

Red_MNIST

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(48, activation='relu'),
    tf.keras.layers.Dropout(0.25),
    tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

1.1 Modelo_básico_TF+Red_MNIST:

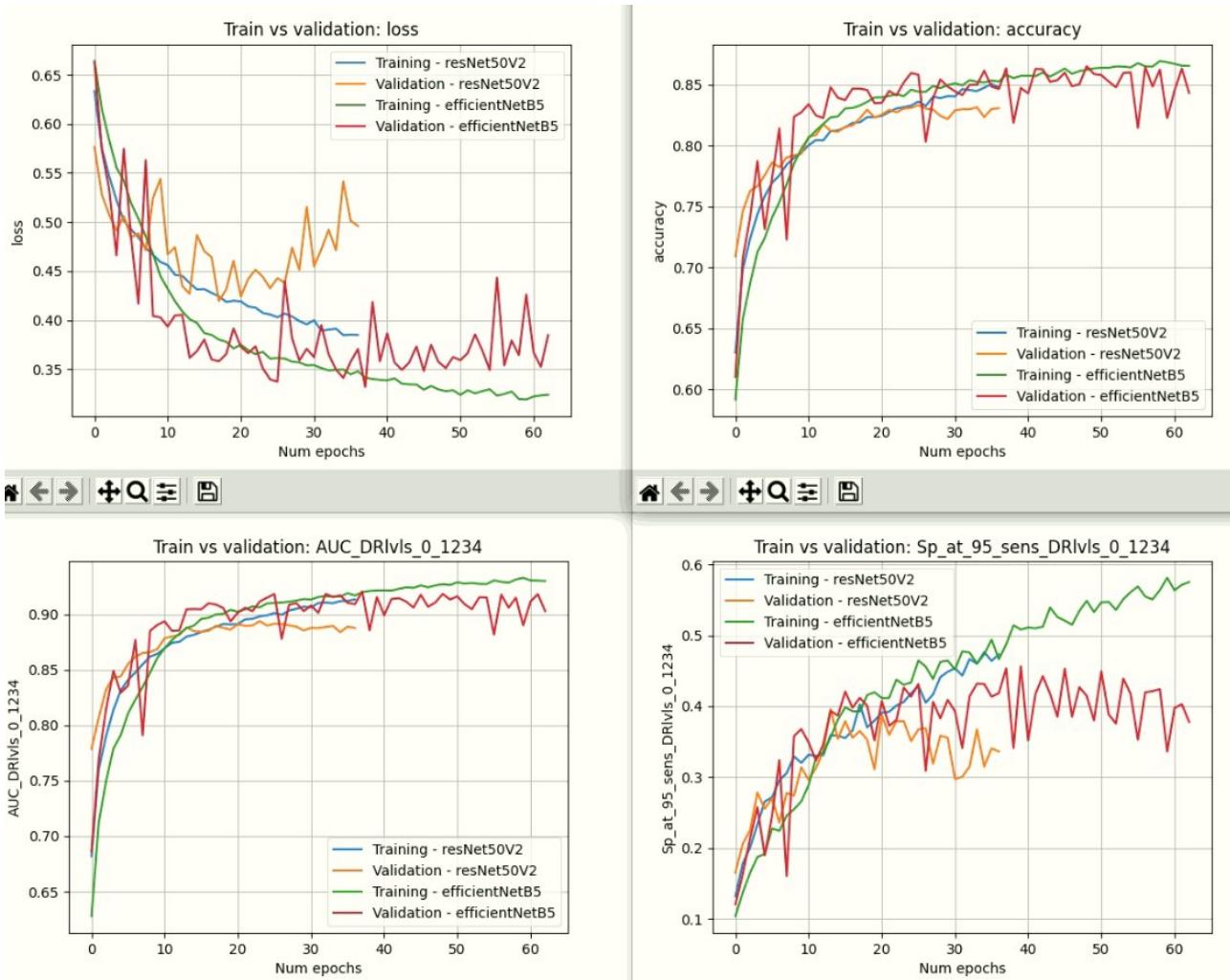
Entrenamiento cargando los datasets y creando la red Red_MNIST tal y como se muestra en el tutorial de TensorFlow: <https://www.tensorflow.org/tutorials/quickstart/beginner?hl=es-419>

Train: 60.000 imágenes

Validación: 10.000 imágenes

Batch size por defecto = 32 → 1.875 iteraciones en entrenamiento.

- Sin data augmentation
- Sin balanceo en ninguno de los dos datasets
- Sin shuffle
- Batch size = 32 para ambos datasets
- Loss function: sparse_categorical_crossentropy -- etiquetado mediante números enteros [0, ..., 9].



1.2 Modelo_ajustado_TF+Red_MNIST:

Se añaden dos leves modificaciones para que esté en igualdad de condiciones que si se cargasen las imágenes utilizando el código propio:

- Como en el otro código las imágenes se leerán del disco, tendrán 3 canales de color, aunque estos sean idénticos. Para tener 3 canales, se replica el mismo canal de intensidad proporcionado originalmente.
- Como el otro código aplica etiquetado 'one-hot', se aplica también aquí, para las 10 clases. Esto implica también sustituir la loss function a 'categorical_crossentropy' en lugar de 'sparse_categorical_crossentropy'.
- El resto se mantiene igual: sin data_aug, sin balanceo ...

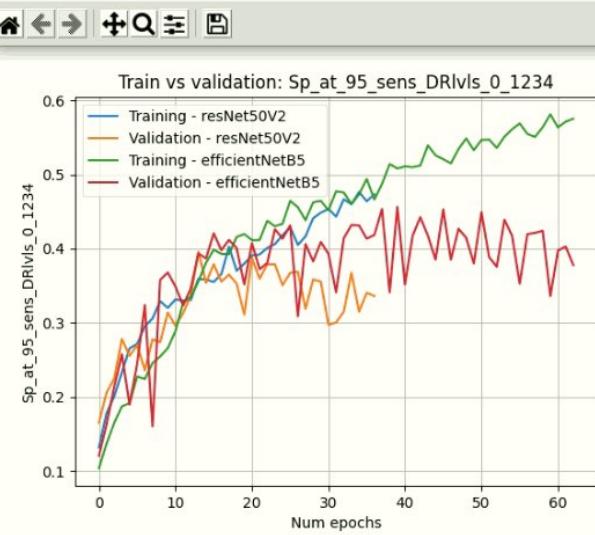
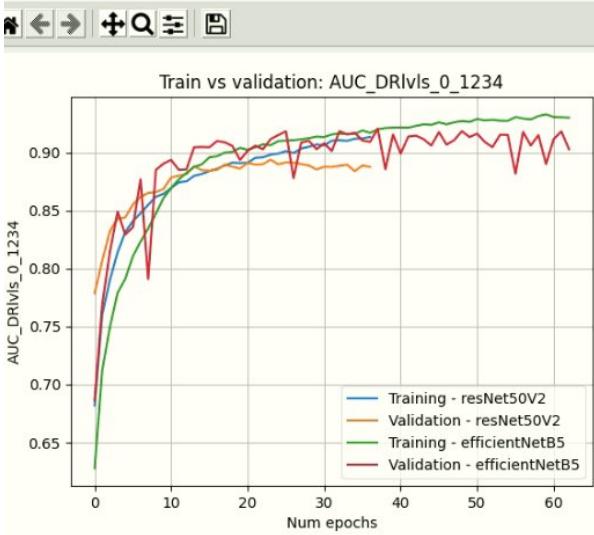
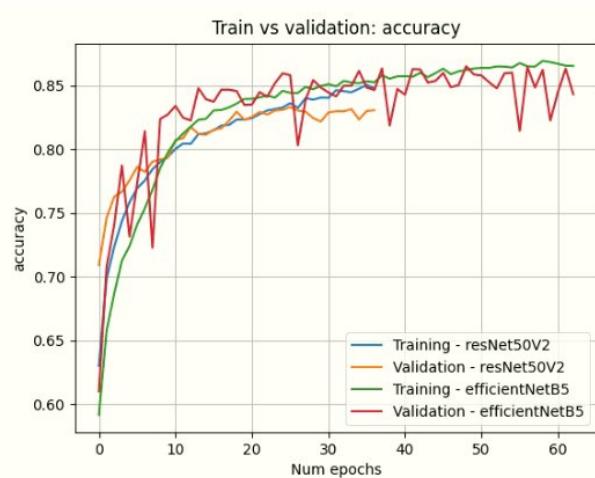
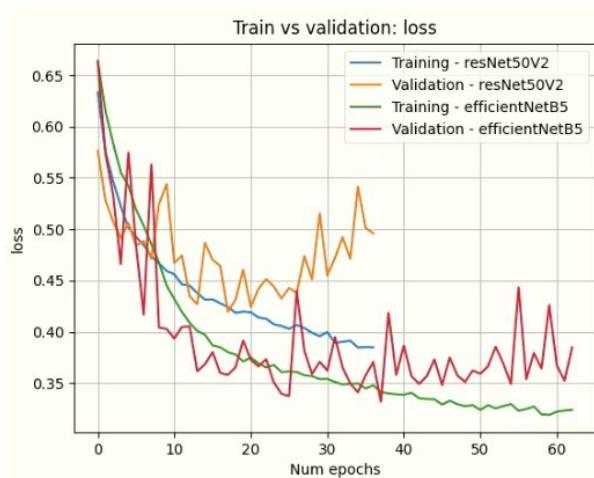
1.3 Modelo_propio_TF+Red_MNIST:

MNIST utilizando código propio de carga de datasets. Las funciones propias se encuentran en:

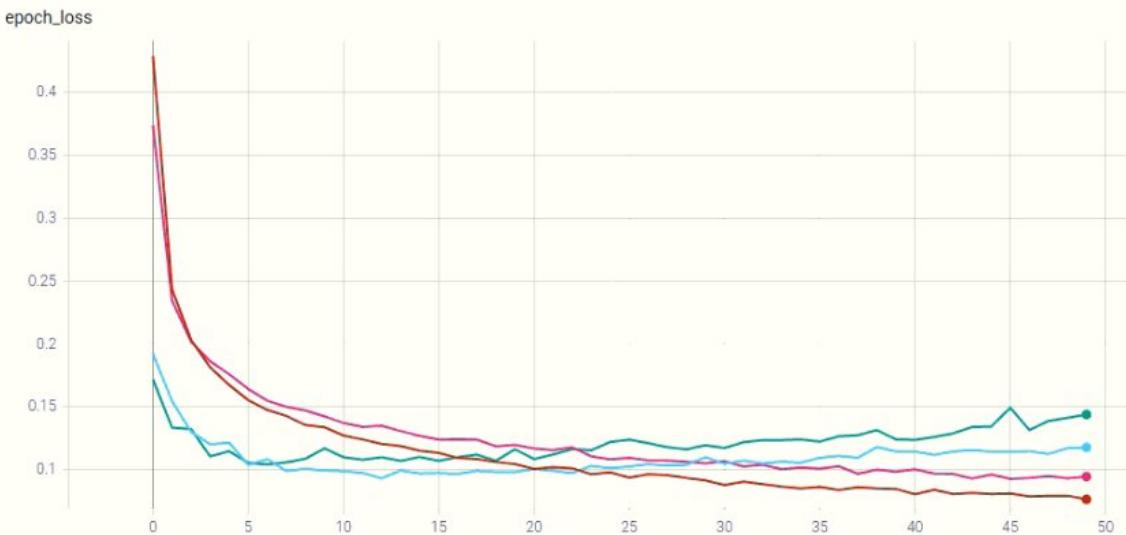
`lib.dataset.prepare_dataset_new()` y sus internas. Generan el dataset como objetos de tipo `tf.data.Dataset`.

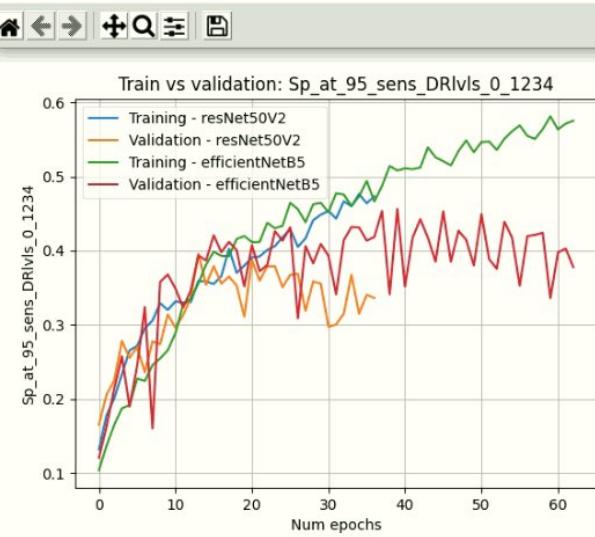
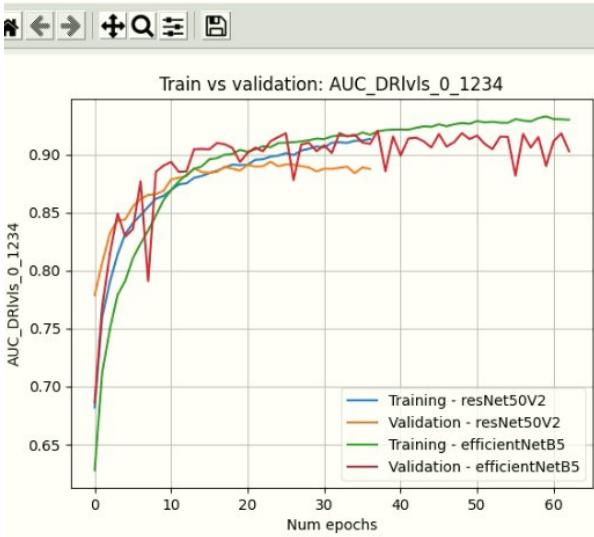
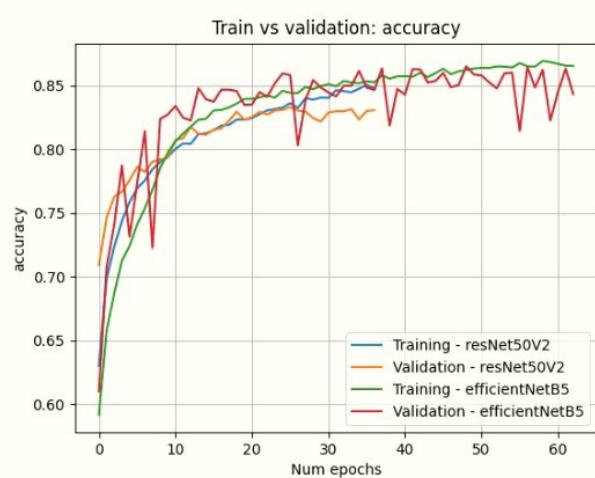
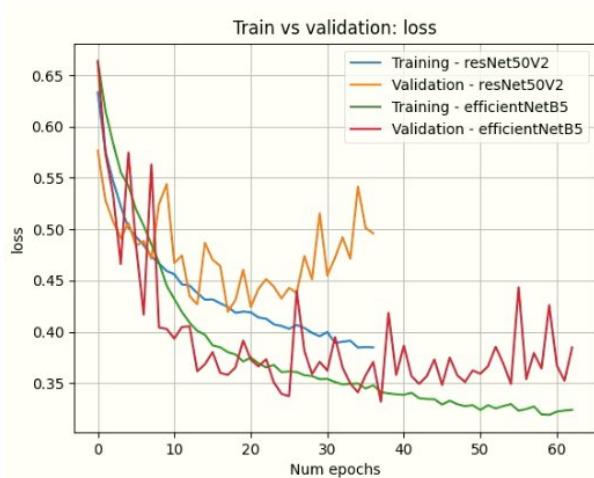
- Sin data augmentation
- Sin balanceo en ninguno de los dos datasets
- Sin shuffle
- Batch size = 32 para ambos datasets

1.4 Gráficas



- MNIST_Modelo_basico_TF/train
- MNIST_Modelo_basico_TF/validation
- MNIST_Modelo_ajustado_TF/train
- MNIST_Modelo_ajustado_TF/validation

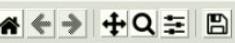
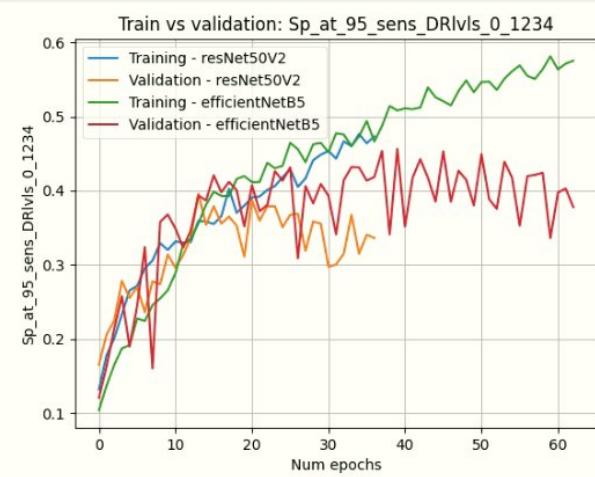
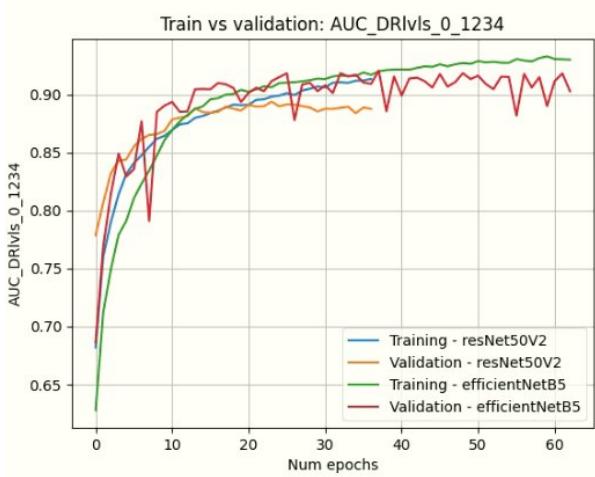
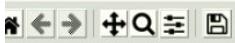
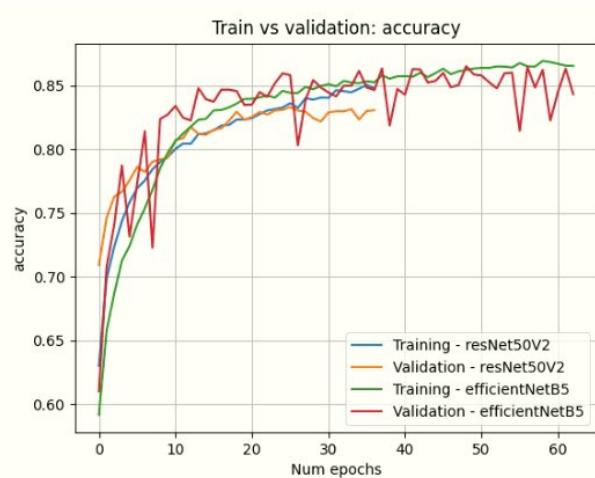
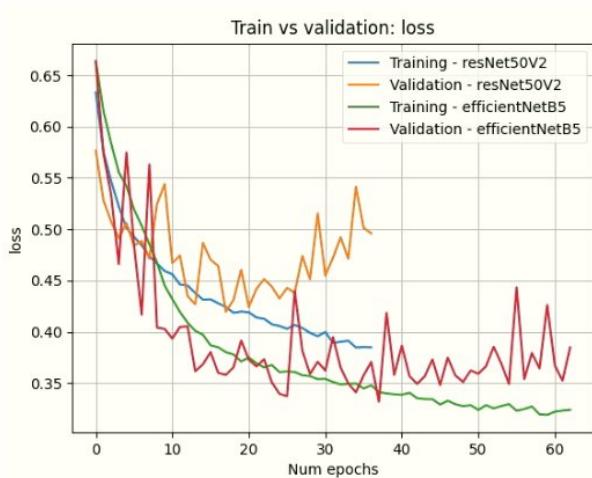




- MNIST_Modelo_propio_TF/train
- MNIST_Modelo_propio_TF/validation
- MNIST_Modelo_basico_TF/train
- MNIST_Modelo_basico_TF/validation

epoch_loss

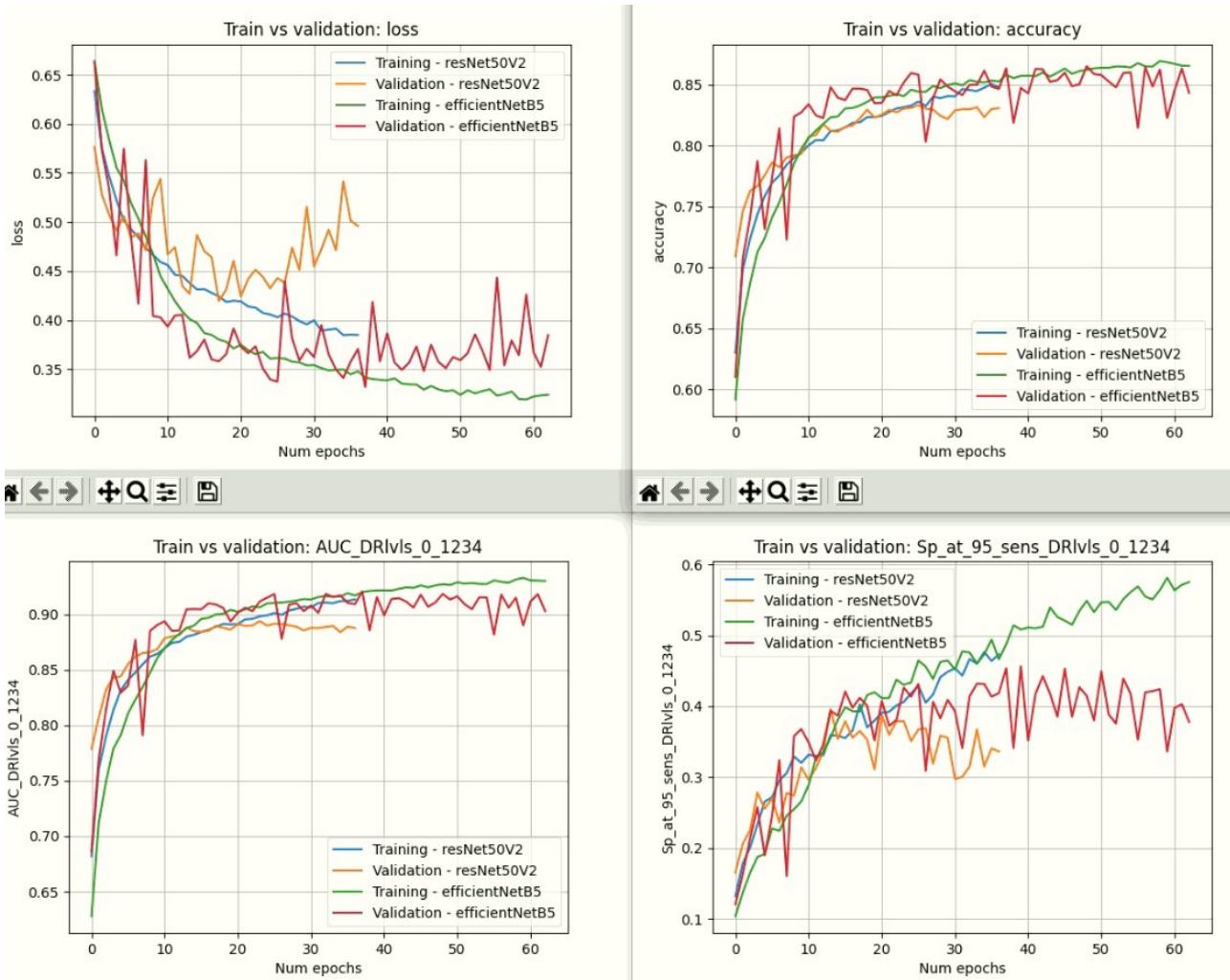




- MNIST_Modelo_propio_TF/train
- MNIST_Modelo_propio_TF/validation
- MNIST_Modelo_basico_TF/train
- MNIST_Modelo_basico_TF/validation
- MNIST_Modelo_ajustado_TF/train
- MNIST_Modelo_ajustado_TF/validation

epoch_loss





Red_Seq

```

model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32,(5,5), strides=2, input_shape=IMAGE_SIZE),
    tf.keras.layers.ReLU(), # 270, 270
    tf.keras.layers.Conv2D(32,(7,7), strides=5),
    tf.keras.layers.ReLU(), # 54, 54
    tf.keras.layers.Conv2D(32,(3,3), strides=2),
    tf.keras.layers.ReLU(), # 27, 27
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(48, activation='relu'),
    tf.keras.layers.Dropout(0.25),
    tf.keras.layers.Dense(len(DR_LEVELS_PER_CLASS), activation='softmax')
])
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

