UNIVERSIDAD ADOLFO IBÁÑEZ
FACULTAD DE INGENIERÍA Y CIENCIAS

Detección automática de tapetes microbianos de *Beggiatoa* en imágenes bajo zonas de acuicultura

Nicolás Martínez Alfaro

Santiago de Chile, 2023.

UNIVERSIDAD ADOLFO IBÁÑEZ
FACULTAD DE INGENIERÍA Y CIENCIAS

Detección automática de tapetes microbianos de *Beggiatoa* en imágenes bajo zonas de acuicultura

Nicolás Martínez Alfaro

Profesor Guía: Sebastián Moreno Araya

Comité: Miguel Carrasco y Vladimir Riffo

Tesis para Optar al

Título de Ingeniería Civil en Informática y

Grado de Magíster en Ciencias en Data Science

Santiago de Chile, 2023.

Resumen del Proyecto

Los tapetes microbianos de *Beggiatoa* son ecosistemas naturales que se encuentran presentes en los fondos marinos de zonas anóxicas. Su presencia en el fondo marino bajo zonas de producción de salmones en la industria acuícola es un indicador de alta contaminación. Actualmente, la búsqueda de *Beggiatoa* en el fondo marino de las salmoneras se realiza a través del análisis manual de videos, lo que no es escalable y está sujeto al error humano. En esta tesis, se propone un modelo de red convolucional *VGG-16* para la detección automática de tapetes microbianos de *Beggiatoa* en imágenes del fondo marino. Para ello, se analizó distintos hiperparámetros de la red, seleccionando un modelo que obtiene una mejora de *F2-score* cercana al 0.60 con respecto al estado del arte actual.

Abstract

Beggiatoa microbial mats are natural ecosystems present on the seabed of anoxic zones. Their presence on the seabed under salmon production areas in the aquaculture industry indicates high contamination. Currently, the search for *Beggiatoa* on the seabed of salmon farms is through manual video analysis, which is not scalable and subject to human error. In this thesis, a convolutional network model *VGG-16* is proposed for the automatic detection of *Beggiatoa* microbial mats in seafloor images. We searched over different network hyperparameters, selecting a model with an improvement over 0.60 in *F2-score*, with respect to the current state of the art.

Key Words: CNN, salmon farms, *Beggiatoa* microbial mats, aquaculture.

Índice general

Índice de cuadros

Capítulo 1

Introducción

El ambiente marino es un sistema complejo y con una alta interacción entre sus componentes. Esto conlleva a un permanente riesgo de exposición a patógenos, como también a variables abióticas tales como temperatura, oxígeno, salinidad, y luminosidad. Respecto a los patógenos, existe un limitado conocimiento sobre la diversidad, distribución, frecuencia, y la forma en que afectan a la fauna marina en un momento dado. Lo anterior, hace que predecir la evolución de patógenos en los procesos de la industria acuícola sea un ejercicio difícil. Esto es necesario, debido a que en Chile, por ley, la industria acuícola debe mantener su fondo marino sin patógenos. Por lo cual, se requiere que se realicen revisiones periódicas del estado de dichos procesos. Actualmente, uno de estos procesos es realizado de forma manual por SERNAPESCA, en el cual tiene que revisar horas de videos del fondo marino en búsqueda de la bacteria *Beggiatoa*. En esta tesis, se propone un modelo basado en redes neuronales para automatizar el proceso de búsqueda de *Beggiatoa* en videos del fondo marino.

1.1. Antecedentes

Una de las principales herramientas que se utilizan hoy para evaluar el nivel de contaminación marina requiere como insumo el análisis de la materia orgánica dispuesta en el espacio bentónico y su comunidad (bentos) como variable proxy [Elliott et al., 2006]. Lo anterior se refiere al sedimento del fondo marino y los organismos que habitan en o sobre él. Se ha estimado que el bentos contiene a más del 90 % de todas las especies encontradas en el océano y constituye la principal fuente de diversidad biológica del planeta. En décadas pasadas, el análisis del nivel de contaminación no se relacionaba con la interacción de los organismos que habitan en el bentos, sólo se enfocaba en la interacción de los componentes físicos/químicos de los sedimentos [Pearson, 1978, Pearson and Stanley, 1979]. Luego,

nuevos estudios comenzaron a analizar el bentos, generando nuevos métodos que consideran marcadores biológicos en organismos [Raffaelli, 1987, Livingston et al., 1994]. Por consiguiente, la investigación se ha enfocado en comprender la interacción entre los distintos organismos bentónicos, sus respuestas frente a las interacciones ambientales, y niveles de contaminación; generados en muchas ocasiones a partir de la intervención humana [Martínez-Alonso and Gaju, 2005]. Específicamente, la comunidad se ha dedicado a estudiar el efecto en: la calidad del agua, la estructura del hábitat, el flujo de agua, las fuentes de energía, hasta las distintas interacciones biológicas del bentos [Harrison et al., 2006].

