAVÉS DE MODELOS ENCODER-DECODER

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL MECÁNICO

FELIPE ESTEBAN SILVA PIÑA

PROFESOR GUÍA:
ENRIQUE LÓPEZ DROGUETT

MIEMBROS DE LA COMISIÓN:
JUAN TAPIA FARIAS
VIVIANA MERUANE NARANJO

SANTIAGO DE CHILE
2019

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL MECÁNICO
POR: FELIPE ESTEBAN SILVA PIÑA
FECHA: 2019
PROF. GUÍA: ENRIQUE LÓPEZ DROGUETT

DETECCIÓN AUTOMÁTICA DE GRIETAS BASADA EN IMÁGENES DE PUENTES DE CONCRETO A TRAVÉS DE MODELOS ENCODER-DECODER

Una de las ramas de más desarrollo y necesidad a lo largo del país es la gestión y confiabilidad de activos físicos, junto con el crecimiento considerable en los últimos años del concepto de deep learning; todos estos temas convergen en un solo punto, la detección temprana de fallas en estructuras y equipos mecánicos. Esto potenciado por el rápido avance tecnológico , permiten obtener una gran cantidad de datos operacionales como lo son las imágenes del estado en que se encuentra una estructura, vibraciones a la que esta siendo sometida , temperaturas presentes, presiones existentes, entre otros. Esto entrega potenciales herramientas a mano de los ingenieros con el fin de poder encontrar modelos y métodos de procesar estas grandes cantidades de datos de manera eficiente con tal de sacar el mayor provecho de estos datos y poder tomar decisiones importantes a la hora de prevenir y gestionar el mantenimiento de los equipos y las estructuras.

La motivación principal para la elaboración de esta memoria es formar parte del desarrollo del deep learning en Chile, aplicándolo directamente en una estructura civil como lo es un puente. El objetivo general es realizar un modelo de detección automático de grietas basado en imágenes en puente de hormigón., para luego analizar los resultados obtenidos y cuantificar las grietas presentes en la estructura civil.

La metodología que se seguirá corresponde principalmente a 5 etapas importantes de trabajo, en primer lugar se tiene el análisis del dataset de imágenes entregados junto con una recopilación bibliográfica sobre la detección en imágenes, luego se usará el modelo de deep learning en tensorflow, el cual involucra la utilización de distintos modelos de segmentación obtenidos en la recopilación bibliográfica, para finalmente analizar los resultados obtenidos y concluir con la detección de la grieta.

Los recursos utilizados son principalmente el dataset de imágenes de las grietas, una computadora GPU con sistema operativo Linux con tensorflow y las reuniones con el profesor guía y coguía para feedbacks y recomendaciones en caso de ser necesarias.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

Junto al increíble aumento en el desarrollo de la tecnología año a año, la forma de enfrentar los problemas cada día va evolucionando, esto viene potenciado por la gran cantidad de información disponible para enfrentarlos y resolverlos.

El manejo de toda esta gran información disponible representa uno de los grandes desafíos de la ingeniería en los últimos años, dado que no existen métodos suficiente que permitan manipular esta gran cantidad de datos de forma practica y eficiente. Es por esto que durante el último tiempo han surgidos conceptos como deep learning, machine learning y redes neuronales, los cuales han entregado otra perspectiva y herramientas para el manejo de gran cantidad de información.

Dado que se sabe que la forma en que son manipulado los datos tienen una repercusión directa en la precisión de los resultados obtenidos, esta precisión siempre es buscada principalmente por industrias que poseen sistemas complejos de equipos o estructuras civiles, en donde una falla pueda generar cuantiosas perdidas económicas y en algunos casos perdidas humanas, es por esto que es importante tener resultados que se ajusten lo mas posible al comportamiento real del sistema y permita monitorear de manera confiable el estado en que se encuentra la estructura.

1.1. Antecedentes básicos

La probabilidad que una estructura civil presente una falla en algún momento de su vida útil aumenta considerablemente producto de su utilización y el tipo de carga que soporta. Esto dado que a medida que pasa el tiempo dicha estructura civil es mas propensa a presentar una falla catastrófica en algún instante. Es por esto que el monitoreo de la salud estructural de las estructuras civiles es una tarea crítica, dadas las posibles consecuencias económicas perjudiciales y el impacto en la vida humana. Muchas estructuras civiles basadas en el concreto están experimentando un deterioro a través del envejecimiento. Además, estas estructuras podrían estar sujetas a sobrecargas, condiciones climáticas adversas, accidentes de impacto y terremotos, y podrían ubicarse en áreas remotas o inaccesibles que impongan un desafío adicional a las actividades de inspección. Lo que hace muy complejo evaluar

adecuadamente la integridad estructural del envejecimiento sobre las estructuras civiles de una manera rentable.

Es por esto que la detección y la inspección han traído una atención importante en los últimos años, y se han propuesto diferentes técnicas para la detección de daño en estructuras pero aún así siguen habiendo desafíos en su aplicación para monitorear la salud de las estructuras civiles en el campo. Un enfoque común usado para identificar y evaluar el daño en una estructura es la inspección visual de la grieta por un inspector experto en el rubro. Pero este tipo de inspección actual consume mucho tiempo llevarla acabo y en ciertas ocasiones no es posible llegar a inspeccionar ciertas áreas inaccesibles para el inspector, como por ejemplo grietas ubicadas en la zona inferior de un puente o en la planta de los pilares ubicados bajo el mar. Además, es una técnica de evaluación altamente subjetiva y depende significativamente del nivel de experiencia del inspector de manera que se notifican grandes discrepancias en los resultados cuando la misma estructura es inspeccionada por diferentes inspectores [1].

Aún cuando se habla de detectar una grieta se refiere a que algo indeseado esta ocurriendo, estas tienen distintos grados de complejidad e importancia dependiendo del tamaño de esta, las sobrecargas que esta sometida, entre otros factores. Por ejemplo no es lo mismo que se encuentre una grieta pequeña a que se encuentre una grieta mas grande, o a que se encuentre una grieta en cierta zona con mucha sobrecarga a una que este sometida a poca carga. Es por esto mismo que se busca estudiar grietas de gran importancia y complejidad, los cuales pueden llegar a ser grietas catastróficas y producir grandes perdidas monetarias, así como también involucrar perdidas de vidas humanas [1].

Dado estos problemas e inconvenientes producto de la inspección visual se entiende que la detección temprana de una grieta debe tomar pasos importantes hacia las inspecciones automatizadas de daños visuales en puente de concreto. Como primer paso importante se tiene la de acceder a una estructura de forma remota, esto quiere decir obtener información visual tomando imágenes remotas de una estructura dada desde diferentes ángulos usando cámaras controladas [2][3][4], adquisición robótica de imágenes [5] o un vehículo aéreo no tripulado (UAV, también llamado dron) equipado con cámaras de alta resolución [6].

Se han propuestos inspecciones de enfoques autónomos basados en imágenes para la detección de grietas en puentes de hormigón, en la cual se proporciona una revisión y evaluación de las etapas más importantes y esenciales para el monitoreo automático del estado de salud, basado en la visión computarizada de la estructura civiles con el objetivo de detectar y localizar las grietas y defectos producidos por la corrosión. Luego de recopilar la información visual adecuada, se deben aplicar métodos de procesamiento de imágenes sólidos y autónomos para analizar las imágenes recopiladas y detectar las fallas [2]. En las últimas décadas se ha seguido investigando y realizando esfuerzos para poder implementar este tipo de tecnología basada en imágenes obtenidas para la detección de grietas en estructuras, en las investigaciones disponibles [5][7] podemos ver ejemplos claro para poder implementar

este tipo de tecnología [1].

En primera instancia los métodos convencionales usados de visión artificial para procesar y detectar grietas en las imágenes, permitían extraer algunas características de la imagen, como la longitud, el tamaño, el color, entre otras, tales características son útil para detectar la deformación, esto fue gracias a que su enfoque se centró principalmente en las técnicas comunes de detección de bordes y en las técnicas morfológicas para la detección de grietas.

Luego se diseñó un modelo para la detección de grietas a partir de imágenes de alto contraste, este permitía detectar grietas en un entorno ruidoso mediante varias operaciones matemáticas de evaluación y morfología como un patrón. La detección se efectuaba definiendo el patrón de grieta para un modelo geométrico preciso. Este se logró realizando un filtrado lineal después de la evaluación de la curvatura para distinguirlos de un patrón de fondo similar. El enfoque de esta investigación identificó las irregularidades en la imagen secuencialmente por la geometría de la grieta basado en sistemas de reconocimiento [8].

Más adelante se propuso una idea adaptativa que otorga a un sistema de inspección robótico la capacidad de cuantificar y detectar grietas en imágenes capturadas desde cualquier distancia hasta el objeto, y con cualquier distancia focal o resolución. En este trabajo utilizó redes neuronales y máquinas de soporte vectorial (conjunto de algoritmos de aprendizaje supervisado) para clasificar grietas a partir de patrones sin grietas [7].

Luego se propuso un algoritmo de detección de grietas el cual se baso principalmente en la tecnología de procesamiento de imágenes digitales, mediante el pre-procesamiento, la segmentación de imágenes y la extracción de características de estas, donde obtuvieron la información sobre la imagen de la grieta. Para lograr esto los investigadores usaron un método de umbral de segmentación después del alisado de la imagen, en la etapa de análisis de las imágenes, los investigadores calcularon el área y el perímetro, junto con definir un índice de redondez. Posteriormente de la comparación de estos índices, se realizó la evaluación de la presencia de una grieta en una imagen [9].

Desde el año 2018 gracias al aumento y masificación de la tecnología la gran mayoría de los métodos empleados para el procesamiento de imágenes utilizan el aprendizaje profundo basado en potentes tarjetas GPU, las cuales permiten procesar miles de imágenes y aplicar los resultados en tiempo real. Esto a repercutido directamente en nuevas propuestas de algoritmo de detección de grietas utilizando un marco de aprendizaje profundo que incorpora información temporal y contextual en la detección de objetos en vídeos [10]. Esta investigación logró uno de los mejores rendimientos del estado del arte en ImageNet para la detección de objetos en vídeos utilizando un detector de objetos por cuadros llamados *tubelets*.

En el mismo año se propuso una idea nueva de aprendizaje por transferencia basado en redes neuronales convolucionales que utilizan métodos de cuellos de botella y ajuste fino con

hasta cuatro clases de daños en estructuras civiles a partir de escenas de terremotos [11]. En esta investigación abre un nuevo enfoque para la detección y evaluación de daños en componentes básicos estructurales mediante el aprendizaje profundo.

Hoy en día el aprendizaje profundo no solo se ha estado utilizando como clasificador y detectores de objetos, si no que también para localizar y extraerla posición exacta del objeto en la imagen. Siendo más claro, ahora se puede construir un detector de objetos más preciso y un segmentador de grietas, utilizando la segmentación semántica, la cual es usada para clasificar cada píxel o grupo de píxeles en las imágenes visuales [1].

Esta segmentación semántica en imágenes ha estado generando cada vez más interés entre los investigadores de visión artificial y aprendizaje automático. Este interés ha crecido junto al creciente interés en otras áreas de aplicación, las cuales también necesitan mecanismos de segmentación precisos y eficientes como por ejemplo la conducción autónoma, navegación interior e incluso sistemas de realidad virtual o aumentada. El interés en este tipo de áreas coincide con la gran cantidad de nuevos enfoques del aprendizaje profundo en casi todos los campos y aplicaciones relacionadas la manipulación y visualización de imágenes digitales, algunos enfoques como la segmentación semántica o la compresión de la escena [12][13][14].

Aún no se han propuesto un método para clasificar y segmentar imágenes de grietas automáticamente en estructuras de hormigón civil utilizando modelos de segmentación semántica como el *Encoder-Decoder with Skip*. Esto puede atribuirse, en gran medida a que los conjuntos de datos que se tienen entregan un número limitado de imágenes para desarrollar un modelo genérico automático para clasificar y segmentar grietas. Otra parte también puede atribuirse a que el etiquetado de los datos requieren un esfuerzo significativo producto del etiquetado manual, el cual es bastante lento de desarrollar y requiere un amplio conocimiento de ingeniería estructural especifico.

En esta memoria se presenta un enfoque automático de detección de grietas en puentes de hormigón. El modelo utilizará dos etapas en primera instancia hará una clasificación de las imágenes, definiendo en primera instancia si esta posee o no posee grieta, luego como segunda etapa segmentará semánticamente todos los píxeles de la imagen que pertenezcan a la grieta.

1.2. Motivación

Teniendo en cuenta la importancia que tiene el estudiar un tema tan nuevo e innovador como lo es la detección automatizada de grietas en un puente de hormigón, trabajar en este tema tiene como principal motivación el buen uso que se le puede dar a la tecnología, permitiendo estar siempre en constante monitoreo y detección de posibles grietas que puedan en un futuro causar daños indeseables de vidas y monetarios. Por otra parte, al hacer este tipo de estudio se tiene la importante labor de dar a conocer a otros que aún no están

familiarizados con este tipo de temas, los cuales en un tiempo más serán el futuro venidero.

Por otro lado, se tiene que al ser capaz de modificar un modelo que pueda segmentar y detectar grietas en imágenes de un puente de hormigón, genera una inmensa responsabilidad y motivación por los futuros avances que esto pueda llevar en el área de detección de grietas.

1.3. Objetivo general

El objetivo general de la presente memoria consiste en segmentar y detectar grietas por medio de un modelo de redes neuronales convolucionales en *tesorflow*.

1.4. Objetivos específicos

- Estudiar y analizar las imágenes de grietas del *dataset* de un puente de hormigón.
- Utilizar y modificar un modelo de deep learning capaz de segmentar y detectar objetos.
- Entrenar, testear y verificar las métricas probabilísticas del modelo.
- Analizar y comparar las imágenes de grietas segmentadas obtenidas.
- Verificar el modelo en un *dataset* externo de imágenes de grietas.

1.5. Alcances

Esta memoria tiene como alcance presentar un modelo automático de detección de grietas en puentes de hormigón. En base a un *dataset* de imágenes de grietas en puentes de hormigón, el cual por medio de dos etapas (clasificación y detección) logrará detectar grietas.

Capítulo 2

Metodología

Con el fin de lograr los objetivos planteados se sigue la siguiente metodología, la cual considera los procedimientos pertinentes para lograr el correcto desarrollo de la memoria, así se plantea la metodología esquematizada en la Figura 2.1 y descrita a continuación de esta.

Figura 2.1: Metodología a seguir

2.1. Analizar datos y recopilación bibliográfica

Luego de obtener el *dataset* de las imágenes obtenidas por un vídeo de la estructura civil de un puente de hormigón, se procede analizar los datos, identificando su organización, su tipo y su contenido, luego se corrobora que cada imagen real de una grieta tenga su imagen de etiqueta. Posteriormente se recopila información para determinar la forma de estudiar y trabajar el *dataset* y se busca información de investigaciones de temas similares de segmentación semántica y los tipos de modelos existentes para realizar un mejor análisis de los datos.

2.2. Utilizar y modificar modelo para detección de grietas

Luego de tener todos esos antecedentes se procede a utilizar y modificar un modelo empleado para la detección de objetos de una carretera en *tensorflow* para el análisis de las imágenes de grietas, utilizando los modelos Encoder-Decoder y Encoder-Decoder with Skip Connections, y comparando con el modelo DeepLabV3+.

2.3. Entrenar, testear y verificar el modelo

Luego de adecuar el modelo a los requerimientos del estudio se procede a entrenar la red usando el *dataset* de imágenes de grietas, para posteriormente testear y verificar si las métricas probabilísticas obtenidas por el modelo son las esperadas, en caso de determinar que el modelo necesita ser modificado, se tabulan los resultados y se repite este paso.

2.4. Analizar los resultados obtenidos

Una vez entrenado, testeado y verificado el modelo se procede a analizar las imágenes segmentadas y gráficos obtenidos por el modelo en cada intento realizado, en este punto se agrupan y se estudian los resultados obtenidos y se toman decisiones sobre cual es la configuración óptima del modelo, dependiendo si esta optimización conseguida se alejan de los resultados esperados, se devuelve a la etapa de Ütilizar y modificar modelo para la detección de grietas"volviendo a repetir el mismo proceso.

2.5. Analizar siguiente modelo

Luego de verificar que los resultados son coherentes con los esperados por la literatura, se procede a estudiar el siguiente modelo con el fin de comparar los resultados obtenidos repitiendo los mismos pasos anteriores.

2.6. Recursos

Para el completo desarrollo de la memoria a realizar solo se identificaron recursos no pecunarios, ya que no se necesitan montos monetarios.

2.6.1. Recursos no pecunarios

- Base de datos de imágenes de grietas de puentes de hormigón.
- Computador GPU, con sistema Linux y TensorFlow 10, este se encuentra ubicado en la misma universidad específicamente en Beauchef 851.
- Reuniones con el profesor guía y co-guía para retroalimentaciones y feedbacks necesarios para el desarrollo de la memoria.

Capítulo 3

Antecedentes

3.1. Conceptos Generales

Algunos conceptos generales utilizados a lo largo de la presente memoria que son importante mencionar y explicar.

3.1.1. Convolutional Neural Network (CNN)

Las redes neuronales convolucionales (CNN) corresponden a otro modelo del Deep Learning. Las CNN están conformadas por neuronas, las cuales reciben una entrada, realizan un producto de los puntos y posteriormente le aplican alguna función de activación. También tiene una función de puntuación la cual representa que tan bien el modelo esta aprendiendo en relación la entrada y los resultados esperados.

