Instituto Tecnológico y de Estudios Superiores de Monterrey.

Campus Querétaro.

Desarrollo de aplicaciones avanzadas de ciencias computacionales (Gpo 301)

Desarrollo de un modelo de segmentación imágenes urbanas basado en una arquitectura UNET

Presenta:

Jaime López Hernández - A00571842

Fecha de entrega:

05 de junio del 2024

Índice

Resumen

Implementación basada en el paper "*Deep Semantic Segmentation of Angiogenesis Images*"

Dentro de los puntos principales que presenta este paper son los siguientes:

En el contexto de las tareas de segmentación de imágenes, especialmente cuando se trata de segmentación binaria (primer plano/fondo), el uso de la pérdida de entropía cruzada por píxel puede ser una opción más adecuada en comparación con la pérdida de entropía cruzada binaria. La pérdida de entropía cruzada por píxel tiene en cuenta los píxeles individuales de la imagen y penaliza el modelo en función de la exactitud de la predicción de cada píxel, por lo que se adapta mejor a la naturaleza de las tareas de segmentación en las que la clasificación de cada píxel es crucial.

Optimización de la función de pérdida: El documento introduce el uso de Focal Loss (FL) para manejar el desequilibrio de clases en el conjunto de datos.

Transfer Learning, en este caso hicieron uso de una EfficientNet.

Data Augmentation

En el caso del problema a tratar y combinado con las particularidades del conjunto de los datos, se lleva a que la segmentación de la imagen sea una clasificación a nivel de píxel. Por lo tanto, las técnicas de aumento de datos como la rotación, el volteo o el escalado pueden alterar las relaciones espaciales entre píxeles, lo que puede dar lugar a clasificaciones incorrectas de los mismos. Esto es especialmente relevante en las imágenes urbanas, donde la disposición espacial de los objetos y las estructuras es crucial para una segmentación precisa.

Pre-procesado

Antes del entrenamiento del modelo, se aplicó un proceso completo de preprocesamiento para transformar los datos brutos en un formato adecuado para la ingestión y optimización eficientes del modelo. Se llevaron a cabo los siguientes pasos

- **Redimensionamiento de imágenes y máscaras:** Para mantener la coherencia espacial y adaptarse a los requisitos de entrada del modelo, las imágenes originales y sus correspondientes máscaras de segmentación se redimensionaron a una resolución uniforme de 192 x 192 píxeles. Este paso garantizó unas dimensiones espaciales uniformes en todo el conjunto de datos, facilitando un procesamiento coherente y minimizando las posibles distorsiones. La operación de redimensionamiento se realizó utilizando la función tf.image.resize de TensorFlow, que aplica técnicas de interpolación eficientes para preservar la calidad de la imagen.

- **Conversión del espacio de color:** El espacio de color nativo de las imágenes, inicialmente codificado en el formato BGR (Azul-Verde-Rojo) según la convención OpenCV, fue convertido al espacio de color RGB (Rojo-Verde-Azul) utilizando la función cv2.cvtColor con la bandera cv2.COLOR_BGR2RGB. Este paso de conversión alinea la representación de la imagen con el formato estándar esperado por la mayoría de los marcos de aprendizaje profundo, lo que permite una integración perfecta con la arquitectura del modelo elegido.

- **Normalización:** Para mitigar el posible impacto de la variación de las condiciones de iluminación y facilitar una convergencia más estable durante el entrenamiento, se normalizaron los valores de píxel de las imágenes de entrada y sus máscaras asociadas. Este proceso de normalización consistió en dividir las intensidades de los píxeles por 255,0, escalando de hecho los valores a un rango entre 0 y 1.

- **Partición de datos:** Debido a la estructura única del conjunto de datos, en el que la imagen original y su correspondiente máscara de segmentación están concatenadas horizontalmente, se implementó un paso de partición. Cada imagen compuesta se dividió en dos mitades: la parte izquierda, la cual representaba la imagen de entrada original, y la parte derecha contenía la máscara de segmentación. Esta partición garantizaba que el modelo recibiera los datos de entrada y de destino adecuados durante el entrenamiento y la inferencia.

- **División de los datos:** Tras el preprocesamiento, el conjunto de datos se dividió en subconjuntos de entrenamiento y validación. Las primeras 2.900 muestras se asignaron al conjunto de entrenamiento (X_train, y_train), mientras que las muestras restantes se asignaron al conjunto de validación (X_valid, y_valid). Esta división facilitó el entrenamiento del modelo y la evaluación del rendimiento en datos no observados, permitiendo un desarrollo robusto del modelo y el ajuste de hiperparámetros.