En la actualidad, los distintos sistemas de vigilancia ambiental consideran como esencial el análisis y control del espacio bentónico, en fondos blandos (arenosos) y duros (rocosos). Esto se debe a que la respuesta generadas sobre éstos, y sus cambios en el tiempo permiten evaluar las condiciones ambientales [Gray and Elliott, 2009]. Por ejemplo, la contaminación marina origina una respuesta de la comunidad bentónica, la cual puede generar variados efectos tales como una mayor mortalidad de especies, o abundancia de especies tolerantes y/u oportunistas (como los tapetes microbianos). De esta forma cambios de la comunidad bentónica, tales como biomasa, abundancia, números de especies, entre otras, implican cambios en los niveles de contaminación [Douglas et al., 2017].

1.2. Tapetes microbianos

Los tapetes microbianos o bacterianos son ecosistemas naturales, parte del bentos, donde microorganismos se agrupan en comunidades de unos pocos milímetros de espesor sobre una superficie o sustrato, y se encuentran presentes en los fondos marinos de zonas anóxicas [Aguzzi et al., 2011, Harrison et al., 2006, Matabos et al., 2011]. Estos dominaron gran parte del planeta durante miles de millones de años. Sin embargo, en los últimos años, se ha postulado que la aparición de tapetes bacterianos en el fondo marino permite proporcionar medidas para determinar la influencia de la contaminación en el lugar [Crawford et al., 2001, Pearson, 1975]. Hoy en día, se propone que los tapetes microbianos surgen en lugares sujetos a contaminación crónica, debido a que se alimentan de los contaminantes en dicha zona [Martínez-Alonso and Gaju, 2005].

Un estudio reciente ha determinado que la presencia de tapetes microbianos en el fondo marino permite, específicamente, evaluar el nivel de contaminación generado bajo zonas de producción de salmones en la industria acuícola [Knight et al., 2021]. Por lo general, los cultivos marinos están compuestos por una gran cantidad de peces ubicados espacialmente en zonas delimitadas o jaulas, los cuales son alimentados en forma externa. Este proceso genera una gran cantidad de fecas y/o alimentos no consumidos, los cuales

finalmente son depositados en el fondo marino. Esta carga orgánica adicional tiene un evidente efecto sobre la fauna y flora que allí habita, la que por lo general utiliza la carga orgánica como alimento; y el restante se descompone generando sulfuro de hidrógeno (H_2S) el cual aumenta la temperatura y acidifica el mar. Lo anterior finalmente redunda en la aparición de un tapete microbiano de *Beggiatoa* [Elliott et al., 2006], bacterias que se alimentan del H_2S. Por esta razón es que a mayor presencia de tapetes microbianos de *Beggiatoa* en el fondo marino, se puede vincular a una mayor contaminación de la zona de acuicultura.

1.3. Detección de *Beggiatoa* en Chile

Actualmente, en Chile, se está utilizando la presencia de *Beggiatoa* como marcador de contaminación en zonas acuícolas sobre fondos duros y semi-duros. Específicamente, la normativa chilena exige que debe tomarse un registro visual (grabación subacuática de los componentes del megabentos) en los centros de categoría N°4 [1]. En este registro se evalua tanto el impacto como el aporte, de la materia orgánica, sobre la epifauna bentónica; además de la presencia de burbujas o tapices bacterianos conformados principalmente por la bacteria *Beggiatoa*. Sin embargo, a pesar de que los registros visuales entregan información valiosa, estos no están ajenos a problemas. Uno de los principales problemas es la interpretación y evaluación de los videos [CIEP, 2010], que en el caso de las acuicolas Chilena, está a cargo del Servicio Nacional de Pesca y Acuicultura (SERNAPESCA).

En particular, SERNAPESCA debe identificar la presencia de los tapetes de *Beggiatoa* en el fondo marino de los Centros de Engorda de Salmónidos (CES), debido a que gran parte de la contaminación acuícola se genera como subproducto de la industria de salmonicultura [Knight et al., 2021]. El proceso de inspección se realiza en dos etapas. Primero, los CES proporcionan videos del fondo marino a Sernapesca, junto a un informe indicando la presencia o ausencia de tapetes microbianos; entre otros elementos. Segundo, Sernapesca analiza el informe, y en el caso que no se encuentran tapetes microbianos, Sernapesca debe certificar el video en su totalidad. De otro modo, se analizan los intervalos en donde se específica la presencia de tapetes microbianos. Finalmente, en caso de que se identifique la presencia de estos, el CES queda inhabilitado para la "siembra" de una nueva producción hasta que las condiciones ambientales vuelvan a la normalidad (las aguas deben volver a un estado aeróbico).