La CNN realiza un tratamiento espacial de la entrada dado que asume que las entradas tienen dependencias espaciales entre sus diferentes componentes (por ejemplo, imágenes o series de tiempo), por lo cual el modelo puede aprovechar la relación espacial entre las características de los datos. Explicado de otra forma ciertas redes neuronales convierten su entrada en vectores, por lo cual se pierde todo tipo de información importante relacionada con la ubicación espacial de cada dato con respecto a los demás. En cambio la CNN no realiza esta transformación inicial en su entrada, por lo cual funciona mucho mejor con datos de imágenes o espectrogramas. A continuación se muestra un esquema del funcionamiento de una CNN a fin de entender mejor el proceso que siguen los datos, cabe destacar que un modelo CNN no puede clasificar (diagnosticar) por si solo, por lo cual en su ultima capa se debe enlazar con una red neuronal clasificadora siguiendo la misma lógica explicada anteriormente, para que los datos puedan ingresar a esta red la CNN en su ultima capa aplana los datos con el fin de que puedan ser procesados.

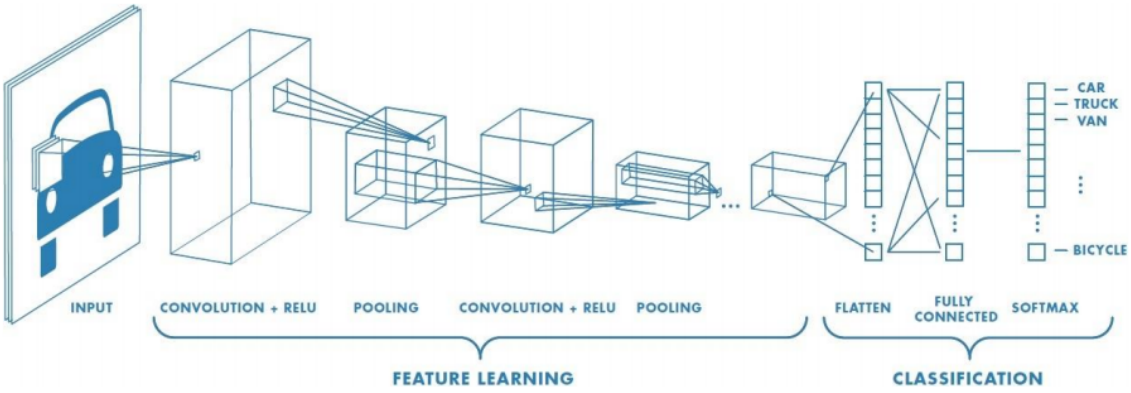

Figura 3.1: Esquema de una CNN[33]

3.1.2. Max-Pooling

Es un proceso de reducción de dimensiones de una muestra . El objetivo es reducir la muestra de una representación de entrada (imagen, matriz de salida de capa oculta, etc.), reduciendo su dimensionalidad y permitiendo suposiciones sobre las características contenidas en las sub-regiones agrupadas.

Este se usa para para evitar el sobreajuste excesivo al proporcionar una forma abstracta de la representación. Además, reduce el costo computacional al reducir el número de parámetros para aprender y proporciona menor variación de la traducción de características de la representación interna [15].

El *max-pooling* se realiza aplicando un filtro a las sub-regiones no superpuestas de la representación inicial. En el ejemplo se puede apreciar de una forma más clara el concepto de max-pooling:

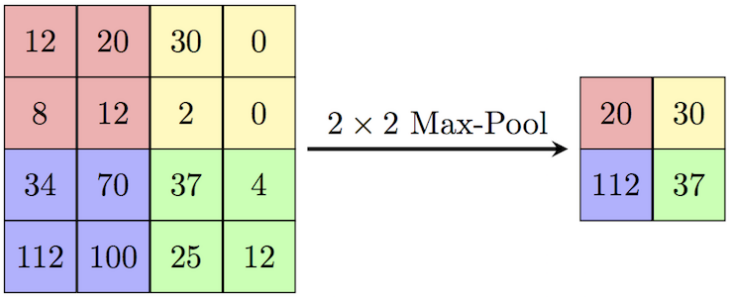

Figura 3.2: Max Pooling [15]

3.1.3. Upsampling

Corresponde a una técnica que aumenta o expande las dimensiones de una muestra. Es decir, es lo inverso del *max-pooling*. Donde cada valor puede estar rodeado de ceros en una sub-región para aumentar las dimensiones de la muestra o también en vez de ceros puede copiar el valor del píxel tantas veces como sea necesario.

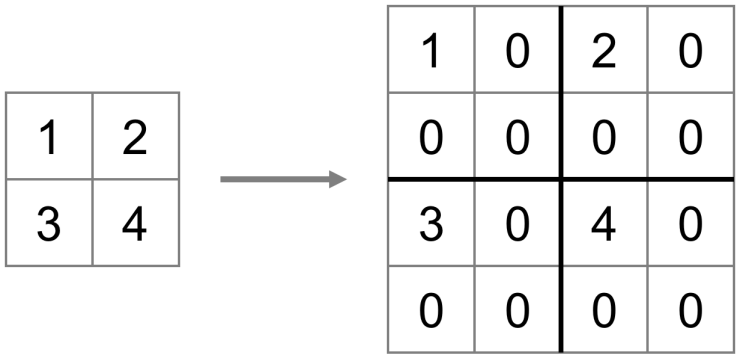

Figura 3.3: Upsampling [16]

Esta técnica se puede mejorar recordando las ubicaciones de la muestra antes del *upsampling* y usándolo al momento de realizar el *upsampling*, como se muestra a continuación:

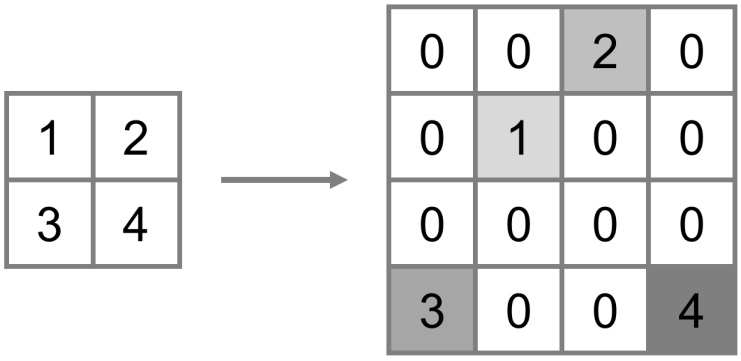

Figura 3.4: Upsampling recordando las ubicaciones [16]

3.2. Métricas probabilísticas

3.2.1. Recall (Sensitivily)

Cuando se habla de *recall* se refiere a la capacidad de un modelo para encontrar todos puntos de datos de interés en un conjunto de datos. Una definición más precisa del concepto *recall* es el número de verdaderos positivos dividido por el número de verdaderos positivos más el número de falsos negativos. Cuando se refiere a los verdaderos positivos se esta hablando de los puntos de datos clasificados como positivos por el modelo pero que en

realidad son positivos (lo que significa que son correctos) y cuando se refiere a los falsos negativos son los puntos de datos que el modelo identifica como negativos pero que en realidad son positivos (incorrectos). El concepto *recall* es efectivo para evaluar cómo un modelo identifica con precisión real las fallas.

$$Recall = \frac{Verdaderos\ Positivos}{Verdaderos\ Positivos + Falsos\ Negativos} \times 100 \tag{3.1}$$

En el caso de que el *recall* fuera el 100 %, no significa exactamente que se tiene un clasificador perfecto, si no como en la mayoría de los conceptos en ciencia de datos, existe una compensación en las métricas que se eligen maximizar. En el caso del *recall*, cuando se aumenta, disminuye la precisión.

3.2.2. Precisión

El concepto de precisión se define matemáticamente como el número de positivos verdaderos dividido por el número de positivos verdaderos más el número de falsos positivos.

$$Precisión = \frac{Verdaderos\ Positivos}{Verdaderos\ Positivos + Falsos\ Positivos} \times 100 \tag{3.2}$$

A pesar de que la precisión y el *recall* parecieran ser similares, no lo son dado que la precisión expresa la proporción de los puntos de datos que el modelo dice que eran relevantes pero en realidad eran relevantes [17]. Mientras que el *recall* muestra la capacidad de encontrar todas las instancias relevantes en un conjunto de datos. En otras palabras, a medida que aumentamos la precisión, disminuimos el *recall*, tal como el ejemplo que se muestra en la siguiente figura.

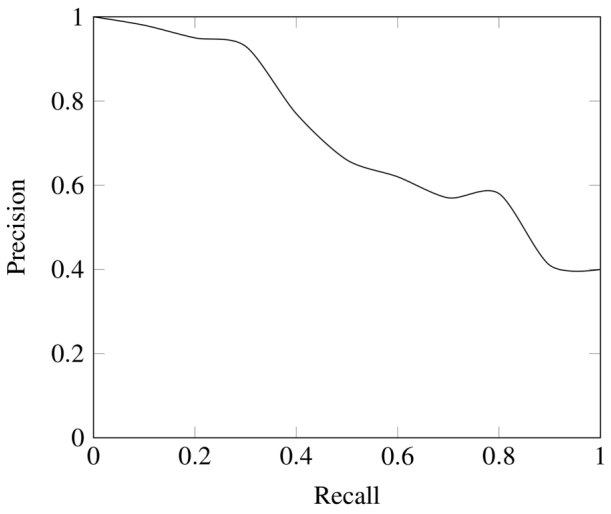

Figura 3.5: Ejemplo de precisión vs recall [19]

Otra forma más clara de entender el concepto de *recall* y precisión es con la siguiente figura:

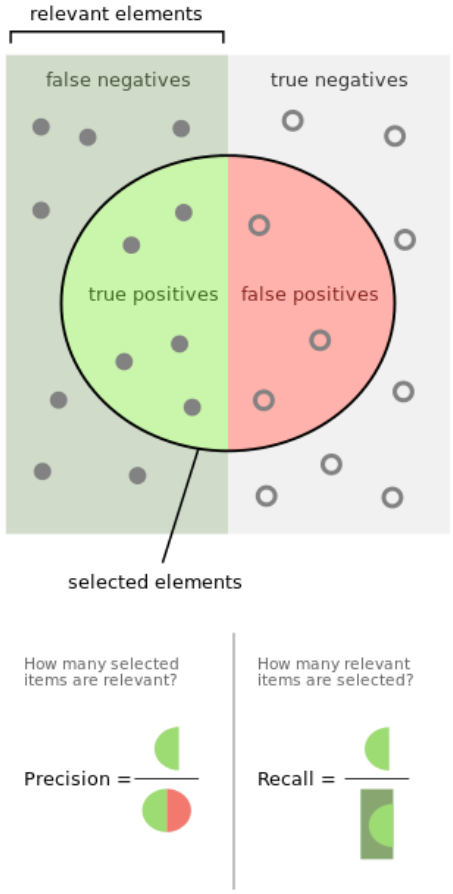

Figura 3.6: Recall y precisión[18]

3.2.3. F1-SCORE (F-Measure)

El concepto F1-*score* corresponde al promedio ponderado de la precisión y el *recall*. Por lo cual este valor tiene en cuenta tanto los falsos positivos como los falsos negativos.

La formula del F-*Measure* es:

$$F_1 = \left(\frac{recall^{-1} + precision^{-1}}{2} \right)^{-1} = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{3.3}$$

En otras palabras la puntuación F_1 es la media armónica de la precisión y recuperación, y esta media armónica se define como el recíproco de la media aritmética de los recíprocos. Debido a eso, el resultado no es sensible a valores extremadamente grandes. Por otro lado, no todos los valores atípicos son ignorados. Los valores extremadamente bajos tienen una influencia significativa en el resultado [20].

Ecuación de la media armónica:

$$H = \frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + ... + \frac{1}{x_n}} = \frac{1}{\sum_{\mathrm{i}-1}^{n} \frac{1}{x_\mathrm{i}}} = \left(\frac{\sum_{\mathrm{i}=1}^{n} x_\mathrm{i}^{-1}}{n} \right)^{-1} \tag{3.4}$$

Interpretando la puntuación puntuación F_1 como métrica, se puede decir que se esta seguro de que si la puntuación F_1 es alta, tanto la precisión como la recuperación del clasificador indican buenos resultados[20].

Esa característica de la métrica permite comparar el rendimiento de dos clasificadores usando solo una métrica y aún así estar seguro de que los clasificadores no están cometiendo algunos errores graves que pasan desapercibidos para el código que califica su salida.

Pero en el caso contrario si se tiene un clasificador cuyo puntuación F_1 es bajo, no se puede decir a ciencia cierta si tiene problemas con falsos positivos o falsos negativos, para este caso sería necesario tomar otro camino, como por ejemplo utilizar la matriz de confusión para determinar el problema [21] .

3.2.4. Accuracy

Corresponde a la medida de rendimiento más intuitiva y simple es una relación entre la observación predicha correctamente y el total de observaciones. Mientras mayor sea *accuracy*, se puede decir que nuestro modelo es mejor, pero esto no pasa siempre dado que en ciertas ocasiones se puede obtener un *accuracy* cercano al 100 %, pero es debido a que el modelo se ha sobre-ajustado a los datos de entrenamiento, es decir, se enamoro de estos

datos. En otras palabras el *accuracy* es una gran medida cuando se tiene un conjuntos de datos simétricos donde los valores de falsos positivos y falsos negativos son casi iguales [22].

La ecuación que define el *accuracy* es:

$$Accuracy = \frac{Verdaderos\ Positivos + Verdaderos\ Negativos}{Verdaderos\ Positivos + Falsos\ Negativos + Falsos\ Positivos + Verdaderos\ Negativos} \tag{3.5}$$

Considerando el accuracy para el caso de píxeles por clases en una imagen, esencialmente se esta evaluando una máscara binaria; un verdadero positivo representa un píxel que se predice correctamente que pertenece a la clase dada (de acuerdo con la máscara de destino) mientras que un verdadero negativo representa un píxel que se identifica correctamente como no perteneciente a la clase dada.

3.2.5. Intersection-over-union (IoU)

El concepto de IoU que en español se traduce como la intersección sobre la unión, también conocida como el índice Jaccard, corresponde a una medida comúnmente utilizada para determinar qué tan precisa es una segmentación de imagen propuesta, en comparación con una segmentación conocida tomada como verdad fundamental. Es decir, es esencialmente un método para cuantificar el porcentaje de superposición entre la imagen objetivo y el resultado de la predicción. Esta métrica está estrechamente relacionada con el coeficiente Dice, que a menudo se usa como una función de pérdida durante el entrenamiento.

La definición matemática del IoU entre una segmentación conocida Y de n pixeles, y un conjunto similar de segmentación predicha $\hat{Y}$ (En el caso binario donde $Y_{\mathrm{i}}, \hat{Y}_{\mathrm{i}} \in \{0, 1\}, \forall \mathrm{i} \in [1, n]$) es la siguiente ecuación:

$$IoU(Y, \hat{Y}) = \frac{Y \cap \hat{Y}}{Y \cup \hat{Y}} = \frac{\sum_{\mathrm{i}=1}^{n} \mathrm{mín}\left(Y_{\mathrm{i}}, \hat{Y}_{\mathrm{i}}\right)}{\sum_{\mathrm{i}=1}^{n} \mathrm{máx}\left(Y_{\mathrm{i}}, \hat{Y}_{\mathrm{i}}\right)} \tag{3.6}$$

Acercando aún más esta definición al caso de la segmentación semántica, el IoU se calcula para cada etiqueta a nivel de píxel como:

$$IoU = \frac{Verdaderos\ Positivos}{Verdaderos\ Positivos + Falsos\ Positivos + Falsos\ Negativos} \tag{3.7}$$

En donde los verdaderos positivos son aquellos píxeles que pertenecen a cierta etiqueta y se predicen correctamente como esa etiqueta, los falsos negativos son los píxeles que pertenecen a cierta etiqueta pero se predicen incorrectamente como una etiqueta diferente y los falsos positivos son los píxeles que pertenecen a una etiqueta diferente pero se predicen como la etiqueta. En la siguiente figura se explica de forma mas práctica el concepto de IoU:

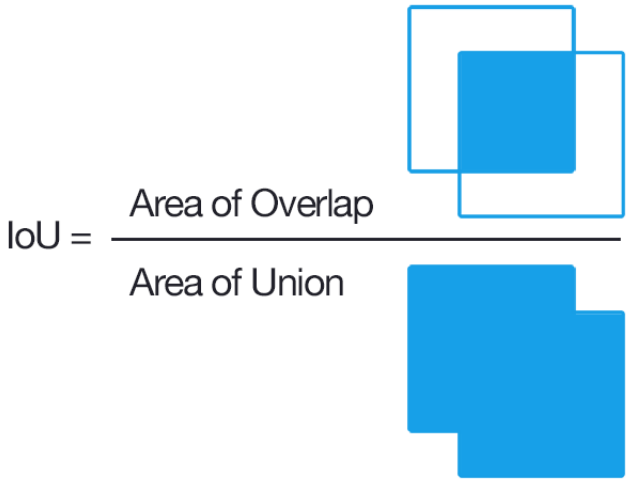

Figura 3.7: Concepto de IoU [23]

Como se puede ya concluir el IoU toma valores entre 0 y 100 %, donde un valor mayor indica que la segmentación es más precisa. Es por esto que se busca que este parámetro sea de guía al momento de utilizar la segmentación semántica, a continuación se presentan unos ejemplos de valores de IoU para cierta segmentación:

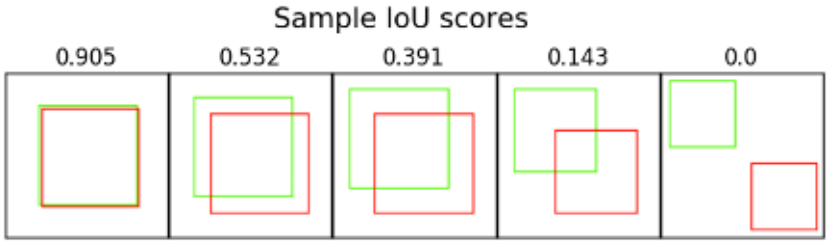

Figura 3.8: Ejemplos de IoU [23]

3.3. Segmentación semántica

La segmentación semántica tiene como objetivo clasificar cada píxel de la imagen en una categoría específica . La diferencia con la clasificación de imágenes estándar es que no clasifica la imagen completa en una clase sino en cada píxel individual. Entonces, se tiene un conjunto de categorías predefinidas y se le quiere asignar una etiqueta a cada píxel de la imagen. De esta forma se hace una asignación en función del contexto de los diferentes

objetos en la imagen y se etiqueta cada píxel con la clase de su objeto o región circundante [34].

