Humanidades, Tecnología y Ciencia, del Instituto Politécnico Nacional
Certificado de Reserva de Derechos al Uso Exclusivo del Título: 04-2010-0326124-14000-203 ISSN: 2007-1957

MODELO BAYESIANO PARA LA ESTIMACIÓN DEL GRADO DE GLEASON EN IMÁGENES DE CÁNCER DE PRÓSTATA USANDO PYTHON

Israel Rivera Zárate
Instituto Politécnico Nacional-CIDETEC
irivera@ipn.mx

Miguel Hernández Bolaños
Instituto Politécnico Nacional-CIDETEC
mbolanos@ipn.mx

Patricia Pérez Romero
Instituto Politécnico Nacional-CIDETEC
promerop@ipn.mx

Abstract

Diagnosis of prostate cancer begins with a blood test called the prostate-specific antigen test and, in many cases, a digital rectal examination to detect abnormalities. If there is suspicion, a biopsy is performed, which involves removing a sample of prostate tissue for microscopic analysis to confirm the presence of cancer cells. The Gleason score is a 2 to 10 scoring system for classifying the aggressiveness of prostate cancer, with a higher score indicating a more aggressive cancer prone to metastasis. This article proposes a Bayesian Variational Autoencoder model for automatic Gleason grade group estimation of prostate cancer lesions from magnetic resonance imaging. It uses the PROSTATEx database not a database of patients or general clinical information, but rather a prostate magnetic resonance imaging (MRI) dataset developed for machine learning challenges and prostate cancer research. This retrospective resource includes multiparametric images (T2-weighted, diffusion-weighted, and dynamic contrast-enhanced) from 346 subjects and can be downloaded from the website https://doi.org/10.7937/K9TCIA.2017.MURS5CL.

Palabras clave: Prostate cancer, Bayesian model, Gleason Grade, Variational Autoencoder.

El cáncer de próstata es un tumor maligno originado dentro del tejido prostático, con un pronóstico heterogéneo. Como lo señala Rebecca L. Siegel (2016), es una creciente preocupación mundial en salud pública, ya que hay un aumento en la población masculina de

la tercera edad. A nivel mundial, de acuerdo con el Observatorio Global de Cáncer, en 2020, el cáncer de próstata ocupó el segundo lugar en incidencia con 7.3% de casos nuevos con una tasa de mortalidad del 3.8% representando una tasa de 7.7 por cada 100,000 habitantes. De acuerdo con la Agencia Internacional de Investigación en Cáncer, para el año 2020 en México, el cáncer de próstata fue el que presentó una mayor incidencia en varones en el mismo año, el porcentaje de mortalidad fue del 8.3%. El Cáncer de Próstata es una patología importante a nivel nacional y mundial, la detección y el tratamiento oportuno disminuye el riesgo de mortalidad asociada a dicha enfermedad, es por eso que debe ser un objetivo de salud enfocarse en tratar de mejorar la atención en los derechohabientes de los sistemas de salud mediante un protocolo bien estructurado, con la finalidad de detectar este padecimiento en etapas tempranas, evitando el sobrediagnóstico y sobretratamiento de la enfermedad y logrando una atención de calidad para esta enfermedad.

I. CLASIFICACIÓN DE GLEASON

El sistema de clasificación de Gleason es ampliamente aceptado como parte de un protocolo estándar para determinar la gravedad del cáncer y está relacionado con el patrón de crecimiento de las glándulas tumorales, tal como lo señala Donald Gleason (1992). El sistema consta de tres grados, del 3 al 5, cada uno de los cuales incluye grupos de patrones glandulares. En la práctica clínica, los patólogos analizan visualmente las biopsias de próstata teñidas para detectar patrones cancerosos. Evaluar manualmente cada muestra es una tarea muy laboriosa y subjetiva. Por esta razón, en los últimos años, se ha popularizado el uso de herramientas de diagnóstico asistido por computadora. Existen tres enfoques principales para el procesamiento de imágenes de histología prostática: predicciones a nivel de imagen, segmentación a nivel de píxel o análisis a nivel de glándula. El procesamiento a nivel de imagen proporciona un patrón canceroso general para una región, sin una delimitación precisa de las estructuras en el tejido, mientras que el análisis a nivel de glándula se limita a patrones cancerosos con estructuras glandulares (es decir, patrones de Gleason 3 o 4). Ver figura 1.