```

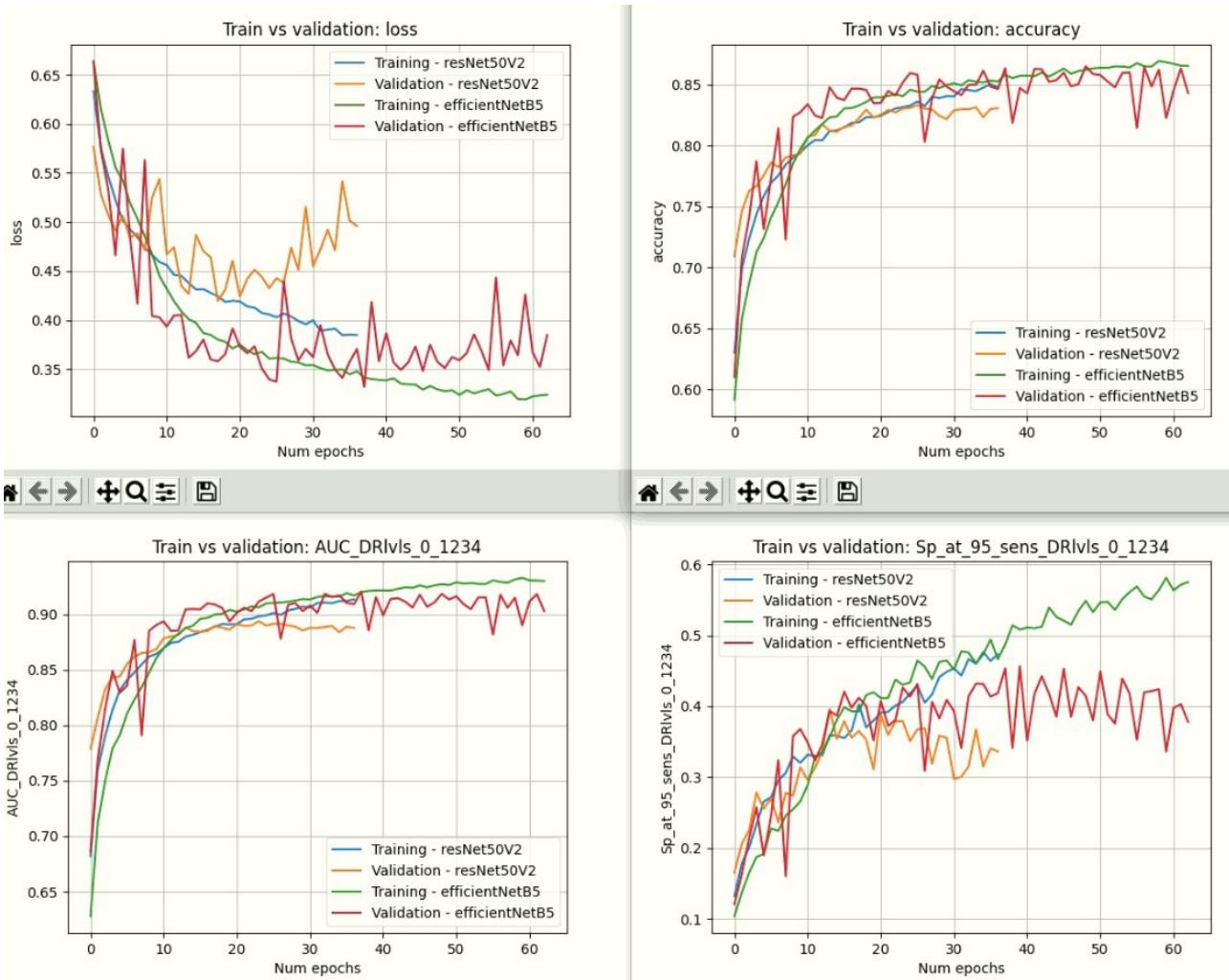
2.1 Modelo_basico_TF+Red_Seq (sin Dropout) + Dataset EYEPACS

Utilizando como datasets de entrenamiento y validación el mismo, DATASET-VALIDATION-10.csv, con 7.103 imágenes, etiquetado en números enteros, por lo que se usa 'sparse_categorical_crossentropy'.

El etiquetado es:

- Etiqueta 0: nivel de RD 0
- Etiqueta 1: niveles de RD 1, 2, 3 y 4

Carga de imágenes realizada en forma de objeto `tf.data.Dataset` y usando dos funciones simples de mapeado (leer imagen y normalizar de [0, 255] a [0, 1]).



- Sin data augmentation
- Sin balanceo en ninguno de los dos datasets
- Sin shuffle
- Batch size = 32 para ambos datasets

2.2 Modelo_ajustado_TF+Red_Seq (sin Dropout) + Dataset EYEPACS

Utilizando como datasets de entrenamiento y validación el mismo, DATASET-VALIDATION-10.csv, con 7.103 imágenes, etiquetado de la misma forma que el anterior, pero en formato 'one-hot'.

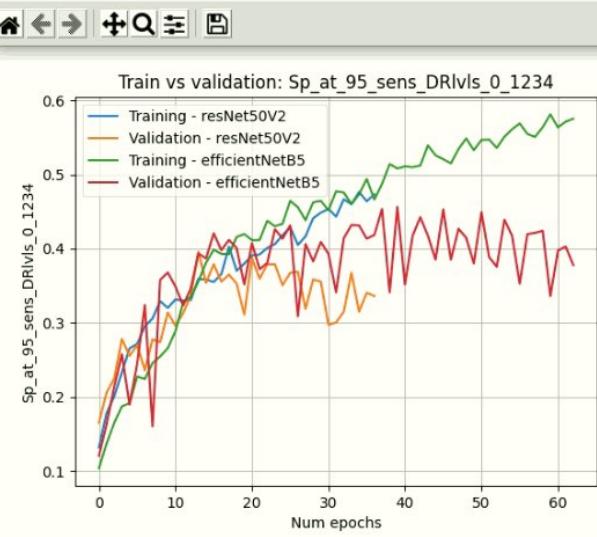
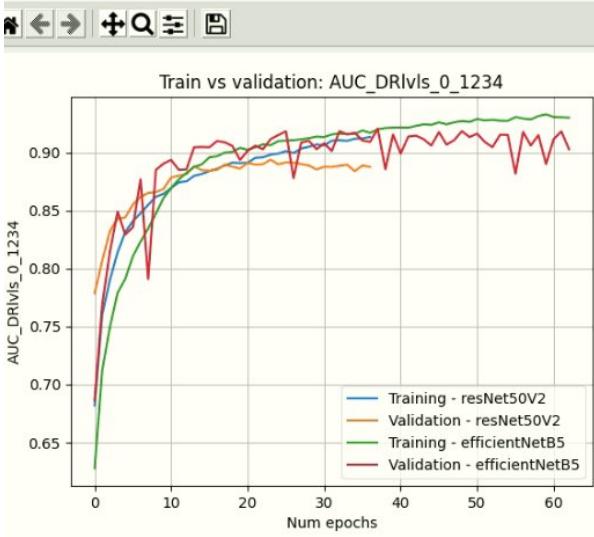
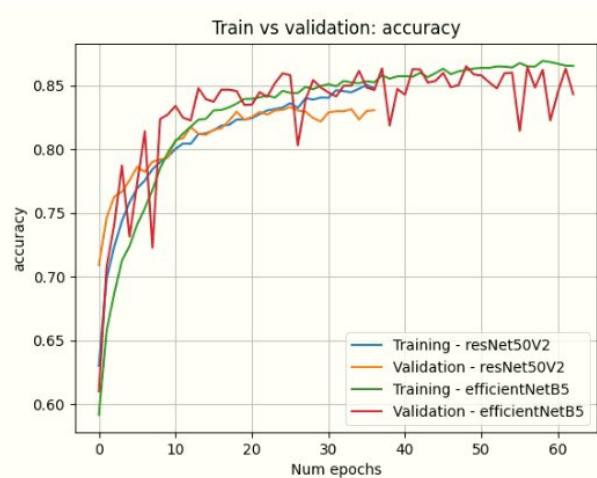
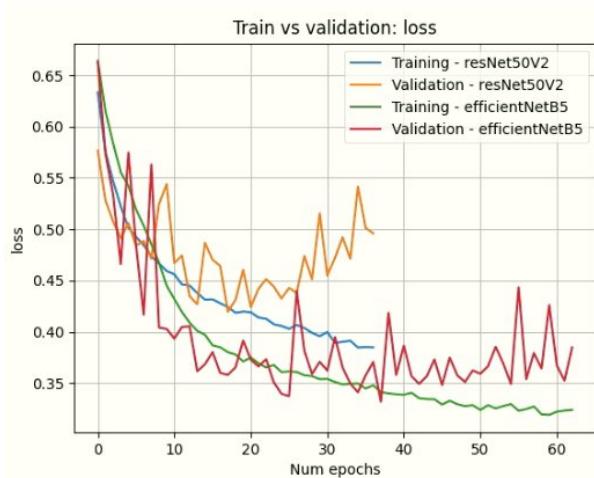
Carga de imágenes realizada en forma de objeto tf.data.Dataset y usando dos funciones simples de mapeado (leer imagen y normalizar de [0, 255] a [0, 1]), convirtiendo a formato 'one-hot' las etiquetas previamente.

- Sin data augmentation
- Sin balanceo en ninguno de los dos datasets
- Sin shuffle
- Batch size = 32 para ambos datasets

2.3 Modelo_propio_TF+Red_Seq + Dataset EYEPACS

Utilizando como datasets de entrenamiento y validación el mismo, DATASET-VALIDATION-10.csv, con 7.103 imágenes, etiquetado de la misma forma que el anterior, en formato 'one-hot'.

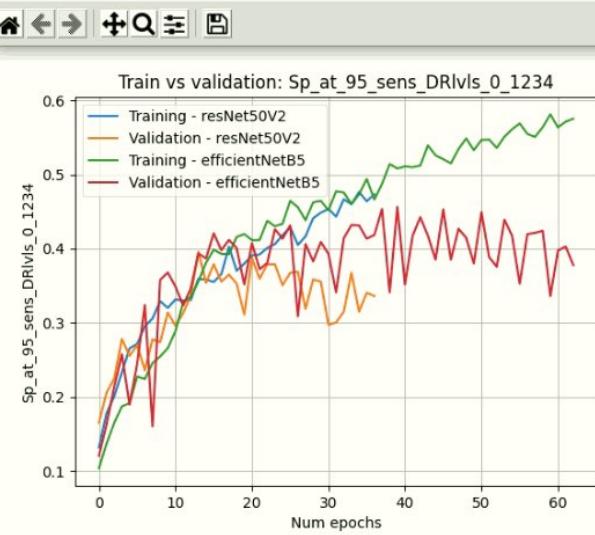
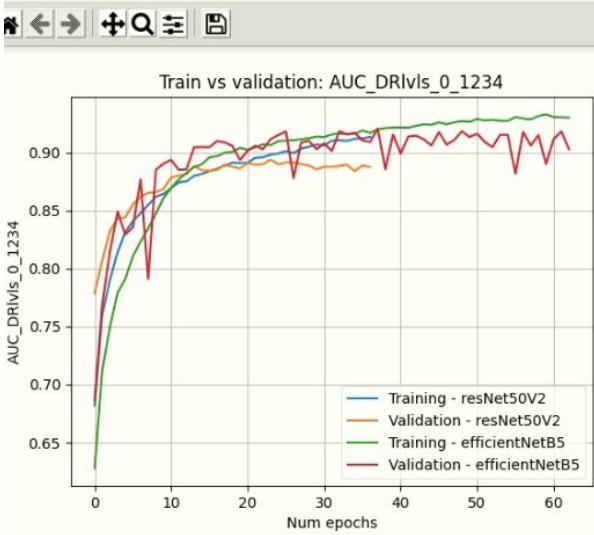
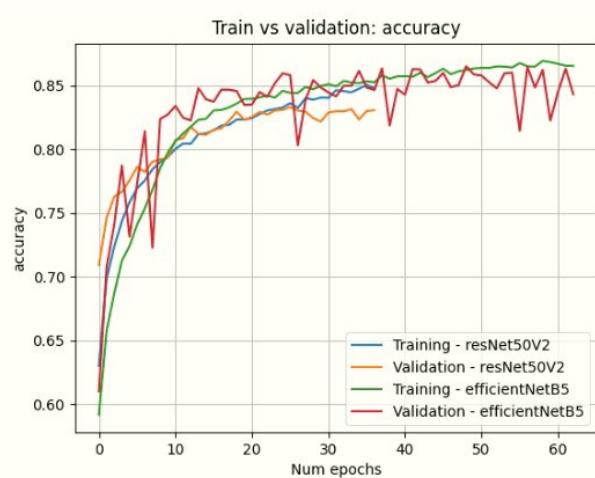
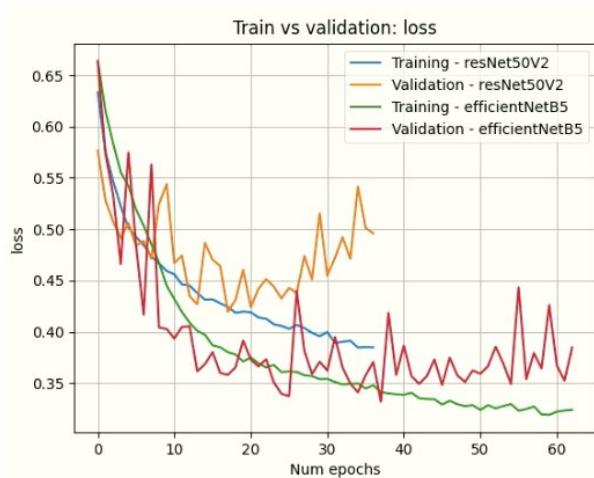
La carga de las imágenes se realiza con funciones propias (lib.dataset.prepare_dataset_new).



- Sin data augmentation
- Sin balanceo en ninguno de los dos datasets
- Sin shuffle
- Batch size = 32 para ambos datasets

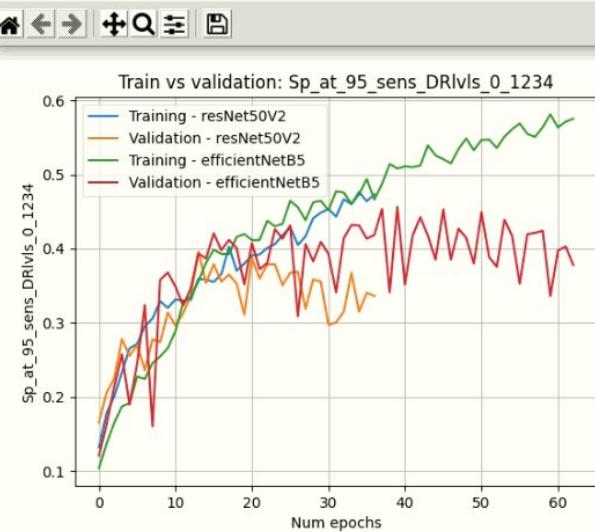
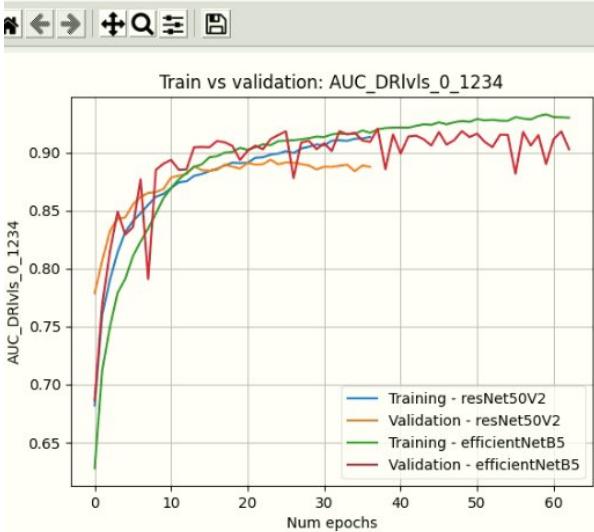
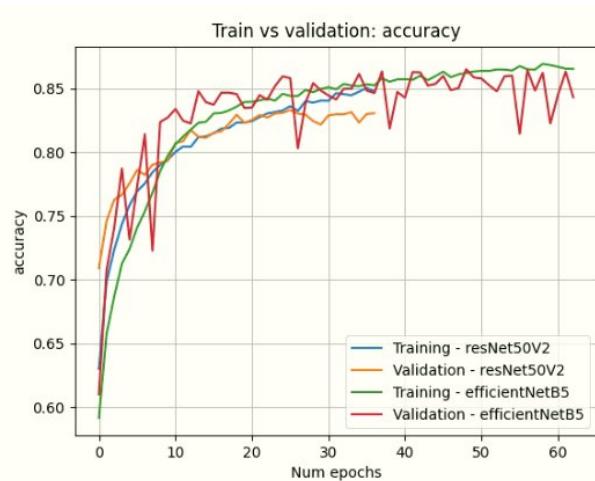
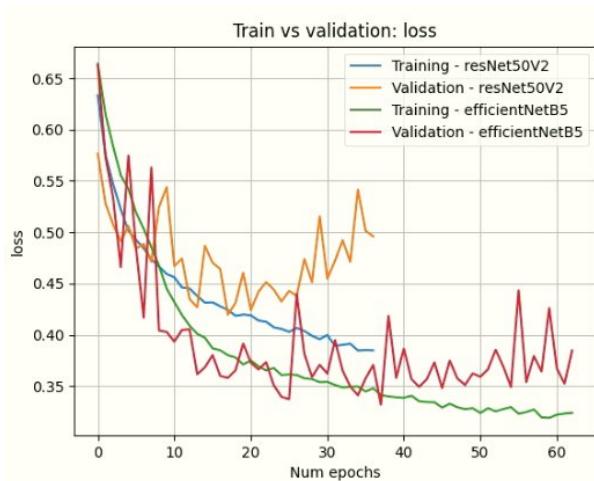
Esta prueba se ha realizado con y sin dropout.

2.4 Gráficas



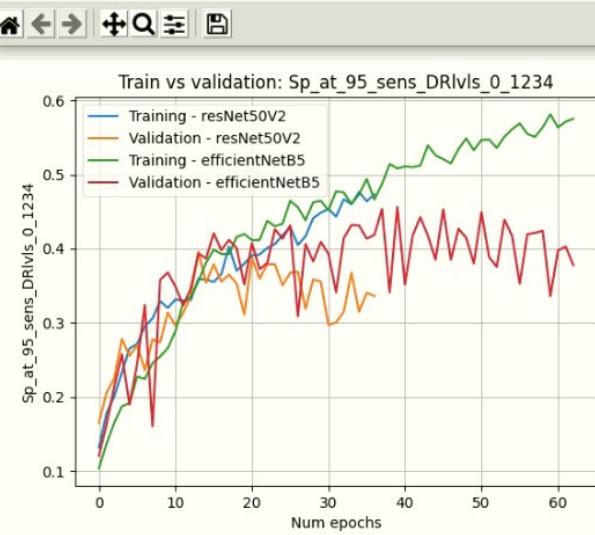
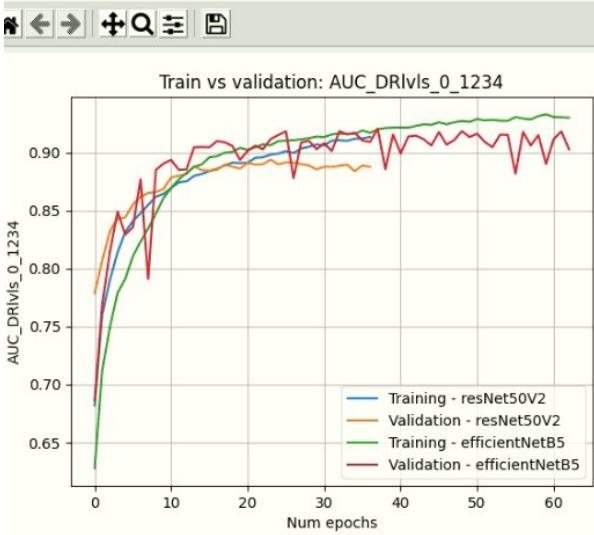
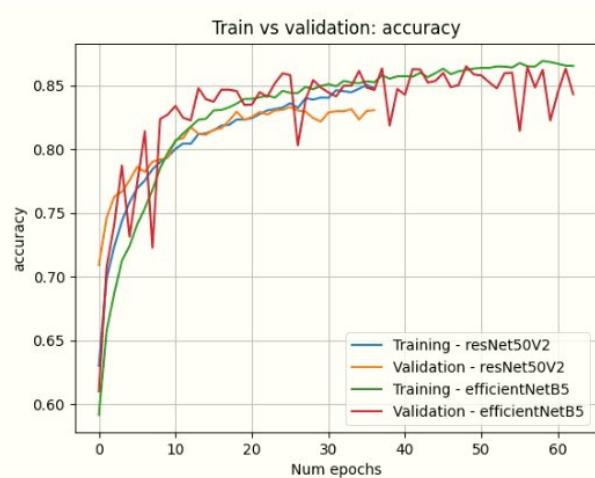
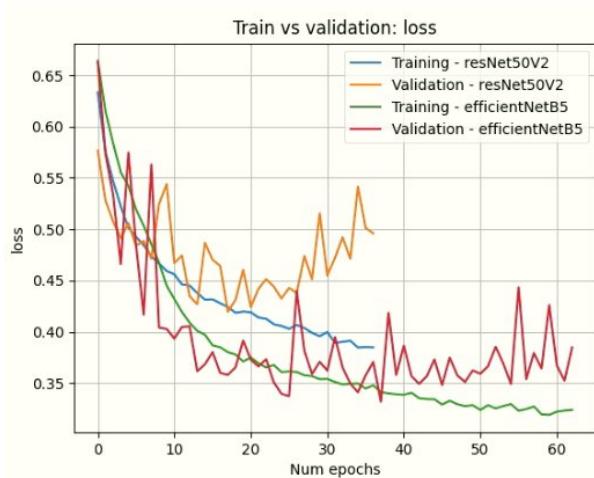
- EYEPACS_Seq_Modelo_basico (sin Dr op)/train
- EYEPACS_Seq_Modelo_basico (sin Dr op)/validation
- EYEPACS_Seq_Modelo_ajustado (sin Dr op)/train
- EYEPACS_Seq_Modelo_ajustado (sin Dr op)/validation