El proceso anterior genera una gran cantidad de videos e informes que deben ser re-

[1]Centros de cultivo con sistemas de producción intensivos que se encuentren en sectores de **fondos duros o semi-duros** y cuyas profundidades sean iguales o menores a 60 metros.

visados meticulosamente. Esta documentación presenta información exhaustiva del fondo marino bajo las zonas de cultivo[2]. Claramente este proceso es complejo de escalar, ya que demanda una excesiva cantidad de horas persona (conllevando un costo monetario muy alto) y se encuentra sujeto al error humano.

Sumado a la problemática anterior, cambios en la legislación vigente especifican que los CES ya no incluirán la presencia de tapetes microbianos dentro de sus informes. Esto implica que SERNAPESCA deberá analizar la duración completa de todos los videos para determinar la presencia de tapetes microbianos, resultando en un mayor gasto de recursos, y también en un menor número de fiscalización sobre los CES.

1.4. Problema

Como se ha mencionado anteriormente, SERNAPESCA realiza un análisis visual de cada uno de los videos para identificar la presencia de tapetes de *Beggiatoa*. Lamentablemente, su sistema actual no es escalable, ya que depende de la revisión manual de cada video por un funcionario. Además, el análisis está sujeto al error humano debido al criterio personal de cada funcionario. Es por estos motivos que surge la necesidad de un algoritmo automático que realice la segmentación temporal de dichos videos a partir de la identificación automática de la presencia de tapetes microbianos de *Beggiatoa* en el fondo marino.

Actualmente, solo se conoce un algoritmo capaz de realizar la tarea anteriormente mencionada. [Castillo, 2023] estudió el uso de la *CNN VGG-16* para crear un modelo de clasificación de imágenes, y un algoritmo de segmentación temporal de videos. Sin embargo, el algoritmo no es íntegramente automático, este requiere de la selección de parámetros para poder realizar el análisis de cada video (sin haber un criterio para seleccionarlos). Además, Castillo menciona que su modelo de clasificación presenta problemas de sensibilidad en imágenes poco claras. En esta tesis se propone realizar un proceso de búsqueda y entrenamiento riguroso con el fin de superar el desempeño del modelo de clasificación anterior, con el objetivo de que este, en un futuro, pueda ser utilizado para detectar en forma automática los tramos de los video con *Beggiatoa*.

[2]Resolución N° 3612 de la Subsecretaría de Pesca y Acuicultura de Chile

Capítulo 2

Estado del Arte

Como se menciono en el capítulo anterior, nuestro objetivo, de forma general, consiste en el análisis de videos para segmentarlos temporalmente. A modo de contexto, la clasificación de objetos dentro de imágenes y videos ha sido un área activa de investigación durante las últimas décadas [Hoeser and Kuenzer, 2020, Zhao et al., 2019, Wu et al., 2020, Jiao et al., 2021, Mahrishi et al., 2021, Chailloux et al., 2008, Ji et al., 2012, Tran et al., 2015]. Esta área de investigación se puede dividir en métodos que utilizan descriptores basados en aprendizaje profundo, y otros métodos basados en descriptores creados específicamente para solucionar la tarea requerida.

Actualmente, para la clasificación de imágenes y análisis de videos se utilizan de forma extensiva las redes neuronales de aprendizaje profundo; también siendo utilizadas para análisis de comportamiento, reconocimiento facial y conducción autónoma [Ngiam et al., 2011, Szegedy et al., 2017, Othmani, 2022, Muhammad et al., 2021]. Hay muchos tipos de redes neuronales de aprendizaje profundo que pueden ser utilizadas para la detección de objetos. En todas estas, el proceso es automatizado a través de un entrenamiento riguroso, en el cual la red a partir de imágenes y su clase respectiva aprende a extraer características importantes. Luego, la red entrenada es capaz de detectar de manera automática el objeto en imágenes no utilizadas en el entrenamiento, sin la necesidad de intervención humana.

En el caso de detección de tapetes microbianas de *Beggiatoa*, a la fecha, no se encontró ningún artículo que detecte este tipo de objetos en videos. Por lo cual se realizaron búsquedas dentro del marco de detección de tapetes microbianos *Beggiatoa* en imágenes, y por otro lado, sobre segmentación temporal dentro de videos a partir de la presencia de un objeto o tipo de imagen.

Desafortunadamente, casi no existe literatura reciente relacionada a la detección de tapetes microbianos de *Beggiatoa* en imágenes. La mayoría de los estudios tienen más

de 10 años de antigüedad [Aguzzi et al., 2011, Chailloux et al., 2008, Jerosch et al., 2007, Matabos et al., 2011], existiendo dos estudios recientes [Chen et al., 2022, Castillo, 2023]. Debido a la escasa literatura, también se analizaron métodos actuales que se utilicen para la detección de objetos y clasificación en imágenes submarinas.

