Daniel H. Martínez S.
Universidad de Antioquia
Fundamentos de *Deep Learning* 2023-2

Proyecto: Detección histopatológica del cáncer

En este proyecto la métrica de validación utilizada fue el AUC (*Area under the receiver operating characteristic curve*) entre la probabilidad predicha y el objeto observado.

Esta métrica es igual a la probabilidad de que un clasificador clasifique una muestra aleatoria positiva más alta que una muestra aleatoria negativa. Para entender esto es importante saber que los **falsos negativos** (muestras con cáncer detectadas como negativas) y los **falsos positivos** (muestras sin cáncer detectadas como positivas), son muestras que se clasificaron incorrectamente. Por el contrario, los **verdaderos negativos** (muestras sin cáncer detectadas como negativas) y los **verdaderos positivos** (muestras con cáncer detectadas como positivas) son muestras que se clasificaron correctamente. Esta información se cuantifica en la matriz de confusión, que en un caso ideal es una matriz diagonal [1-3].

- ✓ **Descripción de la estructura de los notebooks entregados**

Se entregaron los siguientes notebooks:

- **01– Exploración de datos**: Se realizó la exploración de datos, descargando los datos directamente desde *kaggle* y estudiando el número de imágenes y clases. Luego se selecciona una muestra de 5400 imágenes (con balanceo de clases del 50%) y dichos datos se subieron al *github* del proyecto (en la carpeta *data*) con el fin de facilitar su análisis y el posterior entrenamiento de diferentes modelos.
- **02 – Estudio de la métrica AUC**: Se realizó la separación de los datos *train*, *val* y *test*. Se crea un modelo de prueba para evaluar el comportamiento de diferentes métricas de validación para un clasificador biclase como el AUC, el Recall, la matriz de confusión, entre otros.
- **03 – Aplicación de modelos de CNN**: Se prueban diferentes arquitecturas de Redes Neuronales Convolucionales (CNN), variando tanto el número como el tipo capas. Se evalúan 6 modelos (A, B, C, D, E y F) en función de las métricas de validación (en los modelos C, D, E y F se utiliza *transfer learning*).
- **04 – Modelos & datos aumentados**: Con el objetivo de intentar disminuir el *overfiting* encontrado en los resultados del notebook 03, se prueban los 6 modelos en un nuevo *dataset* con datos aumentados. Para lograr esto, se utilizan las imágenes iniciales y se les agrega un *white-padding (wp)* en la región central a las muestras que no tienen células cancerígenas (clase 0), mientras que a las muestras que tienen cáncer (clase 1), se hace el *wp* en la región por fuera del centro. Con estas nuevas imágenes se aumentan los datos del entrenamiento.
- **05 – Modelos & datos aumentados - 02**: Con el objetivo de intentar disminuir el *overfiting* se realiza un procedimiento similar al notebook 04, pero aumentando los datos de entrenamiento solo realizando el *wp* a las muestras que no tienen células cancerígenas (clase 0).

✓ **Descripción de la solución (arquitectura)**

Los modelos trabajados presentan las siguientes arquitecturas:

- Modelo-A: Utiliza capas convolucionales, *dropout*(0.5) y capas densas.

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input (InputLayer)          [(None, 96, 96, 3)]       0

 Conv2D_3 (Conv2D)           (None, 96, 96, 64)        1792

 Conv2D_4 (Conv2D)           (None, 96, 96, 128)       73856

 flatten (Flatten)           (None, 1179648)           0

 densa_1 (Dense)             (None, 120)               141557880

 densa_2 (Dense)             (None, 24)                2904

 dropout (Dropout)           (None, 24)                0

 densa_3 (Dense)             (None, 12)                300

 output (Dense)              (None, 1)                 13

=================================================================
Total params: 141636745 (540.30 MB)
Trainable params: 141636745 (540.30 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

- Modelo-B: Utiliza capas convolucionales, *maxpooling*, *dropout* y capas densas.

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input (InputLayer)          [(None, 96, 96, 3)]       0

 conv2d (Conv2D)             (None, 96, 96, 32)        896

 max_pooling2d (MaxPooling2  (None, 96, 96, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 96, 96, 64)        18496

 max_pooling2d_1 (MaxPoolin  (None, 96, 96, 64)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 96, 96, 128)       73856

