DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico Final

Reconocimiento de patrones en imágenes de células

Reconocimiento de patrones
Primer Cuatrimestre de 2021

Integrante	LU	Correo electrónico
Gómez, Bruno Agustín	428/18	bgomez@dc.uba.ar
Olszanowski, Evelyn	46/11	eolszanowski@fbmc.fcen.uba.ar
Heidenreich, Ana C.	/	heidenreich.ac@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

Índice

1. Introducción

Un desafío presente en biología celular es determinar el estado de agregación de las moléculas dentro de las células. El estado de agregación (u oligomerización) de las moléculas se refiere a la estructura cuaternaria de las mismas, es decir, a la cantidad de sub-unidades presentes. En particular se puede desear conocer este estado de oligomerización en las distintas partes de la célula. Una forma de afrontar este desafío es mediante la utilización de microscopía confocal, estudiando moléculas *taggeadas* fluorescentemente, empleando la técnica *Number and Brightness*.

1.1. Técnica *Number and Brightness analysis* (N&B)

La técnica N&B consiste en la adquisición de decenas de imágenes por microscopía confocal de fluorescencia de células vivas a través del tiempo (en un período de 2 minutos aproximadamente). Una vez adquiridas las imágenes se procede a realizar el cálculo del Brillo Molecular para cada volumen de observación confocal (que podemos considerar equivalente a cada píxel), a partir de el promedio (*average*) y la varianza (*variance*) (Figura 1). El brillo molecular nos da la información del estado de oligomerización de la molécula. El valor de brillo obtenido para cada parte de la célula, en diferentes condiciones experimentales se relativiza contra una condición para la cual el valor de brillo se sabe que corresponde a la molécula en su estado monomérico; es decir, con una sola sub-unidad en su estructura secundaria.

En este trabajo se utilizaron imágenes que registran el receptor de glucocorticoides (GR) taggeado fluorescentemente con una proteína llamada GFP para poder visualizarlo. En las imágenes con las que trabajamos podemos observar que GR se encuentra en el núcleo de la célula. En particular hay un sector de mucha intensidad de fluorescencia donde se encuentra la mayor concentración de GR, a ese sector le llamamos array.

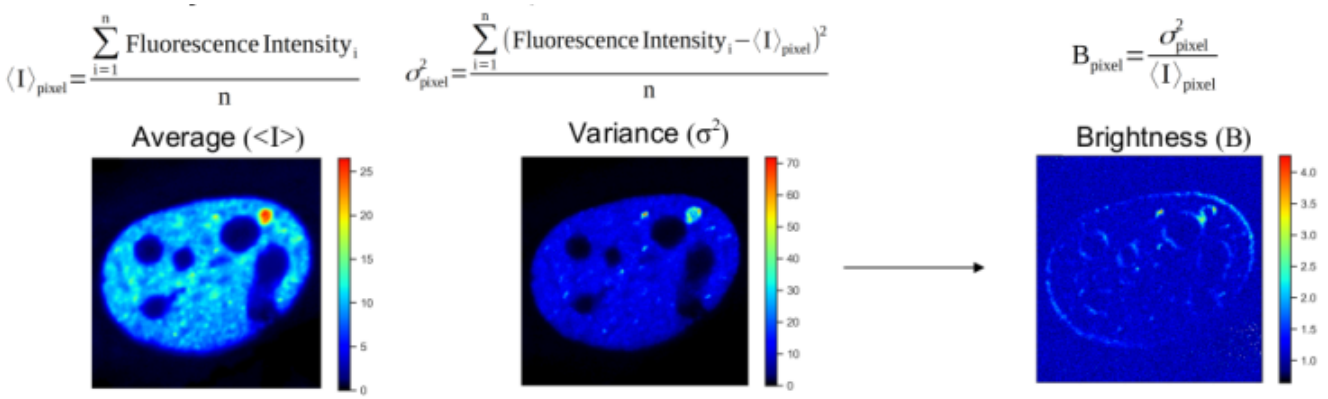

Figura 1: Cálculos realizados en la técnica *Number and Brightness*. Cálculo del brillo (B) a partir de la intensidad de fluorescencia promedio ($\langle I \rangle$) y la varianza (σ^2), en cada píxel.