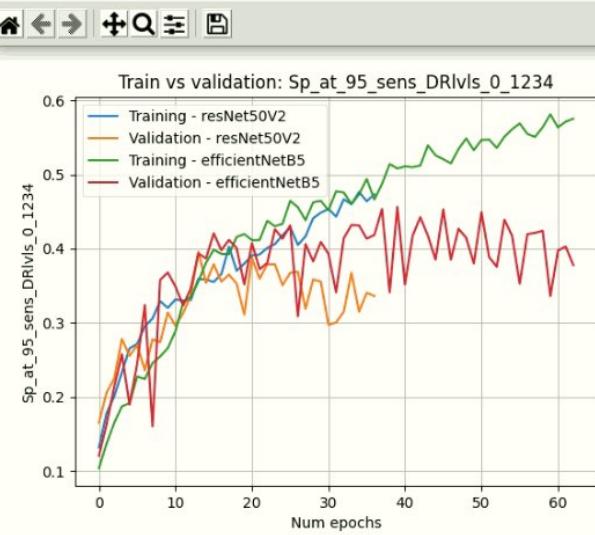
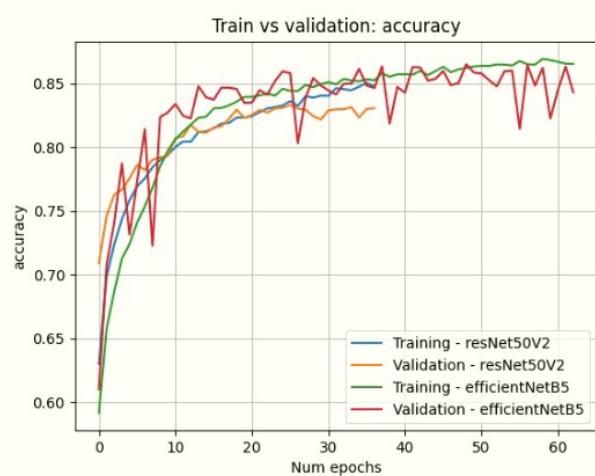
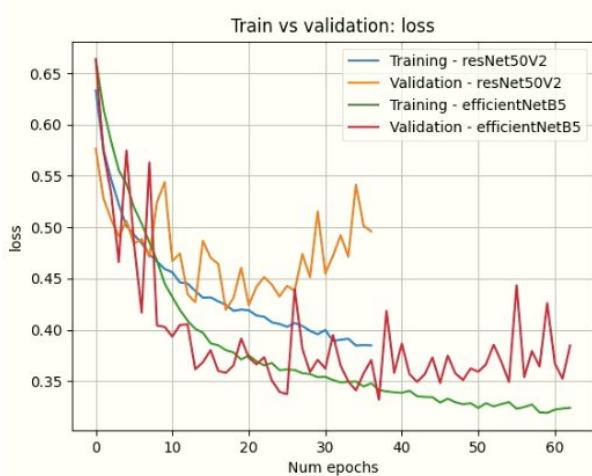
- EYEPACS_Seq_Modelo_propio/train
- EYEPACS_Seq_Modelo_propio/validation
- EYEPACS_Seq_Modelo_basico (sin Dropout)/train
- EYEPACS_Seq_Modelo_basico (sin Dropout)/validation





- EYEPACS_Seq_Modelo_propio/train
- EYEPACS_Seq_Modelo_propio/validation
- EYEPACS_Seq_Modelo_basico (sin Dr op)/train
- EYEPACS_Seq_Modelo_basico (sin Dr op)/validation
- EYEPACS_Seq_Modelo_ajustado (sin Dr op)/train
- EYEPACS_Seq_Modelo_ajustado (sin Dr op)/validation

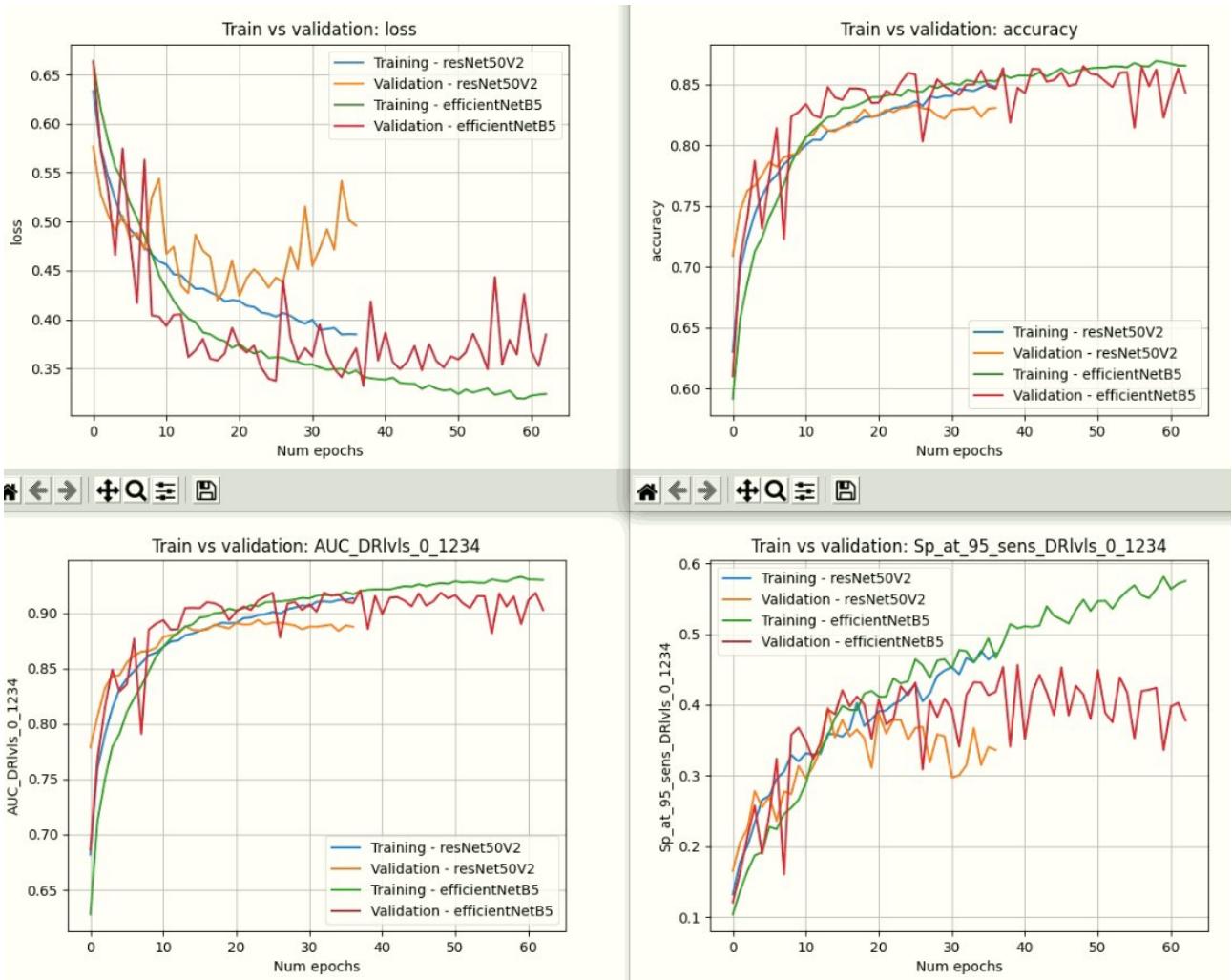




- EYEPACS_Seq_Modelo_propio/train
- EYEPACS_Seq_Modelo_propio/validation
- EYEPACS_Seq_Modelo_propio (sin Drop)/train
- EYEPACS_Seq_Modelo_propio (sin Drop)/validation

epoch_loss





2.5 Modelo_propio_TF+Red_Seq (con Dropout y usando distintos datasets (subconjuntos de EYEPACS))

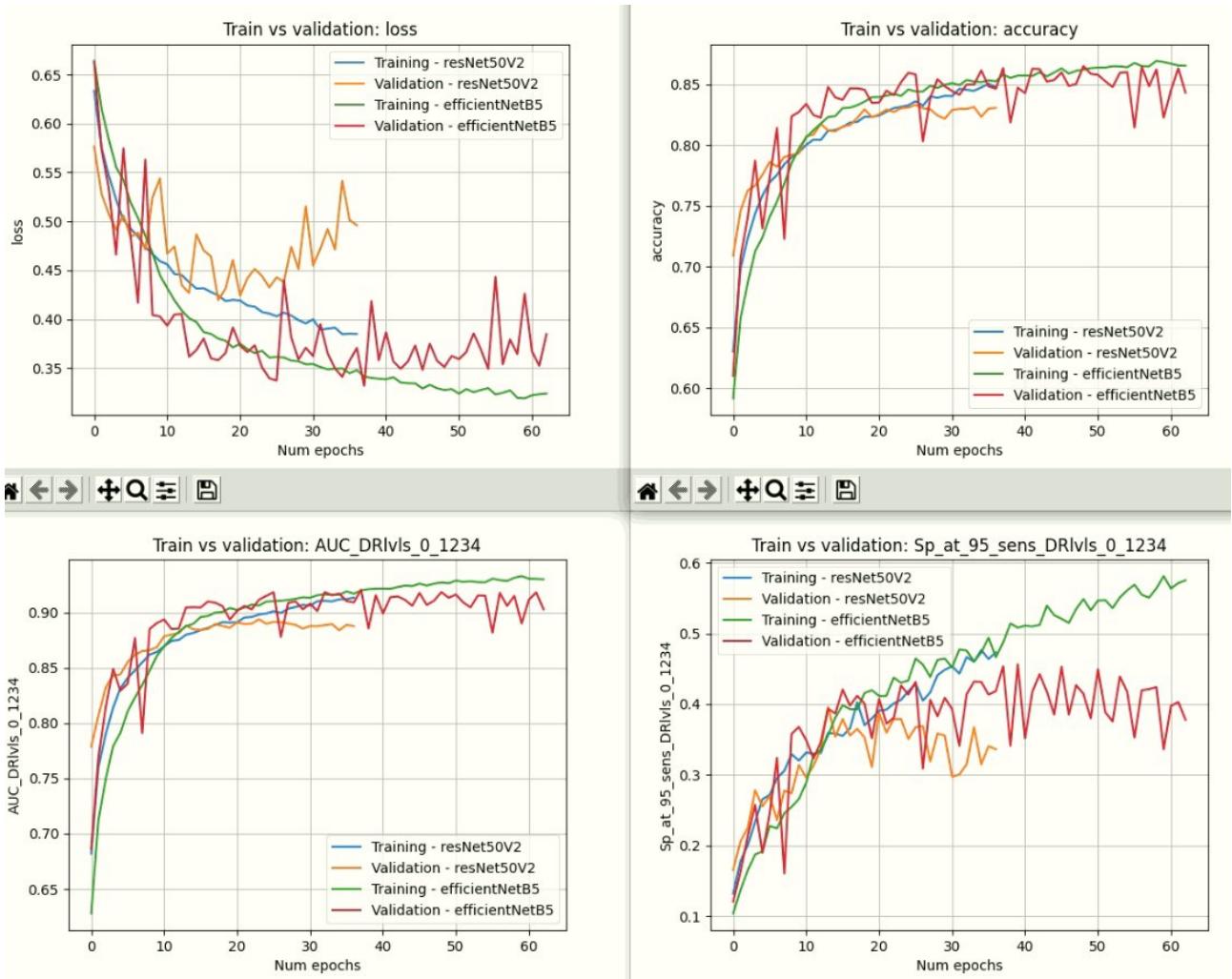
Modelo_propio_TF+Red_EYEPACS_Seq (con Dropout de 0.25) utilizando:

- DATASET-VALIDATION-10.csv (7.103 imágenes) como dataset de entrenamiento
- DATASET-TEST-10.csv (7.102 imágenes) como dataset de validación.

EYEPACS_Seq_Modelo_propio 2/train
 EYEPACS_Seq_Modelo_propio 2/validation



Red_EfficientNetB2



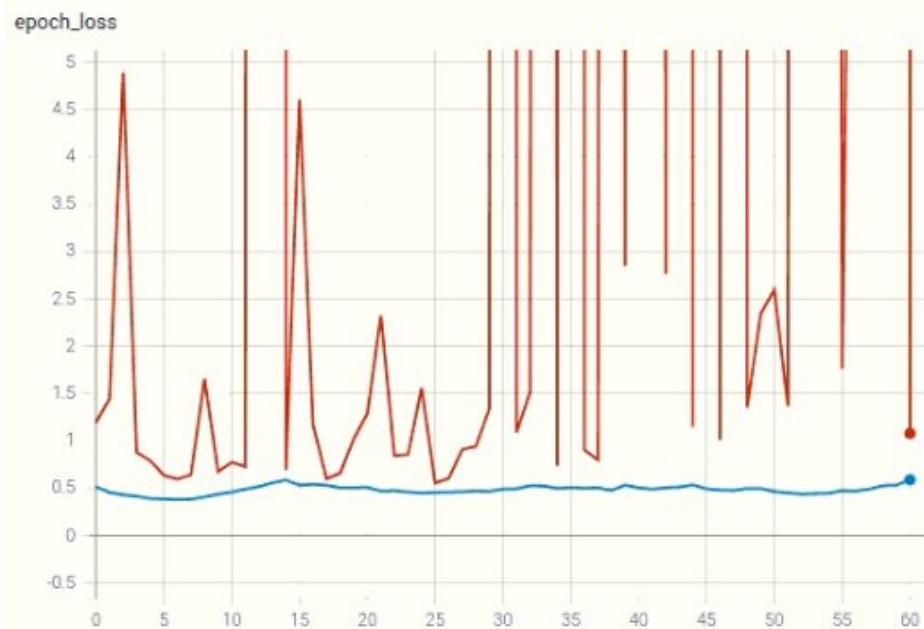
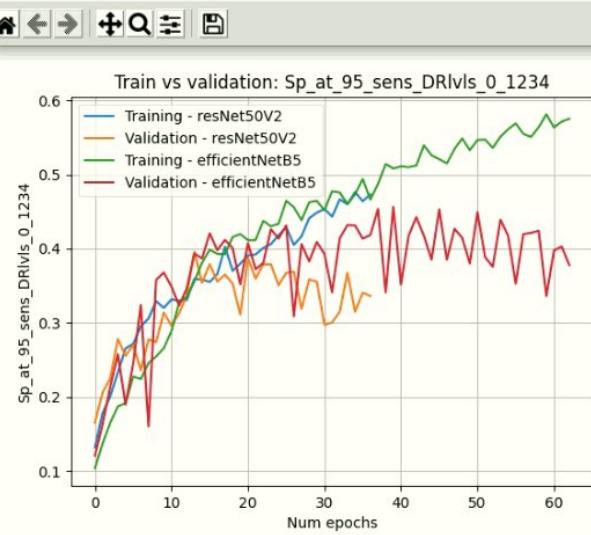
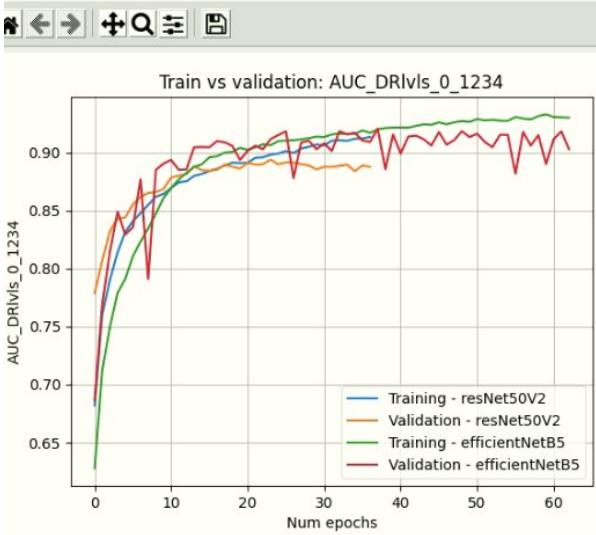
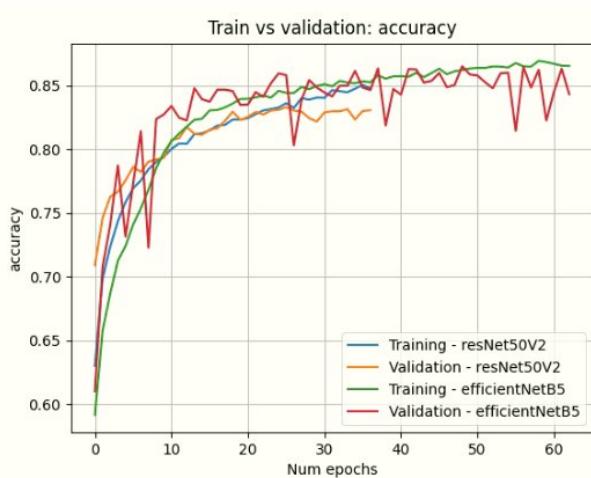
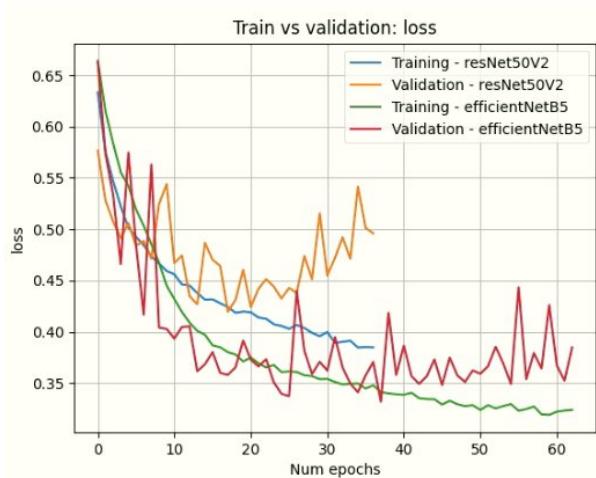
Red declarada utilizando la librería de redes de TensorFlow.

3.1 Modelo_propio + EfficientNetB2 + EYEPACS completo

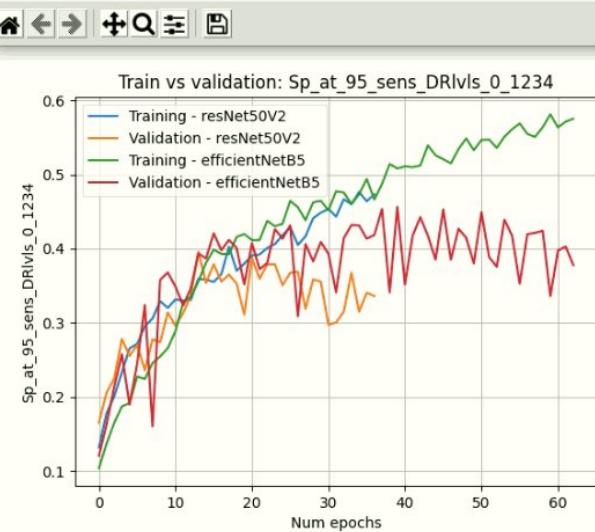
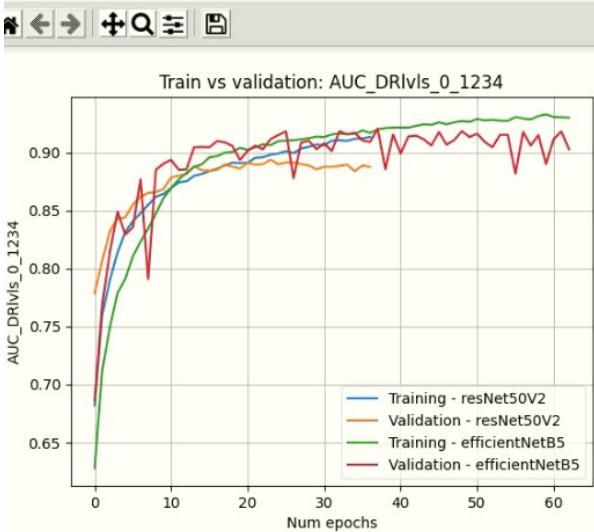
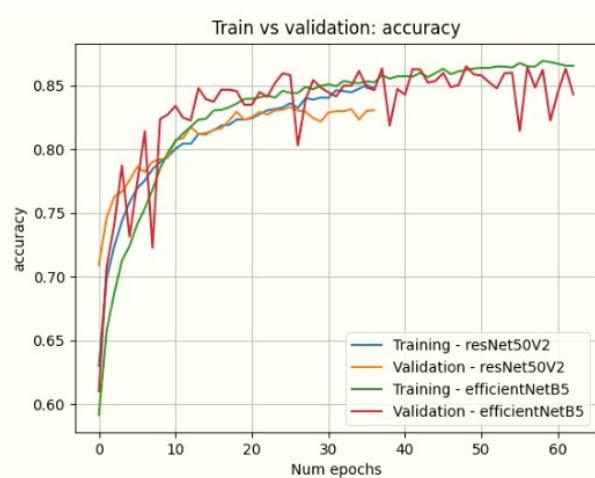
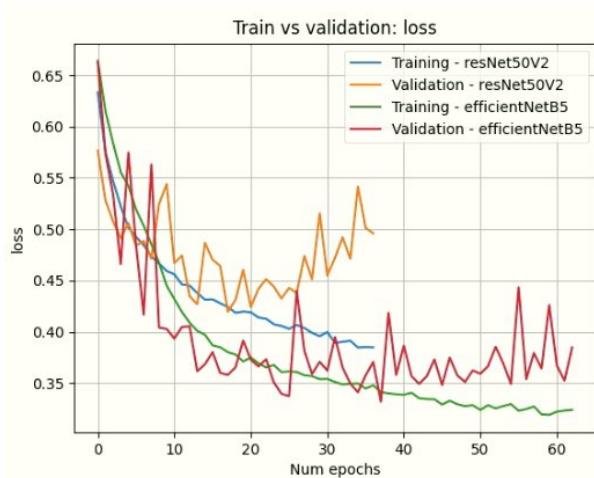
- Dataset de entrenamiento: DATASET-TRAIN-80.csv con aproximadamente 56.000 imágenes.
- Dataset de validación: DATASET-VALIDATION-10.csv con 7.103 imágenes.
- División de clases: 0 vs 1, 2, 3 y 4.

Carga de imágenes realizada haciendo uso de las funciones propias (lib.dataset.create_dataset_new).

- Se aplica data augmentation en dataset de entrenamiento
- Shuffle en el dataset de entrenamiento
- Sin balanceo en ninguno de los dos datasets. Esto hace que aproximadamente el 74% de las imágenes sean de la clase 0 (sano). Ambos datasets tienen la misma distribución.
- Batch size = 4 para ambos datasets.
- Prefetch = 1
- SGD con learning rate = 0.001, momentum = 0.9 y clipnorm = 1.0.



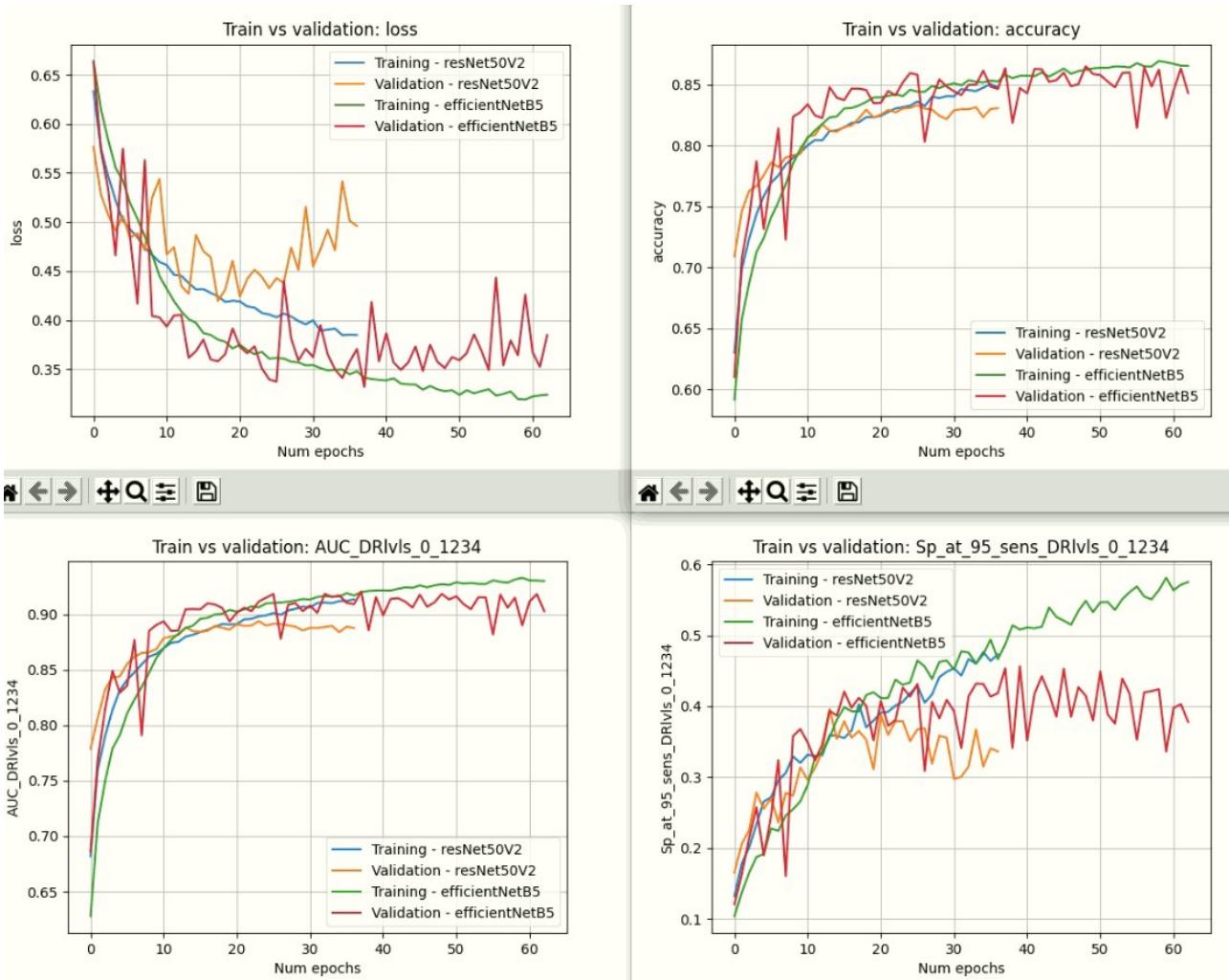
Ajustando la escala en Y:



Data augmentation modificado

El procedimiento de data augmentation ha sido temporalmente modificado debido a su tiempo de ejecución por el siguiente método:

Por cada imagen, se aplicará uno de los siguientes efectos:



- Adición de ruido, que podrá ser uniforme, laplaciano o de poisson, con misma probabilidad,
- Volteos, que podrán ser derecha-izquierda o arriba-abajo, ambos con misma probabilidad,
- Rotación, tomando como valor de giro uno aleatorio definido en su rango,
- Desplazamiento del canal de color (H), tomando como valor uno aleatorio de su rango.