 dropout (Dropout)           (None, 96, 96, 128)       0

 flatten (Flatten)           (None, 1179648)           0

 densa_1 (Dense)             (None, 120)               141557880

 densa_2 (Dense)             (None, 24)                2904

 dropout_1 (Dropout)         (None, 24)                0

 densa_3 (Dense)             (None, 12)                300

 output (Dense)              (None, 1)                 13

=================================================================
Total params: 141654345 (540.37 MB)
Trainable params: 141654345 (540.37 MB)
Non-trainable params: 0 (0.00 Byte)
```

- Modelo-C: Se utiliza *transfer learning* de la arquitectura de Inception-v3, *global-average-pooling*, *dropaout* y capas densas.

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input (InputLayer)          [(None, 96, 96, 3)]       0

 inception_v3 (Functional)   (None, 1, 1, 2048)        21802784

 global_average_pooling2d (  (None, 2048)              0
 GlobalAveragePooling2D)

 dropout (Dropout)           (None, 2048)              0

 dense (Dense)               (None, 120)               245880

 dense_1 (Dense)             (None, 10)                1210

 dense_2 (Dense)             (None, 1)                 11

=================================================================
Total params: 22049885 (84.11 MB)
Trainable params: 22015453 (83.98 MB)
Non-trainable params: 34432 (134.50 KB)
_________________________________________________________________
```

- Modelo-D: Se utiliza *transfer learning* de la arquitectura de Resnet50, *global-average-pooling*, *dropaout* y capas densas.

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input (InputLayer)          [(None, 96, 96, 3)]       0

 resnet50 (Functional)       (None, 3, 3, 2048)        23587712

 global_average_pooling2d (  (None, 2048)              0
 GlobalAveragePooling2D)

 dropout (Dropout)           (None, 2048)              0

 dense (Dense)               (None, 120)               245880

 dense_1 (Dense)             (None, 10)                1210

 dense_2 (Dense)             (None, 1)                 11

=================================================================
Total params: 23834813 (90.92 MB)
Trainable params: 23781693 (90.72 MB)
Non-trainable params: 53120 (207.50 KB)
_________________________________________________________________
```

- Modelo-E: Se utiliza *transfer learning* de la arquitectura de Xception, , *global-average-pooling*, *dropaout* y capas densas.

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input (InputLayer)          [(None, 96, 96, 3)]       0

 xception (Functional)       (None, 3, 3, 2048)        20861480

 global_average_pooling2d (  (None, 2048)              0
 GlobalAveragePooling2D)

 dropout (Dropout)           (None, 2048)              0

 dense (Dense)               (None, 120)               245880

 dense_1 (Dense)             (None, 10)                1210

 dense_2 (Dense)             (None, 1)                 11

=================================================================
Total params: 21108581 (80.52 MB)
Trainable params: 21054053 (80.31 MB)
Non-trainable params: 54528 (213.00 KB)
_________________________________________________________________
```

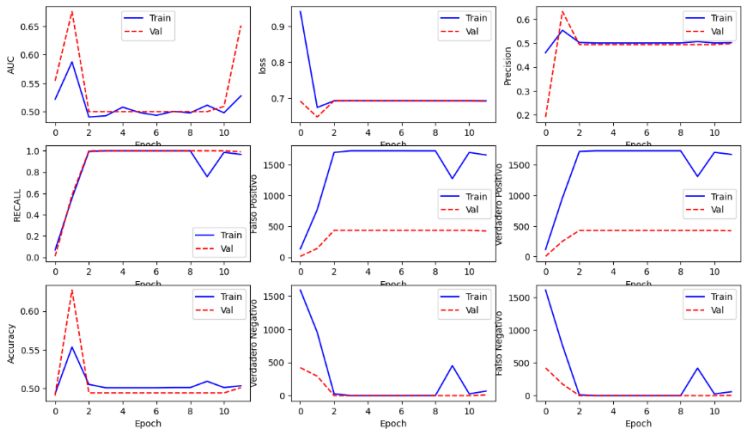
- Modelo-F: Se utiliza *transfer learning* de la arquitectura de Resnet50+Xception, , *global-average-pooling*, *dropaout* y capas densas.

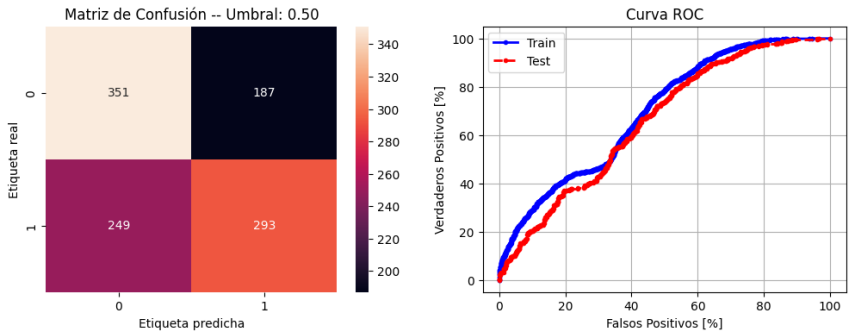
```
Model: "model"
_________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
=================================================================================================
 input (InputLayer)          [(None, 96, 96, 3)]       0         []

 resnet50 (Functional)       (None, 3, 3, 2048)        23587712  ['input[0][0]']

 xception (Functional)       (None, 3, 3, 2048)        20861480  ['input[0][0]']

 global_average_pooling2d (  (None, 2048)              0         ['resnet50[0][0]']
 GlobalAveragePooling2D)

 global_average_pooling2d_1  (None, 2048)              0         ['xception[0][0]']
  (GlobalAveragePooling2D)

 concatenate (Concatenate)   (None, 4096)              0         ['global_average_pooling2d[0][0]',
                                                                  'global_average_pooling2d_1[0][0]']

 dense (Dense)               (None, 120)               491640    ['concatenate[0][0]']

 dense_1 (Dense)             (None, 10)                1210      ['dense[0][0]']

 dense_2 (Dense)             (None, 1)                 11        ['dense_1[0][0]']

=================================================================================================
Total params: 44942053 (171.44 MB)
Trainable params: 44834405 (171.03 MB)
Non-trainable params: 107648 (420.50 KB)
_________________________________________________________________________________________________
```