En la siguiente imagen de ejemplo a cada píxel se le ha asignado a una etiqueta específica y representado por un color diferente. Rojo para personas, azul para automóviles, verde para árboles, etc.

Figura 3.9: Ejemplo de segmentación semántica [34]

3.4. Front-End

En el ámbito de la segmentación semántica cuando se habla de *front-end* se refiere específicamente a una red neuronal previamente entrenada para la clasificación y extracción de características, y al igual que el concepto de *front-end* explicado en el párrafo anterior, es la primera parada a donde ingresan las imágenes que se quieren segmentar, dado que este clasificador entrenado previamente determina cuales imágenes continúan a la siguiente etapa de la segmentación y cuales no. Para el tema en estudio se hicieron uso de dos modelos de extracción de características: ResNet50/101/152 y MobileNetV2.

3.4.1. ResNet50/101/152[26]

ResNet corresponde a la abreviatura de *Residual Networks*, es una red neuronal utilizada como columna vertebral para muchos trabajos de visión computacional. Esta red se dio a conocer por ser introducida por Microsoft, ganando la competición ILSVRC (ImageNet Large Scale Visual Recognition Challenge) en el año 2015.

Este tipo de redes convolucionales profundas han logrado un resultado de clasificación de imágenes a niveles humanos. Esto se explica porque las redes residuales extraen características y clasificadores de nivel medio y alto en una capa múltiple de extremo a extremo, y la

cantidad de capas que posee apiladas enriquecen los "niveles"de características.

Cuando una red profunda comienza a converger se ve expuesta a un problema de degradación, esto quiere decir que al aumentar la profundidad de la red, la precisión se satura y se degrada rápidamente. Esta degradación no es producto del *overfitting* (sobreajuste) o por el hecho de agregar más capas a una red profunda lo que conduzca a un error de entrenamiento mayor [24].

Como solución a esta problemática Microsoft propuso un marco de aprendizaje residual profundo. En lugar de esperar que cada pocas capas apiladas encajen directamente en una asignación subyacente deseada, dejan explícitamente que estas capas se ajusten a una asignación residual. Para esto se utilizaron conexiones de acceso directo las cuales son aquellas que omiten una o más capas, tal como se muestra en la figura. Estas conexiones de acceso directo realizan un mapeo de la identidad y a su salidas se agregan a las salidas de las capas apiladas de la red, obteniendo la formulación de $F(x) + x$ [23].

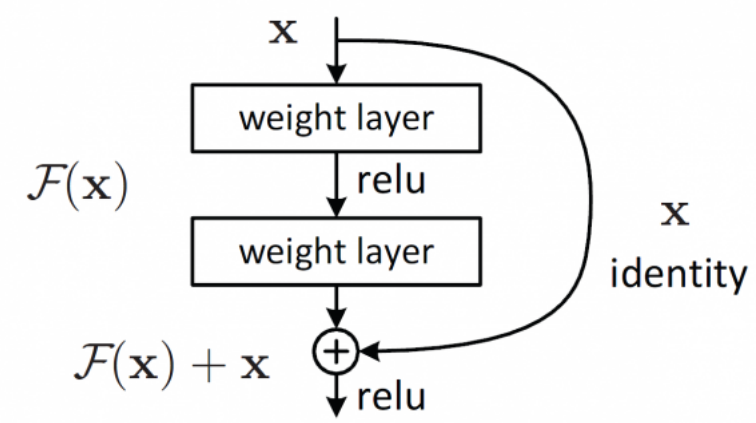

Figura 3.10: Bloque Residual[24][26]

Esto le permite a la red apilar capas adicionales y construir una red más profunda, compensando el gradiente que se desvanece al permitir que la red se salte las capas y sienta que son menos relevantes en el entrenamiento.

Para entrenar esta red los investigadores utilizaron ImageNet, que es un conjunto de datos de millones de imágenes de alta resolución etiquetadas que pertenecen aproximadamente a 22.000 categorías. Estas imágenes fueron recolectadas de internet y etiquetadas de forma manual.

Por otro lado es importante mencionar que la arquitectura de una ResNet cambia dependiendo del número de capas que esta presenta, estos número de capas pueden ser de 18, 34, 50, 101 y de 152. A continuación se presenta una tabla comparativa de las arquitecturas posibles de una ResNet.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figura 3.11: Arquitecturas ResNet[24][26]

Cada bloque ResNet que posee una profundidad de 2 capas (se utiliza en redes pequeñas como ResNet 18, 34) y cada bloque ResNet que posee una profundidad de 3 capas (se utiliza en redes mas grandes como ResNet 50, 101, 152). Tal como se muestra en la siguiente figura:

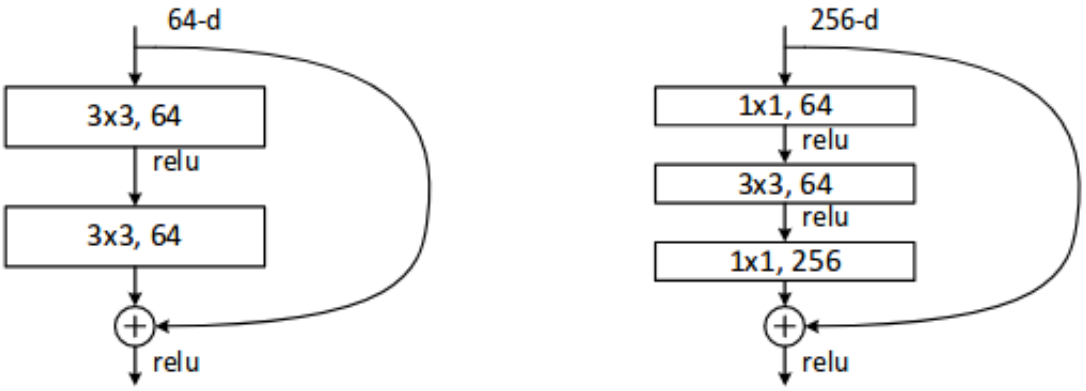

Figura 3.12: Arquitecturas ResNet

3.4.2. MobileNetV2 [27]

El año 2017 un grupo de investigadores de *Google* publicó un artículo que introdujo una arquitectura de red neuronal optimizada para dispositivos móviles. Se esforzaron por un modelo que ofreciera una alta precisión mientras mantenía los parámetros y las operaciones matemáticas lo más bajo posible. Esto con la finalidad de poder llevar las redes neuronales

profundas a los teléfonos inteligentes.

La arquitectura denominada MobileNetV2 corresponde a la segunda versión de la arquitectura MobileNet, esta gira en torno a la idea de utilizar convoluciones separables en profundidad, que consisten en una convolución en profundidad y una en sentido puntual una tras otra.

Principalmente MobileNetV2 sigue un enfoque estrecho->ancho->estrecho. El primer paso reduce la red usando una convolución 1x1 porque la siguiente convolución de profundidad 3x3 ya reduce en gran medida el número de parámetros. Luego otra convolución 1x1 exprime la red para que coincida con el número inicial de canales. En la siguiente imagen se esquematiza esta estructura de MobileNetV2.

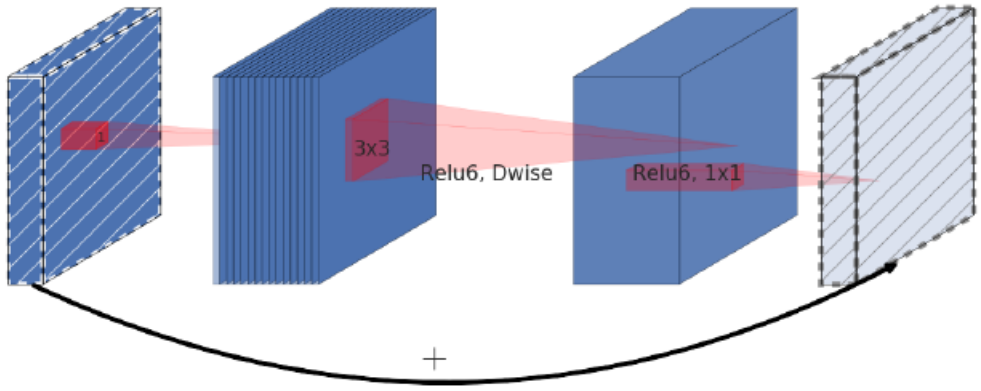

Figura 3.13: Bloque MobileNetV2 [28]

Mirando un poco mas de cerca el bloque MobileNetV2 con respecto al número de canales. La entrada tiene una gran cantidad de canales, que se comprimen con una convolución 1x1 económica. De esa manera, la siguiente convolución 3x3 tiene muchos menos parámetros. Para agregar entradas y salidas al final, el número de canales se incrementa nuevamente usando otra convolución 1x1. De esta forma disminuyen los requerimientos necesarios para que este tipo de redes neuronales puedan ser usadas en teléfonos inteligentes.

En la siguiente imagen se puede observar la arquitectura MobileNetV2:

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Figura 3.14: Arquitectura MobileNetV2 [28]

Se puede ver cómo se organizan los bloques de cuello de botella. En la imagen la columna t representa la tasa de expansión de los canales. Se puede ver que se usó un factor de 6. La columna c representa el número de canales de entrada y la columna n con qué frecuencia se repite el bloque. Por último, s dice si la primera repetición de un bloque usó una zancada de 2 para el proceso de disminución de resolución o una de 1. En general, se tiene que es un conjunto muy simple y común de bloques convolucionales [28].

3.5. Modelos

Los modelos escogidos para realizar este estudio corresponde a *Encoder-Decoder* y *Encoder-Decoder with Skip Connections*, dado que presentan la base de la mayoría de los modelos usados para la detección de objetos por medio de la segmentación semántica, es por eso que es de principal interés el estudiar y entender como trabaja este tipo de modelos, como se optimiza su red y los resultados obtenidos a partir de esto. Por otro lado se compara con otro modelo que usa como base el *Encoder-Decoder* pero varia su estructura al agregarle una convolución de *Atrous*, con el fin de poder ver cuanto varían los resultados del Encoder-Decoder al agregarle este tipo de convolución.

3.5.1. Segnet:Encoder-Decoder[29]

Este modelo Segnet corresponde a una red neuronal profunda totalmente convolucional para la segmentación semántica de píxeles. Este modelo entrenable de segmentación consta como principal motor una red de codificador y una red de decodificador seguida de una capa de clasificación de píxeles.

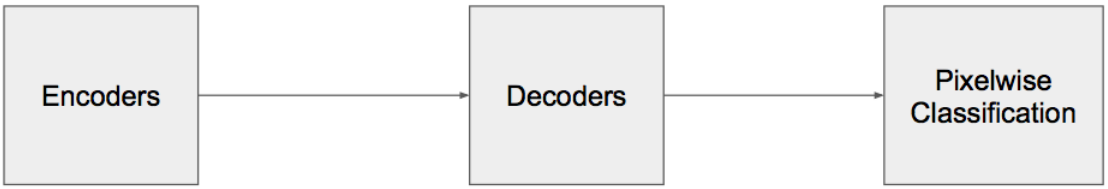

Figura 3.15: Arquitectura general Segnet [29]

Estos quiere decir a modo general que el codificador es una red que mapea la imagen de entrada y emite un mapa de características, el cual puede ser en vector o tensor. Estos vectores de características contienen la información, las características, que representan la entrada. Por otra parte el decodificador que sigue inmediatamente después del codificador, es una nueva red que tiene como función mapear los mapas de características del codificador de baja resolución a los mapas de características de resolución de entrada completa para la clasificación en píxeles, este ultimo hace el etiquetado semántico de estos.

El principal diseño de SegNet surge de la necesidad de asignar características de baja resolución a la resolución de entrada para la clasificación en píxeles. Este mapeo debe producir características que sean útiles para la localización precisa de límites. Esto se logra en la forma que el decodificador realiza el *upsampling* a los mapas de características de entrada de menor resolución a mayor resolución. Específicamente, el decodificador utiliza índices de agrupación calculados en la etapa de *max-pooling* del codificador correspondiente para realizar un *upsampling* no lineal. Lo cual elimina la necesidad de aprender a *upsampling* los mapas de características de baja resolución.

Tal como se menciono anteriormente SegNet tiene una red de codificador y una red de decodificador correspondiente, seguida de una capa de clasificación final de píxeles. La red del codificador consta de 13 capas convolucionales que corresponden a las primeras 13 capas convolucionales de la red VGG16 [20], diseñadas para la clasificación de objetos. Por lo tanto, se puede inicializar el proceso de entrenamiento a partir de pesos entrenados para la clasificación en grandes conjuntos de datos. También podemos descartar las capas completamente conectadas para conservar mapas de características de mayor resolución en la salida más profunda del codificador. Esto también reduce significativamente el número de parámetros en la red del codificador SegNet (de 134 a 14.7 M) en comparación con otras arquitecturas. Cada capa de codificador tiene una capa de decodificador correspondiente y, por lo tanto, la red de decodificador tiene 13 capas. La salida final del decodificador se alimenta a un clasificador soft-max multiclase para producir las probabilidades de cada clase para cada píxel de forma independiente.

3.5.1.1. Arquitectura

Cada codificador en la red del codificador realiza una convolución con un banco de filtros para producir un conjunto de mapas de características. Estas características luego

se normalizan por lotes. Posteriormente, se le aplica la función no lineal ReLu como el máx$(0, x)$. Después de eso, un *max-pooling* con una ventana (22) y un *stride* (zancada) de 2 (ventana no superpuesta) y la salida resultante se le realiza un *upsampling* por un factor de 2. El *max-pooling* se utiliza para lograr que no varié la lectura de la información con los pequeños cambios espaciales de la imagen de entrada. El *Upsampling* da como resultado un gran contexto de imagen de entrada (ventana espacial) para cada píxel en el mapa de características. Si bien varias capas de *max-pooling* y de *Upsampling* pueden lograr una mayor invariancia de traducción para una clasificación profunda correspondiente, existe una pérdida de resolución espacial de los mapas de características.La representación de imagen cada vez más con pérdida (detalle de límites) no es beneficiosa para la segmentación donde la delineación de límites es vital. Por lo tanto, es necesario capturar y almacenar la información de límites en los mapas de características del codificador antes de realizar el submuestreo. Si la memoria durante el proceso no está limitada, se pueden almacenar todos los mapas de características del codificador (después del *upsampling*). Este no suele ser muy eficiente, dado que el tiempo de procesamiento aumentaría en demasía, por lo tanto, para evitar esto se realiza una forma más eficiente para almacenar esta información. Esto implica almacenar solo los índices del *max-pooling* , es decir, las ubicaciones del valor máximo de la característica en cada ventana de agrupación se memorizan para cada mapa de características del codificador. Esto se logra usando 2 bits para cada 22 ventana de agrupación y, por lo tanto, es mucho más eficiente de almacenar en comparación con la memorización de mapas de características.

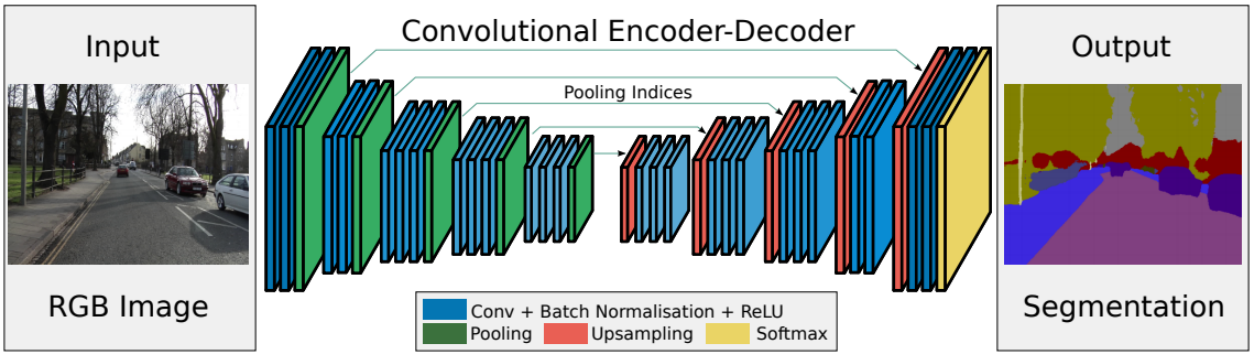

Figura 3.16: Arquitectura Segnet [29]

El decodificador apropiado en la red del decodificador muestrea sus mapas de características de entrada utilizando los índices del *max-pooling* memorizados del mapa de características del codificador correspondiente. Este paso produce mapas de entidades dispersos. Estos mapas de características se combinan con un banco de filtros de decodificador entrenable para producir mapas de características densos. Luego se aplica un paso de normalización por lotes a cada uno de estos mapas. A pesar de que, la entrada del codificador tiene tres canales (RGB), el decodificador correspondiente al primer codificador (el más cercano a la imagen de entrada) produce un mapa de características multicanal. Esto es diferente solo en el primer decodificador, ya que los otros decodificadores en la red que producen mapas de características lo hacen con el mismo número de tamaño

y canales que sus entradas en el codificador. La representación de características de alta dimensión en la salida del decodificador final se alimenta a un clasificador soft-max entrenable. Este soft-max clasifica cada píxel de forma independiente. La salida del clasificador soft-max es una imagen del canal K de probabilidades donde K es el número de clases.