Los siguientes cambios serán utilizando este procedimiento.

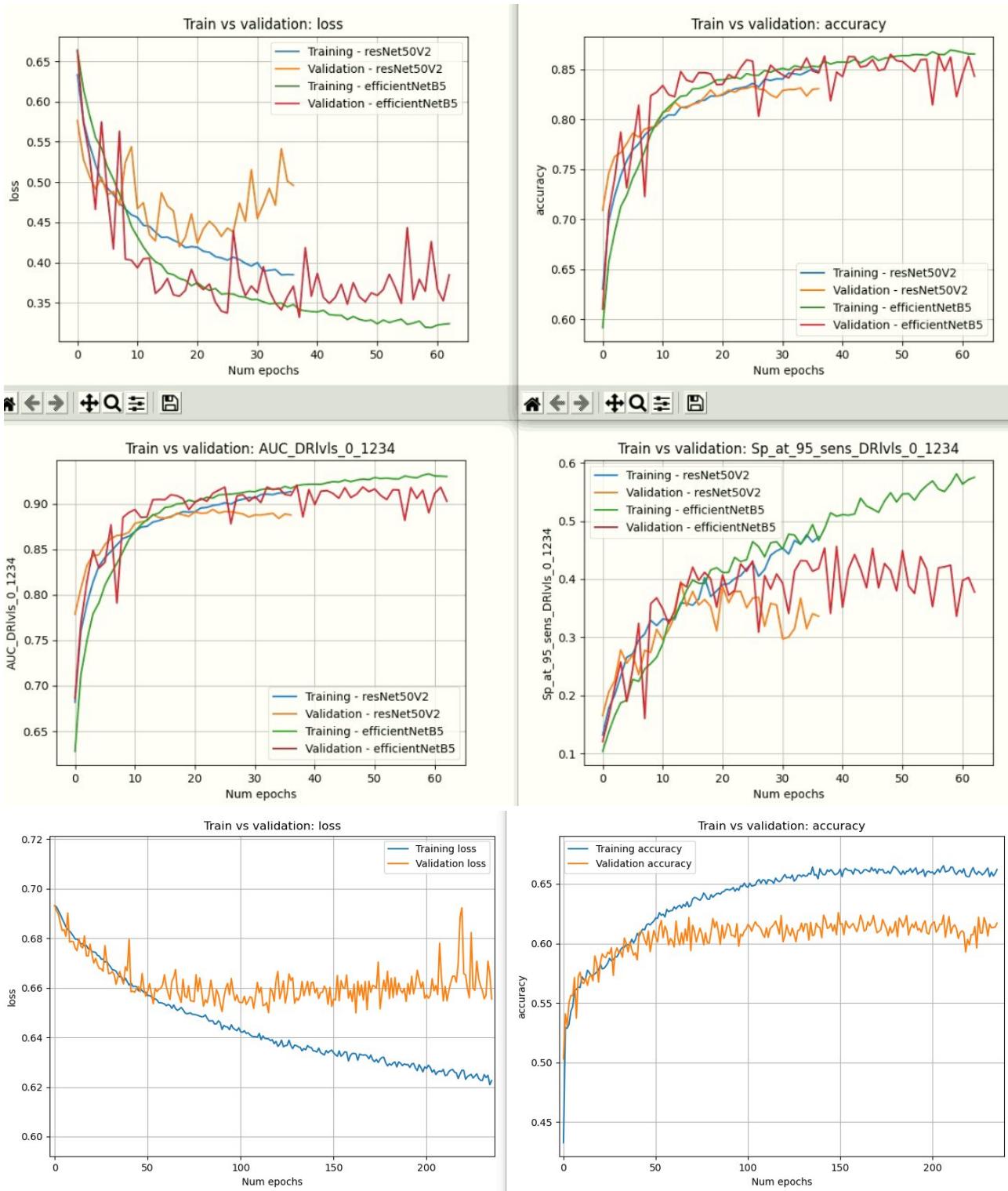
3.2 Modelo_propio + Red_Seq + EYEPACS

Se ha utilizado el dataset completo de entrenamiento de EYEPACS: EYEPACS-TRAIN-80.csv, aplicando únicamente una ponderación a cada clase a la hora de ser evaluada por la loss function.

La ponderación ha sido: clase 0 se pondrá con 0.6741; la clase 1 se pondrá con 1.9363. Estos índices fueron calculados con la función sklearn.utils.class_weight.compute_class_weight.

Para validar, se ha empleado un fichero csv balanceado, para que haya mismos elementos de cada clase, y éstos no varíen en ningún momento: EYEPACS-VALIDATION-10_BALANCED.csv.

- Batch size = 12
- Se aplica data augmentation y shuffle en el dataset de entrenamiento
- Salida en formato categorical y no one-hot (necesario para aplicar el ponderado de clases)
- Sparse_categorical_crossentropy como loss function
- SGD con learning rate = 0.001, momentum = 0.9 y clipnorm = 1.0



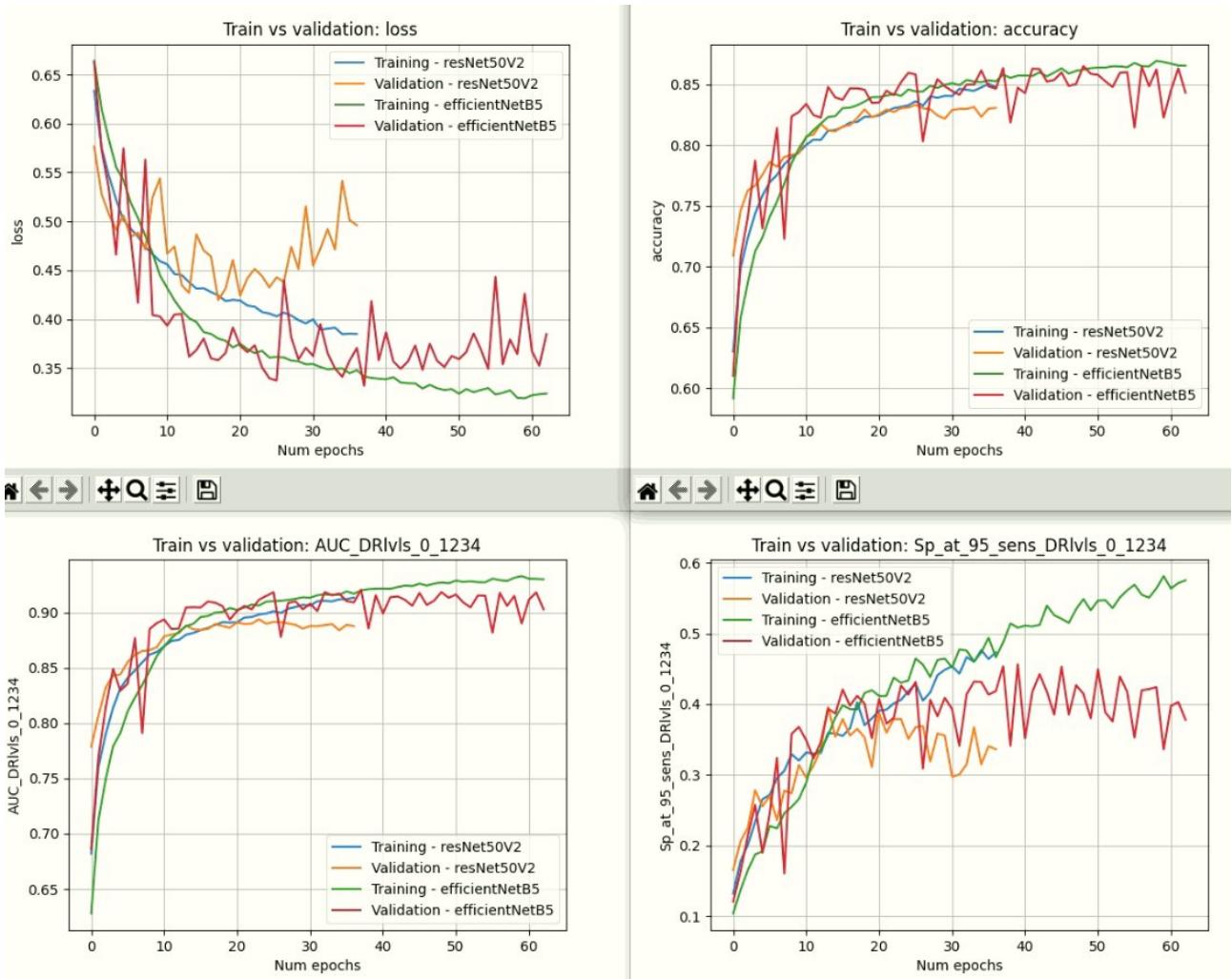
4.1 Modelo_propio + InceptionResNetV2 + EYEPACS

Se ha utilizado el dataset completo de entrenamiento de EYEPACS: EYEPACS-TRAIN-80.csv, aplicando únicamente una ponderación a cada clase a la hora de ser evaluada por la loss function.

La ponderación ha sido: clase 0 se pondrá con 0.6741; la clase 1 se pondrá con 1.9363. Estos índices fueron calculados con la función `sklearn.utils.class_weight.compute_class_weight`.

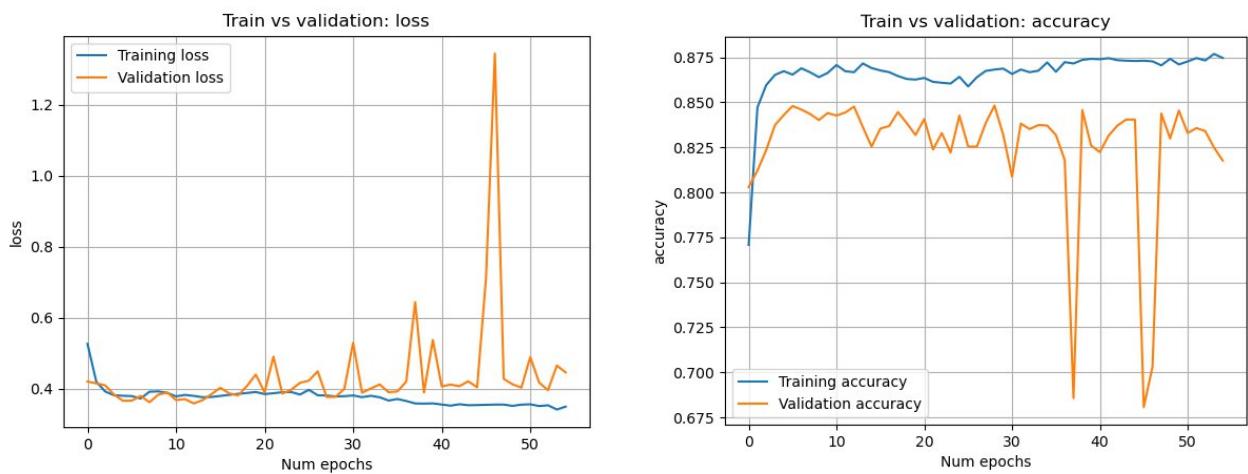
Para validar, se ha empleado un fichero csv balanceado, para que haya mismos elementos de cada clase, y éstos no varíen en ningún momento: EYEPACS-VALIDATION-10_BALANCED.csv.

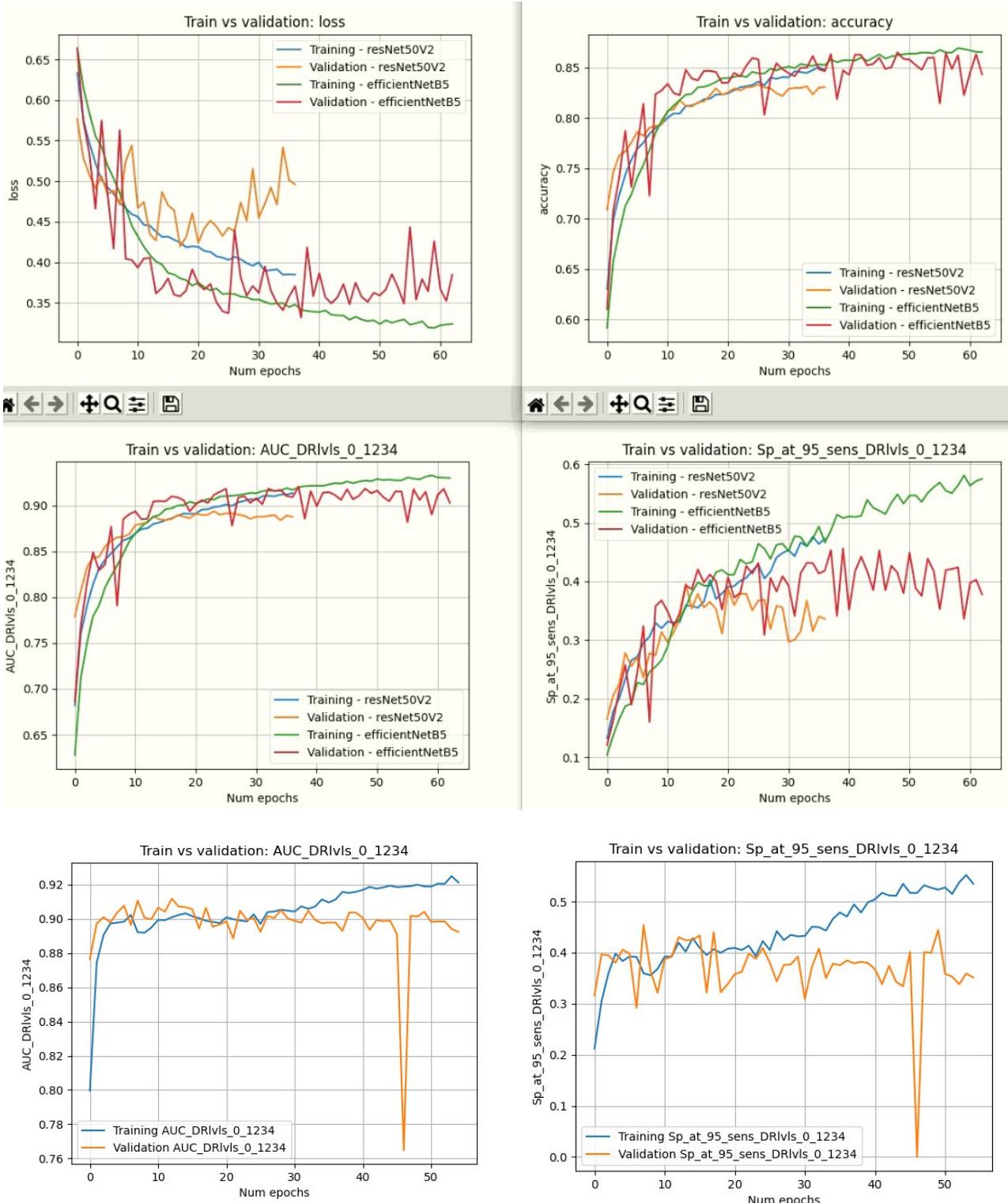
- Batch size = 12
- Se aplica data augmentation y shuffle en el dataset de entrenamiento



- Salida en formato categorical y no one-hot (necesario para aplicar el ponderado de clases)
- Sparse_categorical_crossentropy como loss function
- SGD con learning rate = 0.001, momentum = 0.9 y clipnorm = 1.0

Fichero: prueba12_MPropio_IncepResNetV2_EYEPACS.ipynb



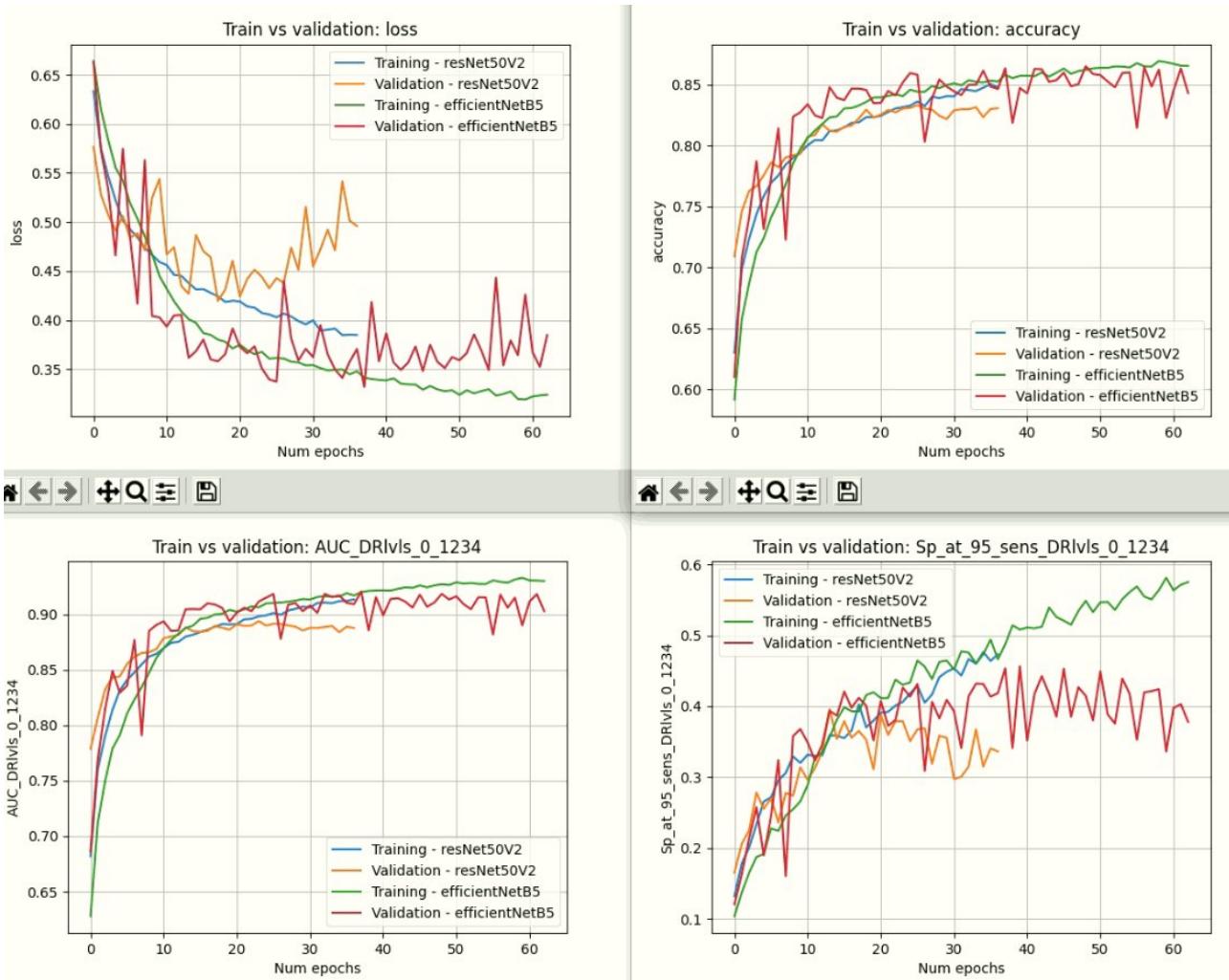


Data augmentation modificado de nuevo

El procedimiento de data augmentation ha sido temporalmente modificado debido a su tiempo de ejecución por el siguiente método:

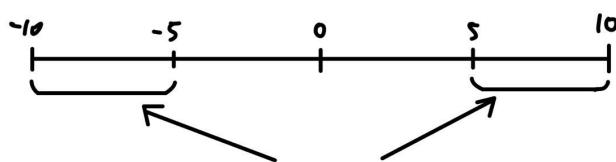
Por cada imagen, se aplicará uno de los siguientes efectos:

- Adición de ruido, que podrá ser uniforme, laplaciano o de poisson, con misma probabilidad,
- Volteos, que podrán ser derecha-izquierda o arriba-abajo, ambos con misma probabilidad,
- Rotación, tomando como valor de giro uno aleatorio definido en su rango,
- Desplazamiento del canal de color (H), tomando como valor uno aleatorio de su rango.



- Se aplica al final una operación Jigsaw, partiendo la imagen aleatoriamente en dos filas y en dos columnas de tamaños no definidos, intercambiándose entre sí.

Los rangos empleados por cada una de las funciones transformadoras no serán continuos, es decir, estarán divididos en valores positivos y negativos, evitando los valores muy cercanos a 0 y que apenas modificarían la imagen. Por cada imagen se tomará o el rango positivo o el rango negativo:



Los siguientes cambios serán utilizando este procedimiento.

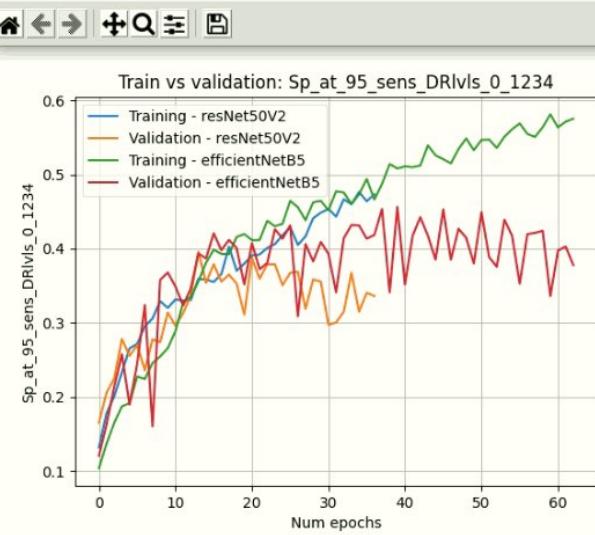
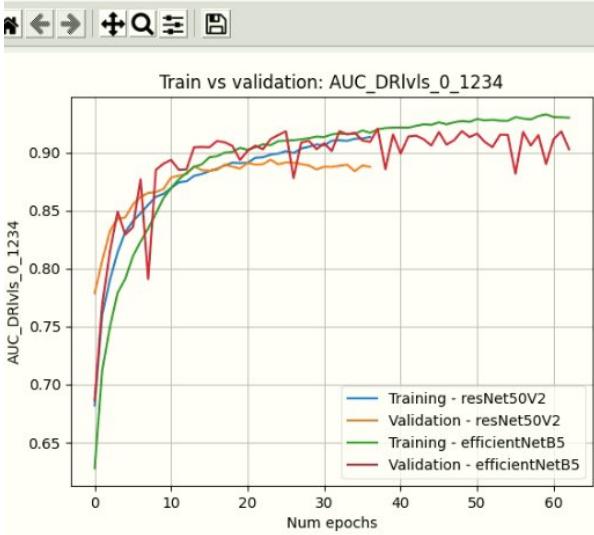
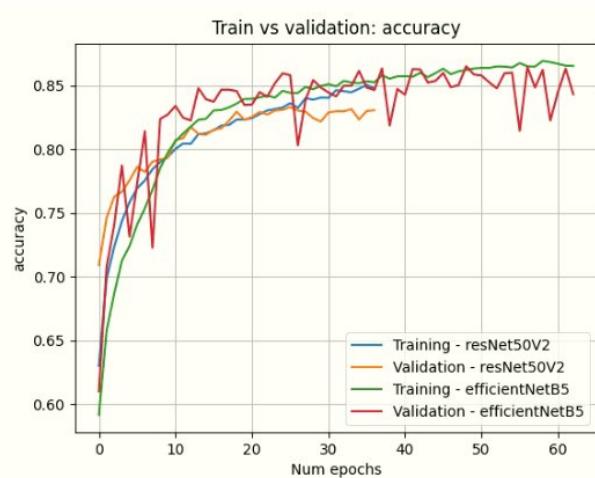
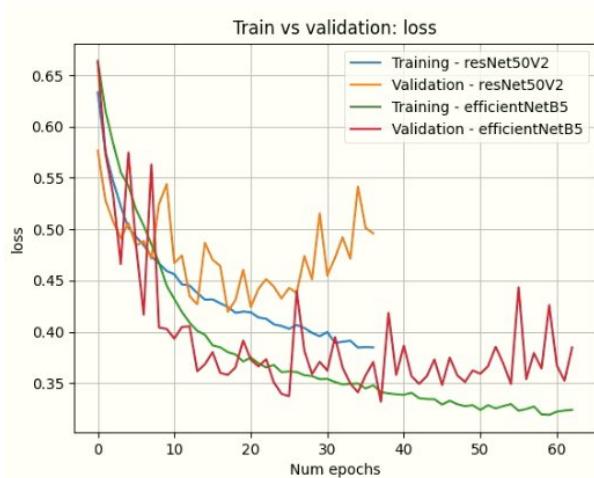
4.2 Modelo_propio + InceptionResNetV2 + EYEPACS + Decremento LR

Se ha utilizado el dataset completo de entrenamiento de EYEPACS: EYEPACS-TRAIN-80.csv, aplicando únicamente una ponderación a cada clase a la hora de ser evaluada por la loss function.