✓ ***Descripción de las iteraciones***

Las iteraciones realizaron por 50 épocas, un tamaño del *batch* de 32 y la función de *early-stopping* con *patience* igual a 10. El *bias* se inicializa de acuerdo a la receta 'init well'[1] que ayuda con la convergencia inicial ($b_0 = \log_e(pos/neg)$).

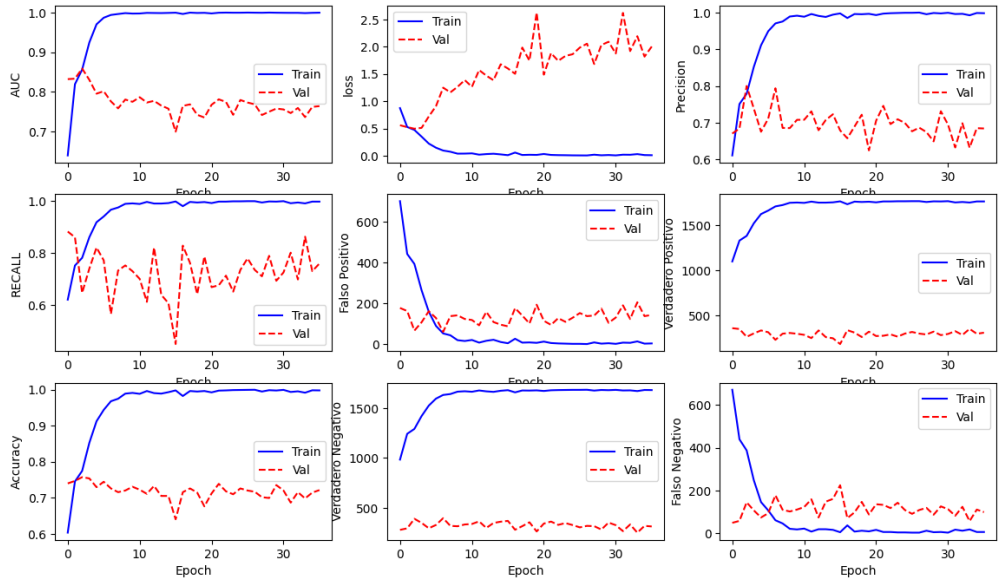
- **Modelo-A**: Este modelo presenta un rendimiento muy bajo en la validación el AUC (~0.5, primera imagen superior izquierda). Presenta mayores problemas con los falsos negativos.