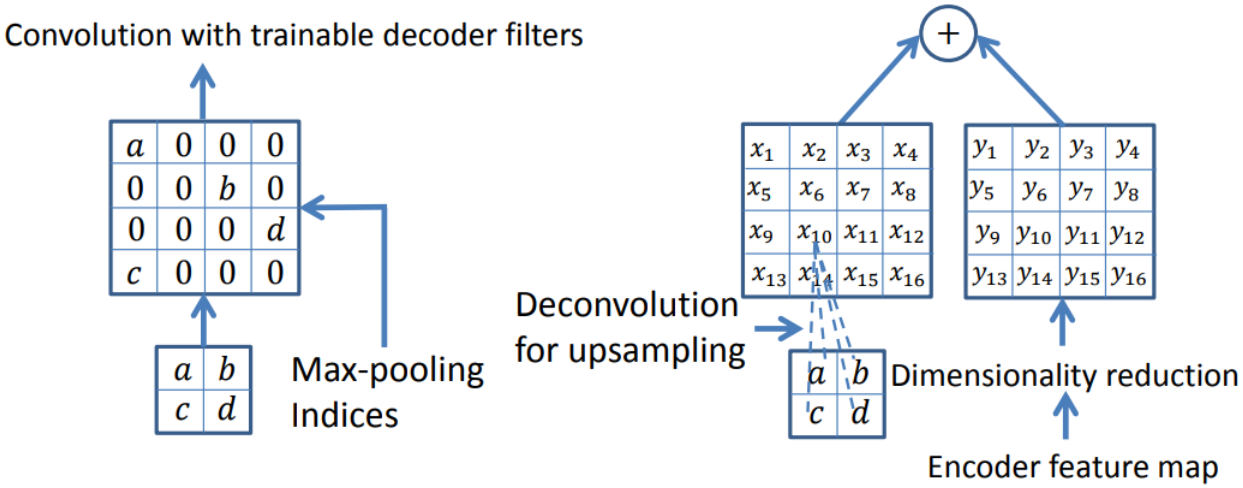

Figura 3.17: Se guardan los índices max-pooling para el upsampling [29]

3.5.2. Segnet:Encoder-Decoder with Skip Connections [29]

El modelo de *Encoder-Decoder with skip connections* es similar al anterior solo que varia en las conexiones entre los bloques de las capas de la red. Esto se debe a que las redes convolucionales que son sustancialmente más profundas, son más precisas y más eficientes para entrenar si contienen salto entre conexiones, es decir, que poseen una conexión más corta entre las capas cercanas a la entrada y las cercanas a la capa de salida.

Para poder comprender un poco más estos se usará el siguiente ejemplo, si se visualiza la superficie de pérdida (el espacio de búsqueda para la pérdida variable de la predicción del modelo), esto se compara como una serie de colinas y valles como muestra la imagen de la izquierda en el diagrama a continuación. La pérdida más baja es el punto más bajo. La investigación ha demostrado que una red óptima más pequeña puede ignorarse incluso si es una parte exacta de una red más grande. Esto se debe a que la superficie de pérdida es demasiado difícil de navegar. Esto significa que al agregar muchas capas profundas a un modelo puede empeorar la predicción.

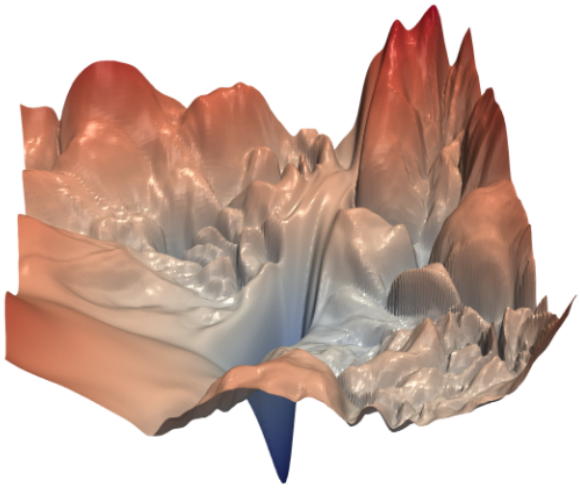

Figura 3.18: Sin skip connections[30]

Una solución que ha sido muy efectiva es agregar conexiones cruzadas entre las capas de la red permitiendo que se salten grandes secciones si es necesario. Esto crea una superficie de pérdida que se parece a la imagen a continuación. Esto hace mucho más fácil para que el modelo sea entrenado con pesos óptimos para reducir la pérdida [30].

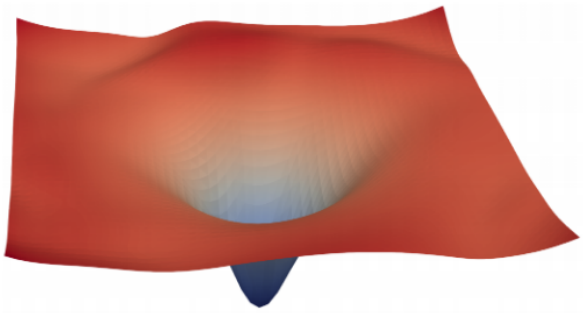

Figura 3.19: Con skip connections[30]

Cada bloque tiene dos conexiones desde su entrada, una que pasa por una serie de convoluciones, normalización por lotes y funciones lineales y la otra conexión que omite esa serie de convoluciones y funciones. Estos se conocen como conexiones de identidad, cruzadas u omitidas. Las salidas de tensor de ambas conexiones se suman [30].

3.5.3. DeepLabV3+ (Encoder-Decoder with Atrous Convolution)[31]

El modelo DeepLabv3+ corresponde a una red neuronal profunda que aplica varias convoluciones *atrous* paralelas con diferentes tasas (llamada específicamente *Atrous Spatial Pyramid Pooling*, o ASPP). Estos es útil dado que la información semántica importante

está codificada en el último mapa de características, y la convolución *atrous* permite extraer mapas de características más densos, esto soluciona la falta de información detallada de la ubicación de los índices de los píxeles debido a las agrupaciones o convoluciones con operaciones de zancada dentro de la red.

La rica información semántica está codificada en la salida de DeepLabv3+, con una convolución atrous que permite controlar la densidad de las características del codificador. Además, el módulo decodificador permite la recuperación detallada de los límites del objeto. En la siguiente imagen se puede observar la arquitectura del modelo de forma general, más adelante se hondeará aún más en esta.

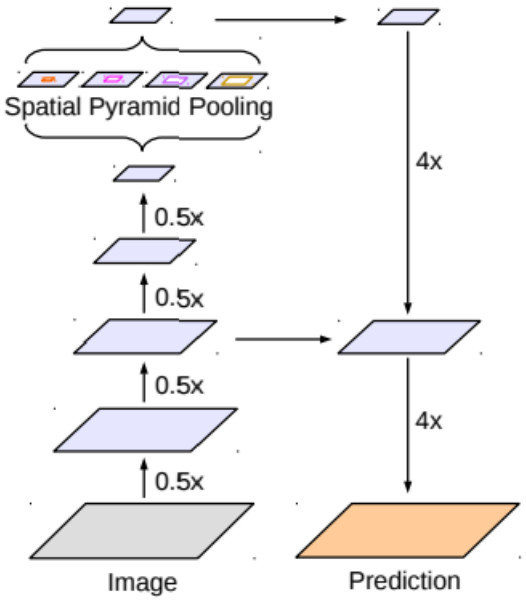

Figura 3.20: Arquitectura General DeepLabv3+ [31]

3.5.3.1. Arquitectura

A continuación se nombrarán y detallarán las partes presentes en la arquitectura del modelo DeepLabv3+.

Una de las partes mas importantes e innovadora de la red, es la convolución *Atrous* la que corresponde a una herramienta poderosa que permite controlar explícitamente la resolución de las características calculadas por las redes neuronales convolucionales profundas y ajusta el campo de visión del filtro para capturar la información de múltiples escalas, es decir, generaliza la operación de convolución estándar. En particular, en el caso de señales bidimensionales, para cada ubicación i en el mapa de características de salida y y un filtro de convolución w, se aplica una convolución atrous sobre el mapa de características de entrada x, siguiendo la ecuación mostrada a continuación [31]:

$$y[\mathrm{i}] = \sum_{k} x[\mathrm{i} + r \cdot k] w[k] \tag{3.8}$$

Donde r representa la tasa *atrous* la que determina el paso con el que es muestrada la señal de entrada. En el caso de $r = 1$ se tiene el caso especial de la convolución estándar. En la siguiente imagen se muestra la convolución de *atrous* para diferentes valores de r [32].

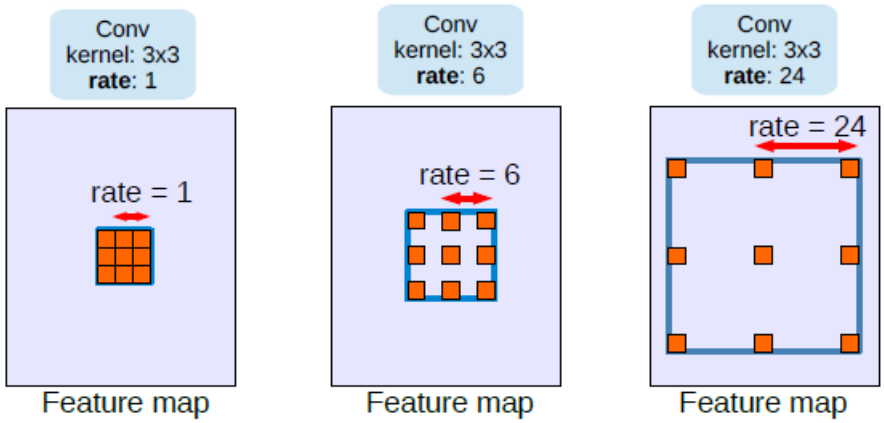

Figura 3.21: Convolución de atrous para diferentes r [31]

Al tratarse de un *Encoder-Decoder* posee un codificador y un decodificador al igual que el modelo SegNet descrito anteriormente. El codificador del modelo DeepLabv3+ es el encargado de la tarea de clasificación de imágenes. La resolución espacial de los mapas de entidades finales suele ser 32 veces menor que la resolución de la imagen de entrada y, por lo tanto, la zancada de salida es de 32. Para la tarea de segmentación semántica, se utiliza la zancada de salida de 16 (o 8) para una extracción de características más densa mediante la eliminación de la zancada en el último o dos últimos bloques y aplicando la convolución *atrous* correspondientemente. Además, se puede aumentar el módulo de la *Atrous Spatial Pyramid Pooling*, que sondea características convolucionales a múltiples escalas mediante la aplicación de convoluciones *atrous* con diferentes velocidades, con las características de nivel de imagen. Se usa el último mapa de características como salida del codificador en la estructura de codificador-decodificador. El modelo cuenta con un mapa de funciones de salida del codificador generalmente de 256 canales [32].

Las características del codificador se muestrean primero bilinealmente por un factor de 4 y luego se concatenan con las características correspondientes de bajo nivel de la red troncal que tiene la misma resolución espacial que decodificador. Luego en el decodificador se le aplica otra convolución 1 1 en las funciones de bajo nivel para reducir el número de canales, ya que las funciones de bajo nivel correspondientes generalmente contienen una gran cantidad de canales (por ejemplo, 256 o 512) que pueden superar la importancia de las funciones de codificador enriquecido (solo 256 canales en nuestro modelo) y dificultan la capacitación. Después de la concatenación, se aplican unas pocas convolucio-

nes 3 3 para refinar las características seguidas de otro muestreo bilineal simple por un factor de 4. En la siguiente imagen se muestra de forma práctica la estructura del modelo [32].

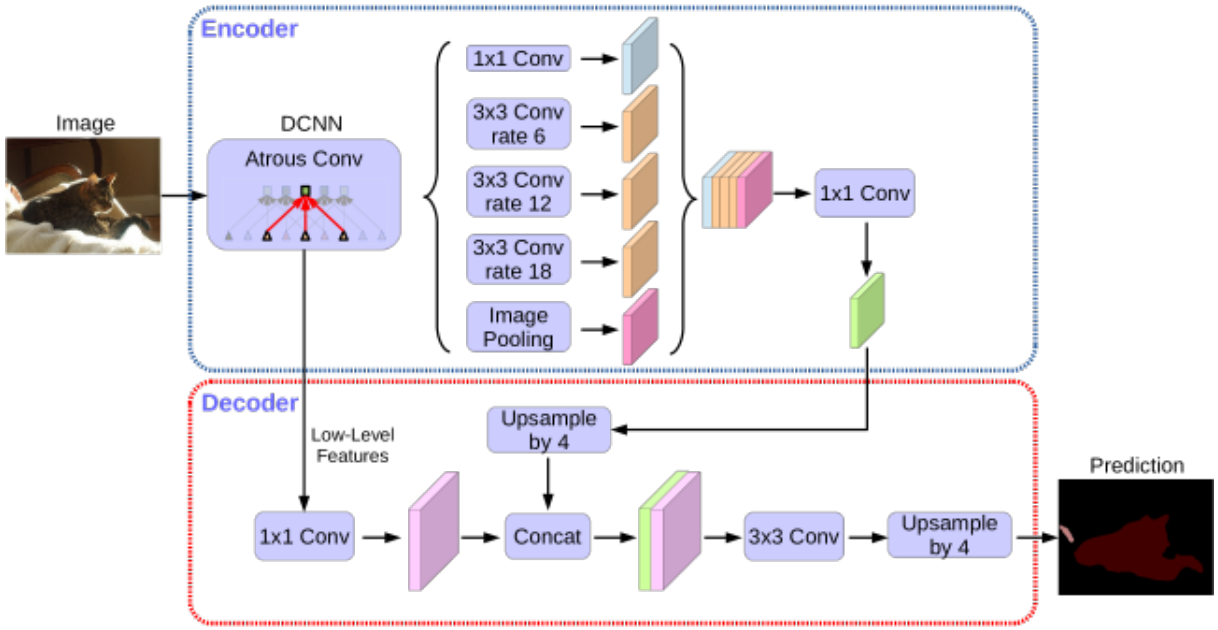

Figura 3.22: Arquitectura DeepLabv3+ [31]

3.5.4. Dataset Externo [35]

En la presente memoria se someterá el modelo optimizado a un dataset externo [35], el cual contiene veinte mil imágenes de grietas de hormigón, con el fin de verificar el modelo.

3.5.5. Experimentos a realizar

En los modelos de redes neuronales se busca siempre poder optimizar una red lo máximo como sea posible, esto se logra por medio del método ensayo y error, esto quiere decir que se prueba cierta configuración óptima de hiperparámetros y dependiendo de que tan cercano da el resultado a lo esperado se determina si se vuelve o no a realizar otra configuración distinta de los hiperparámetros. Para esto es necesario guiarse o compararlo con algún valor o métricas que nos haga una idea de que tan cerca estamos del punto óptimo buscado, es por eso que se compararan las métricas probabilísticas descritas anteriormente entre cada intento por optimizar el modelo. Estas métricas probabilísticas usadas son Recall (*Sensitivily*), F1-Score, Accuracy y el IoU, todas estas métricas se usan como el promedio de estas a lo largo de la cantidad de épocas a la que fue sometida la red.

Se realizan 9 ensayos para el modelo Encoder-Decoder, y posteriormente se escoge la configuración óptima para dicho modelo. Para el modelo Encoder-Decoder with Skip se realizan 4 ensayos y se escoge la configuración óptima. Para el modelo a comparar DeepLabV3+ se expone la configuración óptima encontrada.

Capítulo 4

Resultados

En esta sección se presentaran los resultados logrados por el estudio realizado durante la presente memoria, para esto se usaron datos correspondientes a imágenes de grietas obtenidas de un puente de hormigón por medio de un vídeo de la estructura civil. Por medio de este vídeo se obtuvo un *dataset* de 1358 imágenes reales usadas para el entrenamiento del modelo, 679 para la validación del modelo y 227 usadas para el testeo del modelo, dando un total de 2264 imágenes reales usadas para entrenar el modelo. Cada una de estas imágenes reales usadas para el estudio corresponden a imágenes de $96x96$ píxeles, a continuación se muestran algunas de estas imágenes reales.

Figura 4.1: Imágenes real de grietas del dataset

Por otra parte el *dataset* también tiene imágenes de etiquetas para cada imagen obtenida

por medio del vídeo de la estructura civil, estas fueron obtenidas de forma automática y etiquetan la grieta de color blanco y todo lo que no corresponde a la grieta la etiquetan de color negro, en la escala RGB el color blanco corresponde a la configuración $(256, 256, 256)$ y el color negro corresponde a $(0, 0, 0)$ respectivamente. Como se puede entender cada imagen tanto de entrenamiento, validación y testeo poseen su imagen de etiqueta, es por eso que el numero de imágenes de etiqueta totales corresponden a 2264 de $96x96$ píxeles. A continuación se muestran algunas de estas imágenes de etiquetas.

Figura 4.2: Imágenes de etiquetas de grietas del dataset

4.1. Optimización del modelo

En primera instancia la red de cada modelo se optimizó con un *Front-End* fijo para cada configuración de hiperparámetros. El *Front-End* utilizado corresponde a ResNet101 encargado principalmente de la clasificación y extracción de características.

Se mostrará una tabla resumen obtenida en los más importantes ensayos realizado para cada modelo, al momento de referirse a altura imagen y ancho imagen se refiere exclusivamente a las dimensiones de la imagen que entrara a la red, este valor no necesariamente debe ser igual a las dimensiones de las imágenes de entrenamiento.

También se mostrará otra tabla resumen de las métricas probabilísticas obtenidas por dicha configuración de hiperparámetros, las columnas Crack y No-Crack representan los resultados de la precisión de clasificación para las imágenes de grieta y no grieta respectivamente. Para posteriormente mostrar un ejemplo de una imagen real de grieta segmentada por el modelo para cada ensayo respectivamente.