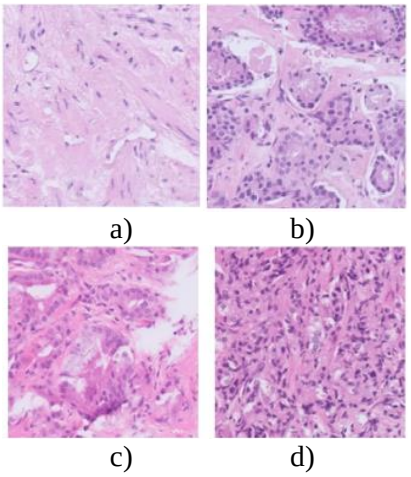

Figura 1. Ejemplos de clasificación: a) No cáncer, b) Gleason 3, c) Gleason 4 y d) Gleason 5. Elaboración propia.

Base de datos

Para el desarrollo y la validación del modelo, se emplearon dos conjuntos de datos principales de resonancia magnética (mpMRI) ProstateX, obtenida en: https://doi.org/10.7937/K9TCIA.2017.MURS5CL por Litiens, G. et al. (2017). Los datos consistieron en un total de 238 mpMRI (una por paciente) que incluían las siguientes secuencias: ponderada en T2 (T2), ponderada

en difusión (DW) con valores b de 50, 400 y 800 s/mm², mapa del coeficiente de difusión aparente (ADC) (calculado a partir de los valores b) y Ktrans (calculado a partir de series ponderadas con contraste dinámico -DCET1). Para cada uno de estos pacientes, se proporcionan de una a cuatro (1,62 por paciente en promedio) ubicaciones de lesiones (es decir, un punto que marca su posición) y su Grado de Gleason (GG). Las ubicaciones de las lesiones fueron informadas por o bajo la supervisión de un radiólogo experto con más de 20 años de experiencia en RM de próstata y confirmadas mediante biopsia guiada por RM. Además, se incluyen 140 mpMRI adicionales como parte del conjunto de pruebas, que incluyen toda la información previa, excepto el GG de las lesiones. Todas las mpMRI se obtuvieron con dos escáneres Siemens de 3 Tesla diferentes.

Modelo Bayesiano

De acuerdo con Xu W. et al. (2019), existe una versión diferente del autoencoder convencional conocida como autoencoder variacional (VAE), el cual va a permitir no solo reconstruir la entrada original, sino además generar nueva información semejante a los datos de entrada, pero no estrictamente reconstrucciones. Esto se logra incorporando una etapa de muestreo aleatorio sobre el espacio latente. En vez de aprender directamente de las variables latentes se podrá tener una media μ y una desviación estándar σ de la distribución de probabilidad de la variable latente. Ver figura 2.

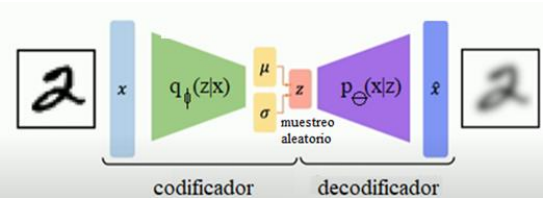

Figura 2. Autoencoder Variacional. Elaboración propia.

El VAE es una red neuronal incompleta ya que varios de sus parámetros son en realidad distribuciones gaussianas. Resulta conveniente abordar la inferencia variacional bayesiana, ya que se observa en el modelo que se cuenta con la distribución a prior de cada clase, $p(z) = (0, I)$, la red neuronal en el codificador, que actúa como el posterior variacional $q_\Phi(z|x) = \mathcal{N}(\mu, I\sigma^2)$. Así como, el decodificador, que actúa como la verosimilitud $p_\theta(x|z) = \mathcal{N}(\hat{\mu}, I\hat{\sigma}^2)$. Dados el posterior y la verosimilitud se puede definir la distribución conjunta como se indica en la ecuación (1).

$$q_\Phi(z, x) = p_\theta(x|z) * p(x) \ldots (1)$$

Los *autoencoders* se entrenan minimizando el error entre la entrada y la salida, que es equivalente a maximizar la verosimilitud gaussiana y añadiendo además un regularizador L2 en θ y Φ e incorporando un prior gaussiano.

