✅ ***Objetivo general:***
Revisar, analizar y comprender los principales parámetros del modelo U-Net que afectan su entrenamiento y rendimiento.

1️⃣ Arquitectura U-Net

Esencialmente la estructura encoder-decoder permite la segmentación de imágenes gracias a que inicialmente se reduce la resolución de la misma (encoder) para extraer su morfología básica, es decir obtener la forma del cuerpo a segmentar. Una vez hecho lo anterior se desea recuperar los detalles de la morfología segmentada por lo que se aumenta la resolución de la imagen (decoder).

Lo anterior se lleva a cabo a través de operaciones (gradiente) de muchas capas. En el cualquier caso de exceso mucho o poco el gradiente afecta más de lo que pudiese ayudar. Para solucionar lo anterior se crearon los saltos de uniones, las capas anteriormente mencionadas se agrupan en n cantidades creando “bloques residuales” de forma que las skip conections pueden realizar atajos de ruta optimizando la cantidad de operaciones a efectuar lo que desde luego reduce el costo de cómputo a la vez que se obtiene mejor calidad en la información.

En resumen, la estructura encoder decoder manipula redes convolucionales (agrupamiento de capas) para obtener la segmentación de una imagen con ayuda de skip conections que mejoran el rendimiento computacional.

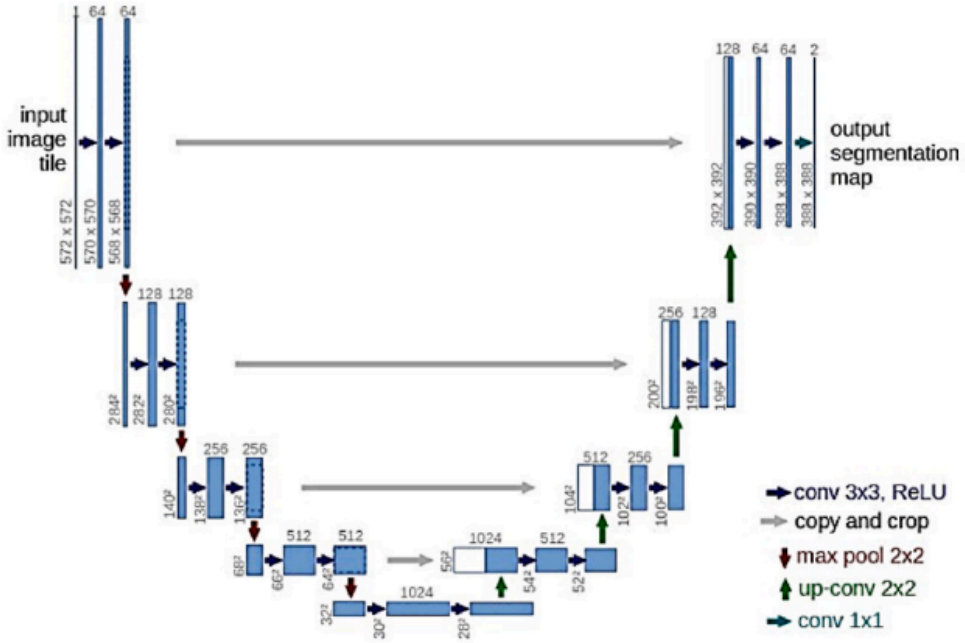


Fig. 2. U-Net model architecture.

El ejemplo por experiencia de red convolucional es U-net, su nombre se debe a la forma de su arquitectura, mientras del lado izquierdo de la U desempeña una trayectoria de contracción para capturar el contexto de la imagen, del lado derecho lleva a cabo la trayectoria de expansión lo que contribuye con el aumento de precisión en la localización de cada pixel de la imagen.

Pero sin duda, la gran ventaja de esta arquitectura se debe a la escasez de datos de entrenamiento (Ronneberger et al., 2015) además de la separación de objetos contiguos de la misma clase tal cómo lo demuestra el artículo de referencia con la segmentación celular. Gracias a su arquitectura Unet segmenta imágenes arbitrariamente grandes de manera rápida.

2️⃣ Los hiperparámetros clave del modelo

La red U-net tiene un total de **23 CNN capas convolucionales** que no están completamente conectadas entre sí permitiendo que la red pueda trabajar con imágenes de cualquier tamaño.

Al ir bajando por el encoder, se reduce la resolución espacial de la imagen (*downsampling* con pooling), pero se **duplica** el **número de filtros**, por ejemplo, en el artículo el primer nivel de encoder consiste a un tamaño espacial de 572x572 correspondiente a 64 filtros, naturalmente si pasamos al siguiente nivel la resolución va a disminuir entonces para compensar esta pérdida se aumenta el número de filtros para conservar o incluso aumentar la información de la imagen. Así sucesivamente se llega a los 1024 filtros, etapa conocida cómo bottleneck.