1.2. Procesamiento manual vs. automatizado

El procedimiento actual estándar para obtener las regiones de interés de una célula consiste en los dos siguientes pasos: obtención de valores de Intensidad Promedio y Brillo, según las imágenes de la figura 1, y mediante la observación del gráfico en el plano para poder discriminar los píxeles correspondientes a las distintas partes de la célula. Como lo indica la figura 2 se pueden observar distintas nubes de puntos, siendo cada punto un píxel. Los mismos se agrupan en función del sector de la célula al cual corresponden. Los sectores de interés en las células son el citoplasma, el núcleo, el array (punto mas brillante dentro del núcleo). Todos los píxeles correspondientes al exterior celular serán catalogados como background. El resto de los píxeles que quedan por fuera de los cuatro recuadros corresponden a los bordes entre los diferentes sectores, no son de interés, por lo tanto no serán seleccionados. Observando este gráfico se pueden segmentar los píxeles, seleccionando los rangos de valores deseados en los ejes cartesianos. Este es el procedimiento manual de segmentación. Dado que los rangos de valores varían entre células, hay que hacerlo de manera individual. Esta manera de analizar los datos tiene las desventajas de que puede demorar mucho tiempo cuando se trabaja con una gran cantidad de células, y además es subjetivo al operador. Estas complicaciones nos llevaron a querer generar un proceso de automatización en el reconocimiento de las partes celulares en este tipo de imágenes de microscopía confocal de fluorescencia, el cual constituye el objetivo del presente trabajo. Nuestra unidad de análisis y de clasificación es cada píxel.

Para poder realizar esta segmentación de manera automática, utilizamos el procedimiento de árboles de clasificación *Random Forest* (RF). Primero debimos clasificar manualmente los píxeles, de manera tal de tener los valores de *True Label* para cada píxel.

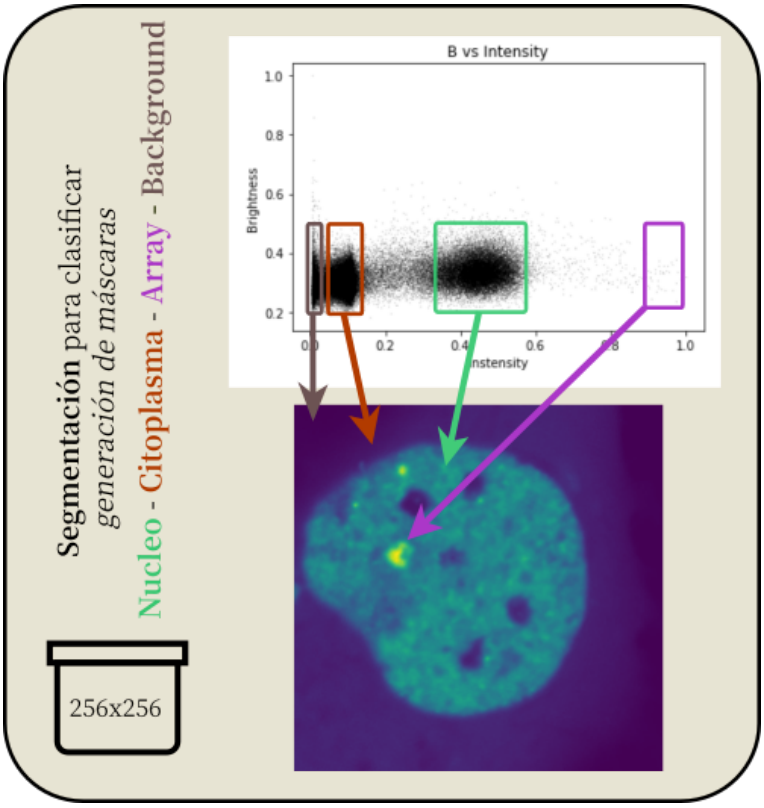

Figura 2: Distribución de los píxeles en el gráfico de Brillo vs Intensidad Promedio. Se muestra a que parte de la célula corresponde cada nube de puntos, donde cada punto es un píxel de la imagen.

1.3. Random Forest

Los métodos basados en árboles, como lo es el algoritmo *Random Forest* (RF) sirven tanto para regresión como para clasificación. En este trabajo se empleó este algoritmo jerárquico para clasificar píxel a píxel las imágenes pre-taggeadas a partir de la generación de máscaras por segmentación, como lo indica el esquema de la figura 2.

Random Forest es un modelo que utiliza internamente múltiples *decision trees* para evitar el problema del overfitting que generan los árboles de forma independiente. Realizando una votación de mayoría que permite incrementar la robustez para el problema mencionado anteriormente y para reducir la varianza obtenida. La idea en general es unir todos los modelos como si fueran un todo. Es llamado un "ensemble method" dado que consiste en utilizar un gran número de otros modelos.

1.3.1. Variación de filas o Bagging

El *Bagging* es una técnica que incorpora el algoritmo de Random Forest entrenando cada *decision tree* con un split de los datos aleatorio, esto implica que para cada *decision tree* utilizado vamos a obtener conjuntos de entrenamiento y prueba distintos. Luego en cada uno de estos se utiliza una función de pérdida sobre la predicción realizada en el conjunto de validación o prueba. Finalmente, cuando se quiere hacer una clasificación de un modelo simplemente se realiza una votación por mayoría sobre los *decision trees* utilizados.