Regularización

Como señala Charte D. et al. (2019), la regularidad del espacio latente depende de los datos en el espacio inicial, la dimensión del espacio latente y la arquitectura del codificador. Si un autoencoder permite codificar y decodificar sin pérdida de información conduce a sobreajuste, por lo

tanto, para lograr un espacio latente suficientemente regular se debe llevar un proceso de regularización explícita. Una vez que la red está lista para ser entrenada a partir de una distribución de datos, se debe asegurar que la distribución no esté muy lejos de la distribución normal estándar. Por lo tanto, se requiere de una etapa de regularización para forzar que la distribución de aprendizaje permanezca cerca de la distribución normal. Esto se lleva a cabo mediante el cálculo de la divergencia de Kullback-Leibler (D_{KL}). Sumando esto a la función de pérdida, esta etapa actuará como una función de regularización que mide qué tan lejos se encuentra la distribución aprendida respecto de la distribución normal y la acerca a la media de 0 y a la desviación estándar de 1. Ver ecuación (2).

$$L(\theta,\Phi) = E_{z\sim q_\Phi(z|x)}[p_\theta(x|z)] - D_{KL}[q_\Phi(z|x)||p_\theta(z)] \ldots (2)$$

Entrenamiento

En el extremo de salida se tendrá una función de pérdida de reconstrucción que medirá que tan lejos está la salida del resultado original. Una vez obtenida la pérdida se puede por retropropagación calcular los gradientes. Sin embargo, al tener en el modelo una capa de muestreo estocástica que recopila datos de la distribución de entrada, no es posible obtener su gradiente. Por su parte Charte D. et al. (2019), han propuesto que en vez de tomar muestras directamente de la distribución se utilice más bien una expresión para generar el vector latente, ver ecuación (3).

$$z = \mu + \sigma \odot \varepsilon \ldots (3)$$

Donde μ y σ son parámetros aprendidos durante la retropropagación y $\varepsilon \sim \mathcal{N}(0, I)$ corresponde con una distribución normal estándar que tiene una media de 0 y una desviación estándar de 1, es un espacio fijo donde se muestrea aleatoriamente y no se aprende durante la retropropagación. El modelo de variable latente se apoya de la inferencia variacional para obtener la distribución a posterior de la variable latente z. Cabe señalar que el VAE se entrena maximizando el límite inferior de la evidencia (ELBO) que en este caso corresponde con la función de pérdida establecida en la ecuación (4). Al maximizar el ELBO se maximiza el logaritmo de la verosimilitud cuando se muestrea el posterior aproximado (pérdidas por reconstrucción). Adicionalmente, se minimiza la divergencia entre la distribución del posterior aproximado y la distribución del prior (regularización del posterior). El ELBO se optimiza típicamente por el gradiente respecto a θ y Φ. Ver ecuaciones (4) y (5).

$$\theta_{t+1} = \theta_t + \eta \nabla_\theta L(\theta_t, \Phi_t) \ldots (4)$$
$$\Phi_{t+1} = \Phi_t + \eta \nabla_\Phi L(\theta_t, \Phi_t) \ldots (5)$$

Donde se tiene que:

$$\nabla_\theta L(\theta,\Phi) = E_{z\sim q_\Phi(z|x)}[\nabla_\theta \, log p_\theta(x|z)] \ldots (6)$$

$$\nabla_\Phi L(\theta,\Phi) = \nabla_\Phi E_{z\sim q_\Phi(z|x)}[f(z)] \ldots (7)$$

Finalmente resulta:

$$L(\theta,\Phi) = E_{z\sim q_\Phi(z|x)}[\log p_\theta(x|z) - D_{KL}[q_\Phi(z|x)||p(z)] \ldots (8)$$

Donde la Divergencia KL tiene una solución analítica y K es la dimensionalidad de la variable latente:

$$D_{KL}[q_{\Phi}(z|x)||p(z)] = \frac{1}{2}\sum_{j=1}^{K}(\mu_j^2 + \sigma_j^2 - \log\sigma_j^2 - 1) \ldots (9)$$

II. DESARROLLO

La implementación del VAE se desarrolló en Python (https://www.python.org, versión 3.6.1) con la ayuda de las librerías gratuitas PyTorch (https://pytorch.org, versión 2.0) y Tensorflow (https://www.tensorflow.org, versión 1.2.1), ampliamente aceptadas por la comunidad. El entorno de desarrollo empleado ha sido el Notebook de Jupyter (http://jupyter.org, versión 5.0.0).