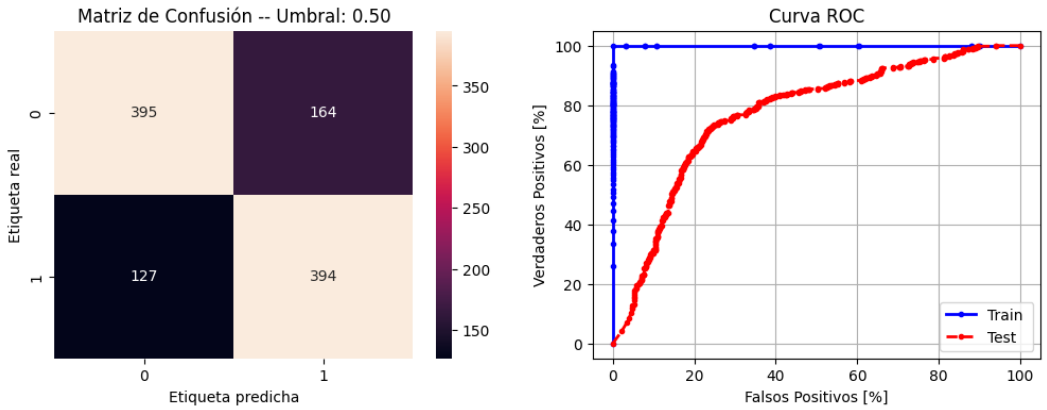




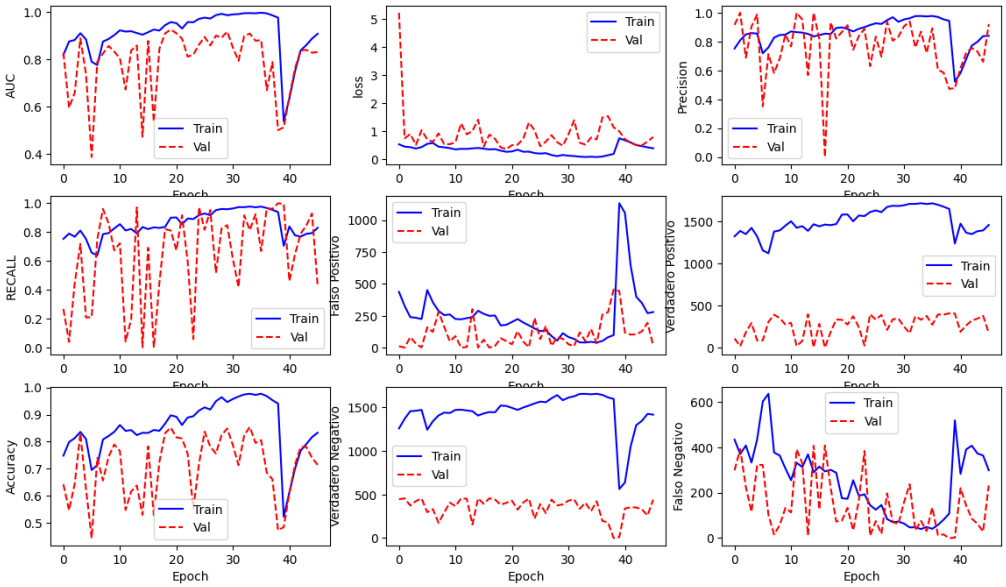
[1] Tomado de: https://karpathy.github.io/2019/04/25/recipe/#2-set-up-the-end-to-end-trainingevaluation-skeleton--get-dumb-baselines

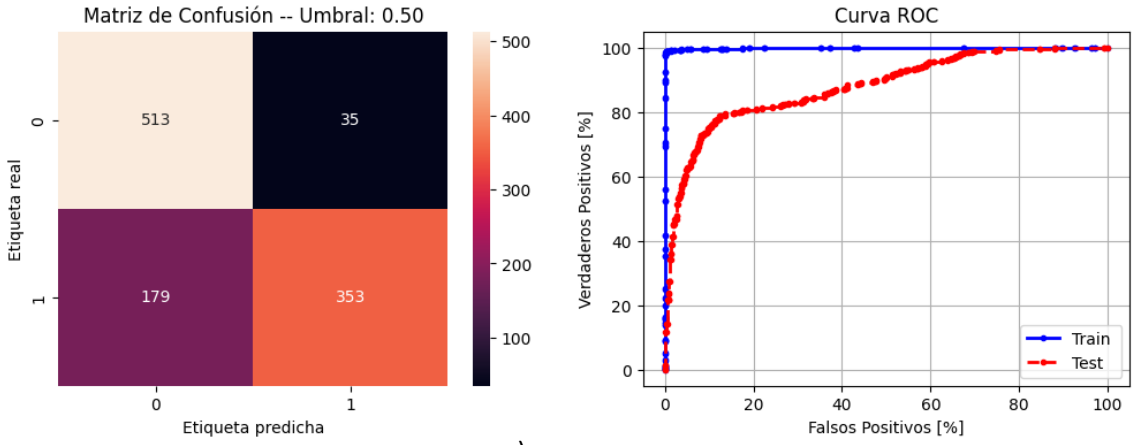
- **Modelo-B**: El AUC mejora respecto al modelo A (~0.75), sin embargo, se puede apreciar un *overfiting*. Un comportamiento similar para el *accuracy* (esquina inferior izquierda).