Modelo Inicial

Componentes de la Arquitectura:

ConvolutionBlock:

- Contiene dos capas convolucionales con kernels de 3x3 y activación ReLU.
- Mantiene las dimensiones espaciales usando padding.

EncoderLayerBlock:

- Compuesto por un ConvolutionBlock seguido de una capa MaxPooling de 2x2.
- Reduce las dimensiones espaciales por un factor de 2, mientras aumenta la profundidad de las características.

DecoderLayerBlock:

- Consiste en una capa Conv2DTranspose de 2x2 para remuestrear.
- Seguido de un ConvolutionBlock para refinar los mapas de características.

UnetAutoencoder:

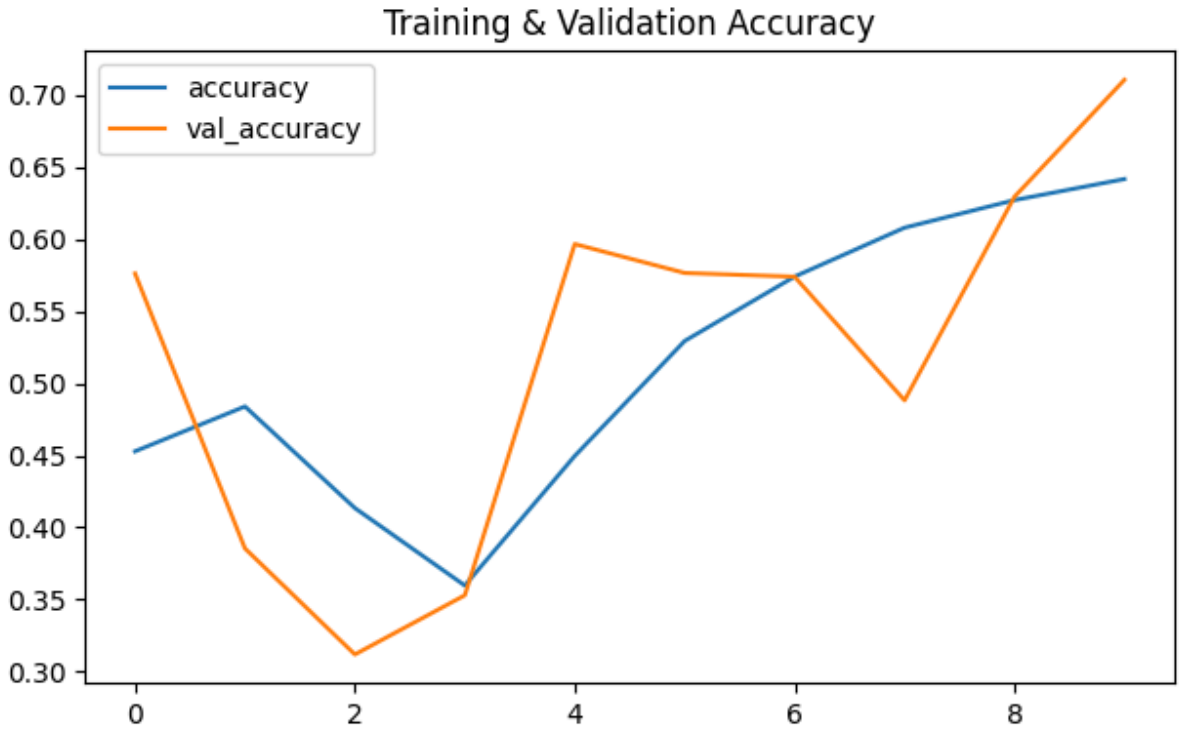
- Ruta del Codificador:
- Cuatro módulos EncoderLayerBlock con canales de salida crecientes: 64, 128, 256 y 512.
- Cuello de botella:
- Un ConvolutionBlock con 1024 canales de salida.
- Ruta del Decodificador:
- Cuatro módulos DecoderLayerBlock con canales de salida decrecientes: 512, 256, 128 y 64.

Capa de Salida:

- Una capa convolucional de 1x1 con 3 canales de salida (para una segmentación de 3 clases) y activación sigmoide para producir el mapa de segmentación final.

Configuración de Entrenamiento:

- Función de Pérdida: Binary Crossentropy.
- Optimizador: Adam.
- Métricas: Accuracy.
- Callback: ShowProgress para visualizar las máscaras predichas y verdaderas después de cada época.