2.1. Detección *Beggiatoa*

Una búsqueda extensa de la literatura presentó solo cinco artículos publicados [Aguzzi et al., 2011, Chailloux et al., 2008, Jerosch et al., 2007, Matabos et al., 2011, Chen et al., 2022], un artículo aceptado [Chen et al., 2024], y una tesis de magíster [Castillo, 2023] relacionados a la detección de tapetes microbianos, específicamente de la especie *Beggiatoa*, en imágenes y/o videos del fondo marino.

[Jerosch et al., 2007] es uno de los primeros métodos de la literatura y se enfocó en la detección y cuantificación de la distribución espacial de tapetes de *Beggiatoa* en imágenes. 2840 imágenes fueron obtenidas a través de la extracción de mosaicos georeferenciados a partir de videos y data de navegación de alta precisión de Remote Operated Vehicle (ROVs) a través del software MATISSE (desarrollado por IFREMER). En el proceso de limpieza, se removieron las zonas donde distintos mosaicos se encontraban sobrepuestos. Luego, para lograr la segmentación de los tapetes de *Beggiatoa* se aplicó transformada de *watershed* [Roerdink and Meijster, 2000] y *relaxation-based labelling* [Kittler and Illingworth, 1985], alcanzado una *precision* mayor a un 90 %; sin embargo, no es un método automático ya que requiere de intervención humana.

A diferencia del trabajo anterior, [Chailloux et al., 2008] se enfocó en la detección y cuantificación automática de tapetes de *Beggiatoa* en imágenes. Las imágenes fueron obtenidas a través de la extracción de *frames* de videos de ROVs, y 3 métodos fueron comparados usando el *classification ratio*. El primer método define puntos semilla de un algoritmo de crecimiento de regiones como centros de regiones de transformaciones de *watershed* basadas en gradientes. El segundo método, también basado en un algoritmo de crecimiento de regiones, se logró mediante una inicialización de puntos semilla determinados a partir de un criterio de similitud, el cual requiere de habilidad humana para determinar regiones homogéneas. Finalmente, el tercer método se basa en análisis de texturas, utilizando parámetros de las matriz de coocurrencia y de filtro de Gabor, y estimando la similaridad entre distintas muestra de imágenes a través de la divergencia de Kullback-Leibler . Como es mencionado en el artículo, los resultados "no son satisfactorios ya que el puntaje alcanzado fue débil".

[Matabos et al., 2011] utilizo un método de cuantificación de tapetes de *Beggiatoa* en

imágenes, con el fin de estudiar el comportamiento de decádopodos demersales (crustáceos bentónicos). Sin embargo, la detección se realiza en condiciones muy especificas e ideales. Todas las imágenes seleccionadas tenían que tener la cámara orientada verticalmente hacia abajo, y con luminosidad uniforme. A pesar de ello, los resultados del modelo fueron insatisfactorios aduciendo que pudieron ser afectados por el sistema de iluminación utilizado al momento de capturar las imágenes. Para mejorar la clasificación realizada, los autores realizaron un nuevo trabajo seleccionando 52 nuevas imágenes [Aguzzi et al., 2011]. En las imágenes seleccionadas se evitó las alteraciones causadas por otras especies bentónicas. Primero, se definió una Región de Interés (RI) constante y se realizó la extracción de fondo mediante un filtro de *Gaussian Blurring*. Después, se realizó un extracción de la imagen dentro de la RI, el cual fue seguido por la suma de las matrices de los canales RGB. Luego, se realizó un proceso de mejoramiento de la imagen y la aplicación de un umbral para la clasificación de los píxeles. Finalmente, el porcentaje de cobertura fue calculado sobre la imagen resultante y la dimensión fractal se estimó usando el método de conteo de cajas. Los resultados de este proceso alcanzaron un valor de correlación de Pearson de 0.67 respecto al porcentaje de cobertura.

Es importante notar que los estudios anteriores tienen más de 10 años de antigüedad, por lo cual las técnicas utilizadas corresponden a modelos clásicos de *machine learning*, u otras técnicas similares de procesamiento de imágenes. Estás metodologías requieren de intervención humana o dependen de un análisis de extracción de características específicas de las imágenes, para poder obtener descriptores con los que se entrenan los modelos, o softwares que facilitan ese proceso.

En los últimos años, y debido a la escasez de datos etiquetados de *Beggiatoa*, [Chen et al., 2022] utilizó un método de *self-training* para estimar el porcentaje de cobertura de *Beggiatoa* en una imagen del fondo marino. El método consiste en una combinación de *Otsu thresholding* [Otsu, 1979] y *Fully Convolutional Networks* (*FCN*), obteniendo una *precision* de 74.35 %. Sin embargo, recientemente, [Chen et al., 2024] propusó un método de clasificación de imágenes basado en *ensemble learning*. Este método consiste en la combinación de 3 modelos para extraer características de las imágenes (uno de ellos una *CNN*) y 4 modelos de clasificación múltiple, obteniendo un *accuracy* de 91.4 %, *precision* de 91.5 %, *recall* de 91.4 % y *f1-score* of 91.4 %.