Problemática por resolver:

En este trabajo, la clasificación del patrón de Gleason de imágenes de próstata se aborda mediante la comparación de diferentes modelos de redes neuronales convolucionales. Estos se basan en arquitecturas reconocidas para la segmentación de imágenes: redes totalmente convolucionales, SegNet y U-Net. Se contrastan los resultados con el modelo Bayesiano que se propone en el presente trabajo. Los modelos propuestos en este trabajo comparten la misma configuración de salida: una capa convolucional con numerosos filtros como clases a predecir. Concretamente, las etiquetas definidas son: tejido no canceroso (NC), patrón de Gleason 3 (GP3), patrón de Gleason 4 (GP4) y patrón de Gleason 5 (GP5). Posteriormente, se utiliza una activación soft-max a nivel de píxel para obtener los mapas de probabilidad. Durante la etapa de inferencia, se obtiene un mapa predicho para cada imagen, asignando la clase con mayor probabilidad a cada salida.

El presente trabajo propone un sistema de detección y clasificación del Grado de Gleason con base en un Autoencoder Variacional que recibe una imagen radiológica de próstata como entrada. El sistema cuenta con una etapa codificadora pre entrenada basada en la red VGG16 utilizada para la extracción de características. Posteriormente, con una etapa de reparametrización para el muestreo de los vectores del espacio latente. Adicionalmente, una etapa decodificadora para la reconstrucción de la imagen de entrada. Finalmente, una etapa de clasificación del Grado de Gleason. Ver figura 3.

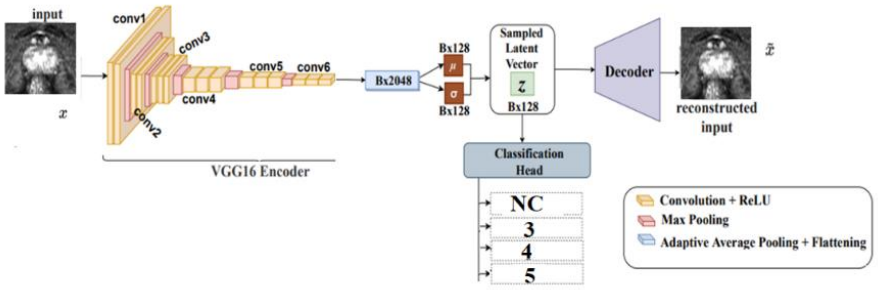

Figura 3. Sistema Autoencoder Variacional propuesto para la detección y clasificación del Grado de Gleason. Adaptado de Addo D. et al. (2022).

Humanidades, Tecnología y Ciencia, del Instituto Politécnico Nacional
Certificado de Reserva de Derechos al Uso Exclusivo del Título: 04-2010-0326124-14000-203 ISSN: 2007-1957

Metodología:

La metodología adoptada en el presente trabajo consiste en los siguientes pasos:

[Paso 1:] Creación del modelo.

[Paso 2:] Aprendizaje de la red.

[Paso 3:] Clasificación.

PASO 1. Creación del Modelo

VGG16 es un modelo de red convolucional que alcanza un 92.7% de precisión habiéndose entrenado con la popular base de datos ImageNet, la cual cuenta con alrededor de 14 millones de imágenes pertenecientes a unas mil clases de objetos y a unas 22 mil categorías. Ver figura 4.

```
# --- Definir el Codificador (Encoder) ---
encoder = Sequential([
    layers.Input(shape=(IMG_HEIGHT, IMG_WIDTH, 3)),
    layers.Conv2D(32, (3, 3), activation='relu', padding='same', strides=2),
    layers.Dropout(0.2),
    layers.Conv2D(64, (3, 3), activation='relu', padding='same', strides=2),
    layers.Dropout(0.2),
    layers.Conv2D(128, (3, 3), activation='relu', padding='same', strides=2),
    layers.Dropout(0.2),
    layers.Flatten(),
    layers.Dense(512, activation='relu'),
    layers.Dropout(0.3),
    layers.Dense(256, activation='relu'),
    layers.Dropout(0.3)
])
# --- Definir el Decodificador (Decoder) ---
decoder = Sequential([
    layers.Dense(25*25*128, activation='relu', input_shape=(256,)),
    layers.Reshape((25, 25, 128)),
    layers.Conv2DTranspose(128, (3, 3), activation='relu', padding='same', strides=2),
    layers.Conv2DTranspose(64, (3, 3), activation='relu', padding='same', strides=2),
    layers.Conv2DTranspose(32, (3, 3), activation='relu', padding='same', strides=2),
    layers.Conv2D(3, (3, 3), activation='sigmoid', padding='same')
])
```

Figura 4. Creación del Modelo. Elaboración propia.