Sin embargo, más allá de la robustez que provee la utilización de *Bagging* puede ocurrir que justo nuestro conjunto de datos sea más propenso a darle importancia a variables que no son importantes, y esto va a influenciar a los modelos que sean entrenados con el mismo. Este fenómeno va a generar que los *decision trees* utilizados estén altamente correlacionados. Como solución a esto surge la técnica de **Random subsets**.

1.3.2. Variación de columnas o Random subsets

Esta técnica consiste en restringir la utilización de variables dentro del conjunto de datos a la hora de entrenar cada *decision tree*. Esto hace que el entrenamiento quede atado a solamente esas variables que le estamos indicando. Esto nos permite generalizar mejor el modelo para evitar el problema mencionado en la sección anterior. Para cada split del **Bagging** vamos a considerar un subset de features seleccionadas de forma aleatoria, en general, dentro de la literatura para los modelos

de clasificación se usa $\sqrt{p}$ siendo p el número de atributos que contiene el dataset. A pesar de que este tipo de modelos no tiene la interpretabilidad que tienen los *decision trees* por si mismos, hay una manera de medir la importancia de las características utilizadas. El procedimiento sería el siguiente:

```
originalRf = RandomForest()

#is a list of tuples with (permutedVariable, actualModel)
rfPermuted = originalRf.variablePermutation()

#dictionary with key: feature, value: importance
variableRelevance = {...}

for model in rfPermuted:
    # average score before the permutation
    previousAvgScore = originalRf.getScore().avg()

    # average score after the permutation
    actualAvgScore = model.getScore().avg()

    variableRelevance[model.permutedVariable] = previousAvgScore - actualAvgScore

return variableRelevance
```

Notar que este procedimiento puede cuantificarse con el normalizando los puntajes con el mayor obtenido y así generar un porcentaje de importancia para cada feature.

1.3.3. Criterios de ramificación

Durante el proceso de la toma de decisiones participan múltiples características y esto hace que sea especialmente relevante cual de ellas es asignada a la raíz para poder comenzar a ramificar hacia abajo. La idea es que a medida que nos movemos hacia abajo en la ramificación obtengamos un menor nivel de impureza e incertidumbre para producir una mejor clasificación. Para poder cuantificar la calidad de la ramificación dentro de este trabajo nos vamos a centrar principalmente en el uso de dos criterios:

- **Information Gain (entropy):** Este criterio es aplicado para cuantificar la ganancia de información que nos puede aportar una feature en términos de la información basada en la clasificación bajo la noción de entropía.

$$Entropy = -\sum_{j}^{n} p_j * log_2(p_j) \tag{1}$$

- **Gini:** El índice de Gini, también conocido como la impureza de Gini, calcula la probabilidad de que una feature en particular sea clasificada incorrectamente cuando se selecciona de forma aleatoria. Al igual que la entropía los valores que puede tomar están entre 0 y 1. Sin embargo, la diferencia principal radica en que cuanto menor es el valor, mayor es la pureza.

$$GiniIndex = 1 - \sum_{j} p_j^2 \tag{2}$$

La diferencia principal entre ambos criterios radica en los siguientes motivos:

- El índice de Gini facilita la utilización de grandes distribuciones haciéndolas mas fáciles de implementar, mientras que Information gain favorece a las distribuciones pequeñas que tienen un menor número de valores múltiples
- Finalmente Gini opera con las variables categóricas en términos de “éxito” o “fracaso” y realiza solamente un split binario, en opuesto a esto, Information Gain computa la diferencia de entropía antes y después del split para indicar la impureza de las clases de elementos.

1.4. Método de K-Fold cross validation

Cross-validation es un procedimiento de remuestreo utilizado para evaluar modelos de aprendizaje automático en una muestra de datos limitada.

El procedimiento tiene un único parámetro llamado K que refiere al número de grupos en los que se dividirá la muestra de datos. El procedimiento a menudo se llama *K-Fold cross-validation*. Cuando se fija el valor de K en 10, por ejemplo, el método se determina como *10-Folds cross-validation*.

Cross-validation se utiliza principalmente en aprendizaje automático aplicado para entrenar un modelo en datos no vistos. Es decir, usar una muestra limitada para estimar cómo se espera que el modelo funcione en general, realizando las predicciones sobre datos que no se usaron durante el entrenamiento del modelo.

Es un método popular debido a su simpleza y porque generalmente da como resultado una estimación menos sesgada de la calidad del modelo que otros métodos que podrían llegar a generar *overfitting*.