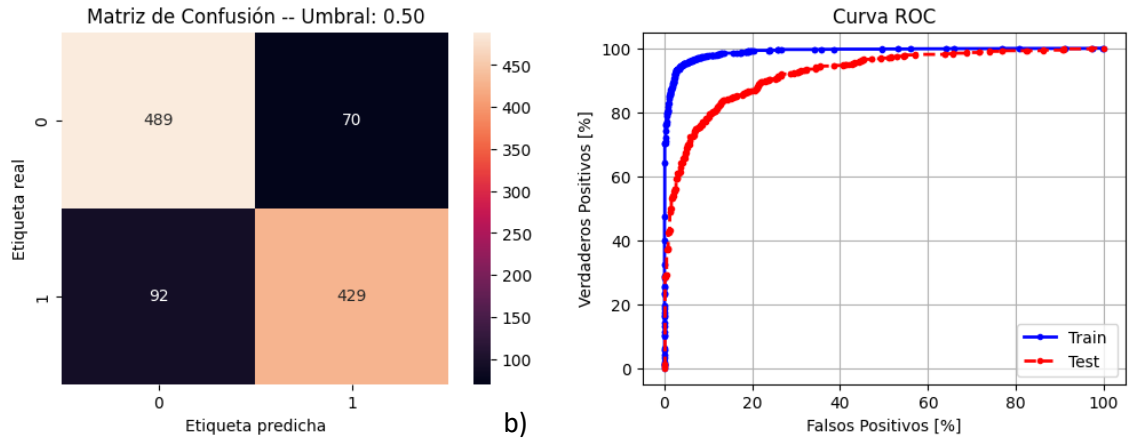


- **Modelo-C**: Tiene un mejor desempeño el AUC (>0.85) en 50 épocas, pero presenta mucha fluctuación. La curva ROC presenta un mejor comportamiento. De acuerdo a la matriz de confusión disminuyen apreciablemente los falsos positivos y los falsos negativos.





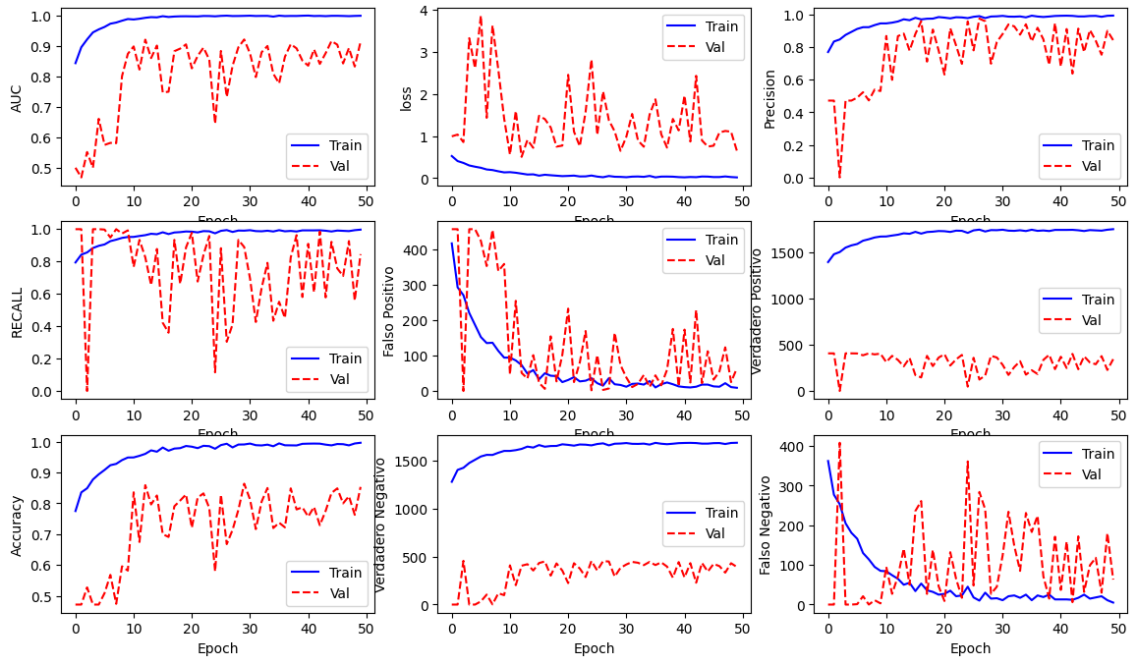
a)

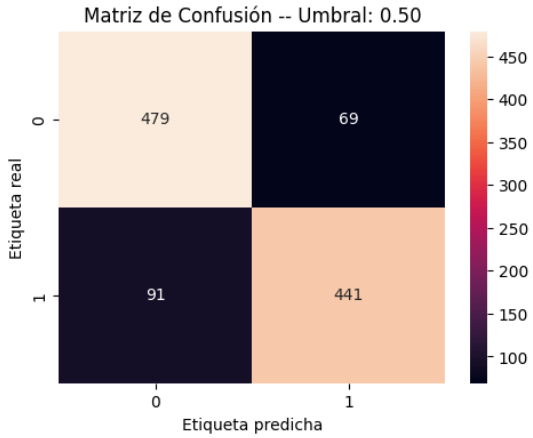


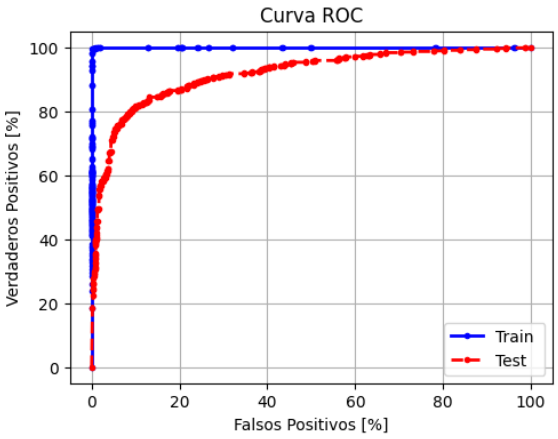
b)

Comparado con pruebas previas (ver Figura-b), al aumentar el modelo con una capa densa con 100 neuronas (Figura-a, modelo actual) se aumentan los falsos negativos, lo que hace disminuir el AUC.