PASO 2. Aprendizaje de la Red

Se congelan todas las capas excepto la última para reemplazar la capa clasificadora. Se establece la función pérdida y se establecen en las últimas capas los parámetros z_mean y z_log_var. Estos son los parámetros de la distribución Gaussiana (media y logaritmo de la varianza) a partir del cual se muestrea las variables latentes. Con estos parámetros se genera épsilon (ruido aleatorio de la distribución normal estándar) y se aplica el truco de reparametrización. Se aplica la optimización por Kullback Leibler. Ver figura 5.

```
# --- 1. Cargar el extractor de características (VGG16) ---
# include_top=False: No incluye las capas densas finales del clasificador de ImageNet
# weights='imagenet': Carga los pesos pre-entrenados en ImageNet
vgg16_base = keras.applications.VGG16(
    input_shape=(IMG_HEIGHT, IMG_WIDTH, 3),
    include_top=False,
    weights='imagenet'
)

# --- 2. Congelar la base de VGG16 ---
# Esto evita que los pesos pre-entrenados se modifiquen durante el entrenamiento
vgg16_base.trainable = False
print("Capas de VGG16 congeladas. No se entrenarán.")
vgg16_base.summary()

# --- 3. Construir el clasificador bayesiano (Monte Carlo Dropout) ---
# Usamos el modelo Sequential para construir las capas clasificadoras
classifier_head = Sequential([
    layers.Flatten(),
    layers.Dense(256, activation='relu'),
    layers.Dropout(0.5), # Capa de Dropout para aproximar la red bayesiana
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.3), # Segunda capa de Dropout
    layers.Dense(num_classes, activation='softmax')
])

# --- 4. Ensamblar el modelo completo ---
model = Sequential([
    vgg16_base,
    classifier_head
])

# --- 5. Compilar el modelo ---
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
              metrics=['accuracy'])

print("\nResumen del Modelo Completo (VGG16 + Clasificador):")
```

Figura 5. Aprendizaje de la Red. Elaboración propia.

PASO 3. Clasificación

Se combina el encoder con la capa sampling y el clasificador. Después de entrenar el modelo, se pasan las imágenes de prueba a través del encoder para obtener los logits de clasificación. Ver figura 6.

Humanidades, Tecnología y Ciencia, del Instituto Politécnico Nacional
Certificado de Reserva de Derechos al Uso Exclusivo del Título: 04-2010-0326124-14000-203 ISSN: 2007-1957

```python
# --- 4. Proponer un modelo VAE con Clasificador y Monte Carlo Dropout ---

# La reparameterización es clave para que el VAE pueda entrenarse con retropropagación
class Sampling(layers.Layer):
    def call(self, inputs):
        z_mean, z_log_var = inputs
        batch = tf.shape(z_mean)[0]
        dim = tf.shape(z_mean)[1]
        epsilon = tf.keras.backend.random_normal(shape=(batch, dim))
        return z_mean + tf.exp(0.5 * z_log_var) * epsilon

# Construir el codificador (encoder)
latent_dim = 128
encoder_inputs = keras.Input(shape=(IMG_HEIGHT, IMG_WIDTH, 3))
x = layers.Conv2D(32, 3, activation="relu", strides=2, padding="same")(encoder_inputs)
x = layers.Conv2D(64, 3, activation="relu", strides=2, padding="same")(x)
x = layers.Conv2D(128, 3, activation="relu", strides=2, padding="same")(x)
x = layers.Flatten()(x)
z_mean = layers.Dense(latent_dim, name="z_mean")(x)
z_log_var = layers.Dense(latent_dim, name="z_log_var")(x)
z = Sampling()([z_mean, z_log_var])

# Conectar el codificador con el clasificador
classifier_outputs = layers.Dense(256, activation='relu')(z)
classifier_outputs = layers.Dropout(0.5)(classifier_outputs) # Monte Carlo Dropout
classifier_outputs = layers.Dense(128, activation='relu')(classifier_outputs)
classifier_outputs = layers.Dropout(0.3)(classifier_outputs)
classifier_outputs = layers.Dense(num_classes, activation='softmax')(classifier_outputs)

# Definir el modelo completo
vae_classifier = Model(inputs=encoder_inputs, outputs=classifier_outputs, name="vae_classifier")
vae_classifier.summary()

# --- 5. Compilar el modelo ---
vae_classifier.compile(optimizer='adam',
                       loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
                       metrics=['accuracy'])
```

Figura 6. Etapa de Clasificación. Elaboración propia.