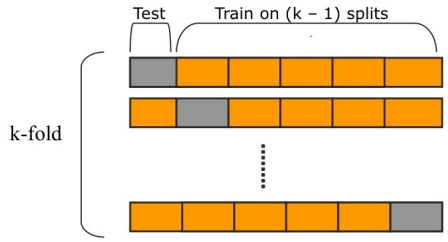

Figura 3: Esquema explicativo del método de *K-Fold cross-validation*.

El procedimiento general es el siguiente:

- Se divide el conjunto de datos en K grupos (cada bloque de la figura 3).
- Se elige el grupo para el conjunto de datos de validación (bloques grises) y se reserva.
- A los grupos restantes se los utiliza como un conjunto de datos de entrenamiento (bloques naranjas).
- Se ajusta el modelo al conjunto de entrenamiento y se lo evalúa en el conjunto de validación.
- Se guarda el puntaje de evaluación y los parámetros asociados pero se descarta el modelo para tener un menor gasto computacional.
- Este procedimiento se realiza para cada split (es decir para cada fila de la figura 3) y luego se obtiene una medida de calidad global del modelo realizando una conjunción de los valores de puntaje para todos los splits, por ejemplo con un promedio.

Es importante destacar que cada observación en la muestra de datos se asigna a un grupo individual y permanece en ese grupo durante la ejecución del procedimiento. Esto significa que cada muestra va a ser utilizada en el conjunto de validación una vez y se usa para entrenar el modelo $K - 1$ veces.

2. Metodología y desarrollo

A continuación se detalla el procedimiento mediante el cual se generaron los sets de datos utilizados para entrenamiento y *testeo*, se normalizaron los mismos, se segmentaron las imágenes para obtener la asignación de clases por píxel (*true label*), se realizó la ingeniería de *features* y por último, la para automatización de la identificación de los *targets*. Para cada píxel se calcularon 8 atributos o *features* (Brillo, intensidad y 6 texturas, las cuales serán explicadas en la sección 2.3).

2.1. Generación de datos

Para la generación del modelo se utilizaron imágenes correspondientes a 44 células. Empleando la función `train_test_split` se reservó un 40 % de las células para realizar el *testeo*. Para el entrenamiento del modelo se utilizó la implementación de `Scikit Learn`, `StratifiedKFold` con 10 splits, con la opción *shuffle*, la cual mezcla de manera aleatoria las muestras de cada clase antes de realizar el split. Además, preserva la proporción de las clases en cada split.

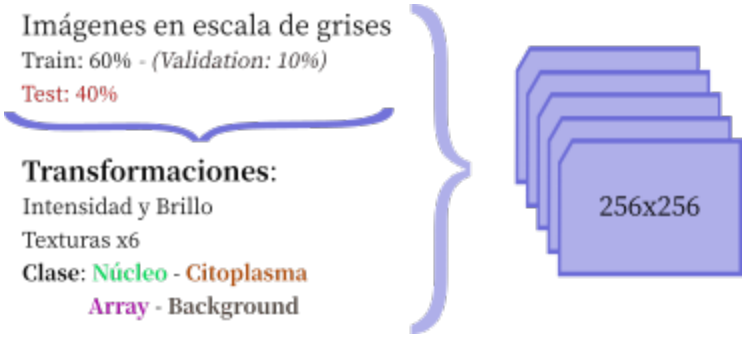

Figura 4: Esquema de tratamiento de las imágenes para el entrenamiento del algoritmo RF para segmentación automática de imágenes

2.2. Segmentación manual de píxeles para la obtención de las *True Labels*

Para poder obtener el valor *True Label* para cada píxel, se procedió de manera que se describe en la sección 1.2. Pero, como se mencionó, los rangos de valores de intensidad de fluorescencia varían significativamente entre células. Para sortear este problema y las que las células sean más comparables entre sí, realizamos una normalización de los valores de intensidad de fluorescencia promedio y brillo, dividiendo el valor de cada píxel por el valor máximo de esa célula. De esta manera, se obtuvieron valores entre 0 y 1, respetando la distribución original de los píxeles observada en el gráfico de Brillo en función de Intensidad Media. Esta normalización solo se realizó en esta etapa, para facilitar la segmentación manual. Para el entrenamiento del modelo se usaron los datos sin normalizar. Una vez normalizados los datos, se definieron los manualmente los cuatro valores de los ejes cartesianos en el gráfico de Brillo vs Intensidad Promedio, que definen el recuadro de selección de píxeles. Se observó la selección en la imagen original de manera de corroborar que la selección sea adecuada para cada región de la célula y luego se generaron los filtros correspondientes a modo de máscaras.

2.3. Feature engineering

Los primeros atributos que se definieron fueron los valores de Intensidad Promedio y Brillo como se describió anteriormente. Con el objetivo de incrementar la cantidad de información disponible para cada píxel de manera tal de enriquecer la clasificación, se calcularon 6 atributos más, denominados texturas (*contrast*, *dissimilarity*, *homogeneity*, *ASM*, *energy* y *correlation*), empleando las funciones que nos provee el módulo de scikitimage.