El **tamaño del kernel** de este modelo es de 3x3 esto permite extraer características de la capa antecesora en la red convolucional durante el decoder, por el contrario en la última capa del decoder el kernel es de 1x1 ya que toma todos los canales de salida del decoder y lo convierte en una predicción final.

Esta red utiliza la **función de activación** Rectified Linear Unit (ReLU)

La **función de pérdida** ayudará a calcular la precisión de la segmentación mediante la función de activación Softmax, un valor alto concluye una alta probabilidad de rendimiento. Por su parte, la entropía cruzada valida si el resultado acertó o no. Ambos son valores dinámicos, no hay un valor obtenido cómo tal.

El **optimizador** es un *algoritmo* que ajusta los pesos de la red neuronal durante el entrenamiento, con el objetivo de **minimizar la función de pérdida** (loss function).

Descenso de gradiente estocástico (SGD), deformaciones elásticas aleatorias, capas de dropout, tasa de aprendizaje.

Referencias

Residual networks and skip conectionns https://youtu.be/Q1JCrG1bJ-A?si=MjRLHAvlU1LvP6yB
U-net CNN for biomedical image segmentation https://link.springer.com/chapter/10.1007/978-3-319-24574-4_28

Parámetro	U-Net Estándar	U-Net++	ResUNet
Arquitectura Base	Encoder-Decoder con skip connections	U-Net con conexiones densas entre bloques (skip pathways mejorados)	U-Net + Bloques Residuales (ResNet)
Skip Connections	Conexiones directas (encoder → decoder)	Conexiones densas anidadas entre múltiples niveles	Conexiones residuales (identidad + convoluciones)
Bloques Convolucionales	Conv2D + ReLU	Conv2D + ReLU + BatchNorm	Residual Blocks: Conv2D + BatchNorm + ReLU + Add
Profundidad	4-5 niveles	4-5 niveles con sub-redes anidadas	4-5 niveles con bloques residuales

Función de Pérdida	`binary_crossentropy` o `Dice Loss`	`binary_crossentropy` + Pérdidas auxiliares en cada sub-red	`binary_crossentropy` o `Dice Loss`
Optimizador	Adam (lr=1e-4)	Adam (lr=1e-4)	Adam (lr=1e-4) o SGD con momentum
Tamaño de Kernel	3x3 (convoluciones), 2x2 (upsampling)	Igual que U-Net	3x3 (convoluciones) + 1x1 (proyecciones residuales)
Ventajas	Simple y efectivo para segmentación binaria	Mejor captura de contextos multi-escala (ideal para objetos pequeños)	Entrenamiento más estable (gradientes fluyen mejor gracias a residuos)
Desventajas	Skip connections rígidas pueden perder detalles	Mayor complejidad computacional	Requiere más memoria por bloques residuales
Aplicación Típica	Segmentación de tumores, células	Segmentación de estructuras jerárquicas (ej: vasos sanguíneos)	Datos médicos con profundidad (ej: MRI 3D)
Exactitud Reportada	98.49% (artículo de referencia)	~99.1% (en datasets como BraTS)	~99.3% (con bloques residuales profundos)

Conclusión: La red U-Net estándar es suficiente para problemas binarios simples cómo lo es la segmentación de tumores que, es el objetivo de principal interés por lo que este modelo es el más conveniente de los tres.

✅ **Creación de red Unet para segmentación de imágenes biomédicas (pill segemntation)**

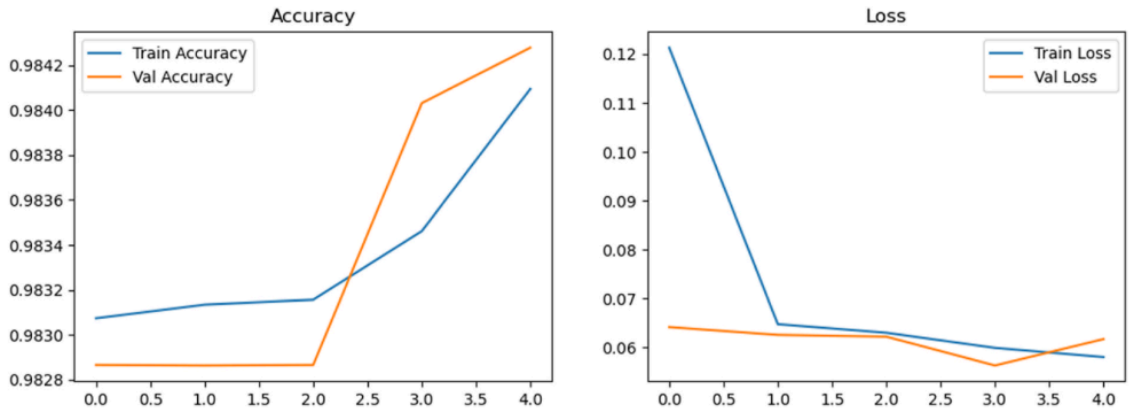
Se decidió replicar los resultados los cuáles se muestran en la Tabla2 del artículo: “Implementation of biomedical segmentation for brain tumor utilizing an adapted U-net model”