III. PRUEBAS Y RESULTADOS

Arquitectura FCN8 (Redes totalmente convolucionales)

Las redes totalmente convolucionales se propusieron como una extensión de las arquitecturas de clasificación clásicas para tareas de segmentación semántica. Las redes neuronales convolucionales para la clasificación de imágenes se componen de una etapa de extracción de características (modelo base) mediante filtros convolucionales apilados y reducción de la dimensión espacial mediante operaciones de agrupamiento máximo, y una fase de clasificación mediante capas totalmente conectadas (modelo superior).

Arquitectura Segnet

La arquitectura SegNet como lo indica Vijay B. et al. (2017), para la segmentación semántica se basa en redes totalmente convolucionales. Tras el modelo base, una rama del decodificador recupera la información espacial mediante bloques convolucionales apilados y operaciones de sobremuestreo. El sobremuestreo se basa en los índices utilizados en el modelo base durante las operaciones de agrupamiento máximo para realizar una reconstrucción no lineal de las dimensiones originales.

Arquitectura U-Net

La arquitectura U-Net es un modelo de segmentación propuesto para aplicaciones médicas en Olaf R. et al. (2015). Su configuración se basa en dos ramas: un codificador encargado de extraer las características relevantes de la imagen y un decodificador que controla la reconstrucción de los mapas de segmentación de probabilidad. La rama del codificador consta de bloques convolucionales apilados con reducción dimensional mediante un operador de agrupación máxima. Cada bloque convolucional duplica el número de filtros, mientras que el operador de agrupación redimensiona la imagen a la mitad. En particular, se utilizan cuatro bloques convolucionales, aumentando el número de filtros de 64 a 1024 y las dimensiones espaciales de 2562 a 142 píxeles. En la rama del decodificador, los bloques convolucionales son seguidos por deconvoluciones que duplican la dimensión de las imágenes y reducen el número de filtros a la mitad. Además, el codificador se conecta al decodificador mediante la concatenación de los mapas de activación de los niveles correspondientes después del filtro deconvolucional.

En el empleo de la base de datos ProstateX con extracción de parches se trabajó con un total de Parches extraídos: 238. Imágenes originales procesadas: 238. Distribución de

clases en parches: Entrenamiento: 142 parches, Validación: 48 parches y Prueba: 48 parches. A_D (Clase 1): 72 parches (30.3%), R (Clase 2): 86 parches (36.1%), G3 (Clase 3): 42 parches (17.6%), G4 (Clase 4): 22 parches (9.2%), G5 (Clase 5): 16 parches (6.7%)

Función de pérdida

La función de pérdida utilizada durante el proceso de entrenamiento es la función Dice, introducida en Fausto M. et al. (2016), para la segmentación volumétrica de imágenes. Esta función establece un equilibrio entre la intersección y la unión de las máscaras predichas y de referencia, siendo apropiada para conjuntos de datos desequilibrados. La función Dice se define de la siguiente manera:

$$Dice = \frac{2\sum_i^N p_i g_i}{\sum_i^N p_i^2 \sum_i^N g_i^2} \quad \ldots (10)$$

Donde pi y gi denotan las etiquetas predichas codificadas en caliente y la verdad fundamental, respectivamente, para un lote de imágenes de la clase i. Nótese que i denota una de las N clases: tejido no canceroso (NC), patrón de Gleason 3 (GP3), patrón de Gleason 4 (GP4) y patrón de Gleason 5 (GP5).

Como factor de mérito, se obtuvo el índice Dice (DI = 1 − Dice) en los mapas de segmentación predichos. Se entrenaron cuatro tipos de modelos de redes neuronales convolucionales para la segmentación semántica de patrones de Gleason. En particular, se utilizó la Red Totalmente Convolucional (FC8) con pesos VGG16 preentrenados como modelo base y stride of 8, Segnet y arquitectura UNet con su configuración estándar y el VAE Bayesiano basado en la VGG16. Los hiperparámetros se optimizaron empíricamente en la cohorte de validación. Los resultados obtenidos en el subconjunto de validación se presentan en la Tabla 1.