La ponderación ha sido: clase 0 se pondrá con 0.6741; la clase 1 se pondrá con 1.9363. Estos índices fueron calculados con la función sklearn.utils.class_weight.compute_class_weight.

Para validar, se ha empleado un fichero csv balanceado, para que haya mismos elementos de cada clase, y éstos no varíen en ningún momento: EYEPACS-VALIDATION-10_BALANCED.csv.

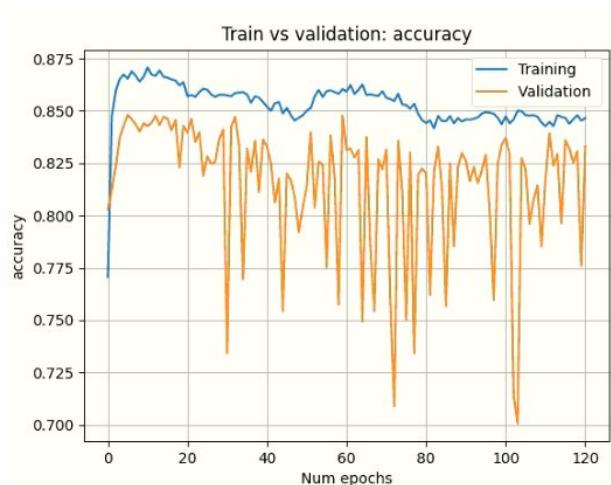
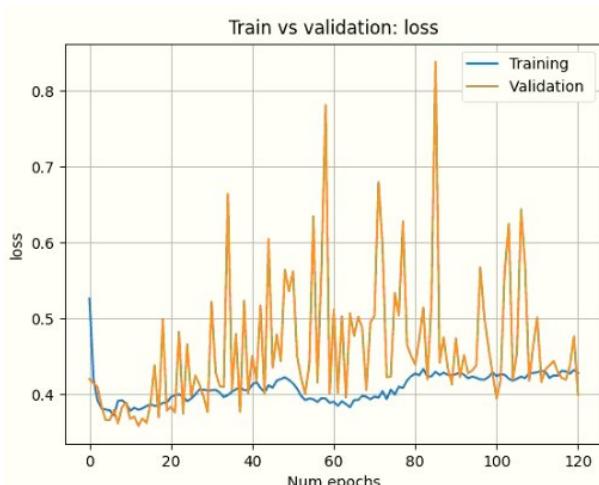
- El modelo de partida es el que mejor valor de AUC obtuvo en validación en el entrenamiento anterior.

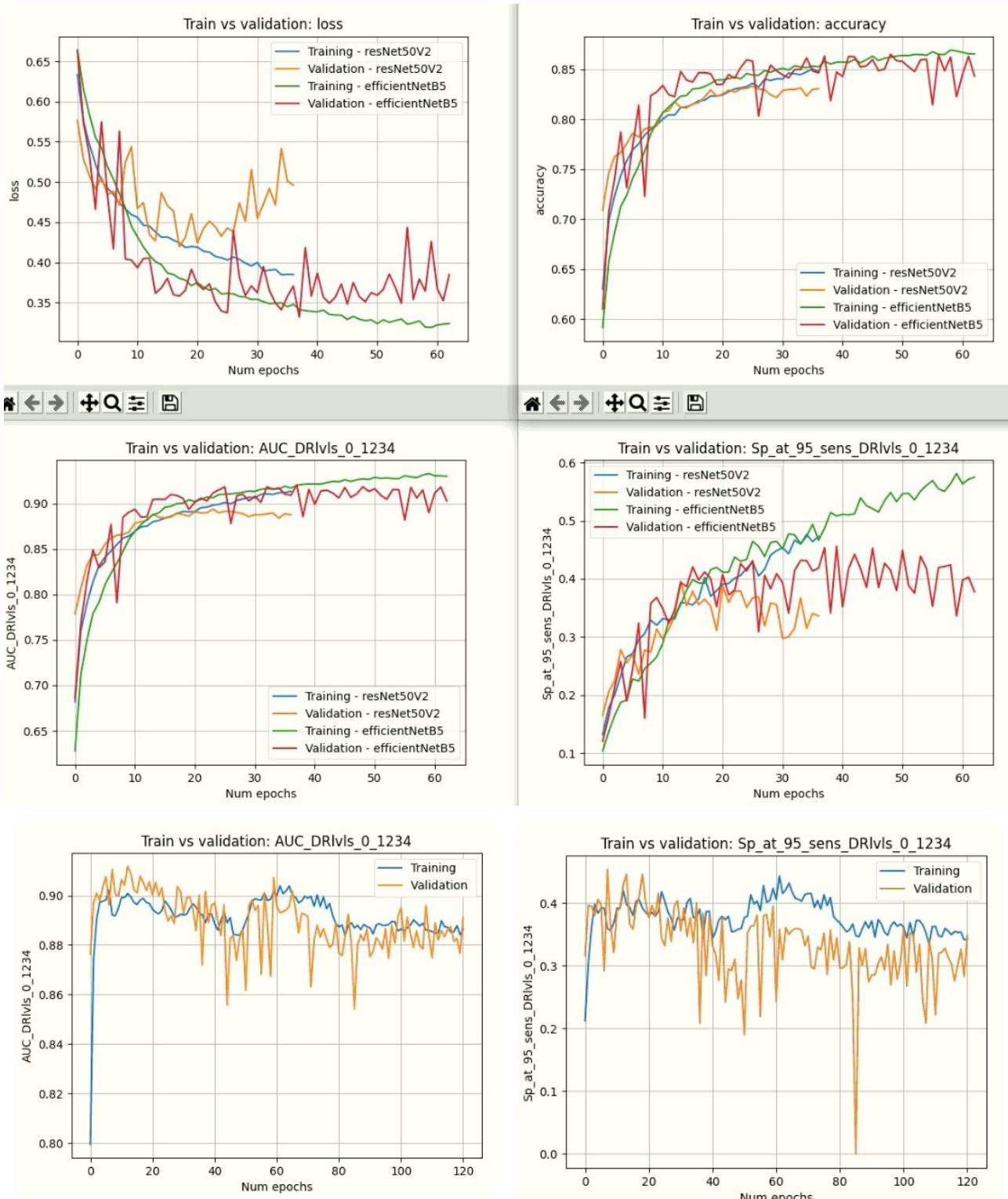


- Batch size = 12
- Se aplica data augmentation y shuffle en el dataset de entrenamiento
- Salida en formato categorical y no one-hot (necesario para aplicar la ponderación de clases)
- Sparse_categorical_crossentropy como loss function
- SGD con learning rate = 0.001, momentum = 0.9 y clipnorm = 1.0
- El learning rate se reduce cada 10 épocas con un factor de 0.9 (cada 10 épocas se multiplica por 0.9)

Fichero: prueba14_MPropio_IncepResNetV2_LrSch_EYEPACS.ipynb

Gráficas agregando 13 épocas de un primer entrenamiento (4.1):

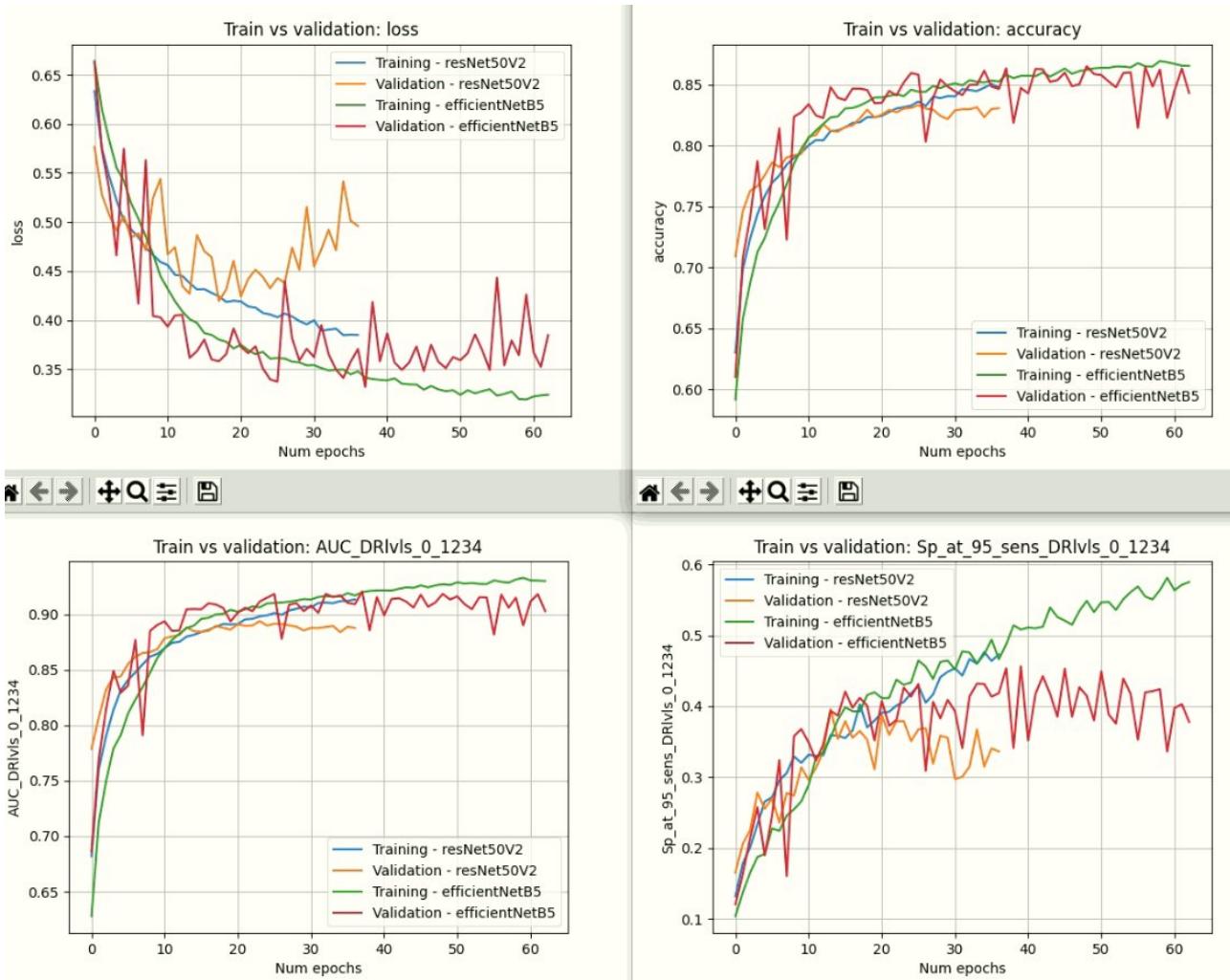




Se ha probado, con el objetivo de conocer si el data augmentation modifica los resultados y tal vez mejora las imágenes de entrenamiento (no se ha comprobado con el data augmentation que usa las funciones de contraste sobre HSI).

Se ha aplicado data augmentation sobre el dataset de validacion (con y sin Jigsaw), y no ha variado ni un solo decimal. Ver salidas guardadas en fichero `prueba14_MPropio_IncepResNetV2_LrSch_EYEPACS.ipynb`

4.3 Modelo_propio + InceptionResNetV2 + EYEPACS 0, 1, 234



Se ha utilizado el dataset completo de entrenamiento de EYEPACS: EYEPACS-TRAIN-80.csv, aplicando únicamente una ponderación a cada clase a la hora de ser evaluada por la loss function.

Las clases se han dividido en:

- 0: DR level 0
- 1: DR level 1
- 2: DR level 2, 3, 4

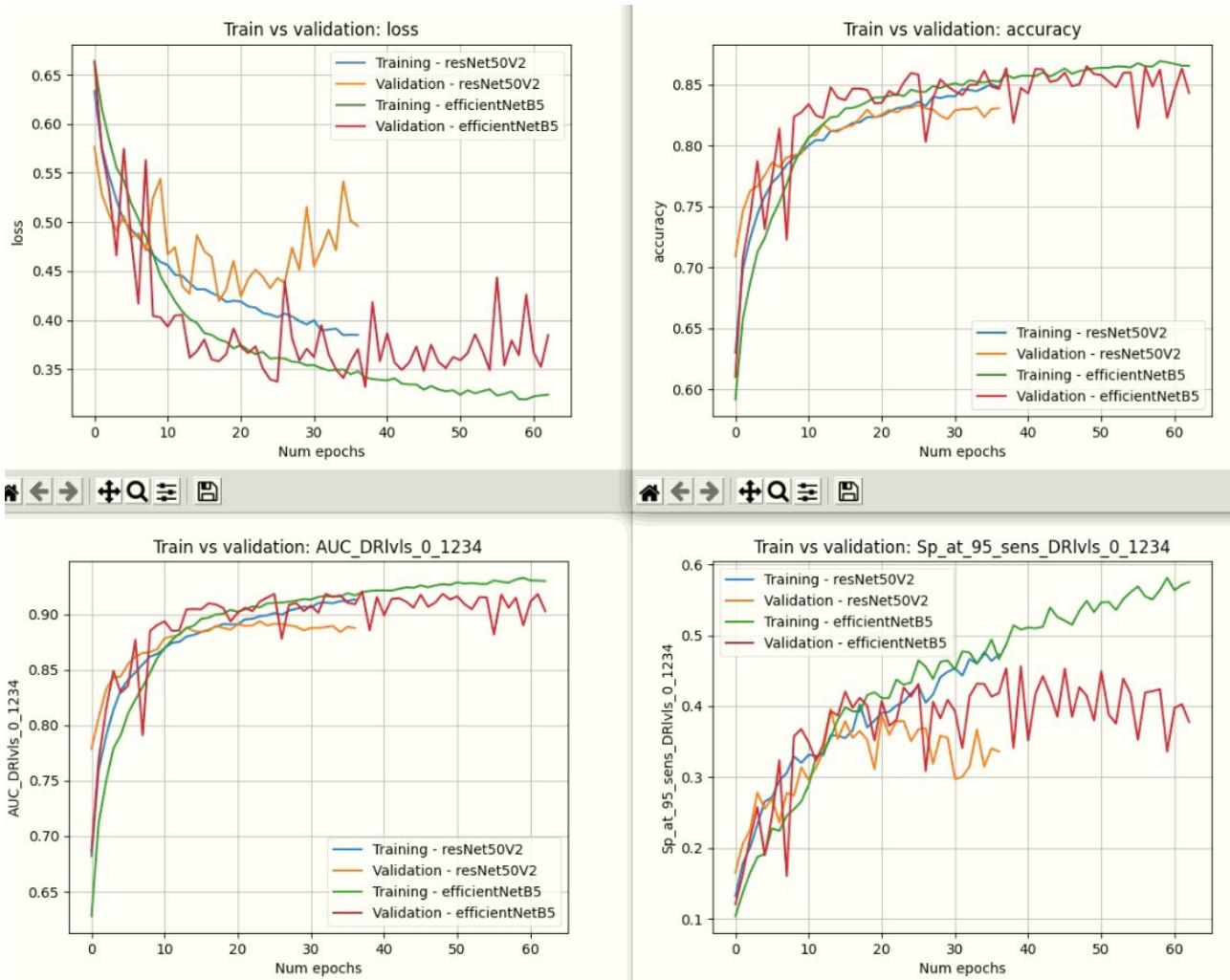
La ponderación ha sido:

- La clase 0 se pondera con: 0.4493740460430062
- La clase 1 se pondera con: 4.677862846793447
- La clase 2 se pondera con: 1.7828187751004017

Estos índices fueron calculados con la función `sklearn.utils.class_weight.compute_class_weight`.

Para validar, se ha empleado un fichero csv balanceado, para que haya mismos elementos de cada clase, y éstos no varíen en ningún momento: VALIDATION_BALANCED_0_1_234.csv. Este fichero se compone de únicamente 1.566 imágenes, 522 por cada clase.

- El modelo de partida será aquel obtenido con una inicialización en ImageNet.
- Batch size = 12
- Se aplica data augmentation y shuffle en el dataset de entrenamiento
- Salida en formato categorical y no one-hot (necesario para aplicar la ponderación de clases)
- Sparse_categorical_crossentropy como loss function
- RMSProp con learning rate inicial = 0.001, rho/decay = 4e-5 y clipnorm = 1
- El learning rate se reduce cada 10 épocas con un factor de 0.9 (cada 10 épocas se multiplica por 0.9)



Fichero: prueba15_Modelo_prop_IncepResNetV2_EYEPACS.py

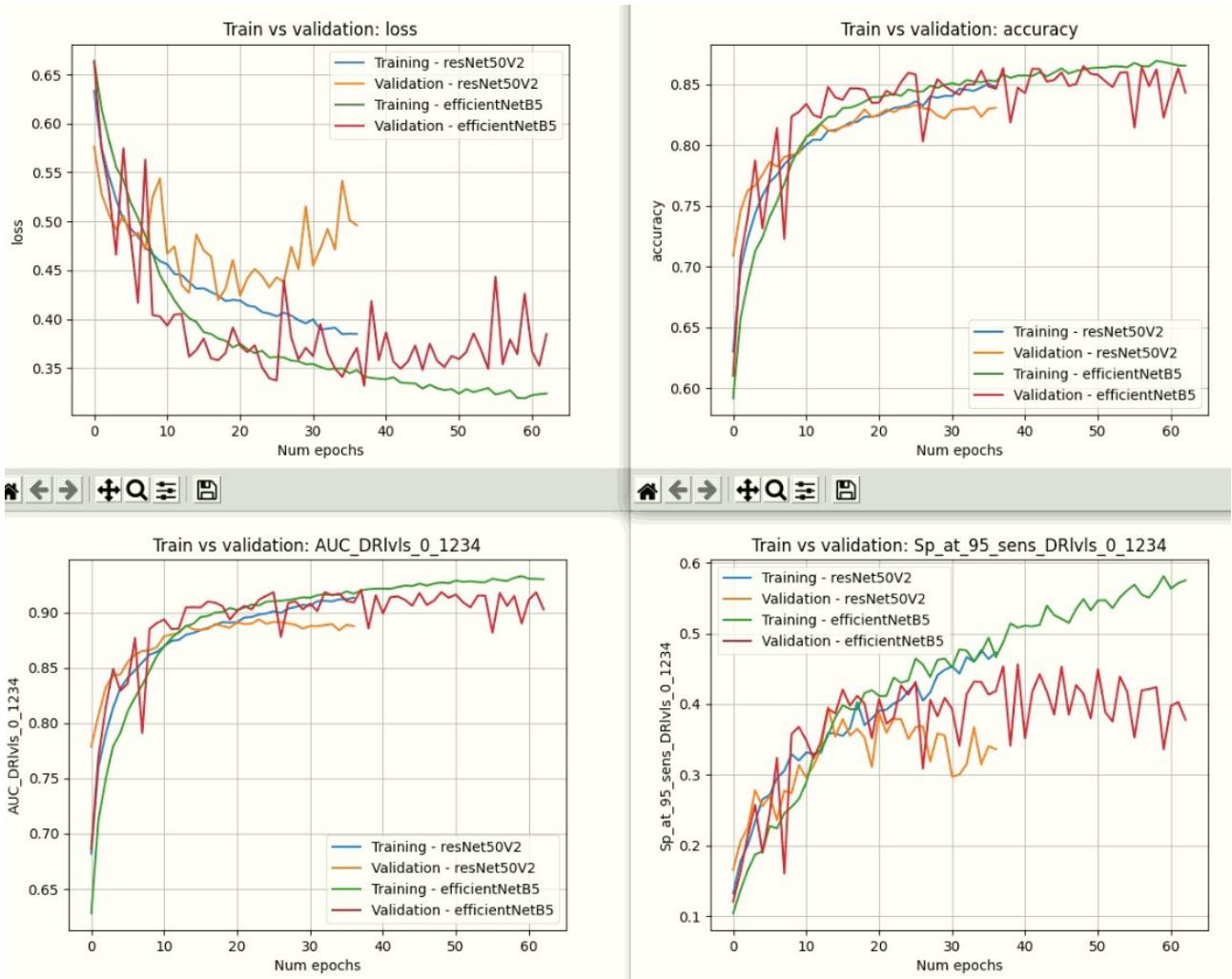
No ejecutado

5.1 Modelo_propio + ResNet50V2 + EYEPACS 0 1234 (ponderación de clases)

Se ha utilizado el dataset completo de entrenamiento de EYEPACS: EYEPACS-TRAIN-80.csv, aplicando únicamente una ponderación a cada clase a la hora de ser evaluada por la loss function.