4.1.1. Encoder-Decoder

4.1.1.1. Ensayo 1

Tabla 4.1: Encoder-Decoder: Configuración Hiperparámetros Ensayo 1

Hiperparámetro	Valor
Épocas	300
Altura Imagen	96
Ancho Imagen	96
Batch Size	14
Nº Imágenes para validación	100
Flip Horizontal	False
Flip Vertical	False
Brillo	0
Rotación	0
Tasa de aprendizaje	0.0001

Tabla 4.2: Encoder-Decoder: Resultados Ensayo 1

Etapa	Crack (%)	No-Crack (%)	Recall (%)	F1-Score (%)	IoU (%)	Tiempo
Entrenamiento	88.28	98.62	97.46	97.46	87.27	4 horas
Testeo	86.59	98.42	97.23	97.20	85.51	-

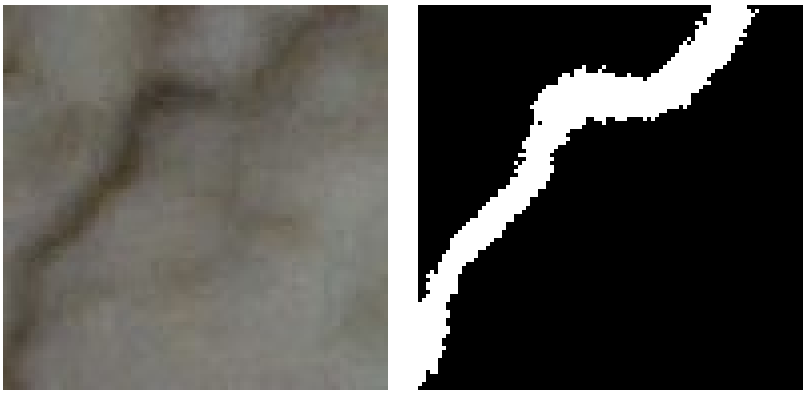

Figura 4.3: Ensayo 1: Grieta real junto a grieta segmentada

4.1.1.2. Ensayo 2

Tabla 4.3: Encoder-Decoder: Configuración Hiperparámetros Ensayo 2

Hiperparámetro	Valor
Épocas	300
Altura Imagen	96
Ancho Imagen	96
Batch Size	14
Nº Imágenes para validación	100
Flip Horizontal	**True**
Flip Vertical	**True**
Brillo	0
Rotación	0
Tasa de aprendizaje	0.0001

Tabla 4.4: Encoder-Decoder: Resultados Ensayo 2

Etapa	Crack (%)	No-Crack (%)	Recall (%)	F1-Score (%)	IoU (%)	Tiempo
Entrenamiento	90.41	98.91	97.89	97.90	89.48	4.5 horas
Testeo	90.51	98.90	98.01	98.00	89.12	-

Figura 4.4: Ensayo 2: Grieta real junto a grieta segmentada

4.1.1.3. Ensayo 3

Tabla 4.5: Encoder-Decoder: Configuración Hiperparámetros Ensayo 3

Hiperparámetro	Valor
Épocas	300
Altura Imagen	96
Ancho Imagen	96
Batch Size	**1**
Nº Imágenes para validación	100
Flip Horizontal	**False**
Flip Vertical	**False**
Brillo	0
Rotación	0
Tasa de aprendizaje	0.0001

Tabla 4.6: Encoder-Decoder: Resultados Ensayo 3

Etapa	Crack (%)	No-Crack (%)	Recall (%)	F1-Score (%)	IoU (%)	Tiempo
Entrenamiento	88.75	99.10	98.05	98.07	89.35	12 horas
Testeo	89.90	98.99	98.12	98.12	89.51	-

Figura 4.5: Ensayo 3: Grieta real junto a grieta segmentada

4.1.1.4. Ensayo 4

Tabla 4.7: Encoder-Decoder: Configuración Hiperparámetros Ensayo 4

Hiperparámetro	Valor
Épocas	300
Altura Imagen	96
Ancho Imagen	96
Batch Size	1
Nº Imágenes para validación	100
Flip Horizontal	**True**
Flip Vertical	**True**
Brillo	0
Rotación	0
Tasa de aprendizaje	0.0001

Tabla 4.8: Encoder-Decoder: Resultados Ensayo 4

Etapa	Crack (%)	No-Crack (%)	Recall (%)	F1-Score (%)	IoU (%)	Tiempo
Entrenamiento	92.74	99.19	98.50	98.50	92.07	12 horas
Testeo	92.13	99.17	98.50	98.49	91.49	-

Figura 4.6: Ensayo 4: Grieta real junto a grieta segmentada

4.1.1.5. Ensayo 5

Tabla 4.9: Encoder-Decoder: Configuración Hiperparámetros Ensayo 5

Hiperparámetro	Valor
Épocas	**100**
Altura Imagen	96
Ancho Imagen	96
Batch Size	1
Nº Imágenes para validación	100
Flip Horizontal	**False**
Flip Vertical	**False**
Brillo	0
Rotación	0
Tasa de aprendizaje	0.0001

Tabla 4.10: Encoder-Decoder: Resultados Ensayo 5

Etapa	Crack (%)	No-Crack (%)	Recall (%)	F1-Score (%)	IoU (%)	Tiempo
Entrenamiento	90.84	99.16	98.30	98.31	90.32	5 horas
Testeo	89.70	99.15	98.25	98.25	90.12	-

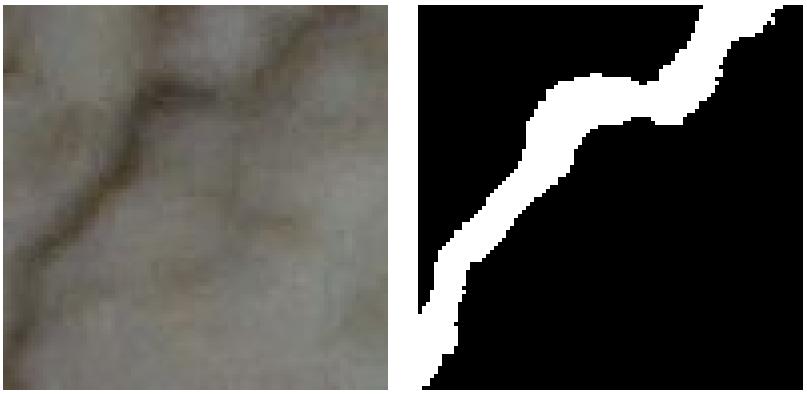

Figura 4.7: Ensayo 5: Grieta real junto a grieta segmentada

4.1.1.6. Ensayo 6

Tabla 4.11: Encoder-Decoder: Configuración Hiperparámetros Ensayo 6

Hiperparámetro	Valor
Épocas	**100**
Altura Imagen	96
Ancho Imagen	96
Batch Size	1
Nº Imágenes para validación	100
Flip Horizontal	**True**
Flip Vertical	**True**
Brillo	0
Rotación	0
Tasa de aprendizaje	**0.001**

Tabla 4.12: Encoder-Decoder: Resultados Ensayo 6

Etapa	Crack (%)	No-Crack (%)	Recall (%)	F1-Score (%)	IoU (%)	Tiempo
Entrenamiento	86.46	99.47	98.11	98.16	89.98	4 horas
Testeo	85.58	99.43	98.10	98.15	89.18	-

Figura 4.8: Ensayo 6: Grieta real junto a grieta segmentada

4.1.1.7. Ensayo 7

Tabla 4.13: Encoder-Decoder: Configuración Hiperparámetros Ensayo 7

Hiperparámetro	Valor
Épocas	100
Altura Imagen	96
Ancho Imagen	96
Batch Size	1
Nº Imágenes para validación	100
Flip Horizontal	True
Flip Vertical	True
Brillo	**30 %**
Rotación	0
Tasa de aprendizaje	**0.00001**

Tabla 4.14: Encoder-Decoder: Resultados Ensayo 7

Etapa	Crack (%)	No-Crack (%)	Recall (%)	F1-Score (%)	IoU (%)	Tiempo
Entrenamiento	92.01	98.96	98.27	98.26	90.15	4 horas
Testeo	91.95	98.95	98.28	98.27	90.44	-

Figura 4.9: Ensayo 7: Grieta real junto a grieta segmentada

4.1.1.8. Ensayo 8

Tabla 4.15: Encoder-Decoder: Configuración Hiperparámetros Ensayo 8

Hiperparámetro	Valor
Épocas	100
Altura Imagen	96
Ancho Imagen	96
Batch Size	1
Nº Imágenes para validación	**300**
Flip Horizontal	True
Flip Vertical	True
Brillo	**60 %**
Rotación	0
Tasa de aprendizaje	**0.0001**

Tabla 4.16: Encoder-Decoder: Resultados Ensayo 8

Etapa	Crack (%)	No-Crack (%)	Recall (%)	F1-Score (%)	IoU (%)	Tiempo
Entrenamiento	90.44	99.19	98.27	98.28	90.58	5.5 horas
Testeo	90.55	99.19	98.24	98.25	90.52	-

Figura 4.10: Ensayo 8: Grieta real junto a grieta segmentada

4.1.1.9. Ensayo 9

Tabla 4.17: Encoder-Decoder: Configuración Hiperparámetros Ensayo 9

Hiperparámetro	Valor
Épocas	100
Altura Imagen	96
Ancho Imagen	96
Batch Size	1
Nº Imágenes para validación	**679**
Flip Horizontal	True
Flip Vertical	True
Brillo	**80 %**
Rotación	0
Tasa de aprendizaje	0.0001

Tabla 4.18: Encoder-Decoder: Resultados Ensayo 9

Etapa	Crack (%)	No-Crack (%)	Recall (%)	F1-Score (%)	IoU (%)	Tiempo
Entrenamiento	91.40	99.03	98.30	98.30	90.61	5.5 horas
Testeo	90.55	99.19	98.24	98.25	90.63	-

Figura 4.11: Ensayo 9: Grieta real junto a grieta segmentada

4.1.2. Configuración óptima Encoder-Decoder

La configuración óptima del modelo Encoder-Decoder corresponde a la mostrada en el ensayo 9, dado que después de varias otras modificaciones de parámetros, ninguno de estos pudo sobre pasar el 91 % de valor de IoU.

También se verifico la calidad del modelo con su configuración óptima con un *dataset* externo de imágenes de grietas, mostrando a continuación algunas de estas segmentaciones obtenidas, junto con los gráficos del *accuracy*, IoU y la función de perdida obtenidas en el entrenamiento y testeo del modelo.

Tabla 4.19: Encoder-Decoder: Configuración Hiperparámetros Óptima

Hiperparámetro	Valor
Épocas	100
Altura Imagen	96
Ancho Imagen	96
Batch Size	1
Nº Imágenes para validación	**679**
Flip Horizontal	True
Flip Vertical	True
Brillo	80 %
Rotación	0
Tasa de aprendizaje	0.0001

Tabla 4.20: Encoder-Decoder: Resultados Óptimos

Etapa	Crack (%)	No-Crack (%)	Recall (%)	F1-Score (%)	IoU (%)	Tiempo
Entrenamiento	91.40	99.03	98.30	98.30	90.61	4.5 horas
Testeo	90.55	99.19	98.24	98.25	90.63	-

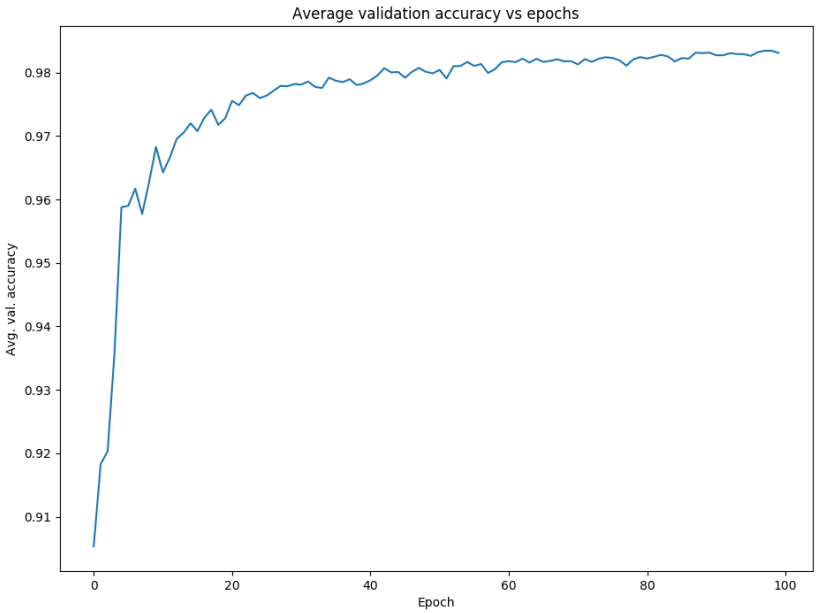

Figura 4.12: Encoder-Decoder Óptimo: Gráfico Accuracy vs Epochs

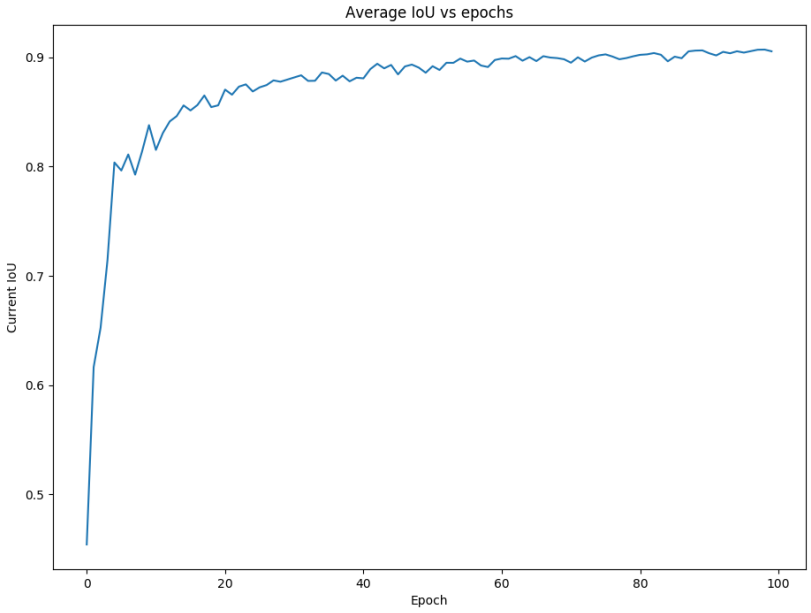

Figura 4.13: Encoder-Decoder Óptimo: Gráfico IoU vs Epochs

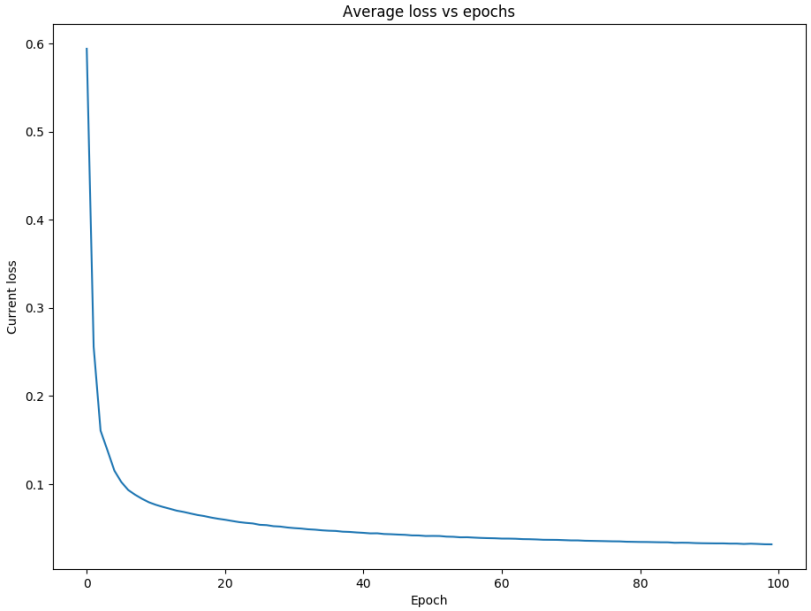

Figura 4.14: Encoder-Decoder Óptimo: Gráfico Loss vs Epochs

Figura 4.15: Encoder-Decoder Óptimo: Grieta real junto a grieta segmentada 1

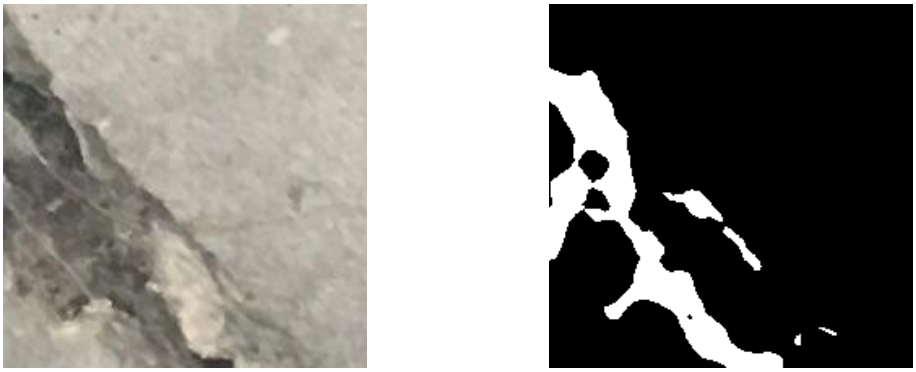

Figura 4.16: Encoder-Decoder Óptimo: Grieta real junto a grieta segmentada 2

Figura 4.17: Encoder-Decoder Óptimo: Grieta real junto a grieta segmentada 3

Figura 4.18: Encoder-Decoder Óptimo: Grieta real junto a grieta segmentada 4

Figura 4.19: Encoder-Decoder Óptimo: Grieta real junto a grieta segmentada 5

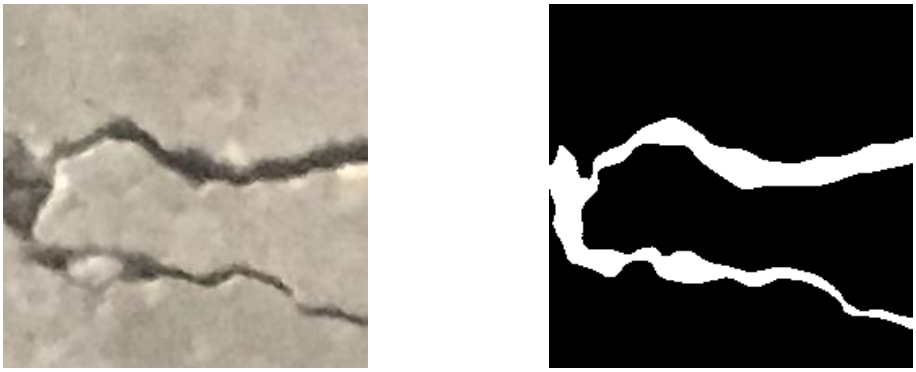

Figura 4.20: Encoder-Decoder Óptimo: Grieta real junto a grieta segmentada 6

4.1.3. Encoder-Decoder with Skip Connections

4.1.3.1. Ensayo 1

Tabla 4.21: Encoder-Decoder with Skip: Configuración Hiperparámetros Ensayo 1

Hiperparámetro	Valor
Épocas	100
Altura Imagen	96
Ancho Imagen	96
Batch Size	14
Nº Imágenes para validación	100
Flip Horizontal	False
Flip Vertical	False
Brillo	0
Rotación	0
Tasa de aprendizaje	0.0001