Tabla 1: Resultados en el subconjunto de validación para los diferentes modelos. NC: no canceroso, GP: patrón de Gleason.

Método	DI_{NC}	DI_{GP3}	DI_{GP4}	DI_{GP5}
$FCN8$	0.087	0.014	0.660	0.008
$SegNet$	0.322	0.706	0.778	0.520
U-Net	0.460	0.468	0.723	0.265
VAEbayes	**0.465**	**0.795**	**0.826**	**0.535**

El modelo FCN8 se entrenó utilizando un optimizador SGD con un momento de Nesterov de 0.9, una tasa de aprendizaje de 0.01 y una tasa de decaimiento de 0.0016. En el modelo U-Net, la tasa de aprendizaje se fijó en 0.00001, y se utilizó Adam como optimizador. Estos modelos se entrenaron durante 200 épocas en una estrategia de minilotes de 16 imágenes. En cuanto a los modelos Segnet y U-Net, también se utilizó el optimizador Adam, pero la tasa de aprendizaje aumentó a 0.01. Estos modelos se entrenaron durante 100 épocas con un tamaño de lote de 8 imágenes.

Las cifras de mérito obtenidas y algunos ejemplos representativos de la segmentación semántica se presentan en la figura 7.

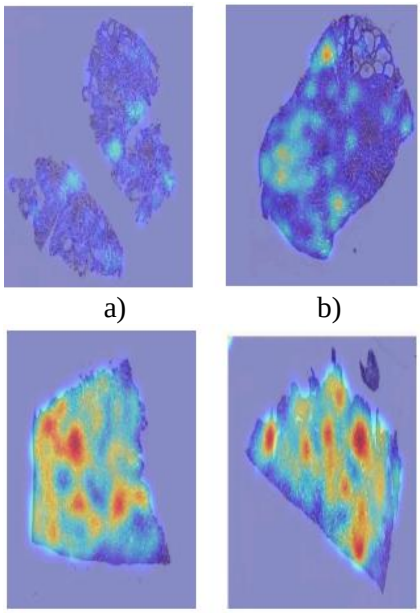

Figura 7. Ejemplos de clasificación: a) No cáncer, b) Gleason 3, c) Gleason 4 y d) Gleason 5. Las áreas en azul representan una probabilidad nula o baja de un grado de Gleason más alto, mientras que las áreas en rojo indican una mayor probabilidad de un grado de Gleason más alto.

En cuanto a los resultados obtenidos en la cohorte de validación, el modelo VAE Bayesiano mostró el mejor rendimiento. El modelo con peor rendimiento fue el FCN8, que solo reconoció correctamente el tejido con patrón de Gleason 4. Se obtuvieron mejores resultados con el modelo Segnet que con la arquitectura básica de U-Net, con un índice Dice promedio para las clases relacionadas con el tejido prostático (es decir, NC, GP3, GP4 y GP5) de 0.5815 y 0.4790, respectivamente. El modelo U-Net, alcanzó un Dice promedio para estas clases de 0.6538. El modelo con mejor rendimiento, VAE Bayesiano, se entrenó en todo el conjunto de entrenamiento y validación, y el modelo resultante se evaluó en la cohorte de prueba. Ver tabla 2.

Tabla 2: Resultados en el conjunto de prueba para los diferentes modelos. NC: no canceroso, GP: patrón de Gleason.

Método	DI_{NC}	DI_{GP3}	DI_{GP4}	DI_{GP5}
VAEbayes	0.840	0.420	0.545	0.195

Los resultados obtenidos en el subconjunto de prueba muestran una ligera disminución en el rendimiento del modelo. El valor promedio de Dice en las clases de tejido cae a 0.4977. Esto podría deberse a la heterogeneidad interna conocida en los grados de Gleason y el desafío de obtener subconjuntos homogéneos en la base de datos durante la etapa de partición. Además, el índice Dice es una métrica rigurosa que no tiene en cuenta que la mayoría de los errores ocurren entre clases adyacentes. Ver figura 8.