Finalmente, [Castillo, 2023] realizó su investigación con el objetivo de construir un modelo de *CNN* que detecte la presencia de *Beggiatoa* en imágenes, y utilizarlo para segmentar videos del fondo marino de forma temporal. Los datos incluyen 64 videos del fondo marino bajo zonas acuícolas previamente segmentados por expertos. El modelo entrenado consiste en una *VGG-16* para extraer descriptores de la imagen, y luego una *FNN*

para realizar la clasificación. En el proceso de entrenamiento se realizó preprocesamiento de datos el cual consistió en técnicas de *data augmentation*, centrar los canales RGB y escalar a 224x224 píxeles. El modelo final reportó un desempeño de 88.58 % *precision*, 88.57 % *recall*, 88.57 % *f1-score*, y 88.57 % *accuracy* en el subconjunto de validación.

2.2. Detección en imágenes submarinas

Se encontraron varios estudios recientes que realizan la tarea de clasificación en imágenes del fondo marino e imágenes bajo el mar ([Langenkämper et al., 2020, Mahmood et al., 2018, Politikos et al., 2021, Rimavicius and Gelzinis, 2017, Villon et al., 2018, Zurowietz et al., 2018, Mahmood et al., 2020, Mittal et al., 2022]). Todos los trabajos encontrados tienen en común que utilizaron el mismo modelo para la clasificación de imágenes, *i.e.*, *Convolutional Neural Networks* (*CNNs*). A continuación se detallan los 3 principales.

[Rimavicius and Gelzinis, 2017] realiza un estudio comparando métodos de aprendizaje profundo para clasificar imágenes del fondo marino. Para ello, utilizó videos de un vehículo operado de manera remota que extrajo imágenes de 960x540 píxeles. Estas imágenes fueron segmentadas y etiquetadas, por un experto, en una o más de las 5 posibles categorías. Para simplificar el problema, a partir de las imágenes originales, se extrajeron 4,859 *sub-imágenes* de 50x50 píxeles, donde cada sub-imagen pertenece a una sola categoría; finalmente, se aplicó rotaciones de 90° obteniendo un conjunto final de 18,356 *sub-imágenes*. Finalmente, al comparar un modelo con características manuales y las obtenidas por una *CNN*, la segunda obtiene los mejores resultados, con un *accuracy* de 88.74 %.

[Mahmood et al., 2018] propone un algoritmo de aprendizaje profundo con el objetivo de realizar anotaciones automáticas de especies de coral en imágenes submarinas. Dentro de su estudio se comparan algoritmos de detección de coral (agrupando todas las especies en una sola clase), lo que corresponde a una tarea de clasificación binaria en imágenes. Para realizar lo anterior, se utilizaron 3,749 imágenes, capturadas por un vehículo submarino autónomo en el oeste de Australia, con 237,923 píxeles etiquetados a lo largo de 3 años (2011-2013). Se utilizaron 2 métodos de extracción de características basados en CNNs: uno con VGG-16 pre-entrenada con Imagenet y *fine tuned* con Benthoz15 ([Bewley et al., 2015]), en la cual se extrajeron los vectores de activación de la primera capa totalmente conectada; y el otro con Resnet-152 pre-entrenada con Imagenet extrayendo el vector de salida de la última capa convolucional llamado *Resfeats*. Además, se utilizaron 3 métodos de clasificación: SVM, FNN (2 capas ocultas), y sCNN (4 capas). Finalmente, se concluye que los modelos que utilizan *Resfeats* para extraer características

tienen un mejor desempeño.

[Mahmood et al., 2020] tiene el objetivo de determinar un nuevo método para extraer características de imágenes submarinas (que son utilizadas para realizar tareas de clasificación). Para ello utilizó los siguientes *datasets* de imágenes submarinas: EILAT y RSMAS [Shihavuddin et al., 2013], MLC [Beijbom et al., 2012], y Benthoz15 [Bewley et al., 2015]. El método propuesto, para extraer descriptores de una imagen, consiste en: Primero, a partir de una red *Resnest50*, preentrenada con *Imagenet*, extraer los vectores de salida de los 3 últimos bloques convolucionales; luego, a cada uno de estos vectores obtenidos realiza *max-pooling*; y, finalmente, los concatena obteniendo un vector descriptor de la imagen el cual llama *cResfeats*. Para medir el desempeño de los descriptores obtenidos utiliza el algoritmo *SVM* en la tarea de clasificación, para así poder compararlos con otros métodos propuestos en la literatura utilizados en tareas de clasificación. Concluye que el uso de *cResfeats* tiene un mejor desempeño que el obtenido en [Mahmood et al., 2018] y que los descriptores obtenidos directamente por *CNNs*.