- `feature.graycomatrix`: Nos permite calcular la matriz de coocurrencia a niveles de grises. Esta no es más que un histograma de los valores de coocurrencia a escala de grises.
- `feature.graycoprop`: La cual nos permite calcular las propiedades de textura utilzando una matriz de coocurrencia a nivel de grises para una ventana de seleccionada. Las texturas se calculan de la siguiente manera:

$$contrast = \sum_{i,j=0}^{levels-1} P_{i,j}(i-j)^2 \tag{3}$$

$$dissimilarity = \sum_{i,j=0}^{levels-1} P_{i,j}(i-j)^2 \tag{4}$$

$$homogeneity = \sum_{i,j=0}^{levels-1} \frac{P_{i,j}}{1+(i-j)^2} \tag{5}$$

$$ASM = \sum_{i,j=0}^{levels-1} P_{i,j}^2 \tag{6}$$

$$energy = \sqrt{ASM} \tag{7}$$

$$correlation = \sum_{i,j=0}^{levels-1} P_{i,j} \left[\frac{(i-\mu_i)\ (j-\mu_j)}{\sqrt{(\sigma_i^2)(\sigma_j^2)}} \right] \tag{8}$$

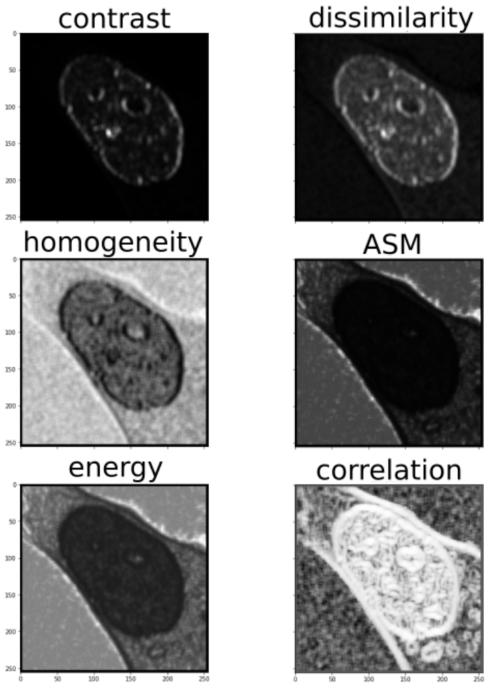

Figura 5: Ejemplo las diferentes texturas calculadas para una célula.

2.4. Funciones de pérdida

2.4.1. Kappa Cohen

Jacob Cohen propuso en 1960 [2] una métrica útil para la evaluación de datos categóricos. La ecuación 9 describe el comportamiento del *score cohen kappa*, donde p_o indica la probabilidad empírica y p_e indica la probabilidad esperada de los dos anotadores que asignan las etiquetas de manera aleatoria.

$$\kappa = (p_o - p_e)/(1 - p_e) \tag{9}$$

Expresado en frecuencias la expresión de la ecuación 9 mejora el costo computacional.

2.4.2. Accuracy

En términos estadísticos, el *accuracy* está relacionado con el sesgo de una estimación. Cuanto menor es el sesgo más exacta es una estimación. Cuando se expresa la exactitud de un resultado, se expresa mediante el error absoluto que es la diferencia entre el valor experimental y el valor verdadero. Y la función se calcula de la siguiente manera.

$$\texttt{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1\ (\hat{y}_i = y_i) \tag{10}$$

Donde $1(x)$ es la función indicador.

2.5. Notebooks utilizadas

A continuación vamos a enumerar las notebooks utilizadas y su función esperada:

- `AtributosYTexturas_bis.ipynb`: Dentro de ella podemos encontrar las funciones implementadas para realizar los cálculos de el brillo y la intensidad promedio, así como las texturas para cada píxel. Además, presenta el procedimiento para generar las etiquetas de las clases a utilizar en el entrenamiento supervisado del modelo. Por último, concatena la información de todas las células pata generar el *data frame* que luego será utilizado para definir los conjuntos proceso de entrenamiento y prueba.

- `RandomForest.ipynb`: En esta notebook podemos encontrar todo el procedimiento y funciones establecidas para poder generar los distintos modelos con los diferentes conjuntos de datos utilizados. Además, luego del entrenamiento se encuentra la parte de medición de performance y visualización de las predicciones realizadas por el modelo, en contraste con los labels correctos.
- `PruebasClustering.ipynb`: Esta notebook contiene el enfoque inicial tomado en este trabajo cuando intentamos utilizar el modelo **K-Means** sobre lo cual vamos a hablar en la sección de conclusiones.