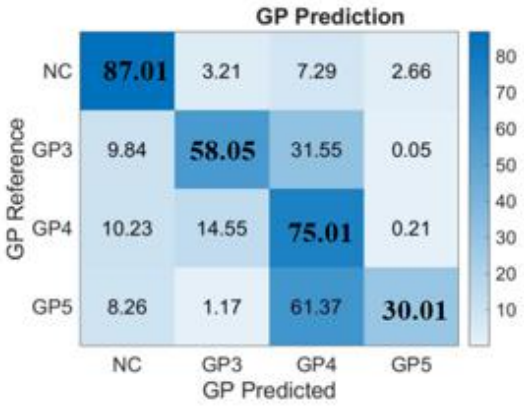

Figura 8. Matriz de confusión de la predicción del grado de Gleason a nivel de píxel en la cohorte de prueba con el modelo VAE Bayesiano propuesto. NC: no canceroso, GP3: patrón de Gleason 3, GP4: patrón de Gleason 4, GP5: patrón de Gleason 5.

IV. CONCLUSIONES

El objetivo de este estudio fue investigar la posibilidad de que una estimación altamente precisa de la GS a partir de imágenes de resonancia magnética mediante un Autoencoder Bayesiano, que utiliza segmentación semántica para aislar las imágenes y limitar el rango de evaluación a la próstata, permitiera la determinación del grupo de grado sin biopsia prostática invasiva, reduciendo así la carga física del paciente. Con base en esta hipótesis, este estudio se propuso construir un sistema para estimar la GS de forma no invasiva a partir de imágenes diagnósticas de resonancia magnética. Utilizando funciones cuantitativas de resonancia magnética, un sistema de estimación de la GS basado en IA no solo puede automatizar y facilitar el flujo de trabajo de los radiólogos, sino también reducir la carga de trabajo del personal médico.

En esta investigación, se propuso una arquitectura VAE Bayesiano capaz de realizar la clasificación de los patrones cancerosos en imágenes de próstata según el sistema de gradación de Gleason. Este rendimiento se encuentra al nivel de trabajos previos para la gradación de patrones de Gleason a nivel de imagen, pero nuestro modelo ofrece una delimitación más precisa de los patrones cancerosos en el tejido.

En este trabajo, exploramos la detección y gradación automáticas de patrones de crecimiento tumoral de próstata mediante un Autoencoder Variacional Bayesiano que gracias a la selección probabilística de características que brinda el enfoque Bayesiano y su capacidad de reconstrucción detallada de la etapa decodificadora luego de la etapa del muestreo del espacio latente pretende obtener tasas de precisión en la clasificación que resulten competitivas con los modelos existentes.

Estudios futuros se centrarán en una comparación exhaustiva de los tres enfoques principales para el análisis de imágenes de próstata para la gradación completa de Gleason: métodos de segmentación a nivel de píxel basados en imagen y análisis a nivel de glándula.

V. REFERENCIAS

Charte D., Herrera F., & Charte F. (2019). Ruta: implementations of neural autoencoders in R. Knowledge-Based Systems.

Donald Gleason, “Histologic grading of prostate cancer: A perspective, human pathology,” (1992).

Fausto Milletari1, Nassir Navab1, and Seyed-Ahmad Ahmadi (2016) “V-net: Fully convolutional neural networks for volumetric medical image segmentation,” Vol. 39, pp. 2481–2495.

Litjens, G., Debats, O., Barentsz, J., Karssemeijer, N., & Huisman, H. (2017). SPIE-AAPM PROSTATEx Challenge Data (Version 2) [dataset]. The Cancer Imaging Archive. https://doi.org/10.7937/K9TCIA.2017.MURS5CL

Olaf Ronneberger, Philipp Fischer, and Thomas Brox (2015) “U-net: Convolutional networks for biomedical image segmentation,” Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 9351, pp. 234–241.

Rebecca L. Siegel, Kimberly D. Miller, and Ahmedin Jemal (2016), “Cancer statistics, 2016,” CA: A Cancer Journal for Clinicians, vol. 66, no. 1, pp. 7–30.

Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla (2017) “SegNet: A Deep Convolutional EncoderDecoder Architecture for Image Segmentation,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 12, pp. 2481–2495.

Xu W., Keshmiri S., & Wang G. (2019). Adversarial approximated autoencoder for image generation and manipulation. IEEE Transactions on Multimedia, 21(9), 2387-2396.