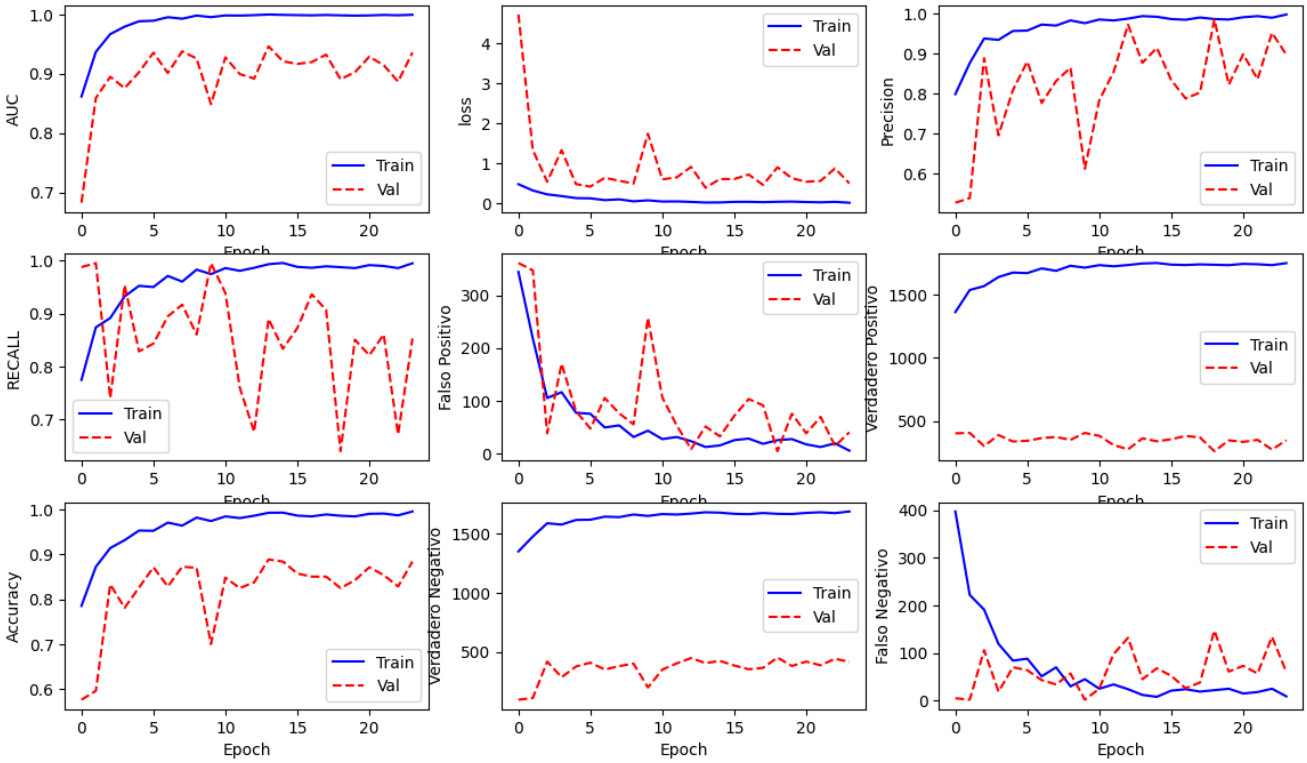
- **Modelo-D**: Se aumenta el AUC (>0.9) pero sigue presentando mucha fluctuación en las diferentes épocas.