2.6. Datasets generados

Un enfoque que nos resultó interesante analizar, es el tipo de características que presentan los datos que estamos utilizando para alimentar el modelo, y de que manera esto afecta al mismo. Para esto establecimos 3 criterios, basados en la homogeneidad, para generar los datos:

- Conjunto de datos para células con todas las células utilizadas (M1): este es el que mayor heterogeneidad presenta de los 3 dado que utiliza células con distinta cantidad de clases y por consecuente con una proporción que varía demasiado entre clases.
- Conjunto de datos para células sin *array* (M2): este conjunto presenta la mayor homogeneidad en cuando a la proporción de datos por etiqueta y cantidad.
- Conjunto de datos para células con *array* (M3): dentro de estos datos vamos a poder encontrar una menor homogeneidad que el anterior. Esto se debe a que al tener células que presentan *array* el conjunto de píxeles que caen dentro de esta etiqueta es mucho menor en proporción a la cantidad de datos de las otras clases.

En la figura 6 se visualiza un ejemplo del *data-frame* generado para alimentar el modelo *Random Forest*. Cabe destacar que los atributos utilizados fueron todas las columnas que se observan en la figura, excepto el ID de la célula y el target, el cual es la variable que se quiere predecir, y la que se uso para entrenar y testear el modelo.

	Mean	Brightness	target	Celula_ID	contrast	dissimilarity	homogeneity	ASM	energy	correlation
0	0.013480	0.096431	BACKGROUND	06/	0.0	0.0	0.0	0.0	0.0	0.0
1	0.012337	0.096662	BACKGROUND	06/	0.0	0.0	0.0	0.0	0.0	0.0
2	0.013023	0.102565	BACKGROUND	06/	0.0	0.0	0.0	0.0	0.0	0.0
3	0.011652	0.115798	BACKGROUND	06/	0.0	0.0	0.0	0.0	0.0	0.0
4	0.015764	0.100085	BACKGROUND	06/	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...
2883579	0.015555	0.230726	BACKGROUND	Y551A 01.oif.files/	0.0	0.0	0.0	0.0	0.0	0.0
2883580	0.014832	0.254196	BACKGROUND	Y551A 01.oif.files/	0.0	0.0	0.0	0.0	0.0	0.0
2883581	0.008561	0.220544	BACKGROUND	Y551A 01.oif.files/	0.0	0.0	0.0	0.0	0.0	0.0
2883582	0.018449	0.216743	BACKGROUND	Y551A 01.oif.files/	0.0	0.0	0.0	0.0	0.0	0.0
2883583	0.015073	0.222209	BACKGROUND	Y551A 01.oif.files/	0.0	0.0	0.0	0.0	0.0	0.0

Figura 6: Vista previa del *data-frame* utilizado.

Las features utilizadas en el conjunto de datos son: *Mean, Brightness, target, Celula_ID, contrast, dissimilarity, homogeneity, ASM, energy, correlation*

3. Resultados y Discusión

3.1. Parámetros óptimos

A la hora de seleccionar los parametros optimos del modelo intentamos varias configuraciones manualmente:

- **n_estimators**: En primer lugar intentamos variar la cantidad de estimadores utilizados dentro del modelo. En este caso no se pudo observar una mejora significativa en el desempeño del modelo. Pero sí un incremento en el costo computacional del entrenamiento. Considerando que en la utilización de 100 estimadores el costo temporal era de 8 minutos y al utilizar 1000 estimadores el costo incrementaba a 100 minutos presentando una mejora del orden de 10^{-3}. La decisión final fue utilizar 100 estimadores.
- **criterion**: Este parámetro permite configurar la utilización del criterio de **Gini** o **Information Gain** para la ramificación. En este caso la diferencia era significativa en el desempeño del modelo decrementando un 10-20 % el puntaje obtenido al evaluar el modelo cuando se utilizaba **Information Gain**. Finalmente optamos por la utilización del criterio de **Gini**.
- **n_job**: Este parámetro permite indicar la cantidad de jobs que pueden correr en paralelo durante el entrenamiento. Comparamos el cambio en la complejidad temporal en la utilización de uno o más jobs y obtuvimos una gran mejora. Sin embargo, dadas las características del procesador utilizado al utilizar más de 5 *threads* de ejecución los tiempos se mantuvieron constantes. Cómo decisión final, en este caso, utilizamos 5 jobs.
- **n_splits**: Por último, este parámtro indica la cantidad de particiones utilizadas en el **Cross validation**. Optamos por utilizar el valor de 10 particiones dado que el costo computacional se incrementaba demasiado pero a su vez no presentaba una gran mejora. Esto posiblemente se atribuye a la cantidad de datos utilizados para el entrenamiento.