Pérdida de entrenamiento

La pérdida de entrenamiento representa el error promedio del modelo en el conjunto de entrenamiento. En este caso, la pérdida de entrenamiento comienza en un valor alto y disminuye

gradualmente a medida que el modelo se entrena. Esto indica que el modelo está aprendiendo a realizar la tarea de segmentación cada vez mejor.

Pérdida de validación

La pérdida de validación representa el error promedio del modelo en un conjunto de datos independiente que no se utilizó para entrenar el modelo. En este caso, la pérdida de validación también disminuye a medida que el modelo se entrena, pero a un ritmo más lento que la pérdida de entrenamiento. Esto indica que el modelo está aprendiendo a generalizar bien a datos nuevos y no vistos.

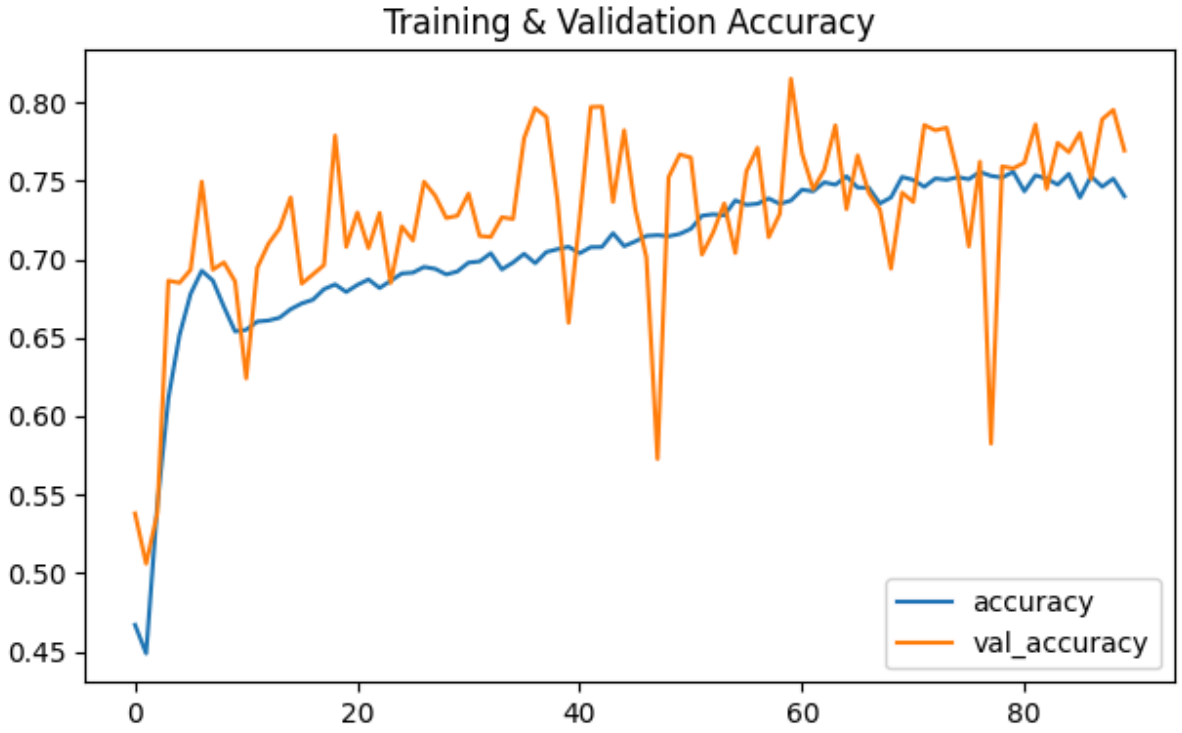
(Test Loss): Un valor de 0.5688 indica que el modelo aún tiene margen para mejorar en la predicción precisa de las etiquetas de las imágenes.