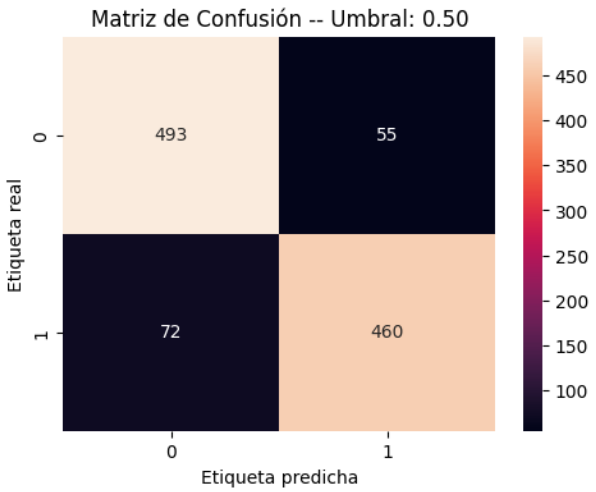


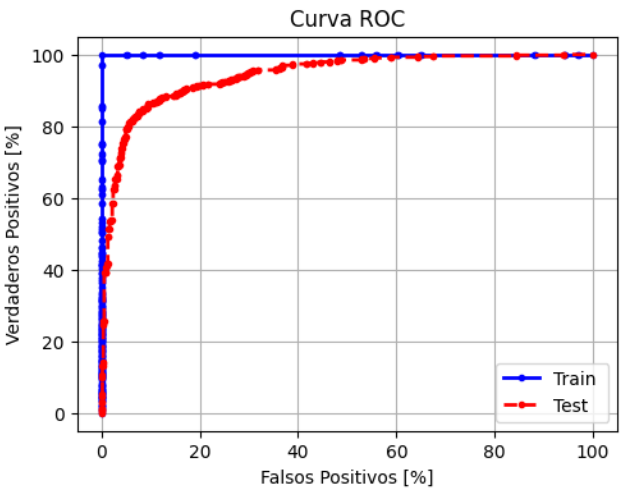




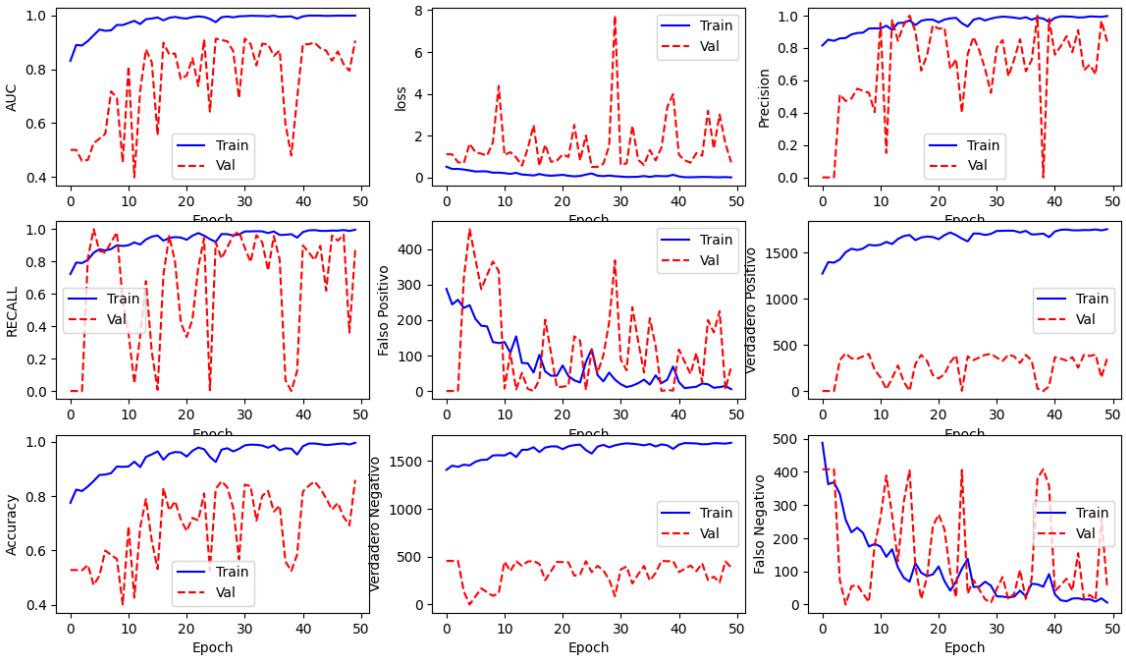
- **Modelo-E**: Respecto al modelo-D se aumentan los verdaderos positivos y disminuyen los falsos positivos, alcanzando el AUC más alto del 0.937.