3.2. Matrices de confusión

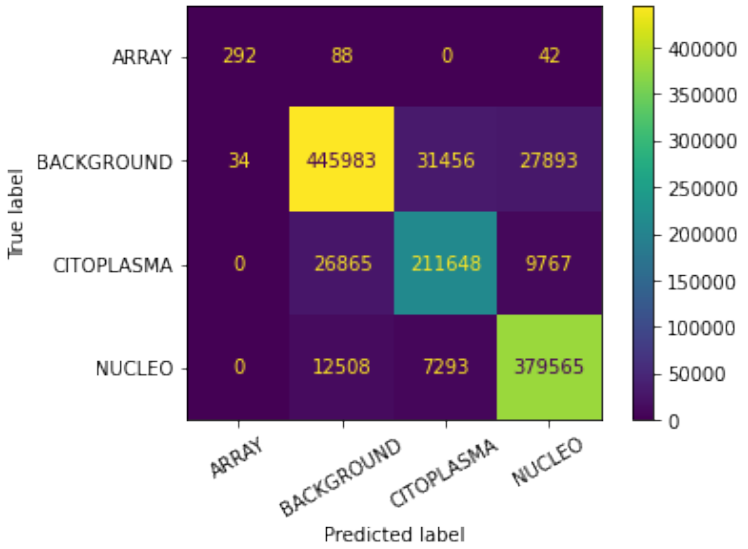

Figura 7: Matriz de confusión obtenida para el modelo entrenado con todos los datos generados.

3.3. Células con o sin *Array*

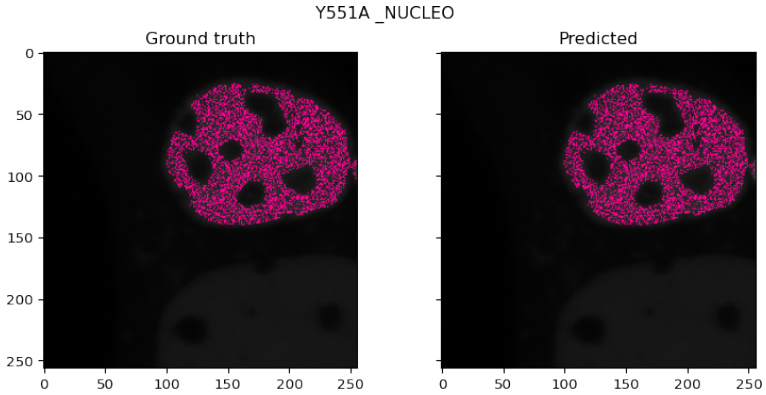

Figura 8: Contraste entre la prediccion realizada sobre el **NUCLEO** por el modelo entrenado solamente con datos sin *array*

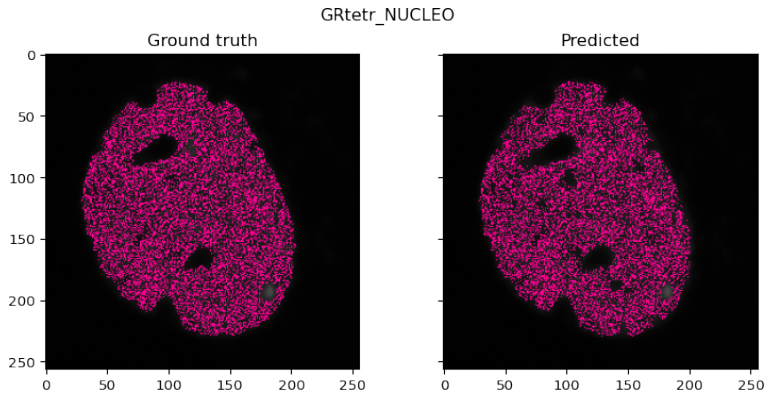

Figura 9: Contraste entre la prediccion realizada sobre el **NUCLEO** por el modelo entrenado solamente con datos con *array*.

3.4. Predicción obtenida sobre las imágenes

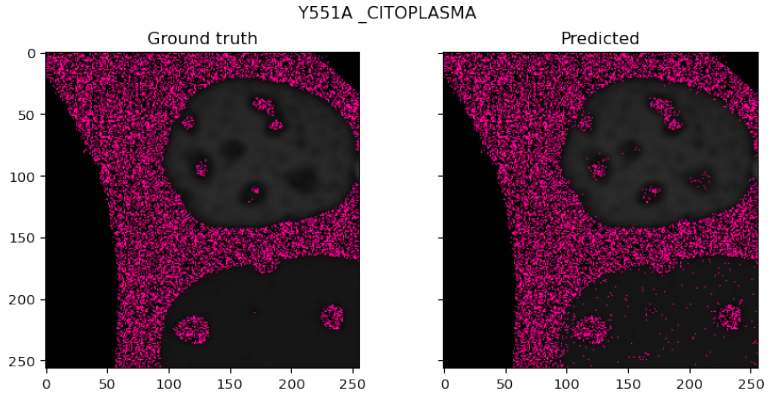

Figura 10: Contraste entre la prediccion realizada sobre el **CITOPLASMA** por el modelo entrenado solamente con datos sin *array*.