Tabla 4.22: Encoder-Decoder with Skip: Resultados Ensayo 1

Etapa	Crack (%)	No-Crack (%)	Recall (%)	F1-Score (%)	IoU (%)	Tiempo
Entrenamiento	92.44	99.23	98.31	98.34	91.27	3 horas
Testeo	92.29	99.40	98.55	98.56	92.05	-

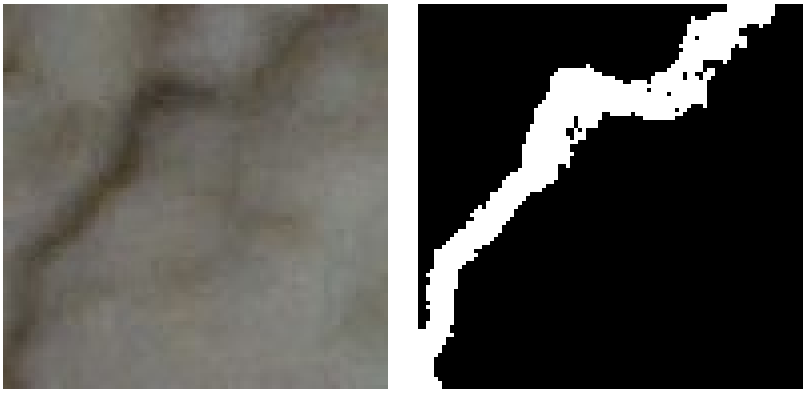

Figura 4.21: Ensayo 1: Grieta real junto a grieta segmentada

4.1.3.2. Ensayo 2

Tabla 4.23: Encoder-Decoder with Skip: Configuración Hiperparámetros Ensayo 2

Hiperparámetro	Valor
Épocas	**300**
Altura Imagen	96
Ancho Imagen	96
Batch Size	**1**
Nº Imágenes para validación	300
Flip Horizontal	False
Flip Vertical	False
Brillo	0
Rotación	0
Tasa de aprendizaje	0.0001

Tabla 4.24: Encoder-Decoder with Skip: Resultados Ensayo 2

Etapa	Crack (%)	No-Crack (%)	Recall (%)	F1-Score (%)	IoU (%)	Tiempo
Entrenamiento	96.41	99.58	99.25	99.25	95.71	13 horas
Testeo	95.86	99.60	99.22	99.23	95.53	-

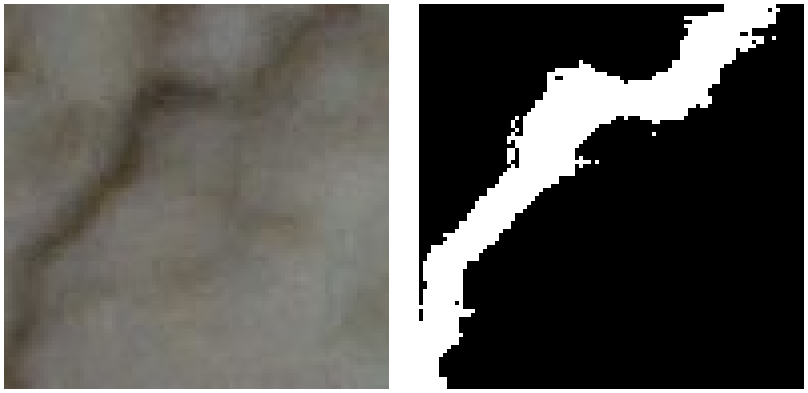

Figura 4.22: Ensayo 2: Grieta real junto a grieta segmentada

4.1.3.3. Ensayo 3

Tabla 4.25: Encoder-Decoder with Skip: Configuración Hiperparámetros Ensayo 3

Hiperparámetro	Valor
Épocas	**100**
Altura Imagen	96
Ancho Imagen	96
Batch Size	1
Nº Imágenes para validación	679
Flip Horizontal	**True**
Flip Vertical	**True**
Brillo	**40 %**
Rotación	0
Tasa de aprendizaje	0.0001

Tabla 4.26: Encoder-Decoder with Skip: Resultados Ensayo 3

Etapa	Crack (%)	No-Crack (%)	Recall (%)	F1-Score (%)	IoU (%)	Tiempo
Entrenamiento	96.17	99.61	99.23	99.23	95.70	5.5 horas
Testeo	96.29	99.59	99.25	99.25	95.72	-

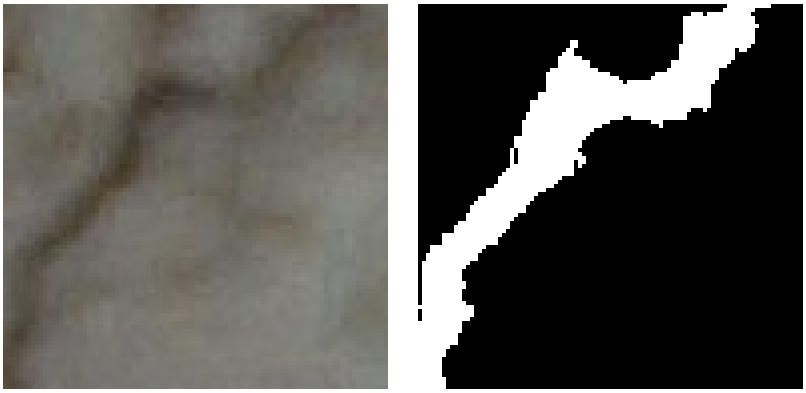

Figura 4.23: Ensayo 3: Grieta real junto a grieta segmentada

4.1.3.4. Ensayo 4

Tabla 4.27: Encoder-Decoder with Skip: Configuración Hiperparámetros Ensayo 4

Hiperparámetro	Valor
Épocas	100
Altura Imagen	96
Ancho Imagen	96
Batch Size	1
Nº Imágenes para validación	**679**
Flip Horizontal	True
Flip Vertical	True
Brillo	**80 %**
Rotación	0
Tasa de aprendizaje	0.0001

Tabla 4.28: Encoder-Decoder with Skip: Resultados Ensayo 4

Etapa	Crack (%)	No-Crack (%)	Recall (%)	F1-Score (%)	IoU (%)	Tiempo
Entrenamiento	96.30	99.62	99.25	99.25	95.84	5.5 horas
Testeo	96.60	99.54	99.25	99.25	95.82	-

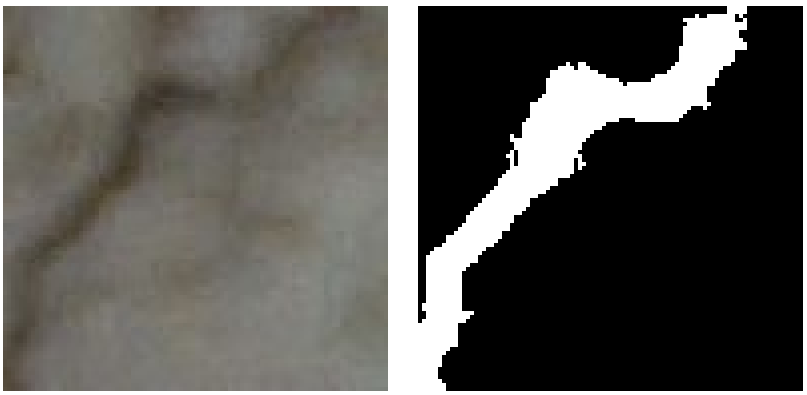

Figura 4.24: Ensayo 4: Grieta real junto a grieta segmentada

4.1.4. Configuración óptima Encoder-Decoder with Skip

La configuración óptima del modelo Encoder-Decoder with Skip corresponde a la mostrada en el ensayo 4, dado que después de varias otras modificaciones de parámetros, ninguno de estos pudo sobre pasar el 96 % de valor de IoU.

También se verifico la calidad del modelo con su configuración óptima con un *dataset* externo de imágenes de grietas, mostrando a continuación algunas de estas segmentaciones obtenidas, junto con los gráficos del *accuracy*, IoU y la función de perdida obtenidas en el entrenamiento y testeo del modelo.

Tabla 4.29: Encoder-Decoder with Skip: Configuración Hiperparámetros Óptimo

Hiperparámetro	Valor
Épocas	100
Altura Imagen	96
Ancho Imagen	96
Batch Size	1
Nº Imágenes para validación	**679**
Flip Horizontal	True
Flip Vertical	True
Brillo	**80 %**
Rotación	0
Tasa de aprendizaje	0.0001

Tabla 4.30: Encoder-Decoder with Skip: Resultados Óptimo

Etapa	Crack (%)	No-Crack (%)	Recall (%)	F1-Score (%)	IoU (%)	Tiempo
Entrenamiento	96.30	99.62	99.25	99.25	95.84	4.5 horas
Testeo	96.60	99.54	99.25	99.25	95.82	-

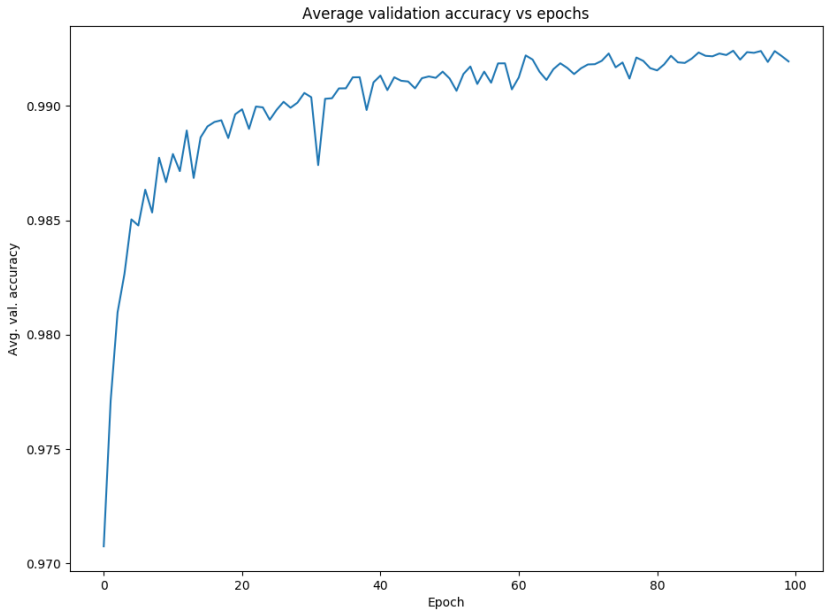

Figura 4.25: Encoder-Decoder with Skip Óptimo: Gráfico Accuracy vs Epochs

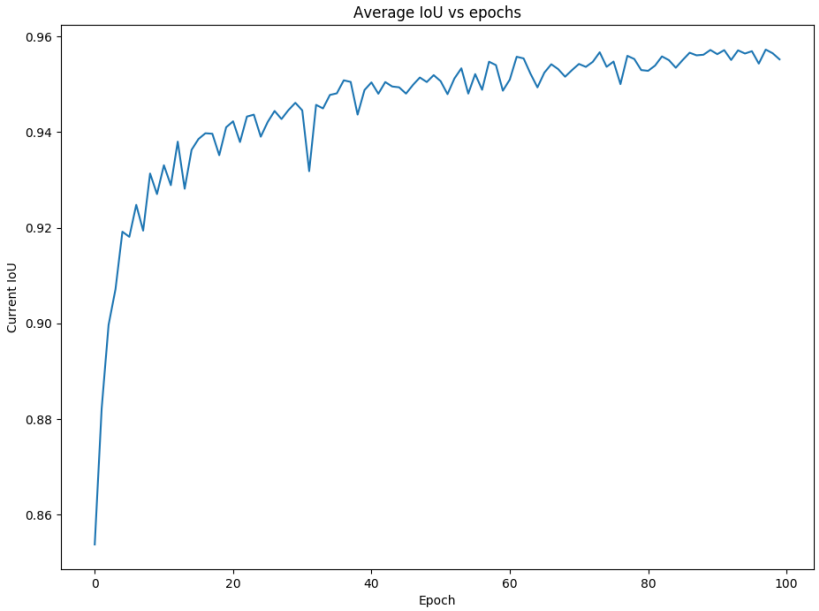

Figura 4.26: Encoder-Decoder with Skip Óptimo: Gráfico IoU vs Epochs

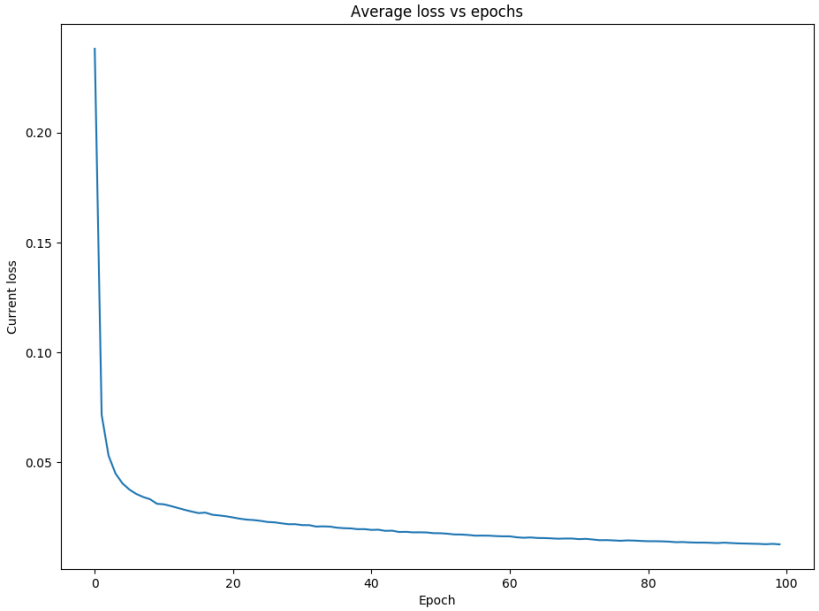

Figura 4.27: Encoder-Decoder with Skip Óptimo: Gráfico Loss vs Epochs

Figura 4.28: Encoder-Decoder with Skip Óptimo: Grieta real junto a grieta segmentada 1

Figura 4.29: Encoder-Decoder with Skip Óptimo: Grieta real junto a grieta segmentada 2

Figura 4.30: Encoder-Decoder with Skip Óptimo: Grieta real junto a grieta segmentada 3

Figura 4.31: Encoder-Decoder with Skip Óptimo: Grieta real junto a grieta segmentada 4

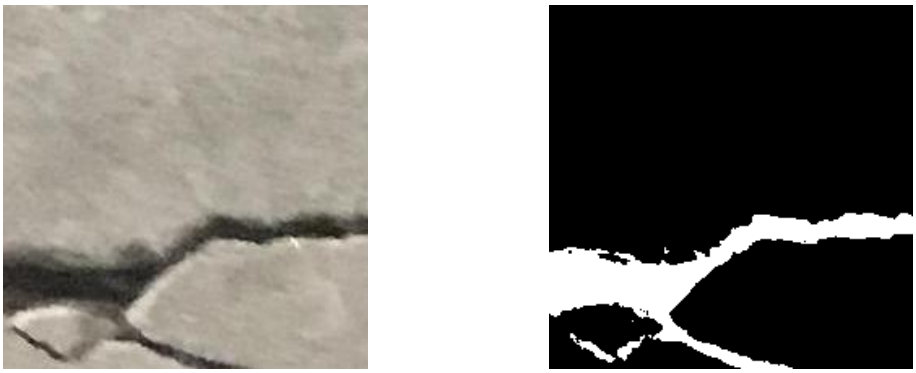

Figura 4.32: Encoder-Decoder with Skip Óptimo: Grieta real junto a grieta segmentada 5

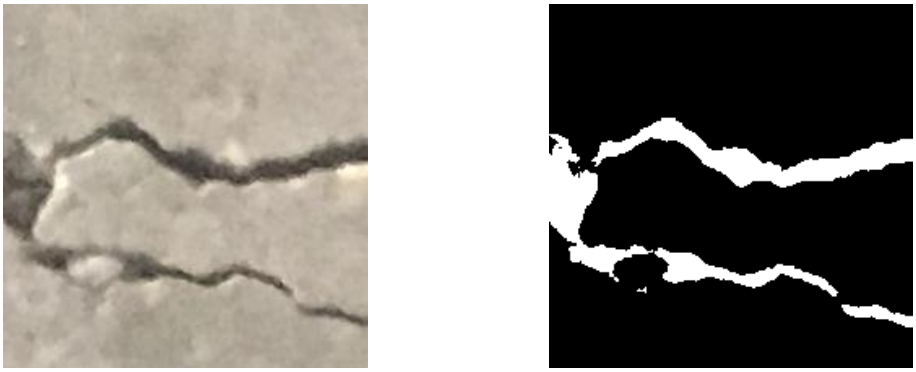

Figura 4.33: Encoder-Decoder with Skip Óptimo: Grieta real junto a grieta segmentada 6

4.1.5. Configuración óptima DeepLabV3+

Dado que el modelo DeepLabV3+ es usado solo para comparación se mostrará solo la solución óptima que se obtuvo luego de realizar varios intentos de ensayo y error.