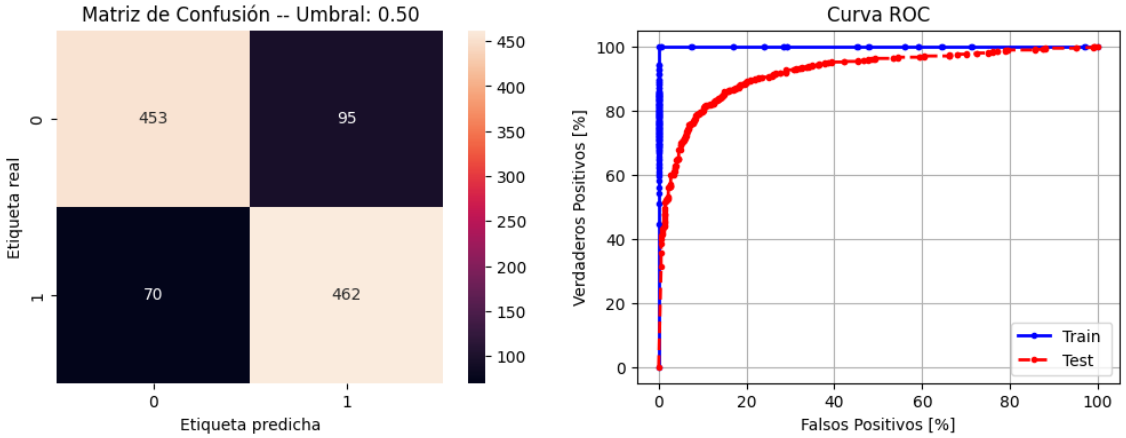






- **Modelo-F**: Similar al modelo D y E el AUC de entrenamiento es prácticamente igual a 1, pero el AUC de validación no supera al modelo E, de igual forma el *loss* aumenta.





✓ ***Descripción de Resultados***

En general los modelos presentan *overfiting*. De acuerdo a las ideas expresadas en los notebooks **04** y **05**, se aumentó el tamaño de los datos de entrenamiento. Las iteraciones anteriores arrojaron los siguientes resultados respecto a la métrica AUC.

Tabla1. Resultados del AUC.

Modelos	AUC[1]	AUC[2]	AUC[3]
A	0.650	0.719	0.680
B	0.717	0.784	0.758
C	0.868	0.891	0.830
D	0.902	0.732	0.863
E	**0.937**	0.913	0.914
F	0.905	0.884	0.912

Donde **AUC^1** corresponde a la métrica de los datos originales **(resultados del notebook 03**), **AUC^2** a los datos aumentados con *wp* tanto en la región central de la clase 0 como en la región por fuera del centro en la clase 1 (**resultados del notebook 04**), y **AUC^3** corresponde a los datos aumentados solo con *wp* para la región central de la clase 0 (**resultados del notebook 05**).

Estos resultados nos muestras que con la estrategia de *wp* se mejora el *score* solo en los modelos A, B, C para AUC^2 y el modelo F para AUC^3 (respecto a AUC^1 y AUC^2 simultáneamente). Sin embargo, estos resultados no superan el score del modelo E, el cual obtuvo el mejor desempeño de acuerdo a la métrica AUC^1=0.937. Algunas ideas para continuar mejorando el score podrían ser el aumento de color, normalización de la tinción o alineación de dominios con una GAN (*Generative Adversarial Networks*) [4-8].

✓ *Conclusiones*

- Se obtuvo un AUC de 0.937 con el Modelo-E correspondiente a la implementación de *transfer learning* del modelo *Xception*, superando una implementación análoga con los modelos *Resnet50* e *InceptionV3*.
- La metodología implementada de aumento de datos, tuvo mejor desempeño en la variante de White-padding a ambas clases, pero no superaron el score del modelo-E utilizando los datos originales.

✓ *Referencias*

[1] M. N. Gurcan, L. E. Boucheron, A. Can, A. Madabhushi, N. M. Rajpoot, and B. Yener, "Histopathological Image Analysis: A Review," *IEEE Rev Biomed Eng*, vol. 2, pp. 147–171, 2009, doi: 10.1109/RBME.2009.2034865.

[2] American Cancer Society, "Lymph Nodes and Cancer." Accessed: Sep. 27, 2023. [Online]. Available: https://www.cancer.org/content/dam/CRC/PDF/Public/7902.00.pdf

[3] H. West and J. Jin, "Lymph Nodes and Lymphadenopathy in Cancer," *JAMA Oncol*, vol. 2, no. 7, p. 971, Jul. 2016, doi: 10.1001/jamaoncol.2015.3509.

[4] B. E. Bejnordi *et al.*, "Stain Specific Standardization of Whole-Slide Histopathological Images," *IEEE Trans Med Imaging*, vol. 35, no. 2, pp. 404–415, Feb. 2016, doi: 10.1109/TMI.2015.2476509.

[5] B. S. Veeling, J. Linmans, J. Winkens, T. Cohen, and M. Welling, "Rotation Equivariant CNNs for Digital Pathology," Jun. 2018.

[6] C. Szegedy *et al.*, "Going Deeper with Convolutions," Sep. 2014.

[7] W. Cukierski, "Histopathologic Cancer Detection." Kaggle, 2018. [Online]. Available: https://kaggle.com/competitions/histopathologic-cancer-detection

[8] Stacke, K., Eilertsen, G., Unger, J., & Lundström, C. (2021). "Measuring Domain Shift for Deep Learning in Histopathology". IEEE Journal of Biomedical and Health Informatics, 25(2), 325–336. https://doi.org/10.1109/JBHI.2020.3032060