Capítulo 3

Objetivo

El objetivo principal de esta tesis es crear un modelo para la detección de tapetes microbianos de *Beggiatoa* en imágenes del fondo marino, para que, en un futuro, pueda ser utilizado para segmentar temporalmente videos en busca de *Beggiatoa*.

Como objetivos secundarios para poder completar el desarrollo de esta tesis:

- Obtener imágenes del fondo marino bajo zonas de acuicultura etiquetadas, es decir, indicando si tienen o no la presencia de tapetes microbianos de *Beggiatoa*.
- Generar particiones de datos que contengan videos independientes para evitar sesgos en la evaluación.
- Entrenar modelos, que no presentan sobreentrenamiento, con distintos hiperparámetros, arquitectura, *fine tuning*, y umbrales.
- Evaluar y comparar el modelo entrenado, con el modelo actual del estado del arte.

Capítulo 4

Metodología

Para el desarrollo de esta tesis, es decir, la creación de un modelo capaz de detectar la presencia de tapetes microbianos de *Beggiatoa* en imágenes del fondo marino, se utilizó la metodología descrita en [Feelders et al., 2000]. Esta metodología fue adaptada para nuestro caso específico de aprendizaje profundo en *computer vision*, obteniendo las siguientes etapas: adquisición y procesamiento de datos de datos, y aplicación y evaluación de modelos. A continuación se procederá a describir en forma genérica cada una de estas etapas, y luego en la subsección correspondiente se explicarán los detalles.

El primer paso fue la adquisión de los videos y la identificación manual de los intervalos de interés descrita en informes. Con esta información se extrajeron cuadros (imágenes) de los videos con su etiqueta respectiva, correspondiente a si se encuentra la presencia de tapetes de *Beggiatoa* (Verdadero ó Falso). En este proceso también se mantuvo cuidado en mantener una equidad de las clases y evitar la replicación de imágenes consecutivas.

Luego, en la etapa de aplicación y evaluación de modelos, comienza la búsqueda de hiperparámetros y arquitectura de red neuronal con el fin de obtener el mejor desempeño posible. Para evaluar correctamente el modelo se aplicó *k-folds cross validation*, y se analizó que los modelos aprendidos no sufran de sobreentrenamiento comparando los errores de entrenamiento y prueba.

4.1. Adquisición de videos

Los datos otorgados por SERNAPESCA consisten en 114 videos de los años 2020 y 2021 de 30 centros de acuicultura ubicados a lo largo Chile. Los registros visuales se realizaron por medio de grabación subacuática de los componentes del mega bentos, con especial atención en la presencia de bacterias filamentosas (cubiertas de microorganismos visibles) y/o burbujas de gas, conforme a los procedimientos definidos por SERNAPESCA

que se indican a continuación:

1. Equipo:
    - La grabación subacuática se podrá realizar por buceo o por sistema remoto; y
    - El equipo debe contar con lente gran angular (120º ó más) y con la capacidad de grabar con buena luminosidad (natural o artificial) y foco adecuado. Además se deberá contar con un sistema de iluminación auxiliar y con un sistema fotométrico automático para regular la abertura del foco y permitir el ingreso de la luz adecuada.
    - Las cámaras a ser utilizadas en la filmación subacuática deben ser cámaras digitales a color, con una alta definición de al menos HD o full HD.
    - Se deberán realizar mantenciones periódicas a los equipos que aseguren la buena calidad de las imágenes de la cámara.
2. Tomas de datos:
    - Durante la filmación se deberá mantener una distancia de filmación entre cámara y fondo, así como una velocidad de arrastre de la cámara tal, que se asegure una filmación de calidad del fondo, que permita la observación nítida y adecuada para la correcta distinción de los distintos componentes del mega-bentos y la identificación de cubiertas de microorganismos y burbujas de gas.
    - La grabación no debe ser editada y se debe entregar una copia con la grabación ininterrumpida desde la superficie antes de la inmersión. Para el caso de la caracterización preliminar de sitio, las transectas deberán tener, como mínimo, 10 minutos de grabación efectivos del sustrato (no se considerará el tiempo de grabación del descenso y ascenso de la cámara).

Para mayor información respecto con lo anterior ver la resolución exenta N° 3612 de la subsecretaria de pesca del ministerio de economía del gobierno de Chile, en donde se encuentran todos los detalles respecto a la información anterior.

4.2. Transformación a set de datos

En primer lugar, expertos realizaron informes de los videos (sección 4.1) describiendo en que instantes encuentran momentos de interés, es decir, presencia de tapetes microbianos

de *Beggiatoa*. Luego, expertos de SERNAPESCA verificaron dichos informes con los videos y en caso de ser necesario re-definieron los momentos de interés en un nuevo documento.