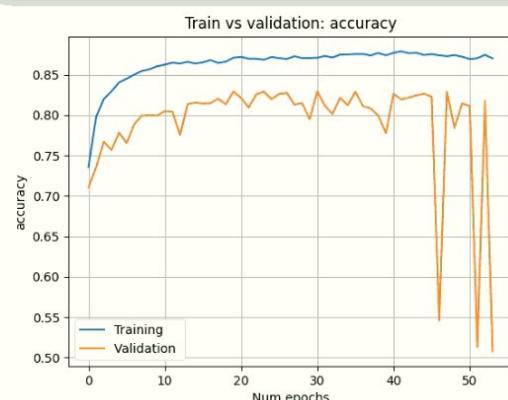
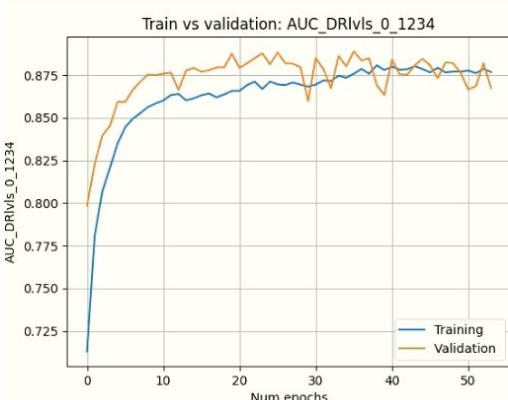
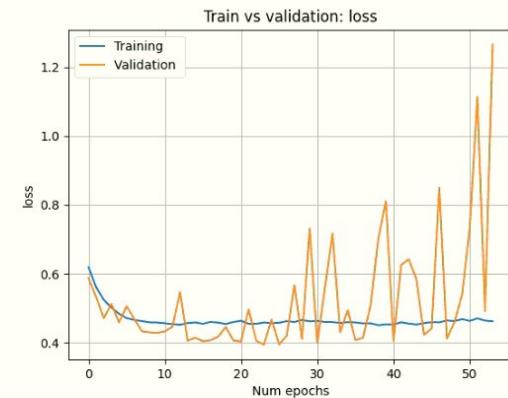
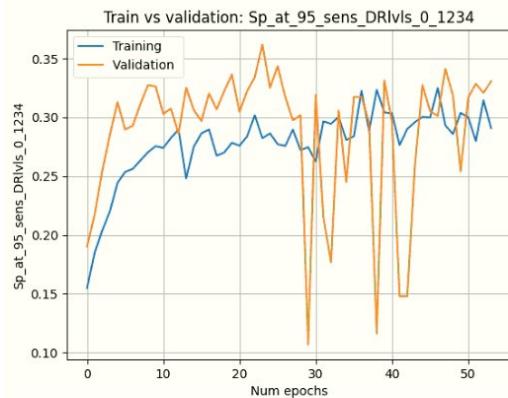
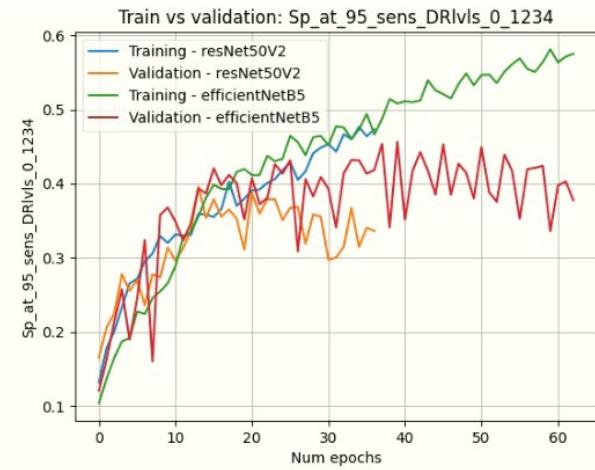
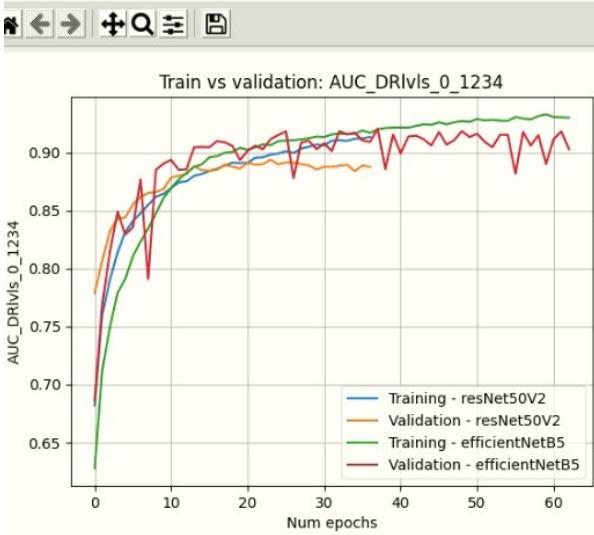
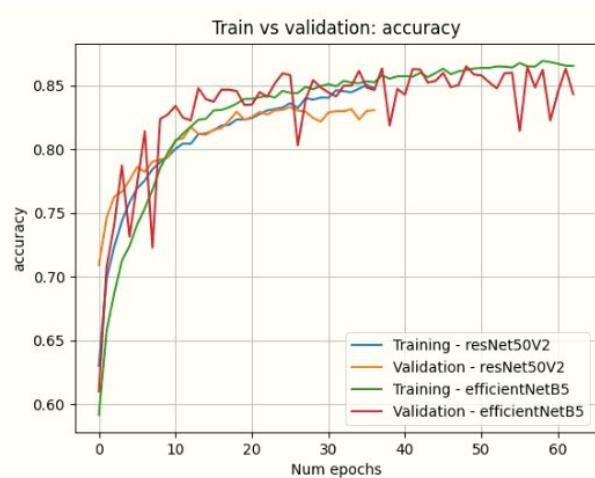
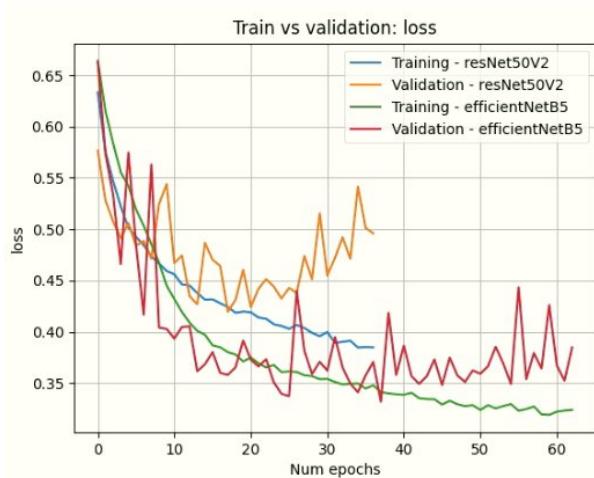
La ponderación ha sido: clase 0 se pondrá con 0.6741; la clase 1 se pondrá con 1.9363. Estos índices fueron calculados con la función sklearn.utils.class_weight.compute_class_weight.

Para validar, se ha empleado un fichero csv balanceado, para que haya mismos elementos de cada clase, y éstos no varíen en ningún momento: EYEPACS-VALIDATION-10_BALANCED.csv.

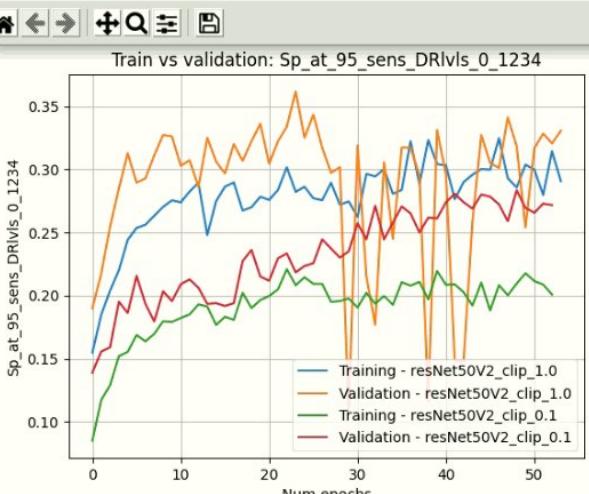
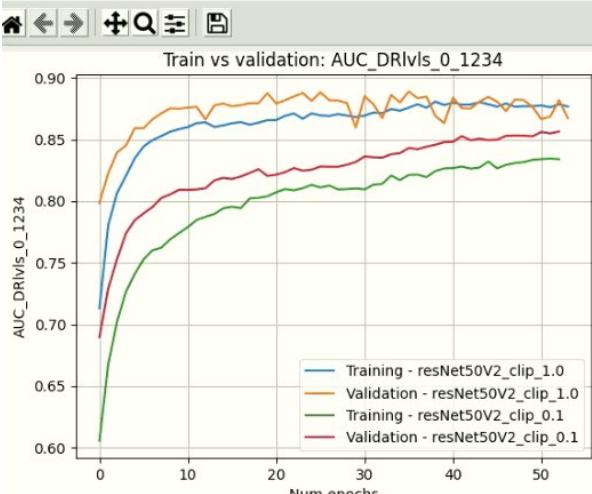
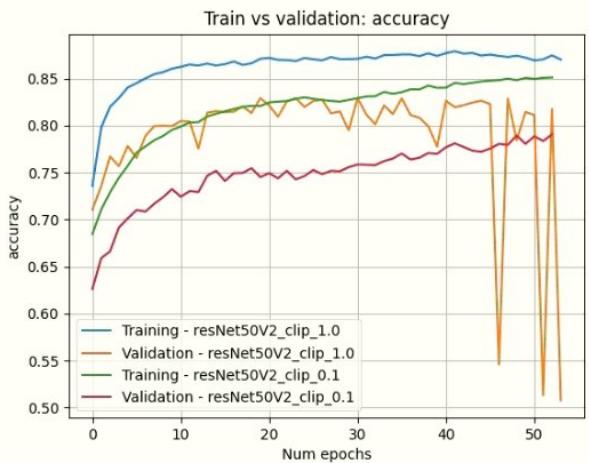
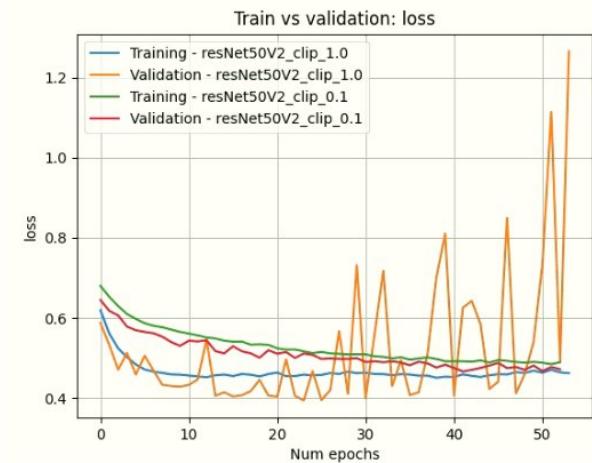
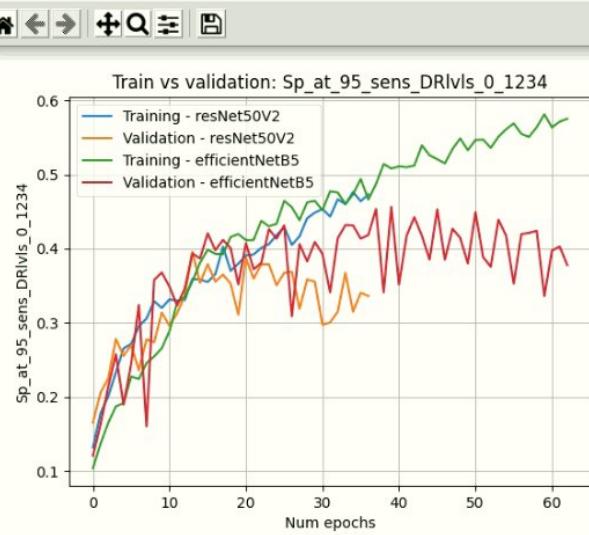
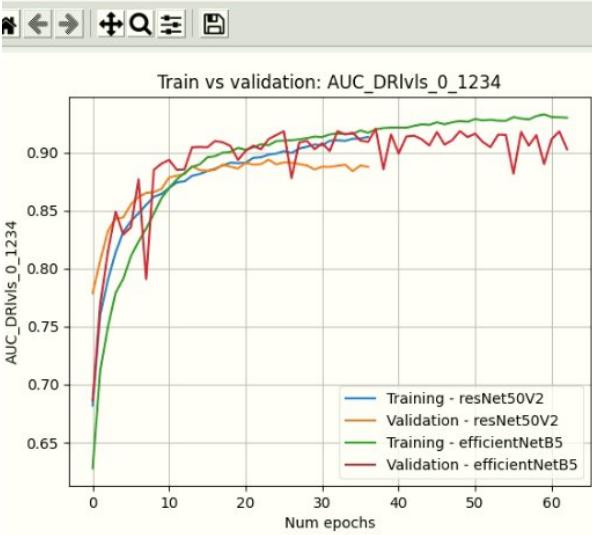
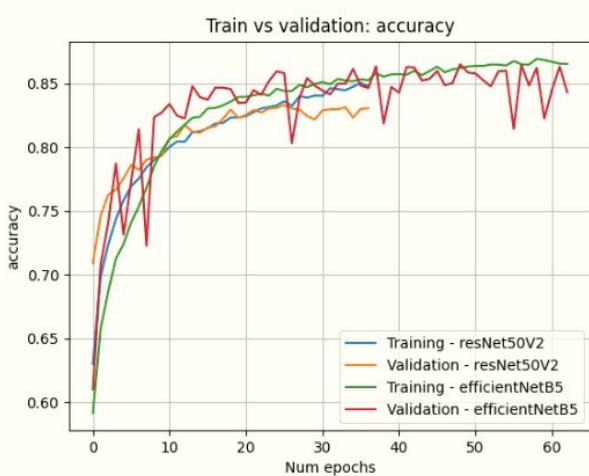
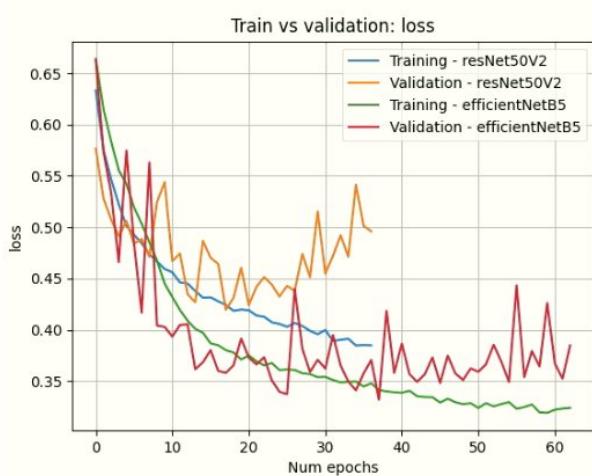


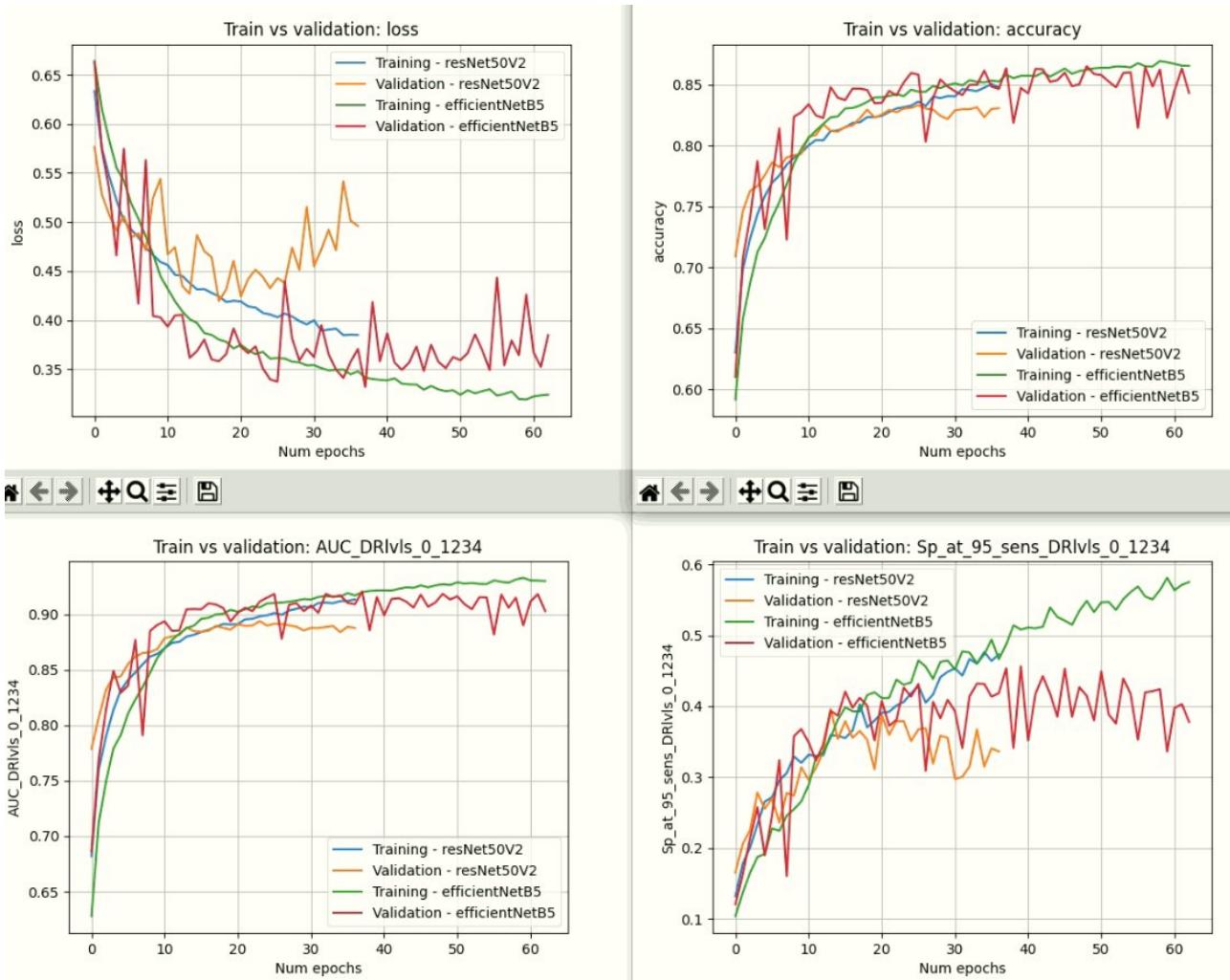
- Batch size = 6 para train, 12 para validation
- Se aplica data augmentation y shuffle en el dataset de entrenamiento
- Salida en formato categorical y no one-hot (necesario para aplicar el ponderado de clases)
- Sparse_categorical_crossentropy como loss function
- SGD con learning rate = 0.0001, momentum = 0.9 y clipnorm = 1.0

Fichero: prueba16_MPropio_ResNet50V2_EYEPACS.ipynb



La siguiente imagen es con los mismos parámetros, pero estableciendo clipnorm a 0.1:





5.2 Modelo_propio + ResNet50V2 + EYEPACS 0 1234 con balanceo

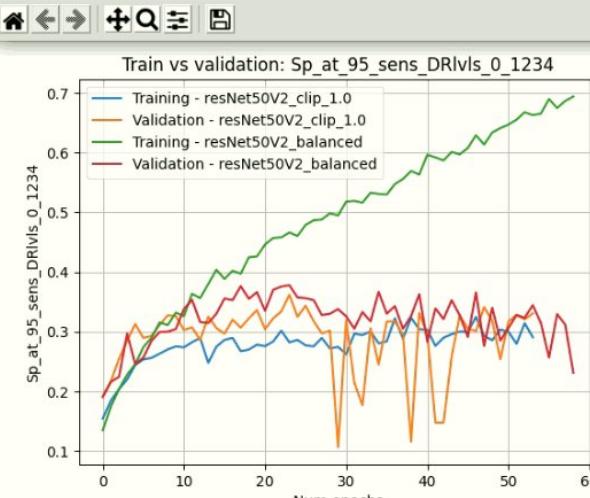
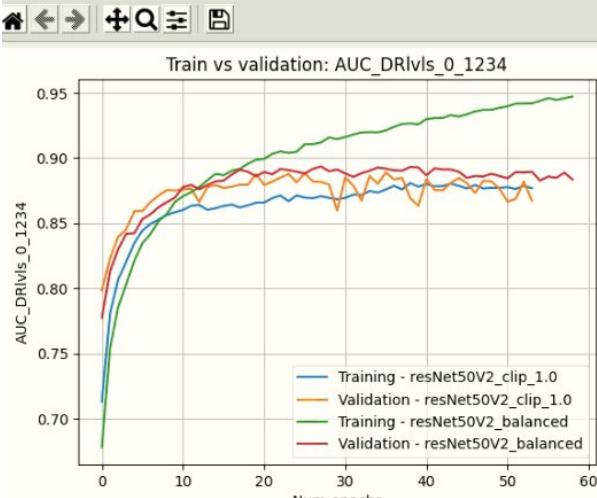
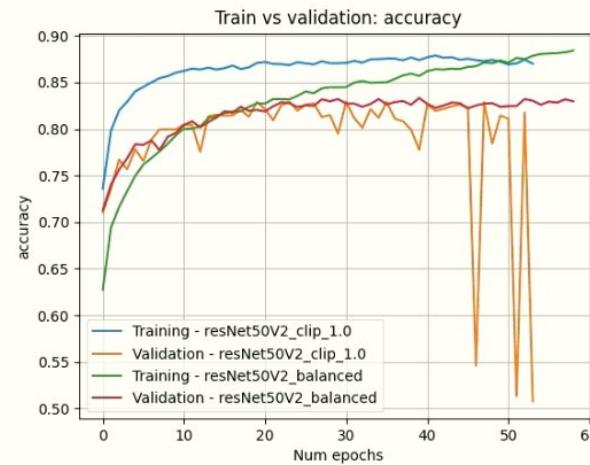
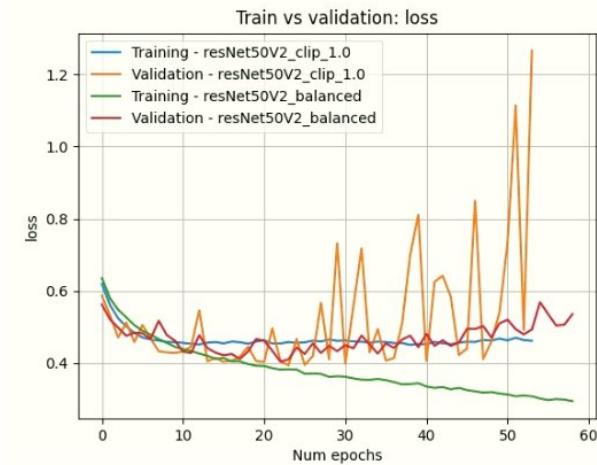
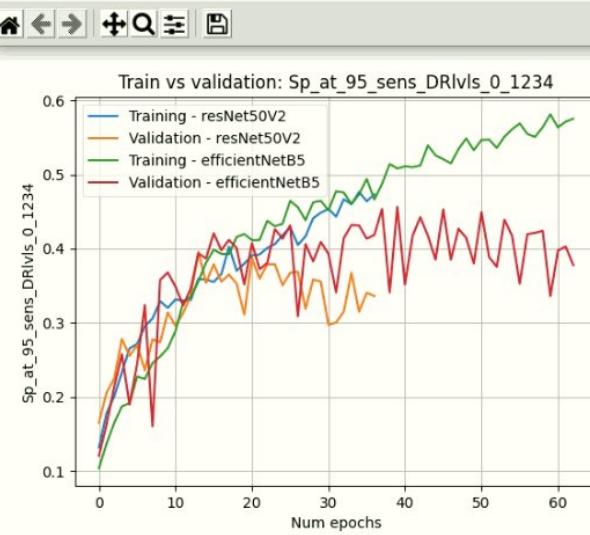
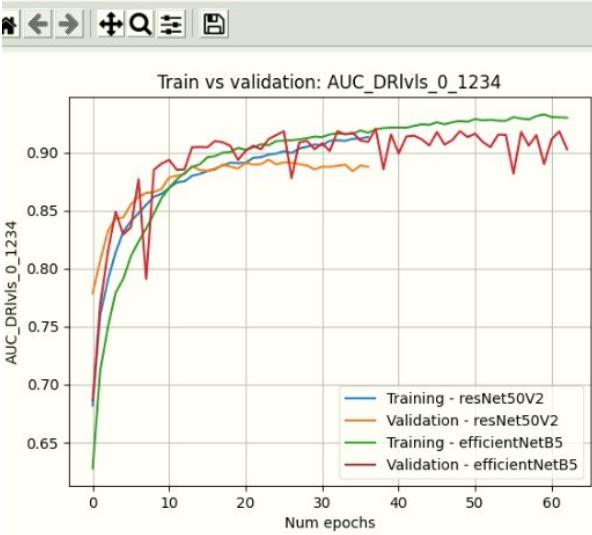
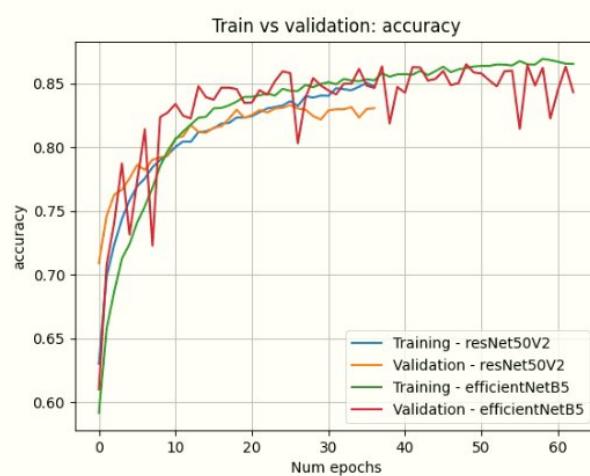
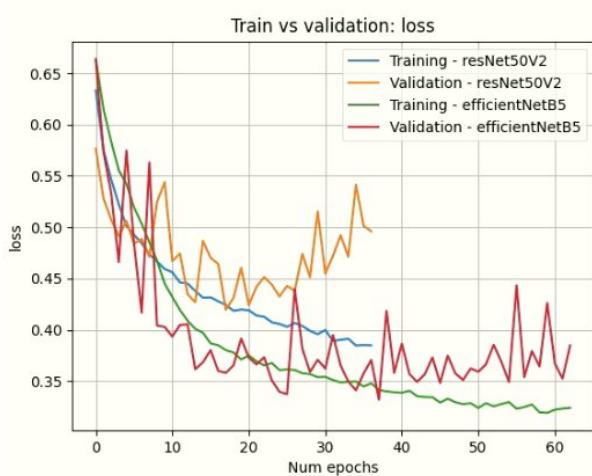
Se ha utilizado el dataset de entrenamiento balanceado, de forma que en cada época, haya siempre el mismo número de imágenes de cada clase, y se ha comprobado, iterando varias veces sobre el dataset de entrenamiento, que aproximadamente en el 97% de los batches aparecen imágenes de las dos clases definidas.