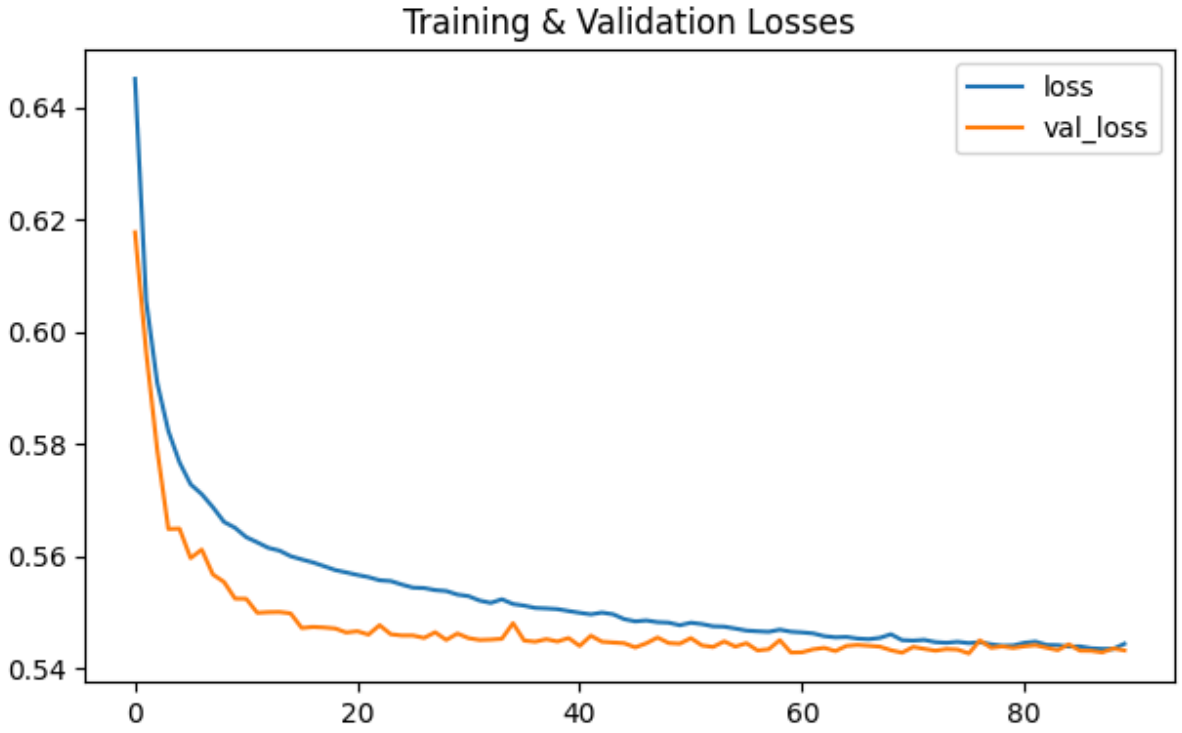
(Test Accuracy): Una precisión de 0.7107 sugiere que el modelo acierta en la clasificación de aproximadamente el 71% de las imágenes.

IoU (Mean Intersection over Union): Un IoU promedio de 0.9297 indica que el modelo predice con alta precisión la superposición entre las áreas segmentadas y las áreas reales. Un IoU cercano a 1 indica una coincidencia casi perfecta entre las segmentaciones predichas y reales.

Coeficiente de Dice: Para esta métrica se obtuvo un coeficiente de Dice de 0.4273. El coeficiente de Dice se enfoca en la evaluación de la coincidencia de volumen entre las segmentaciones, y un valor cercano a 1 indica una alta coincidencia de volumen.

Modelo Mejorado





Perdida entre Entrenamiento y Validación

El gráfico indica que el modelo Unet encoder mejorado tiene un buen rendimiento tanto en los datos de entrenamiento como en los datos de validación. El modelo converge rápidamente y encuentra un buen equilibrio entre el ajuste a los datos de entrenamiento y la generalización a los datos de validación.

Precisión Validación y Entrenamiento

Rendimiento del modelo:

La precisión de entrenamiento muestra una mejora constante a lo largo de las épocas, lo que indica que el modelo está aprendiendo de los datos de entrenamiento.

La precisión de validación fluctúa más que la precisión de entrenamiento, pero también muestra una tendencia general al alza, lo que sugiere que el modelo generaliza razonablemente bien los datos de validación.

Indicadores de overfit:

La precisión de validación presenta fluctuaciones significativas, mientras que la precisión de entrenamiento es más estable. Esta discrepancia sugiere un posible sobreajuste, en el que el modelo funciona bien en los datos de entrenamiento, pero tiene dificultades para generalizar de forma consistente en los datos de validación.

Intersección media sobre la Unión (IoU): 0.9297

La métrica de intersección media sobre la unión (IoU) mide la superposición entre las máscaras de segmentación previstas y las máscaras reales. Una puntuación IoU de 0.9297 indica un alto nivel de concordancia entre las máscaras de segmentación predichas y las reales, con una parte significativa de la máscara predicha que se solapa con la máscara real.