Con esta información, a través de un *script* en *Python*, se realizó la extracción automática de cuadros con su etiqueta correspondiente (ver pseudocódigo en algoritmo 1). Lo anterior fue realizado extrayendo para cada video sus *Frames Per Second* (FPS) y los cuadros del video (imágenes que conforman el video). Luego se recorrieron los cuadros del video, si el cuadro contiene presencia de *Beggiatoa* se avanza FPS cuadros en el video; en el caso contrario se avanzaban $5 \cdot FPS$ cuadros. Esto se utilizó para generar distancia suficiente entre las imágenes para evitar un posible sobreentrenamiento.

```
1  para cada video hacer
2      abrir video;
3      extraer FPS y cuadros del video;
4      mientras no se acaben los cuadros del video hacer
5          si cuadro tiene Beggiatoa entonces
6              guardar cuadro con etiqueta Verdadero;
7              avanzar el video (FPS) cuadros;
8          en otro caso
9              guardar cuadro con etiqueta Falso;
10             avanzar el video (5 · FPS) cuadros;
11         fin
12     fin
13 fin
```

Pseudocódigo 1: Extracción de cuadros y su etiqueta

4.3. Evaluación de modelos

Para evaluar los modelos, estimaremos la media y el error de desviación estándar a través de validación cruzada de *k-fold*, utilizando diferentes medidas de desempeño. La validación cruzada de *k-fold* nos ayuda a estimar la distribución de errores al dividir los conjuntos de datos en k subconjuntos que tenga una distribución similar en las clases. Después se seleccionan $k-1$ folds de entrenamiento y se prueba en el restante. Este proceso se repite hasta que todos los subconjuntos se utilizan para las pruebas, y la estimación del error viene dada por el promedio y la desviación estándar de los obtenidos en cada set.

En esta tesis se utilizó validación cruzada *k-fold* con 5 subconjuntos de cuadros. Para evitar *data-leaking* (uso de datos de test en los datos de entrenamiento), se tomo en

consideración el video al que pertenecen los cuadros realizar la división. Ya que una separación completamente aleatoria de los datos podría mezclar imágenes similares de un mismo video. Para esto cada video y sus cuadros obtenidos en el paso anterior, fue asociado a un subconjunto (*fold*) 1, 2, 3, 4 ó 5 siguiendo un orden alfabético, es decir, el primer video al 1, el segundo al 2, el tercero al 3, el cuarto al 4, el quinto al 5, el sexto al 1, y así sucesivamente. Lo cual será utilizado para los siguientes experimentos.

Las medidas de desempeño que se utilizarán en esta tesis son: *Accuracy*, *Recall*, *Precision*, y *F2-score*. Estas medidas se basan en la matriz de confusión, que es una tabla que muestra las rendimiento de un modelo de clasificación utilizando cuatro valores: Verdaderos Positivos (VP) es el número de cuadros con presencia de *Beggiatoa* clasificados correctamente; Verdaderos Negativos (VN) es el número de cuadros de datos sin presencia de *Beggiatoa* correctamente clasificados; Falsos positivos (FP) es el número de cuadros sin presencia de *Beggiatoa* de datos clasificados incorrectamente como con presencia de *Beggiatoa*; y Falso Negativos (FN) que es el número de cuadros con presencia de *Beggiatoa* clasificados incorrectamente como sin presencia de *Beggiatoa*.

Accuracy (ecuación 4.1), una de las medidas más básicas de evaluación de modelos, indica el porcentaje de cuadros correctamente clasificados (VP+VN), sobre el número total de datos. El *Accuracy* varía entre 0 y 1, donde un alto *Accuracy* implica que el modelo puede predecir la mayoría de los cuadros de datos correctamente.

$$Accuracy = \frac{VP + VN}{VP + VN + FP + FN} \tag{4.1}$$

La *Precision* (ecuación 4.2) es el número de datos con *Beggiatoa* predichos correctamente sobre el total de cuadros clasificados con presencia de *Beggiatoa* (VP+FP). La *Precision* varía entre 0 y 1, donde una alta *Precision* implica que la mayoría de los cuadros clasificados con presencia de *Beggiatoa* se clasifican correctamente. Sin embargo, si tenemos un alto valor de FN este no disminuye el *Precision*.

$$Precision = \frac{VP}{VP + FP} \tag{4.2}$$

El *Recall* (ecuación 4.3) es el número de datos con *Beggiatoa* predichos correctamente sobre el total de cuadros con presencia de *Beggiatoa* (VP+FN). El *Recall* varía entre 0 y 1, donde un alto *Recall* implica que la mayoría de los cuadros con presencia de *Beggiatoa* están correctamente clasificados. Sin embargo, si tenemos un alto valor de FP este no disminuye el *Recall*.

$$Recall = \frac{VP}{VP + FN} \tag{4.3}$$

Para resolver los problemas de *Recall* y *Precision* (altos valores omitiendo uno de los errores), utilizamos el F_β-*score* (ecuación 4.4). El F_β-*score* es una medida que armoniza *Precision* y el *Recall*, buscando un balance entre ambas dado por el valor de β. El valor de β se elige de modo que el *Recall* se considere β veces más importante que la *Precision*.