También se verifico la calidad del modelo con su configuración óptima con un *dataset* externo de imágenes de grietas, mostrando a continuación algunas de estas segmentaciones

obtenidas, junto con los gráficos del *accuracy*, IoU y la función de perdida obtenidas en el entrenamiento y testeo del modelo.

Tabla 4.31: DeepLabV3+: Configuración Hiperparámetros Óptima

Hiperparámetro	Valor
Épocas	100
Altura Imagen	96
Ancho Imagen	96
Batch Size	1
Nº Imágenes para validación	**679**
Flip Horizontal	False
Flip Vertical	False
Brillo	0
Rotación	0
Tasa de aprendizaje	0.0001

Tabla 4.32: DeepLabV3+: Resultados Óptimos

Etapa	Crack (%)	No-Crack (%)	Recall (%)	F1-Score (%)	IoU (%)	Tiempo
Entrenamiento	93.45	99.25	98.68	98.67	92.46	4.25 horas
Testeo	91.65	99.50	98.75	98.76	92.70	-

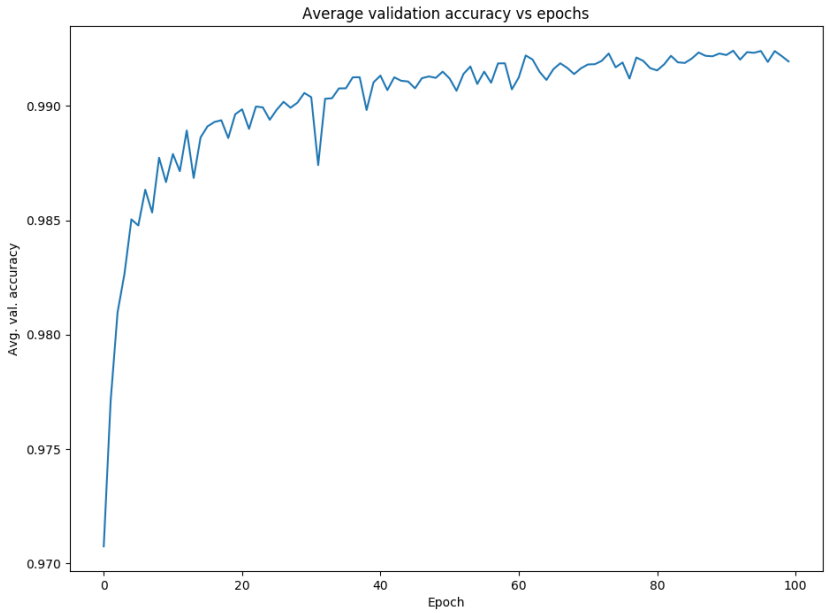

Figura 4.34: DeepLabV3+ Óptimo: Gráfico Accuracy vs Epochs

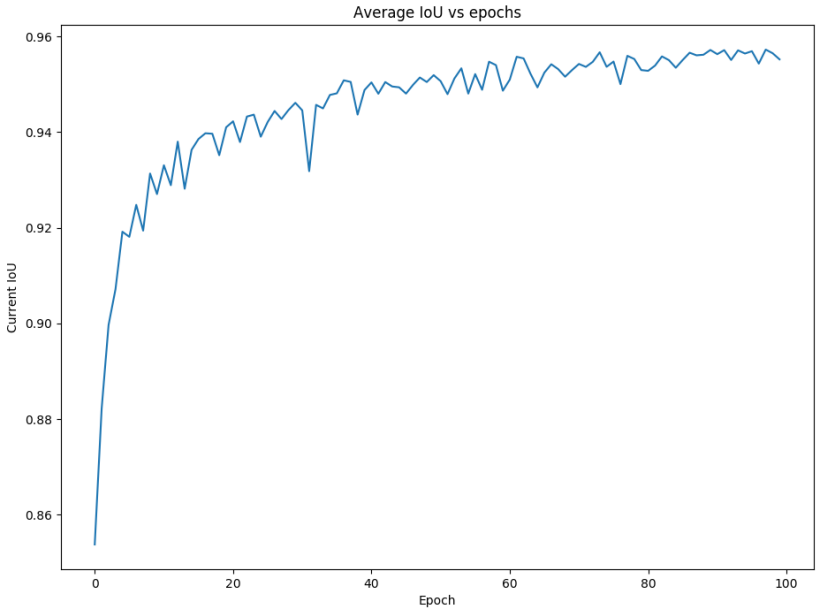

Figura 4.35: DeepLabV3+ Óptimo: Gráfico IoU vs Epochs

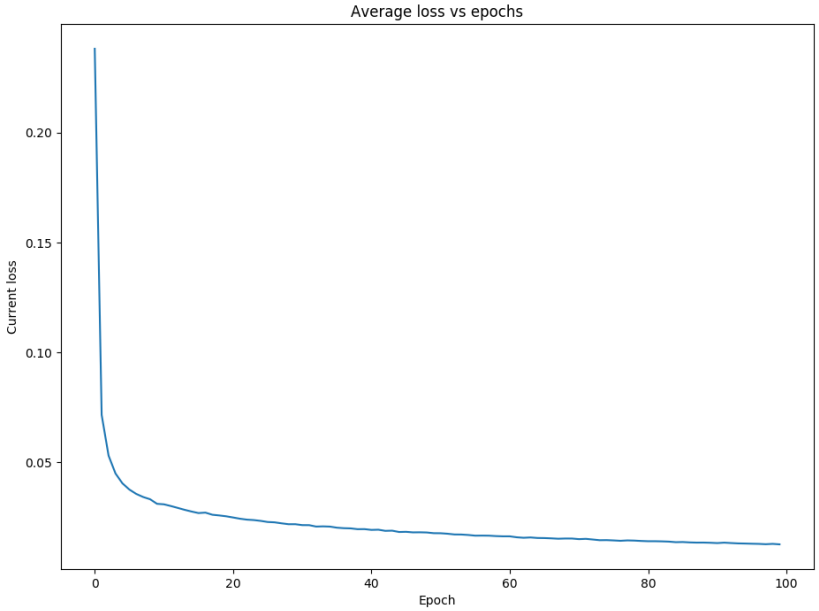

Figura 4.36: DeepLabV3+ Óptimo: Gráfico Loss vs Epochs

Figura 4.37: DeepLabV3+ Óptimo: Grieta real junto a grieta segmentada 1

Figura 4.38: DeepLabV3+ Óptimo: Grieta real junto a grieta segmentada 2

Figura 4.39: DeepLabV3+ Óptimo: Grieta real junto a grieta segmentada 3

Figura 4.40: DeepLabV3+ Óptimo: Grieta real junto a grieta segmentada 4

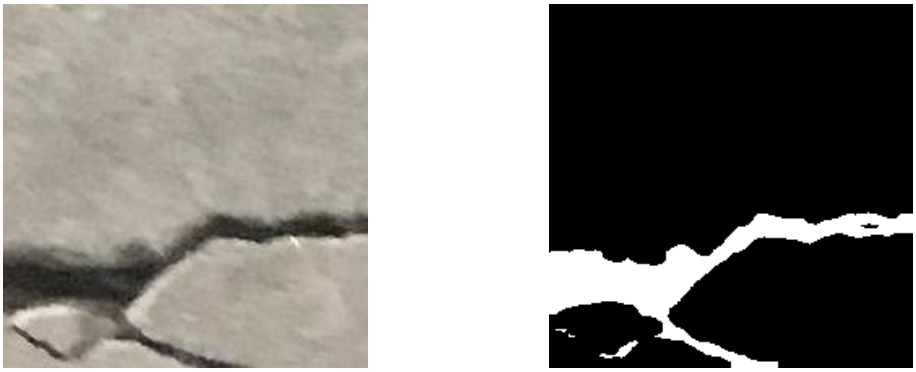

Figura 4.41: DeepLabV3+ Óptimo: Grieta real junto a grieta segmentada 5

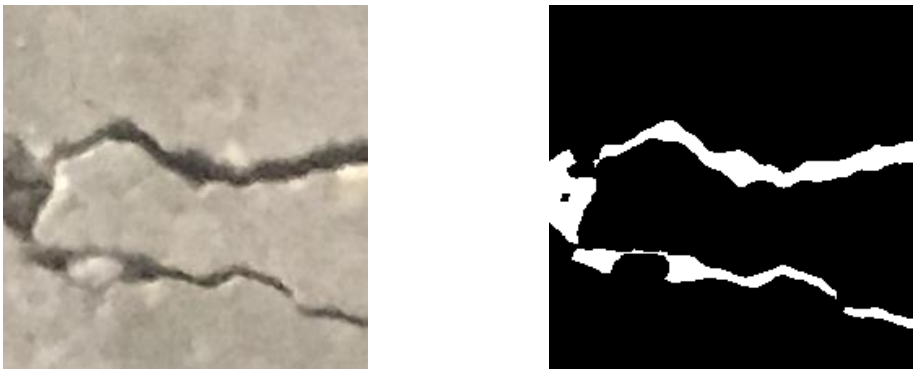

Figura 4.42: DeepLabV3+ Óptimo: Grieta real junto a grieta segmentada 6

4.2. Comparación de Front-Ends

En la presente sección se mostrarán los resultados obtenidos al variar el Front-Ends para cada configuración óptima de los modelos Encoder-Decoder y Encoder-Decoder with Skip respectivamente.

4.2.1. Encoder-Decoder

4.2.1.1. ResNet50

Tabla 4.33: Encoder-Decoder: Resultados ResNet50

Resnet50						
Etapa	Crack (%)	No-Crack (%)	Recall (%)	F1-Score (%)	IoU (%)	Tiempo
Entrenamiento	92.23	98.91	98.29	98.27	90.49	4.0 horas
Testeo	92.67	98.86	98.28	98.26	90.44	-

4.2.1.2. ResNet101

Tabla 4.34: Encoder-Decoder: Resultados ResNet101

Resnet101						
Etapa	Crack (%)	No-Crack (%)	Recall (%)	F1-Score (%)	IoU (%)	Tiempo
Entrenamiento	91.40	99.03	98.30	98.30	90.61	4.5 horas
Testeo	90.55	99.19	98.24	98.25	90.63	-

4.2.1.3. ResNet152

Tabla 4.35: Encoder-Decoder: Resultados ResNet152

Resnet152						
Etapa	Crack (%)	No-Crack (%)	Recall (%)	F1-Score (%)	IoU (%)	Tiempo
Entrenamiento	91.46	99.10	98.37	98.36	90.86	5.7 horas
Testeo	91.92	99.07	98.38	98.38	90.90	-

4.2.1.4. MobileNetV2

Tabla 4.36: Encoder-Decoder: Resultados MobileNetV2

MobileNetV2						
Etapa	Crack (%)	No-Crack (%)	Recall (%)	F1-Score (%)	IoU (%)	Tiempo
Entrenamiento	89.17	99.21	98.24	98.26	90.12	5.75 horas
Testeo	89.31	99.23	98.28	98.29	90.20	-

4.2.2. Encoder-Decoder whit Skip

4.2.2.1. ResNet50

Tabla 4.37: Encoder-Decoder with Skip: Resultados ResNet50

Resnet50						
Etapa	Crack (%)	No-Crack (%)	Recall (%)	F1-Score (%)	IoU (%)	Tiempo
Entrenamiento	95.45	99.66	99.27	99.27	95.78	4.0 horas
Testeo	95.83	99.59	99.25	99.25	95.65	-

4.2.2.2. ResNet101

Tabla 4.38: Encoder-Decoder with Skip: Resultados ResNet101

Resnet101						
Etapa	Crack (%)	No-Crack (%)	Recall (%)	F1-Score (%)	IoU (%)	Tiempo
Entrenamiento	96.30	99.62	99.25	99.25	95.84	4.5 horas
Testeo	96.60	99.54	99.25	99.25	95.82	-

4.2.2.3. ResNet152

Tabla 4.39: Encoder-Decoder whit Skip: Resultados ResNet152

Resnet152						
Etapa	Crack (%)	No-Crack (%)	Recall (%)	F1-Score (%)	IoU (%)	Tiempo
Entrenamiento	96.53	99.56	99.28	99.28	95.88	5.75 horas
Testeo	96.13	99.51	99.19	99.19	95.89	-

4.2.2.4. MobileNetV2

Tabla 4.40: Encoder-Decoder: Resultados MobileNetV2

MobileNetV2						
Etapa	Crack (%)	No-Crack (%)	Recall (%)	F1-Score (%)	IoU (%)	Tiempo
Entrenamiento	96.75	99.56	99.27	99.27	95.83	5.5 horas
Testeo	96.42	99.56	99.25	99.24	95.59	-

4.3. Configuración óptima resumen

Configuración Óptima		
Modelo	IoU Entrenamiento	IoU Testeto
Encoder-Decoder	90.61	90.63
Encoder-Decoder with Skip	95.84	95.82
DeepLabV3+	92.46	92.70

Capítulo 5

Discusión y Análisis

5.1. Encoder-Decoder

De los resultados obtenidos mostrados en la sección anterior de la presente memoria, se comenzara discutiendo los resultados obtenidos para el modelo Encoder-Decoder.

En los primeros dos ensayos, se comenzó probando para un número *batch size* de 14 imágenes, las cuales son usadas en el entrenamiento de la red, este valor quiere decir que por cada época se toman lotes de 14 imágenes y se inyectan en la red, de forma que de un total de 1358 imágenes usadas para el entrenamiento, ingresan un total de 97 lotes de imágenes por cada época, donde cada lote posee 14 imágenes, esto produce que la red disminuya su tiempo de procesamiento dando un tiempo aproximado de 4 horas, pero disminuya la precisión del modelo lo cual se ve reflejado en su valor del IoU menor al 90 %.

En los ensayos posteriores se aprendió de esto y se dejo estable un valor del *batch size* de 1, lo cual se traduce que en cada época entran 1358 lotes de una imagen, lo cual hace mas preciso el modelo, dado que se obtiene el mapa de característica de imagen por imagen, no de un promedio del lote, aparte de la mejora sufrida por el IoU del modelo otro factor se ve afectado directamente como el tiempo de procesamiento, el cual aumenta en sobre manera pasando de un total de 4 horas a 12 horas de procesamiento.

Este gran tiempo hace difícil el análisis dado la gran cantidad de horas que hay que esperar para obtener un solo resultado del modelo, es por esto que buscando disminuir este tiempo en los siguientes ensayos se observo que al modificar las épocas de 300 a 100 disminuía considerablemente el tiempo llegan a ser 5 horas aproximadamente.

Luego de esto se modificaron los valores del Flip horizontal y del Flip vertical a verdadero, esto crea una mayor cantidad de imágenes para el procesamiento de la red dado que crea imágenes rotadas horizontal y verticalmente de las imágenes del *dataset*. Este aumento del *dataset* durante en el entrenamiento debería esperarse que tuviera un impacto positivo en los resultados obtenidos por el modelo para cada ensayo, en el ensayo 4 con 300 épocas tuvo un impacto positivo llegando aumentar el IoU cercano al 92 % promedio, pero este resultado fue descartado por el amplio tiempo de procesamiento de la red.

En el ensayo 6 se permitieron los Flip tanto vertical y horizontal de la red, para los cuales se espera un aumento de las métricas probabilísticas pero para probar el impacto de la tasa de aprendizaje en la red esta se modifico de 10^{-4} a 10^{-3}, esta valor entre 0 y 1 es único para cada red mientras mayor sea este valor más el modelo mas rápido.ªprende pero puede ser que no minimice la función de perdida como se espera, si no que en vez de encontrar el mínimo global se quede en un mínimo local, y para el caso contrario mientras menor sea este valor más "lento.ªprender el modelo, esto puede provocar que la función de perdida del modelo se quede en un mínimo local sin ser capaz de llegar al mínimo global, es decir, se queda antes de camino, para el caso donde se aumento el valor de la tasa de aprendizaje los valores del IoU promedio fueron cercanos a 90 % aproximadamente. Y luego para el caso en que este se disminuyo a 10^{-5} el valor del IoU es un poco mayor pero sigue siendo cercano al 90 %, es por esto que en base a lo obtenido de las tasas de aprendizaje se opto por decidir el valor entre medio de las tasas de aprendizaje probadas de 10^{-4}, donde posiblemente este mas cerca del valor óptimo de la tasa de aprendizaje de la red.

Luego en el ensayo 8 y 9 se comenzó aumentando ciertos valores no modificados anteriormente, como el brillo el cual a tratarse de una imagen casi con puros tonos negros y grises al aumentar el brillo puede que el modelo no se confunda con ciertas áreas sin importancia, frente a estos cambios se obtuvo un leve aumento del valor del IoU, llegando a un valor de 90,62 % promedio para el ensayo 9, el mayor valor obtenido. Esto se puede verificar también al observar las imágenes predichas para cada ensayo donde para los primeros ensayos las imágenes no eran tan detalladas en sus bordes, pero a medida que aumentaba el IoU el detalle de los bordes de las grietas aumentaron considerablemente su calidad. Esto se puede ver reflejado en el resultado descartado debido a la grande demora del tiempo de procesamiento del modelo en el ensayo 4, como de detallada es la imagen segmentada.

Observando los gráficos obtenidos para la configuración óptima se puede ver que la curva del accuracy aumenta a medida que aumentan las épocas. También se puede ver que se estabiliza llegando casi a ser constante para valores mayores al 98 %. En la curva del IoU se puede observar que a medida que aumentan las épocas este valor aumenta, llegando al igual que en el gráfico anterior a estabilizarse en valores cercanos al 90 %. Por otro lado en la función de perdida se ve como disminuye considerablemente a medida que aumentan las épocas, comportándose como una función hipérbola que se acerca cada vez mas al valor 0 sin alcanzar a tocar este.

Al observar las imágenes segmentadas del *dataset* externos con el cual se probó el modelo para la configuración óptima, se puede observar que las segmentaciones no son tan precisas como la imagen real, por lo cual esto se explica con el valor del IoU obtenido, si fuera un mayor valor esta segmentaciones serian aun más precisas y finas en sus detalles.