Dataset del artículo: https://www.kaggle.com/datasets/ashkhagan/figshare-brain-tumor-dataset

Modelo: Standard U-Net

Epoch	Loss	Accuracy	Val Loss	Val Accuracy
1	0.1213	0.9831	0.0642	0.9829
2	0.0648	0.9831	0.0626	0.9829
3	0.0630	0.9832	0.0622	0.9829
4	0.0599	0.9835	0.0563	0.9840
5	0.0580	0.9840	0.0617	0.9843

Tabla 2



Para replicar el resultado del modelo estándar U-Net, la información proporcionada en las fuentes detalla la arquitectura general y algunos parámetros clave, pero no especifica todos los valores numéricos detallados de las capas (como el número exacto de filtros o el tamaño de los kernels para cada capa convolucional) que se necesitarían para una replicación completa del modelo *estándar* que fue probado.

A continuación, se presentan los parámetros y componentes del modelo estándar U-Net mencionados en las fuentes:

• **Arquitectura General**: El modelo U-Net estándar se describe como una arquitectura de capas de Red Neuronal Convolucional (CNN) que toma la forma de una "U". Se compone de tres partes principales:

- **Codificador (Ruta de Contracción)**:
  - En esta parte, se extraen características y se reducen las dimensiones espaciales.
  - Está compuesto por **capas CNN**, **funciones de activación** y **capas de Pooling**.
  - **Función de Activación**: Utiliza la función **Rectified Linear Unit (ReLU)**.
  - **Pooling**: Se realiza **Max pooling** con un **stride de (2 × 2)**.
  - Las operaciones de convolución se describen como `Conv(X) =W*X + b`, donde W son los pesos, X es la entrada y b es el sesgo.
- **Cuello de Botella (Bottleneck)**:
  - Es la parte más profunda del modelo y contiene un alto nivel de características, sirviendo como puente entre el Codificador y el Decodificador.
  - En esta parte, solo se encuentran **capas CNN sin capas de pooling**.
- **Decodificador (Ruta de Expansión)**:

▪ En esta parte, las características se reconstruyen gradualmente hasta alcanzar el tamaño de la imagen de entrada.

▪ Se utilizan **capas de up-sampling** (capas de convolución traspuesta) y **capas de conexión de salto (skip connections)**.

▪ La operación de up-sampling se describe como `UpSampling(X) =WT*X + b`.

▪ Las conexiones de salto se utilizan para transferir mapas de características directamente de la ruta de contracción a la ruta de expansión, y su operación se describe como `SkipConnection(S,Z)= [S,Z]`.

Parámetros Específicos Mencionados y sus Valores:

• **Función de Activación**: **ReLU** (`Max(0,X)`).

• **Tipo de Pooling**: **Max pooling**.

• **Stride de Pooling**: **(2 × 2)**.

• **Operación de Up-sampling**: **Convoluciones traspuestas**.

El modelo estándar U-Net fue entrenado durante **cinco épocas**

Los datos se dividieron en:

▪ **80% para entrenamiento**

▪ **20% para validación**

Asignación para Azucena, Lupita e Ian:

Es necesario contar con la librería Numpy, cv2, os, opencv así cómo con Tensorflow (únicamente permitido para versiones de python 3.11 y anteriores) adjunto un video para realizar la instalación https://youtu.be/rbsWdaZYahE?si=rOWBa5zLa_2IGafP

Plan (sugerido) de trabajo:

1. Definir arquitectura U-Net

- Crear el **bloque de codificación** (encoder): varias capas convolucionales y *max pooling* que extraen y reducen la representación.

- Crear el **bloque de decodificación** (decoder): capas que hacen *upsampling* (o transposed convolution) y recuperan la resolución original.

- Añadir las **skip connections**: conexiones directas entre capas del encoder y decoder que ayudan a recuperar detalles finos.

- Terminar con una capa de salida adecuada (por ejemplo, `Conv2D` con 1 filtro + activación `sigmoid` para segmentación binaria).

2. Definir y compilar el modelo

   - Elegir la *loss function* (por ejemplo, `binary_crossentropy` para segmentación binaria)

– Elegir el *optimizer* (por ejemplo, `Adam`)

– Definir métricas para monitorizar (por ejemplo, `accuracy`, `dice coefficient`, etc.)

3. **Carga de datos (imágenes y máscaras)**
4. **Dividir los datos en conjuntos de entrenamiento y validación**
5. **Evaluación del modelo**
6. **Visualización de las predicciones**