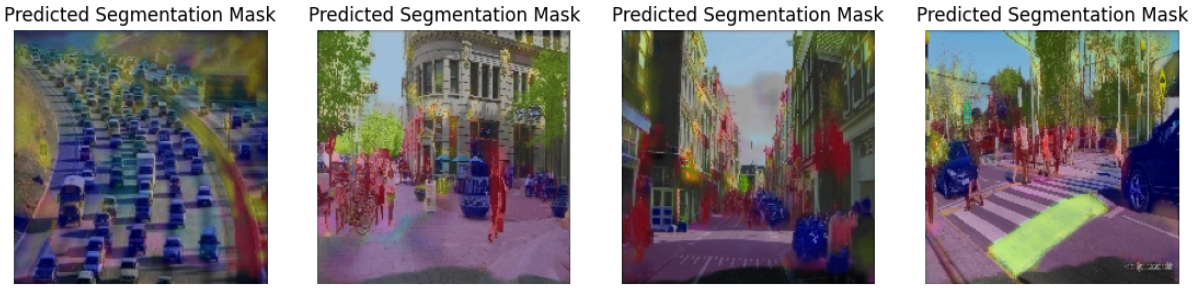
Coeficiente Dice: 0.4587

El coeficiente Dice es otra métrica utilizada habitualmente para evaluar la similitud entre dos conjuntos. Un coeficiente Dice de 0.4587 indica que alrededor del 45.87% de los píxeles de la máscara predicha coinciden con los píxeles de la máscara real. Aunque este valor es inferior en comparación con el IoU, sigue indicando un nivel moderado de concordancia entre las máscaras predicha y real.

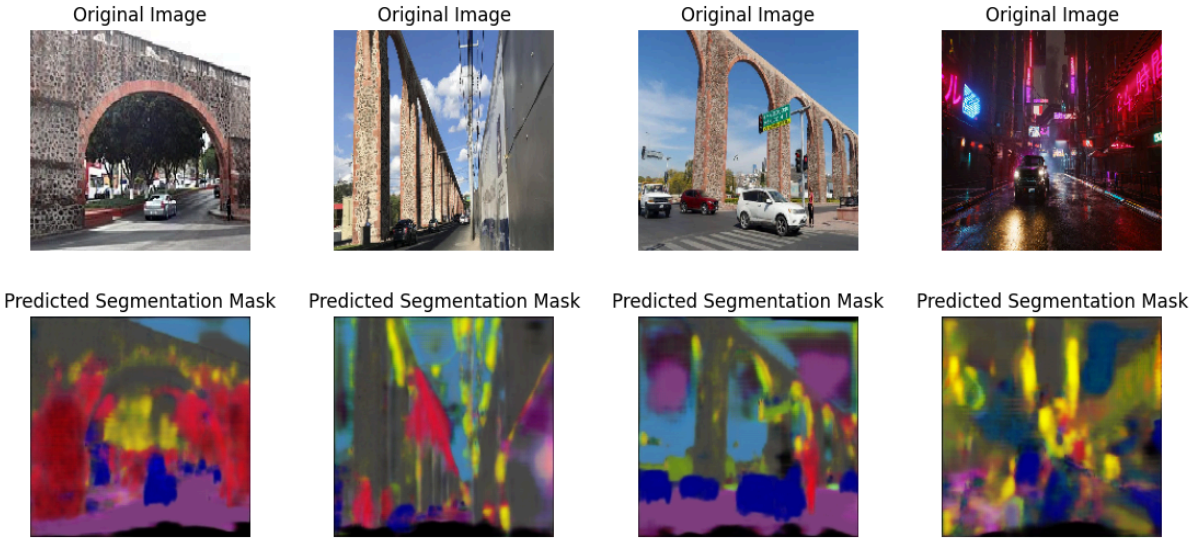
Utilizando el Modelo

Utilizando la versión base del unet autoencoder:

Utilizando la versión mejorada del unet autoencoder



Utilizando la versión mejorada del unet autoencoder con las máscaras separadas



Resultados finales

Principales Cambios con Respecto al Modelo Base

Optimizador: He utilizado el optimizador Adam con una tasa de aprendizaje de 0.001. Esto ayuda a mejorar el rendimiento del modelo mediante la adaptación dinámica del aprendizaje.

Número de épocas: He aumentado el número de épocas a 90, lo que permite que el modelo se entrene durante un período más largo y consiga potencialmente un mejor rendimiento.

Técnicas de regularización: He utilizado dropout para introducir aleatoriedad durante el entrenamiento y evitar el sobreajuste. Esta técnica evita que el modelo memorice los datos de entrenamiento y generalice mal a nuevos datos.

Inicializador del núcleo: He utilizado el inicializador de kernel 'he_normal' para inicializar los pesos de las capas convolucionales. Esto ayuda a evitar que el modelo se sobreajuste