5.2. Encoder-Decoder with Skip

De los resultados obtenidos mostrados en la sección anterior de la presente memoria, se discutirán los resultados obtenidos para el modelo Encoder-Decoder with Skip.

Al igual que para el modelo anterior se comenzó probando para un *batch size* de 14 para el cual se obtuvo un valor del IoUde 91.27 % para el entrenamiento y un valor del IoU

de 92.05 % para el testeo, en primera instancia se puede observar que estos valor son muy superiores a los obtenidos para el óptimo del modelo Encoder-Decoder with Skip.

Luego se aumentaron las épocas a 300, pero disminuyendo el *batch size*, se obtuvo un valor del IoU sobre el 95.5 % pero en 13 horas procesamiento, lo cual es un valor bastante elevado pero esto se hizo con el fin de saber a cuanto valor del IoU es posible llegar.

Luego disminuyendo la cantidad de épocas y modificando el número de imágenes para la validación, junto usar Flip vertical y horizontal, y aumentando el brillo al 40 % se obtuvo un IoU del 95.7 % por encima del obtenido en en el ensayo anterior para un tiempo de 5.5 horas, muy por menor de las 13 horas del ensayo anterior. En la misma linea para el ensayo 4 se aumento aun mas el brillo a un 80 % se obtuvo un IoU del 95.83 promedio en las mismas 5.5 horas. Esto dejo un logro muy importante para obtener la mayor precisión posible en las segmentaciones de grietas reales, este valor y configuración es considerada la configuración óptima del modelo Encoder-Decoder with Skip. Esto se puede explicar por el salto de conexiones explicado en los antecedentes, el cual permite que el aprendizaje de la red sea mucho mejor.

En los gráficos obtenidos para la configuración óptima del modelo con salto de conexiones, se puede ver que la curva del accuracy aumenta a medida que aumentan las épocas. También se puede ver que se estabiliza llegando casi a ser constante para valores mayores al 99 %, estos *peaks* pequeños presentes se notan mucho mas que en mismo gráfico para el Encoder-Decoder. En la curva del IoU se puede observar que a medida que aumentan las épocas este valor aumenta, llegando al igual que en el gráfico anterior a estabilizarse en valores cercanos al 95.8 %. Al igual que en el gráfico anterior los *peaks* pequeños existentes son mucho mas notorio que el modelo anterior. Por otro lado en la función de perdida se comporta de forma similar a la del modelo anterior y se ve como disminuye considerablemente a medida que aumentan las épocas, comportándose como una función hipérbola que se acerca cada vez mas al valor 0 sin alcanzar a tocar este.

Al observar las imágenes segmentadas del *dataset* externos con el cual se probó el modelo para la configuración óptima, se puede observar que las segmentaciones son bastante precisas como la imagen real y mucho mas que el modelo anterior, por lo cual esto se explica con el valor del IoU cercano al 96 % obtenido, lo que hace que estas segmentaciones sean más precisas y finas en sus detalles.

5.3. Comparación con el modelo DeepLabV3+

Al comparar las configuraciones de ambos modelos Encoder-Decoder y Encoder-Decoder with Skip versus el modelo DeepLabV3+, se puede comparar que el DeepLabV3+ posee un IoU del 92.58 % promedio mayor que el valor del IoU de 90.62 % obtenido en el Encoder-Decoder, para un poco menor de tiempo de diferencia cercano a 15 minutos. Esto se ve reflejado en la calidad de la segmentación por parte del modelo DeepLabV3+ en comparación del Encoder-Decoder, más específicamente en la Figura 4.1.2 donde se puede observar que la imagen segmentada no es capaz de segmentar correctamente al medio de la grieta, por el contrario en la Figura 4.1.5 en la cual es capaz de segmentar en cierta forma detalladamente al medio de la grieta, pero por otro lado en el modelo Encoder-Decoder whit Skip en la misma

Figura 4.1.4 correspondiente a la misma segmentación se puede observar como esta es mucho mas detallada pero cae en una leve sobre-presición ya que segmenta otros puntos anexos a la grieta que no pueden ser considerado grietas por el diminuto tamaño que presentan, esto no quiere decir que sea malo si no que al contrario pero es importante nombrar este punto.

El modelo DeepLabV3+ en comparación con el Encoder-Decoder with Skip posee un porcentaje del IoU bastante menor que el valor del IoU obtenido para el Encoder-Decoder whith Skip cercano 96 %, pero este es obtenido en una hora menos de entrenamiento, al ser un tiempo tan poco comparada con casi 3 % mas de precisión en la segmentación de las imágenes, este valor es insignificante, dado que la precisión en este caso toma una mayor relevancia.

A modo de resumen en esta comparación el modelo Encoder-Decoder with Skip es bastante superior a los otros dos modelos, pero el DeepLabV3+ no se queda atrás y es mejor que el Encoder-Decoder, en este ámbito se puede explicar que al tener salto de conexiones en la red produce que aumente el rendimiento del modelo, y también al agregar la convolución *atrous* como en el caso del DeepLabV3+ también produce un aumento del rendimiento del modelo, se llega esto dado que los tres modelos tienen como base una red de Encoder-Decoder.

5.4. Comparación de Front-Ends

Al comparar los resultados obtenidos para cada variación del Front-End en el Encoder-Decoder, se puede observar que el mayor valor corresponde al obtenido usando la ResNet152 en 5.7 horas. Esto se puede entender debido a que posee mas neuronas usadas para la clasificación y obtención de características específicamente 152 neuronas, el tiempo de procesamiento es un poco mayor.

Por otro lado se tiene otro aspecto importante el Front-End MobileNetV2 enfocado para la inteligencia artificial en dispositivos móviles logra un valor del IoU muy cercanos a los demás Front-End, pero la diferencia de esto es que el tamaño del modelo obtenido va enfocado a memoria de celulares es decir es mucho menor.

Ahora comparando los Front-Ends para el modelo Encoder-Decoder with Skip se tiene algo similar a lo discutido en los párrafos anteriores para el Encoder-Decoder dado que se puede observar que el mayor valor del IoU corresponde al obtenido usando la ResNet152 en 5.75 horas cercano al 95.88 %. El mayor tiempo de procesamiento comparado con la ResNet101 es un poco más de una hora, es por esto para lo poco que aumenta el IoU no es tan rentable usar este Front-End.

Observando el Front-End MobileNetV2 para el modelo Encoder-Decoder with Skip se obtiene un valor del IoU un poco menor que incluso el óptimo encontrado, lo cual confirma que el potencial que tiene este Front-End dado que no solo casi alcanza los valores de IoU casi óptimo si no que a demás entrega un modelo de tamaño mucho menor que los otros tipos de Front-End.

Capítulo 6

Conclusiones

De acuerdo al análisis de los resultados obtenidos, es posible desglosar las siguientes conclusiones.

Gracias a la estructura del modelo el *dataset* pudo ser entrenado, testeado y verificado de forma correcta junto a la flexibilidad que entrega tensorflow para la optimización del modelo. Logrando obtener resultados relevantes para el estudio de detección, tales como los gráficos obtenidos, junto con las métricas probabilísticas y la mas importante de todas las imágenes segmentadas.

Se lograron cumplir los objetivos planteados, en primer lugar el objetivo general se cumplió satisfactoriamente dado que se logró segmentar y detectar grietas por medio de un modelo de redes neuronales (Encoder-Decoder, Encoder-Decoder with Skip y DeepLabV3+) en *tensorflow*, por otra parte se cumplieron los objetivos específicos dado que se estudió el *dataset* de imágenes de grietas al momento de utilizar este en un modelo de *deep learning*, se logró entrenar, testear y verificar los resultados obtenidos en las métricas probabilísticas, se analizaron y compararon las imágenes de grietas segmentadas de cada modelo y se verifico el modelo con un *dataset* externo.

Se logró obtener un valor del IoU bastante satisfactorio, los cuales se acercan a los resultados esperados en las investigaciones recopiladas al momento de realizar el estudio, esto deja la puerta abierta para futuras investigaciones del rubro o para nuevos objetivos que se quieran implantar en el ámbito de la detección automática de grietas en estructuras civiles.

Por otro lado se logró verificar en un *dataset* externo los modelos, permitiendo de esta forma salir de las imágenes de estudio y probar con otro tipo de imágenes reales, de otro tipo de puentes de hormigón de diversas dimensiones y áreas de grietas dando la posibilidad de poder generalizar los modelos estudiados.

Capítulo 7

Bibliografía

[1] JUAN TAPIA, ENRIQUE LOPEZ, CLAUDIO YANEZ, RUBEN BOROSCHEK. 'Automatic Crack Detection and Quantification Based on Concrete Bridges Images'. Chile. Mayo 2019.

[2] Jahanshahi, M. R., Masri, S. F., and Sukhatme, G. S. (2011). Multi-image stitching and scene reconstruction for evaluating defect evolution in structures. Structural Health Monitoring.

[3] Ko, J. M., Ni, Y. Q., Zhou, H. F.,Wang, J. Y., and Zhou, X. T. (2009). Investigation concerning structural health monitoring of an instrumented cable-stayed bridge. Structure and Infrastructure Engineering.

[4] Wang, P. and Huang, H.(2010). Comparison analysis on present image-based crack detection methods in concrete structures. In 2010 3rd International Congress on Image and Signal Processing, volume 5.

[5] Ho, H.-N., Kim, K.-D., Park, Y.-S., and Lee, J.-J. (2013). An effient image-based damage detection for cable surface in cable-stayed bridges. NDT E International,

[6] Min, Y. C. and J., D. S. (2015). Vision-based automated crack detection for bridge inspection. Computer-Aided Civil and Infrastructure Engineering.

[7] Jahanshahi, M. R., Masri, S. F., Padgett, C. W., and Sukhatme, G. S. (2013). An innovative methodology for detection and quantification of cracks through incorporation of depth perception. Machine Vision and Applications.

[8] Iyer, S. and Sinha, S. K. (2005). A robust approach for automatic detection and segmentation of cracks in underground pipeline images. Image and Vision Computing.

[9] Yiyang, Z. (2014). The design of glass crack detection system based on image preprocessing technology. In 2014 IEEE 7th Joint International Information Technology and Artificial Intelligence Conference.

[10] Chen, F. and Jahanshahi, M. R. (2018). Nb-cnn: Deep learning-based crack detection using convolutional neural network and naıve bayes data fusion. IEEE Transactions on Industrial Electronics.

[11] Gao, Y. and Mosalam, K. M. (2018). Deep transfer learning for image-based structural damage recognition. Comp.-Aided Civil and Infrastruct. Engineering.

[12] Garcia-Garcia, A., Orts, S., Oprea, S., Villena-Martinez, V., and Rodríguez, J. G. (2017). A review on deep learning techniques applied to semantic segmentation.

[13] Badrinarayanan, V., Kendall, A., and Cipolla, R. (2015). Segnet: A deep convolutional encoder-decoder architecture for image segmentation.

[14] J égou, S., Drozdzal, M., Vázquez, D., Romero, A., and Bengio, Y. (2016). The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation.

[15] MR.MACKENTY, 2017. Max-pooling / Pooling. [en linea]<https://computersciencewiki.org/index.php/Max-pooling_/_Pooling>[consulta: 10 de agosto 2019]

[16] RAJALINGAPPAA SHANMUGAMANI, 2018. Machine Learning Algorithms for semantic segmentation. [en linea] <https://codealogic.org/2018/05/30/machine-learning-algorithms-for-semantic-segmentation/>[consulta: 2 de septiembre 2019]

[17] SHRUTI SAXENA, 2018. Precision vs Recall. [en linea] <https://towardsdatascience.com/precision-vs-recall-386cf9f89488>[consulta: 30 de agosto 2019]

[18] Precision and recall. [en linea] <https://en.wikipedia.org/wiki/Precision_and_recall>[consulta: 30 de agosto 2019]

[19] KOEHRSEN WILL, 2018. Beyond Accuracy: Precision and Recall. [en linea] <https://towardsdatascience.com/beyond-accuracy-precision-and-recall-3da06bea9f6c>[consulta: 30 de agosto 2019]

[20] CALEB ROBINSON, 2018. Understanding intersection-over-union. [en linea] <https://calebrob.com/ml/2018/09/11/understanding-iou.html>[consulta: 31 de agosto 2019]

[21] BARTOSZ MIKULSKI, 2019. F1 score explained. [en linea] <https://www.mikulskibartosz.name/f1-score-explained/>[consulta: 30 de agosto] item LÓPEZ, E. 2018. "Deep Neural Networks for fault diagnostics and prognostics". [Diapositivas]. Facultad de ciencias físicas y matemáticas, Universidad de Chile. Texto en inglés. 14p.

[22] HIGHAM DAVID, 2018. How we use image semantic segmentation.[en linea]<https://medium.com/digitalbridge/how-we-use-image-semantic-segmentation-e85fac734caf>[consulta: 31 de agosto 2019]

[23] JAY PRAKASH, 2018. Understanding and Implementing Architectures of ResNet and ResNeXt for state-of-the-art Image Classification: From Microsoft to Facebook [Part 1]. [en linea] <https://medium.com/@14prakash/understanding-and-implementing-architectures-of-resnet-and-resnext-for-state-of-the-art-image-cf51669e1624>[consulta: 12 de agosto 2019]

[24] MUNEEB UL HASSAN, 2019. ResNet (34, 50, 101): Residual CNNs for Image Classification Tasks. [en linea] <https://neurohive.io/en/popular-networks/resnet/>[consulta: 12 de agosto 2019]

[25] DWIVEDI PRIYA, 2019. Understanding and Coding a ResNet in Keras. [en linea] <https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33>[consulta: 12 de agosto 2019]

[26] KAIMING HE, XIANGYU ZHANG, SHAOQING REN, JIAN SUN. "Deep Residual Learning for Image Recognition", Diciembre 2015.

[27] MARK SANDLER, ANDREW HOWARD, MENGLONG ZHU, ANDREY ZHMOGINOV, LIANG-CHIEH CHEN. "MobileNetV2: Inverted Residuals and Linear Bottlenecks", Google Inc, Enero 2018.

[28] PRÖVE PAUL-LOUIS, 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks. [en linea] <https://towardsdatascience.com/mobilenetv2-inverted-residuals-and-linear-bottlenecks-8a4362f4ffd5>[consulta: 20 de agosto 2019]

[29] VIJAY BADRINARAYANAN, ALEX KENDALL, ROBERTO CIPOLLA. "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation". Noviembre 2015.

[30] THOMAS CHRISTOPHER. U-Nets with ResNet Encoders and cross connections. [en linea]<https://towardsdatascience.com/u-nets-with-resnet-encoders-and-cross-connections-d8ba94125a2c>[consulta: 30 de julio 2019]

[31] LIANG-CHIEH CHEN, YUKUN ZHU, GEORGE PAPANDREOU, FLORIAN SCHROFF, HARTWIG ADAM. 'Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation'. Google Inc. Febrero 2018

[32] SIK-HO TSANG, 2019. Review: DeepLabv3 — Atrous Convolution (Semantic Segmentation). [en linea] <https://towardsdatascience.com/review-deeplabv3-atrous-convolution-semantic-segmentation-6d818bfd1d74>[consulta: 28 de julio 2019]

[33] PRABHU. Understanding of Convolutional Neural Network (CNN) — Deep Learning. [en linea]<https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148>[consulta: 22 de abril 2019]

[34] SERGIOS KARAGIANNAKOS. Semantic Segmentation in the era of Neural Networks. [en linea] <https://sergioskar.github.io/Semantic_Segmentation/>[consulta: 28 de abril 2019]

[35] LEI ZHANG; FAN YANG; YIMIN DANIEL ZHANG; YING JULIE ZHU. Road crack detection using deep convolutional neural network". Phoenix, USA. Septiembre 2016

Capítulo 8

Anexos

8.1. Nuevas imágenes procesadas del dataset externo por el modelo Encoder-Decoder with Skip

Figura 8.1: Anexo: Encoder-Decoder with Skip Óptimo: Grieta real junto a grieta segmentada

Figura 8.2: Anexo: Encoder-Decoder with Skip Óptimo: Grieta real junto a grieta segmentada

Figura 8.3: Anexo: Encoder-Decoder with Skip Óptimo: Grieta real junto a grieta segmentada

Figura 8.4: Anexo: Encoder-Decoder with Skip Óptimo: Grieta real junto a grieta segmentada

Figura 8.5: Anexo: Encoder-Decoder with Skip Óptimo: Grieta real junto a grieta segmentada

Figura 8.6: Anexo: Encoder-Decoder with Skip Óptimo: Grieta real junto a grieta segmentada

Figura 8.7: Anexo: Encoder-Decoder with Skip Óptimo: Grieta real junto a grieta segmentada

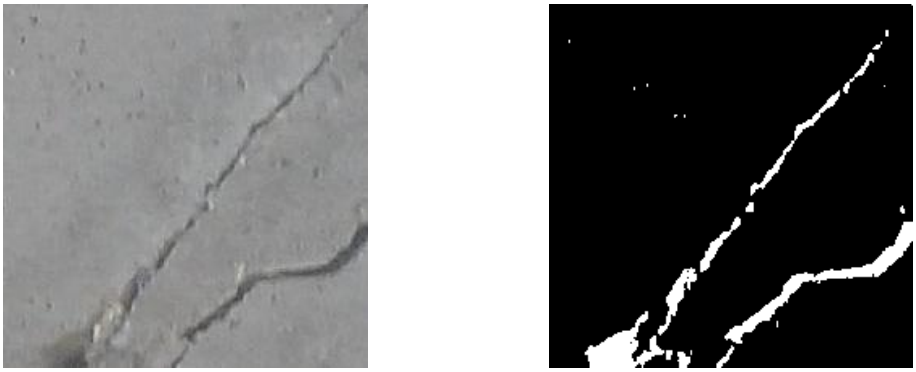

Figura 8.8: Anexo: Encoder-Decoder with Skip Óptimo: Grieta real junto a grieta segmentada

Figura 8.9: Anexo: Encoder-Decoder with Skip Óptimo: Grieta real junto a grieta segmentada