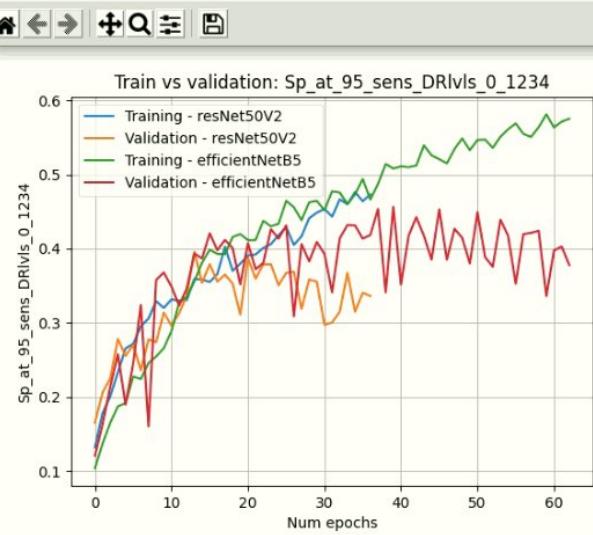
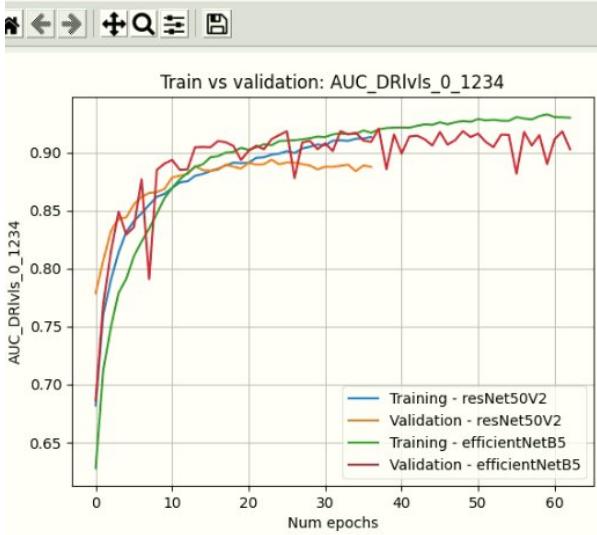
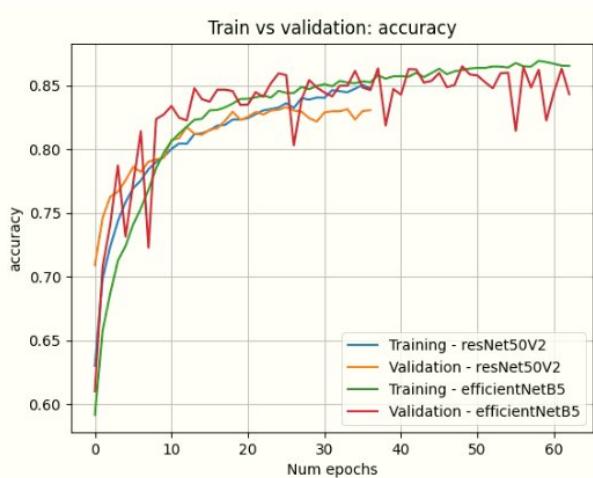
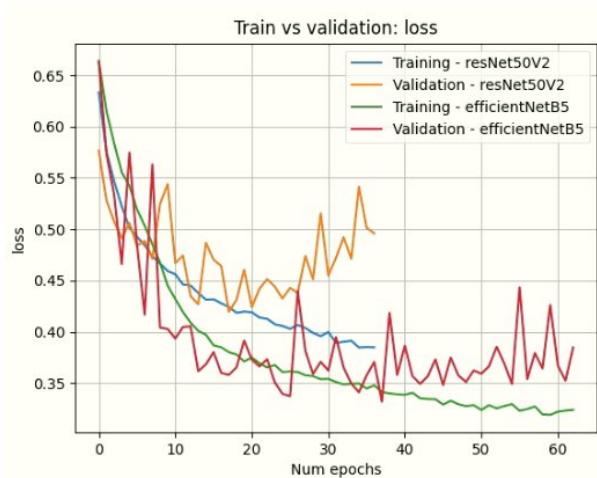
Esto hace que el dataset de entrenamiento se componga en cada época de 29.346 imágenes, siendo cada mitad (14.673) correspondiente a una clase (la clase mayoritaria, 0, tendrá imágenes distintas en cada época).

Para validar, se ha empleado un fichero csv balanceado, para que haya mismos elementos de cada clase, y éstos no varíen en ningún momento: EYEPACS-VALIDATION-10_BALANCED.csv.

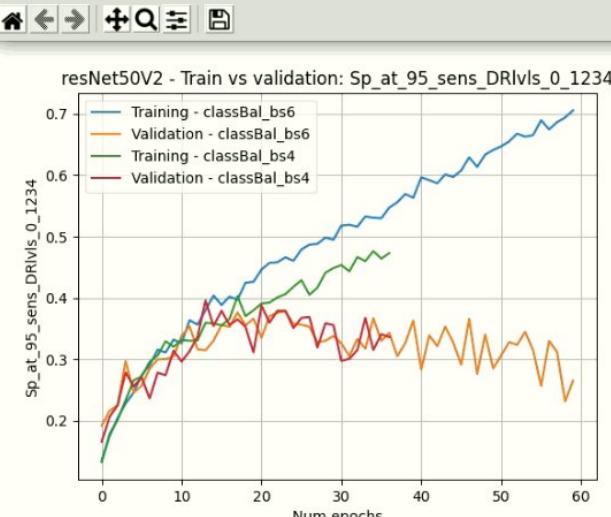
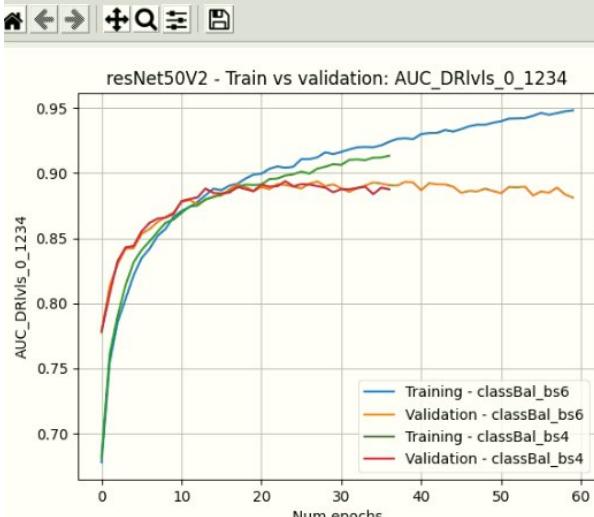
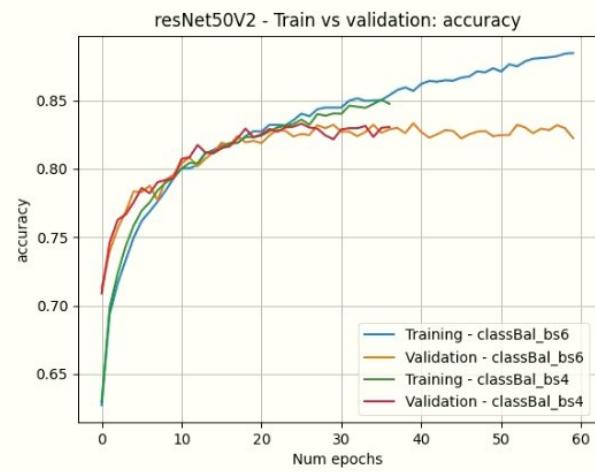
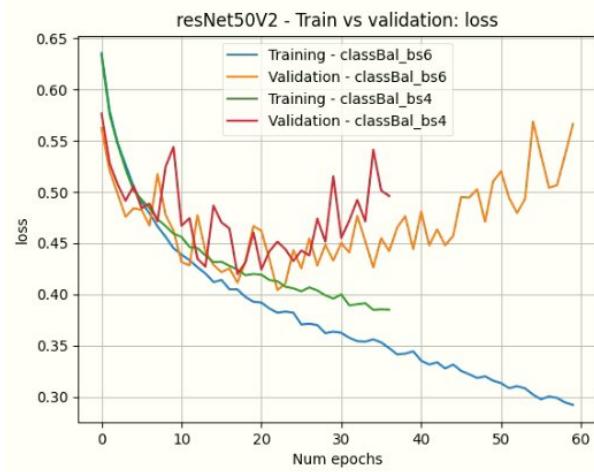
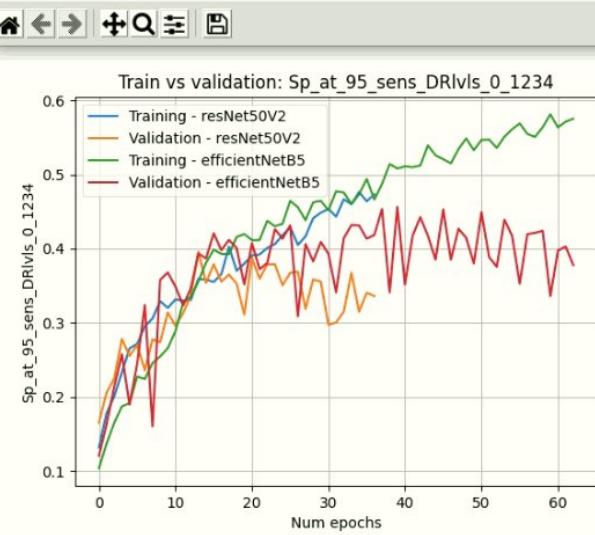
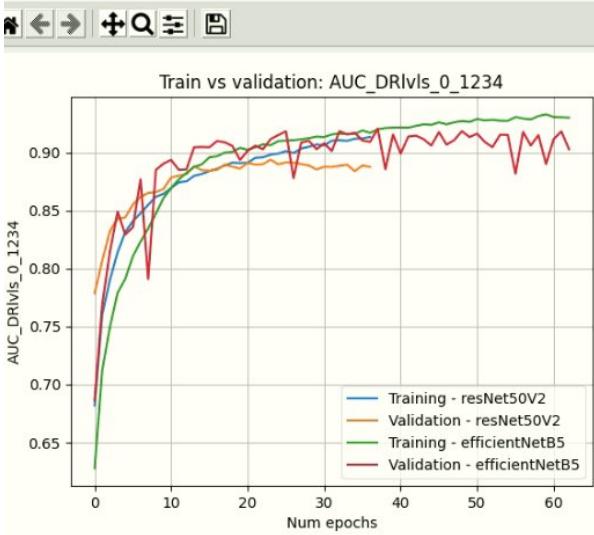
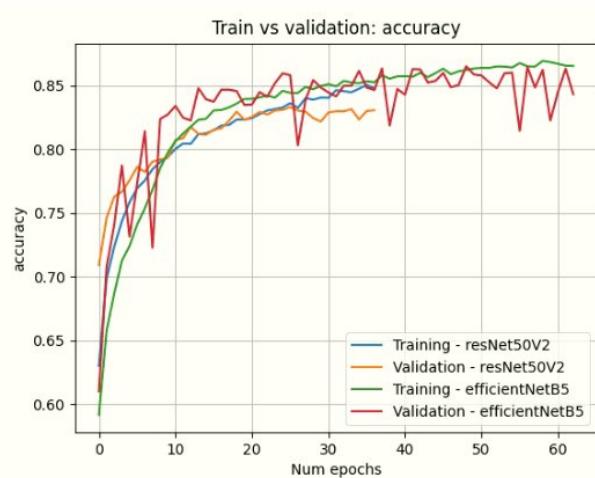
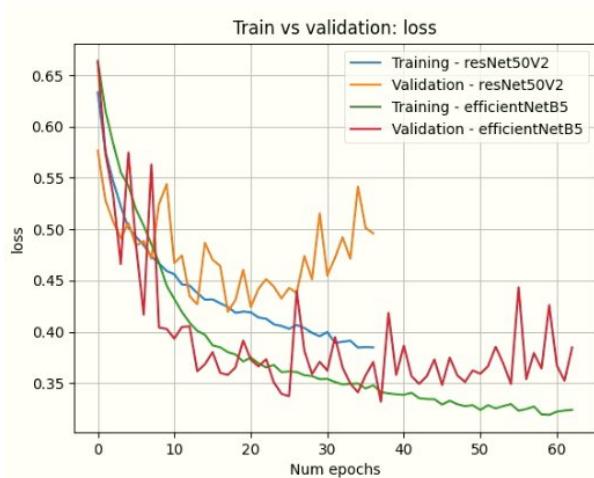
- Batch size = 6 para train, 12 para validation
- Se aplica data augmentation y shuffle en el dataset de entrenamiento
- Salida en formato categorical y no one-hot (necesario para aplicar el ponderado de clases)
- Sparse_categorical_crossentropy como loss function
- SGD con learning rate = 0.0001, momentum = 0.9 y clipnorm = 1.0

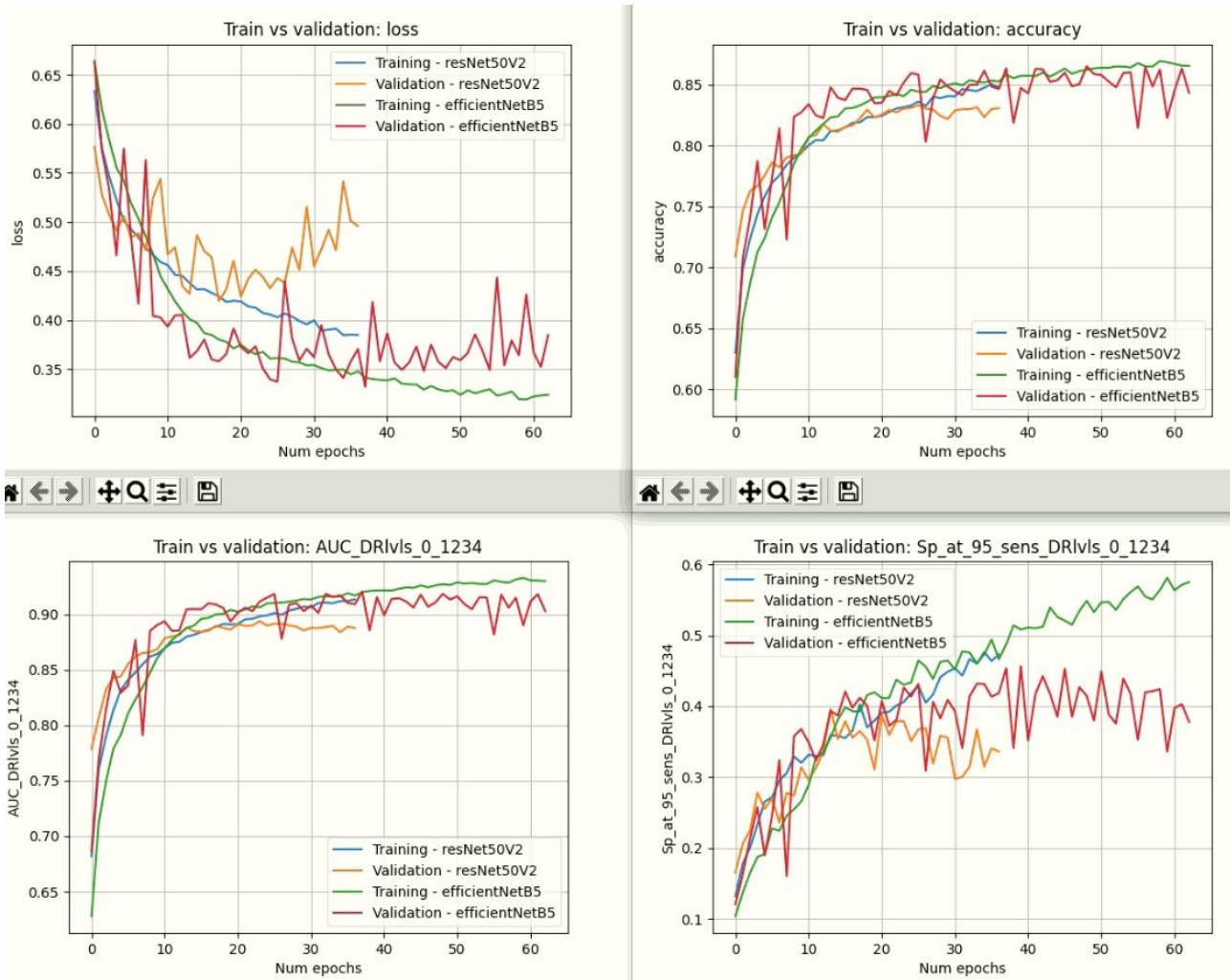
En la siguiente imagen se muestra la evolución que ha seguido:





Y la siguiente imagen muestra su comportamiento estableciendo el batch size a 4:





Comportamientos muy similares.

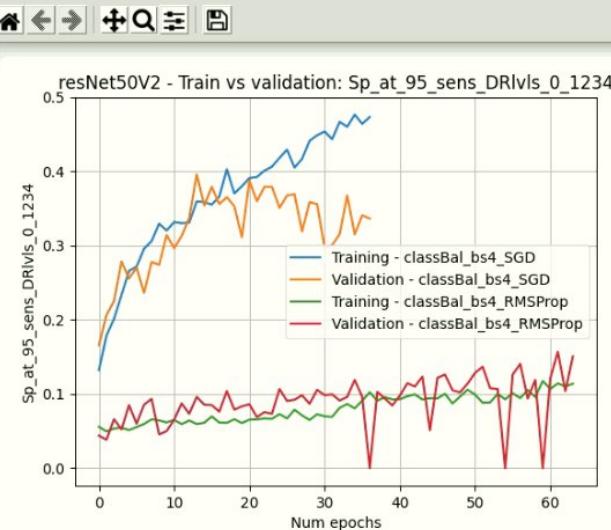
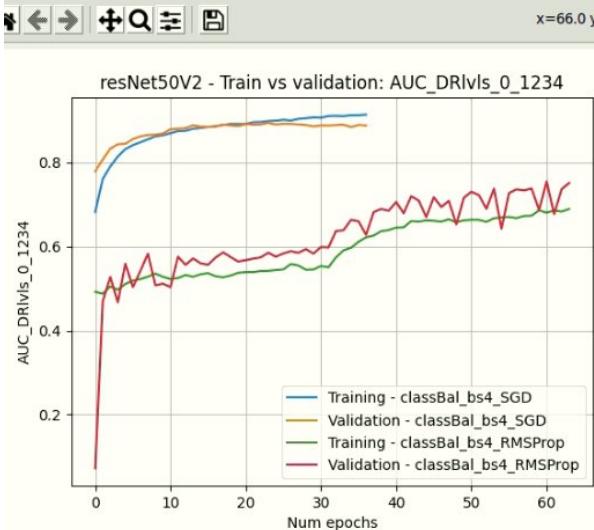
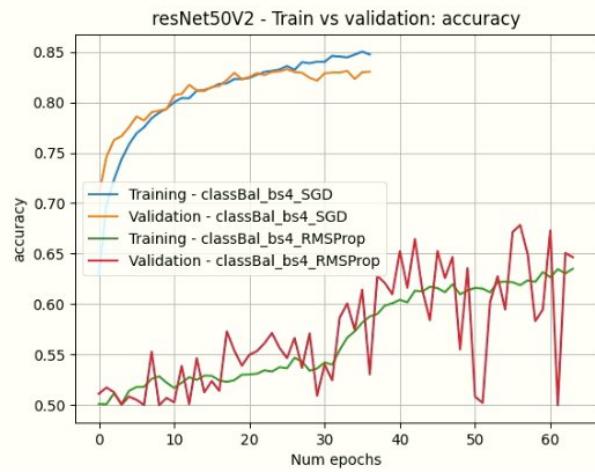
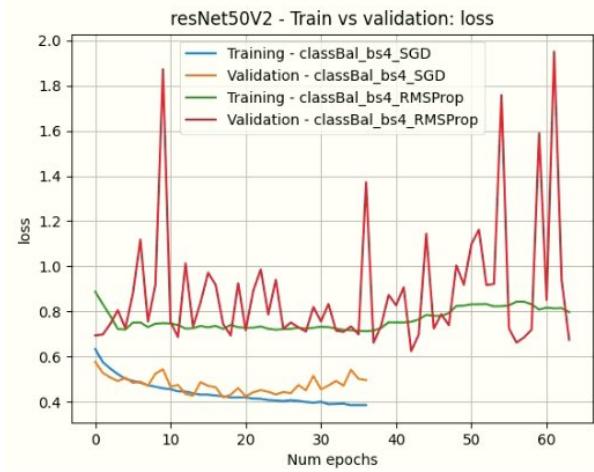
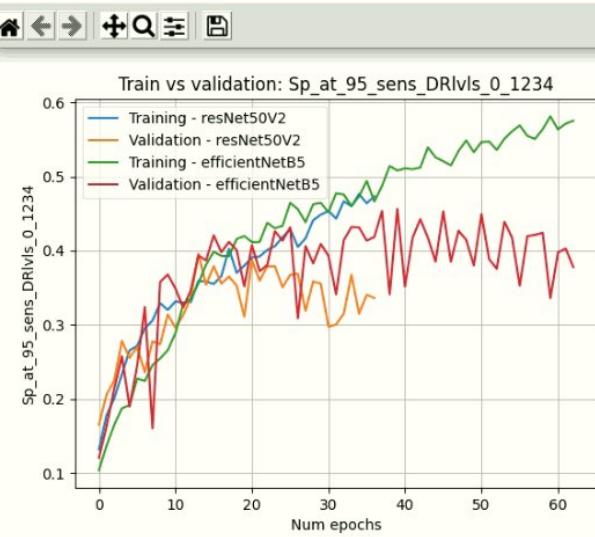
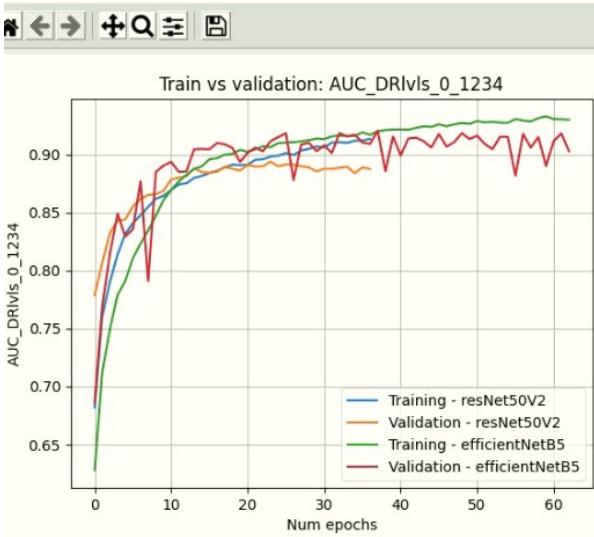
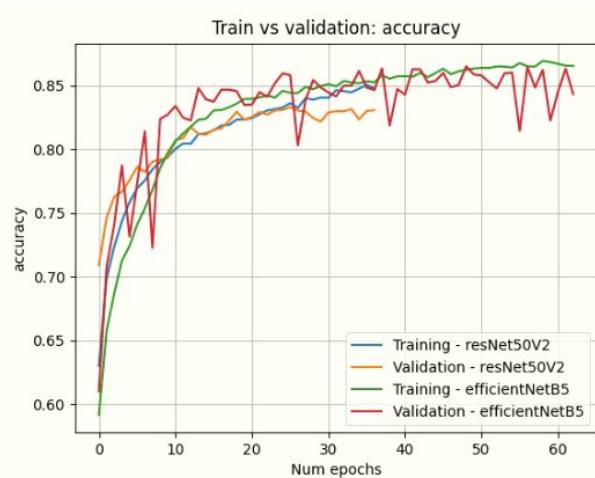
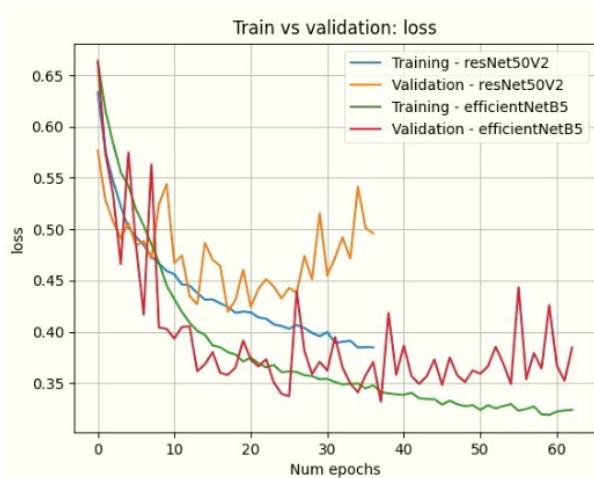
5.3 Modelo_propio + ResNet50V2 + EYEPACS 0 1234 con balanceo RMSProp