3.5. Calidad de clasificación

Como se mencionó en la sección 2.4.1, una de las métricas utilizadas para evaluar la calidad de clasificación fue el Kappa Cohen. En la tabla 1, se muestran los valores obtenidos de esta métrica para los diferentes data-sets utilizados. Como se puede observar, los valores más altos de clasificación fueron obtenidos con el data-set M2 (células sin *array*).

Data-set	Mejor Cohen Kappa	Promedio Cohen Kappa
M1	84.55	84.35
M2	94.88	94.7
M3	89.05	88.85

Tabla 1: KapaCohen

3.6. Importancia de los atributos

En la presente sección se muestran los resultados obtenidos para la importancia de cada atributo utilizado en el algoritmo de RF. Se puede observar que el atributo más importante para la clasificación es la intensidad de fluorescencia promedio (*Mean*), dado que es el atributo que presenta mayor porcentaje de importancia en los tres data-sets probados. Además, se ve que la diferencia de porcentaje con el segundo atributo más importante se encuentra al rededor de un 50 %. Esto demuestra que este atributo es decisivo para la clasificación. Por otro lado, el Brillo se encuentra entre el segundo y tercer puesto de importancia, demostrando también su gran relevancia para la clasificación. Es interesante notar que los data-sets M1 y M3 presentan un orden de atributos más similar entre sí que a M2, siendo éste el que logró un *score* de clasificación mejor.

Atributo	% Importancia
Mean	58.61
Brightness	7.89
correlation	7.48
contrast	7.4
homogeneity	5.31
dissimilarity	4.74
energy	4.54
ASM	4.04

(a) M1 (con y sin *array*)

Atributo	% Importancia
Mean	51.04
contrast	8.85
Brightness	8.17
dissimilarity	7.23
ASM	6.8
energy	6.68
homogeneity	6.54
correlation	4.7

(b) M2 (sin *array*)

Atributo	% Importancia
Mean	71.48
Brightness	6.76
correlation	6.01
homogeneity	3.85
contrast	3.57
dissimilarity	2.88
ASM	2.82
energy	2.63

(c) M3 (con *array*)

Tabla 2: Importancia de cada atributo para los tres data-sets utilizados.

4. Conclusiones y trabajo a futuro

Como indica la figura 2, vemos la generación de máscaras con ventanas rectangulares en el plano Brigthness-Intensity. La importancia del experto en el área es de suma delicadeza dado que la performance es extrictamente dependiente de esto. Dado que solo se emplearon tres ventanas.

Como se mencionó en Resultados, el data-set que mejor *performance* obtuvo fue M2, es decir en ausencia del array. Esto puede deberse a que la cantidad de píxeles para esa clase es muy baja, quedando el resto de las clases extremadamente desbalanceadas respecto de ésta. Por lo tanto, este algoritmo no logró poder clasificar de manera correcta esta clase.

De cara al futuro se pueden probar diferentes modelos y algoritmos a RF para poder comparar su poder de clasificación global, pero prestando mayor atención al array, que es donde RF tuvo mala *performance*.

Un enfoque que intentamos probar inicialmente es el de utilizar un algoritmo de clustering, como **K-Means**, por sobre los píxeles para luego clasificarlos apropiadamente. El problema que encontramos acá es que los clusters que se formaban al aplicar el algoritmo no eran útiles en nuestro problema. Pensamos que esto podría ser causado por *La maldición de la dimensión* dadas las dimensiones de las imágenes con las que trabajamos y para esto estaría bueno utilizar alguna técnica de reducción de dimensión, cómo **PCA**, previo a la clusterización para ver si se logran obtener mejores resultados.

Por otro lado, creemos que la propuesta ideal para este problema sería la utilización de segmentación semántica sobre las imágenes con una red de convolución, cómo las **R-CNN**. Más allá de haberlo intentado, nos encontramos con varias limitaciones técnicas y teóricas de nuestro lado que nos impidieron concretar el objetivo con esta propuesta.

Referencias

[1] Digman, M. A., Dalal, R., Horwitz, A. F., & Gratton, E. (2008). Mapping the number of molecules and brightness in the laser scanning microscope. Biophysical journal, 94(6), 2320-2332. doi:10.1529/biophysj.107.114645.

[2] Cohen, J. (1960). A coefficient of agreement for nominal scales. Educational and psychological measurement, 20(1), 37-46. doi:10.1177/001316446002000104.