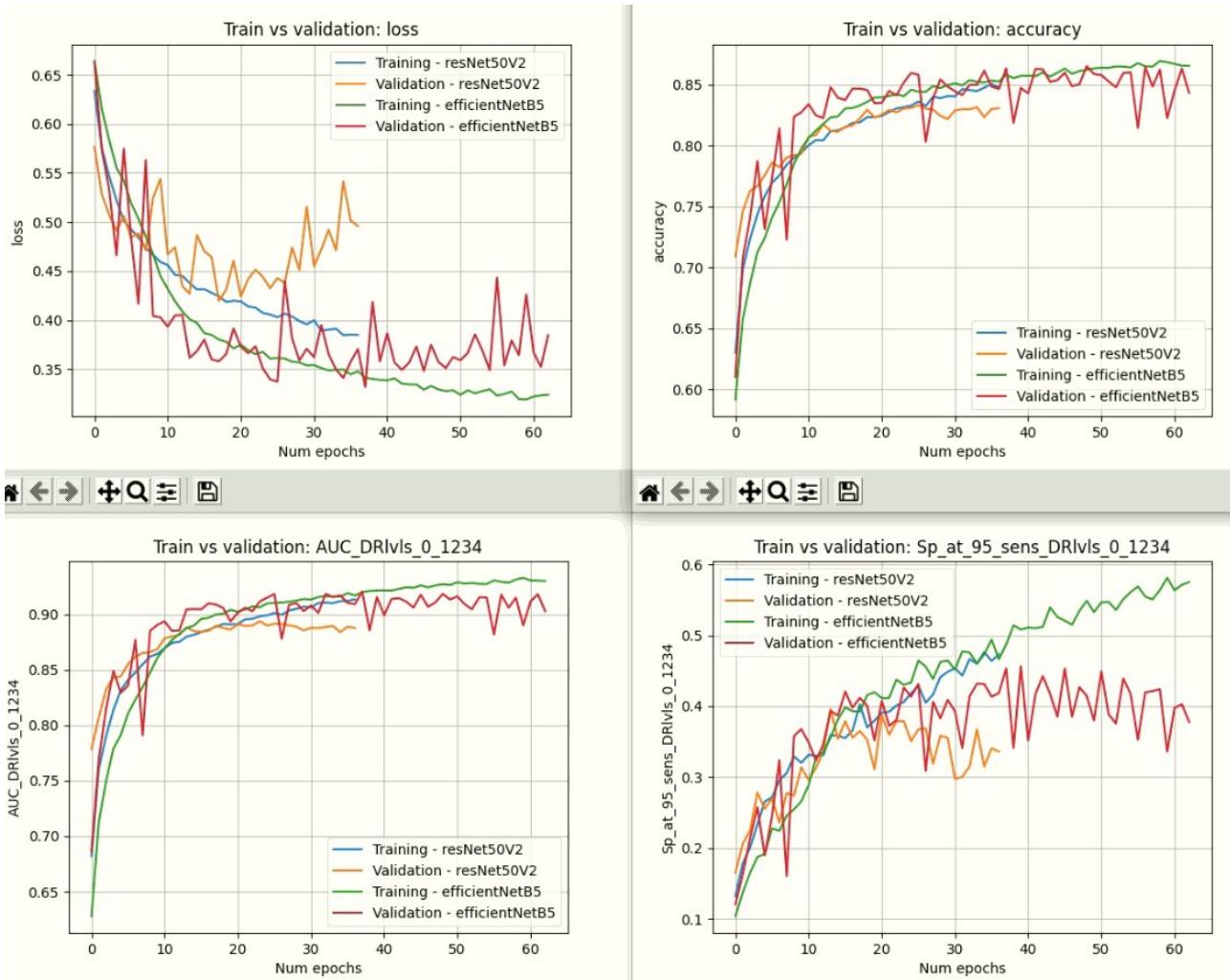
Los datasets utilizados siguen la misma forma que la prueba anterior. Los parámetros de esta prueba son los siguientes:

- Batch size = 4 para train, 12 para validation
- Se aplica data augmentation y shuffle en el dataset de entrenamiento
- Salida en formato categorical y no one-hot (necesario para aplicar el ponderado de clases)
- Sparse_categorical_crossentropy como loss function
- RMSProp con learning_rate=0.001, rho/decay=4e-5, clipnorm=1.0

Fichero de prueba: prueba18_MPropio_ResNet50V2_EYEPACS_Bal_RMSProp.ipynb

En la siguiente imagen se muestra la evolución que ha seguido:





Ha seguido una evolución notablemente peor, sin tendencia clara.

6.1 Modelo_propio + EfficientNetB5 + EYEPACS 0 1234 con balanceo + SGD

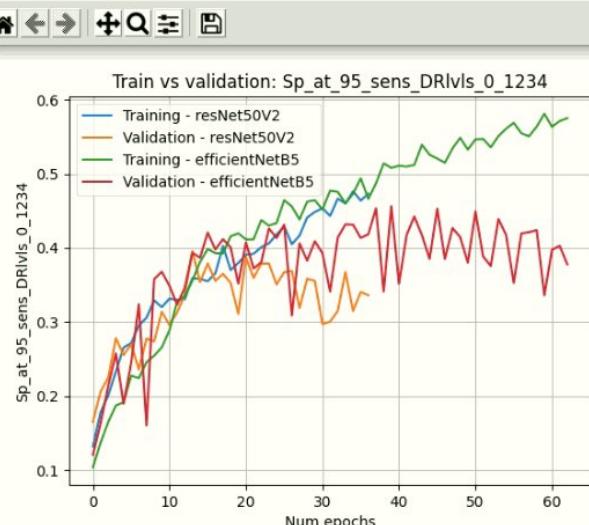
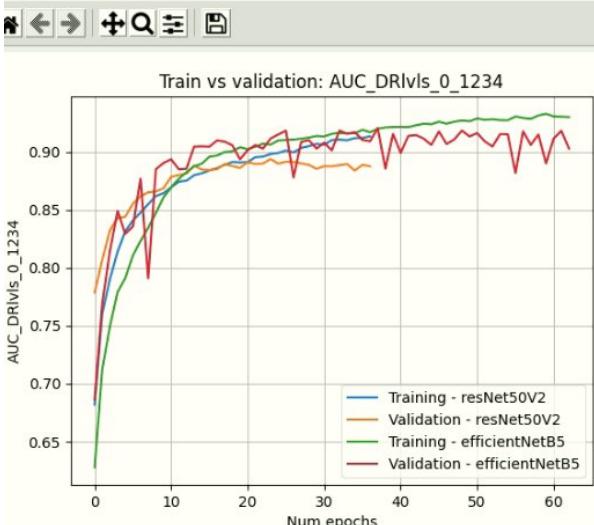
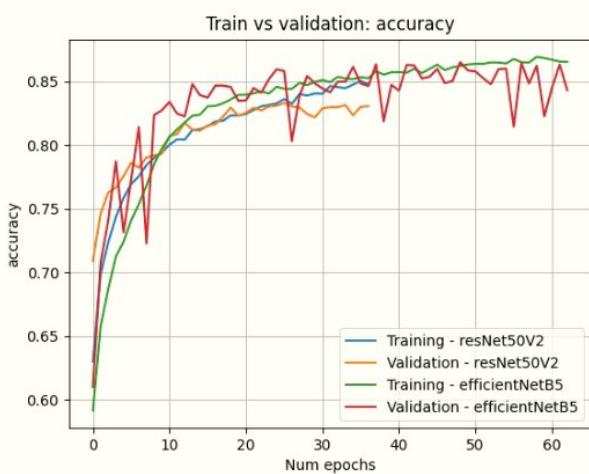
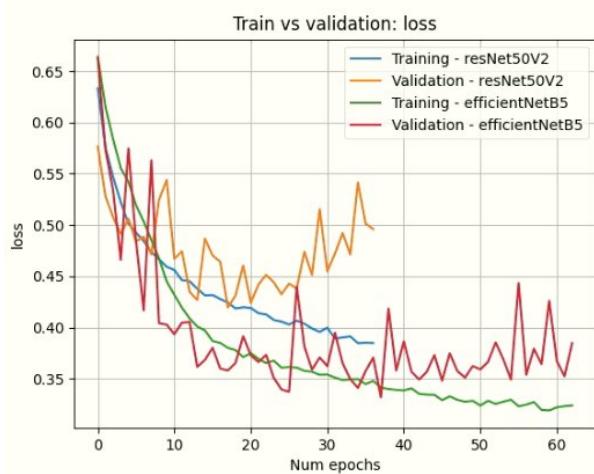
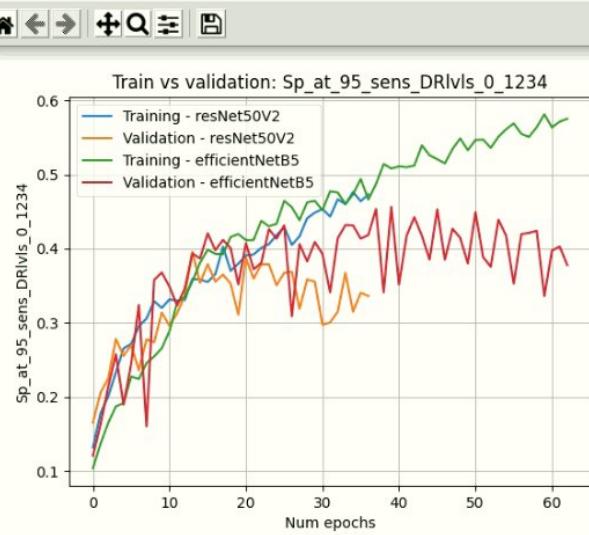
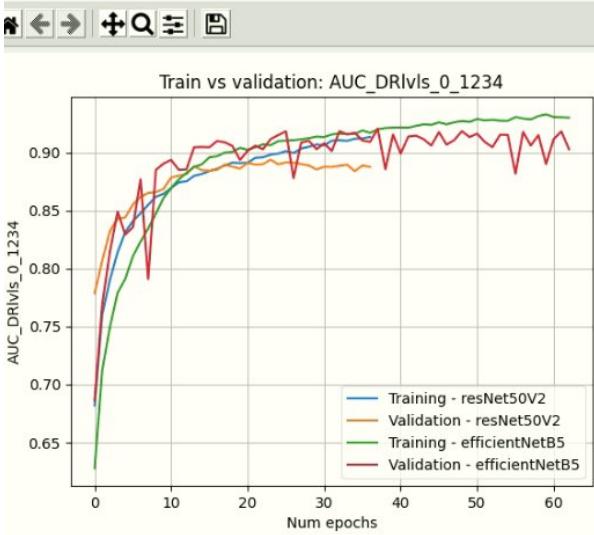
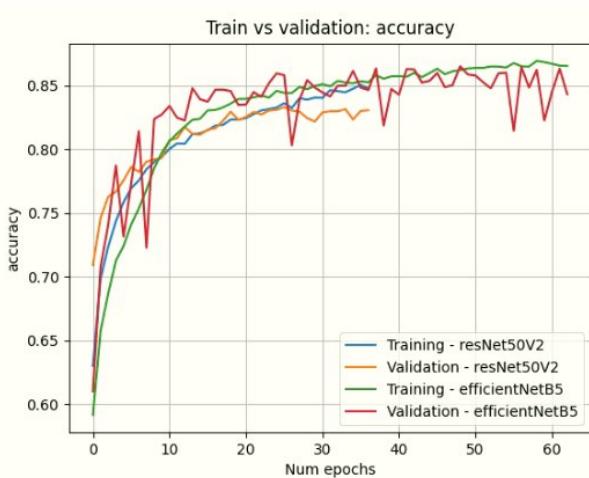
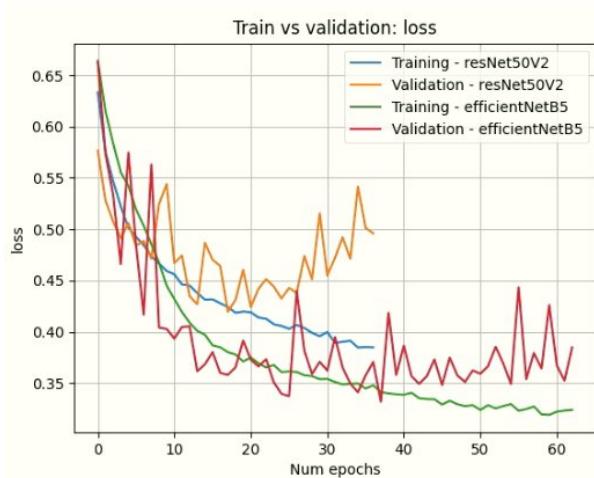
Se ha utilizado el dataset de entrenamiento balanceado, de forma que en cada época, haya siempre el mismo número de imágenes de cada clase, y se ha comprobado, iterando varias veces sobre el dataset de entrenamiento, que aproximadamente en el 97% de los batches aparecen imágenes de las dos clases definidas.

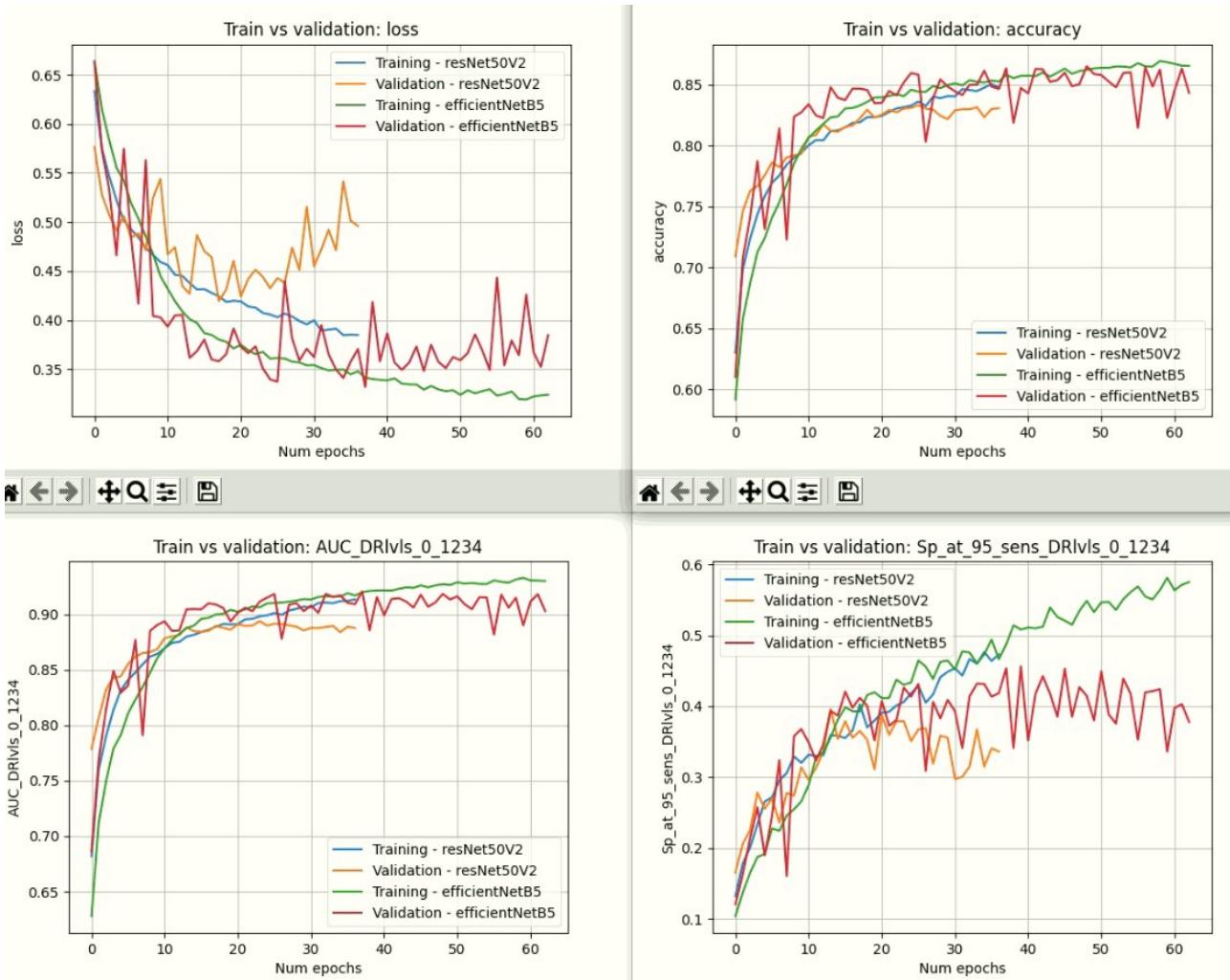
Esto hace que el dataset de entrenamiento se componga en cada época de 29.346 imágenes, siendo cada mitad (14.673) correspondiente a una clase (la clase mayoritaria, 0, tendrá imágenes distintas en cada época).

Para validar, se ha empleado un fichero csv balanceado, para que haya mismos elementos de cada clase, y éstos no varíen en ningún momento: EYEPACS-VALIDATION-10_BALANCED.csv.

- Batch size = 4 para train, 12 para validation
- Se aplica data augmentation y shuffle en el dataset de entrenamiento
- Salida en formato categorical y no one-hot (necesario para aplicar el ponderado de clases)
- Sparse_categorical_crossentropy como loss function
- SGD con learning rate = 0.0001, momentum = 0.9 y clipnorm = 1.0

En la siguiente imagen se muestra la evolución que ha seguido:





Los resultados son mejores que los anteriores. Se compara con el entrenamiento anterior con los mismos parámetros pero utilizando ResNet50V2.

Fichero: prueba19_MPropio_EfficientNetB5_SGD_EYEPACS_Bal.ipynb

6.2 Modelo_propio + EfficientNetB5 + EYEPACS 0 1 234 con balanceo + SGD

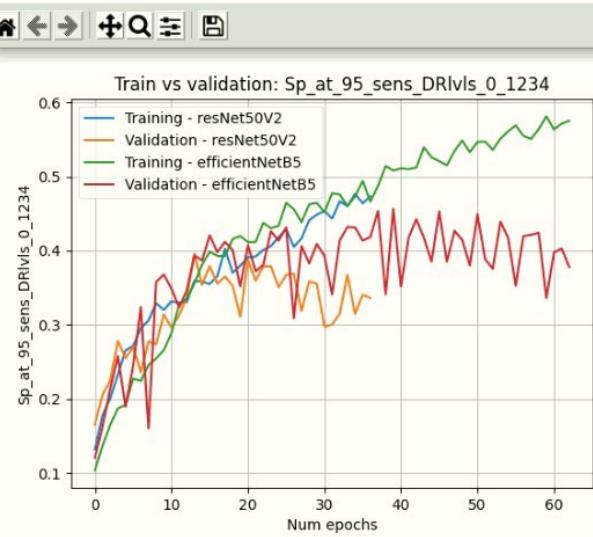
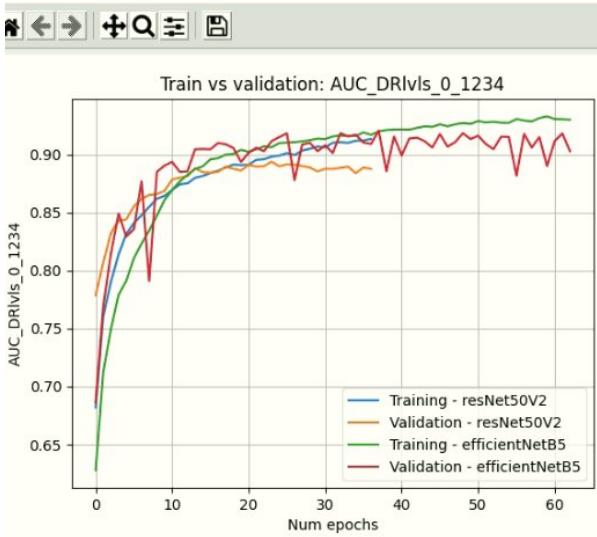
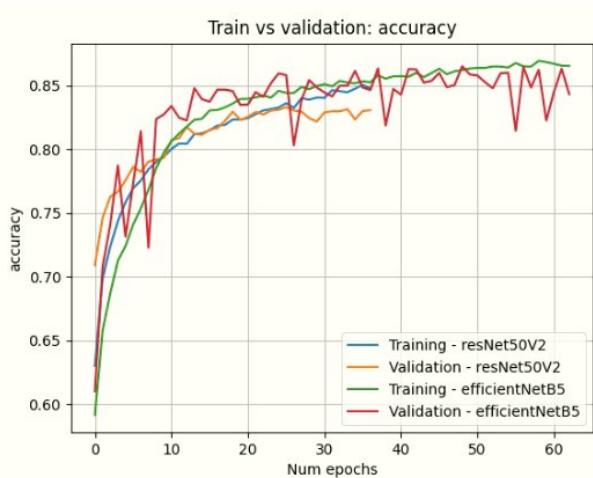
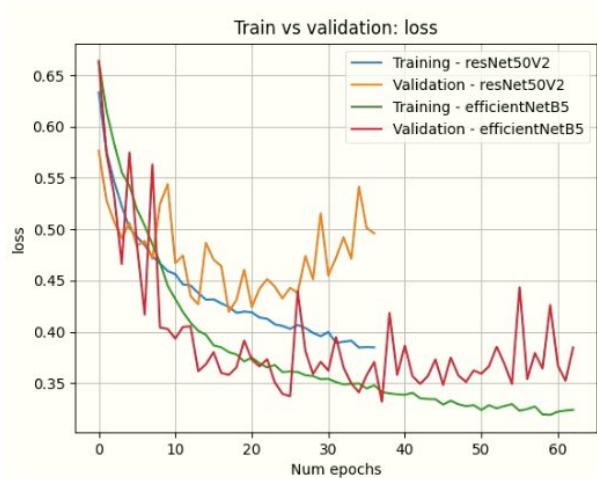
Se ha utilizado el dataset de entrenamiento balanceado, de forma que en cada época, haya siempre el mismo número de imágenes de cada clase, y se ha comprobado, iterando varias veces sobre el dataset de entrenamiento, que aproximadamente en el 97% de los batches aparecen imágenes de las dos clases definidas.

Esto hace que el dataset de entrenamiento se componga en cada época de 12.147 images, teniendo 4.049 por clase (sólo la clase minoritaria mantendrá siempre las mismas imágenes).

Para validar, se ha empleado un fichero csv balanceado, para que haya mismos elementos de cada clase, y éstos no varíen en ningún momento: EYEPACS-VALIDATION-10-BALANCED-0-1-234.csv.

- Batch size = 4 para train, 12 para validation
- Se aplica data augmentation y shuffle en el dataset de entrenamiento
- Salida en formato categorical y no one-hot (necesario para aplicar el ponderado de clases)
- Sparse_categorical_crossentropy como loss function
- SGD con learning rate = 0.0001, momentum = 0.9 y clipnorm = 1.0
- El learning rate se multiplica por 0.9 cada 10 épocas

En la siguiente imagen se muestra la evolución que ha seguido:



Fichero: prueba20_Modelo_prop_EfficientNetB5_SGD_EYEPACS_Bal_0-1-234.py