El F_β-*score* varía entre 0 y 1, donde un F_β-*score* alta implica que el modelo puede clasificar la clase positiva y, lo más importante, generar un número bajo de FN y FP. Aunque VP está asociado con la clase con menos etiquetas, reportaremos el F_β-*score* usando ambas clases como VP, evitando interpretaciones subjetivas de los errores.

$$F_{\beta-score} = (1+\beta^2) \cdot \frac{Precision \cdot Recall}{(\beta^2 \cdot Precision) + Recall} \tag{4.4}$$

Debido a la naturaleza del problema, es de vital importancia que los FN (imágenes donde hay presencia de *Beggiatoa* pero el modelo las clasifica como sin) sean lo menor posible. En consecuencia, el *Recall* debe tener mayor importancia que la *Precision*. Por lo cual se utilizó el *F-score* con $\beta = 2$ para medir el desempeño de los modelos, ya que así el *Recall* se considera el doble de importante que *Precision*.

4.4. Modelos de clasificación

En esta tesis se utilizó el modelo *VGG-16* correspondiente a una red neuronal convolucional (*CNN*, [Simonyan and Zisserman, 2014]) para realizar la extracción de características de las imágenes. Posterior al proceso de extracción de características, la CNN utliza una red neuronal feed forward (*FNN*) que aprende de las características extraídas para finalmente generar la clasificación de las imágenes. Por ello, se realizó una búsqueda de arquitectura de la red FNN y de sus hiperparámetros. Una vez seleccionado el mejor modelo (arquitectura e hiperparámetros), se procedió a entrenar nuevamente el modelo final utilizando *fine tunning*. Es decir, también se ajustaron los pesos de la CNN correspondiente al proceso de extracción de características. Finalmente se realizó una búsqueda del umbral de clasificación buscando mejorar las métricas de desempeño.

A continuación se detalla cada uno de los subprocesos de la búsqueda del modelo de clasificación mencionados antes.

4.4.1. Extracción de características

En forma resumida, las CNN realizan un proceso de extracción de características de las imágenes, las que posteriormente son aprendidas por una FNN para poder determinar la clase a las que pertenecen. En esa tesis se utilizó la *CNN VGG-16* pre-entrenada con

las imágenes de *Imagenet* [Deng et al., 2009], por lo que extrae patrones aprendidos de las imágenes de *Imagenet*. A partir de esta, se utilizó el vector de salida de la última capa convolucional como vector de características, de dimensión [7,7,512], para después realizar la clasificación.

4.4.2. Red neuronal feed-forward

A partir de los vectores de características obtenidos en la subsección 4.4.1 se utilizó una *FNN* para realizar la clasificación de las imágenes. Para esto, se realizó una búsqueda de hiperparámetros, incluyendo la arquitectura de la *FNN*. Específicamente, se modificó la cantidad de neuronas, cantidad de capas, y cantidad de épocas. La capa se salida siempre consistió en una neurona con función de activación sigmoide.

Hiperparámetros

En la primera búsqueda de hiperparámetros se fijo una arquitectura de 128 neuronas en la capa oculta, seguida de un *dropout* de 0.5, y un máximo de 20 épocas de entrenamiento, variando los siguientes hiperparámetros: a) *learning rate*: 0,001 y 0,0001; b) *batch size*: 16, 64 y 256; c) *data augmentation*: con y sin data augmentation; y d) *optimizer*: Adam y SGD. Esta búsqueda resultó en 24 modelos distintos mostrados en el cuadro 4.4.1. El la columna de la derecha, se muestra un identificador de cada modelo para poder hacer refereencias de los mismos.

Para seleccionar la cantidad de épocas de cada modelo, con el fin de evitar el sobre entrenamiento, se utilizó la función de perdida. Para ello, se seleccionó la época en que el valor para el conjunto de entrenamiento era menor que el de prueba (cuando se cruzaban gráficamente) o cuando en las dos épocas siguientes no se encuentre una mejora superior a 0.01 para la métrica en el conjunto de prueba.

Arquitectura

Luego, a partir de la mejor selección de hiperparámetros (modelo **X**), se realizaron cambios en cuanto a las capas ocultas y sus neuronas, pero se mantuvo siempre la función de activación *relu* y posterior a cada capa oculta se encontraba un *dropout* de 0.5. Se realizó un búsqueda considerando un valor para la primera capa desde 32 hasta 1,024 aumentando exponencialmente; y, en cuanto a la segunda capa, se considero: su ausencia, una cantidad de neuronas igual a la primera capa o la mitad de esta. En resumen, se analizaron 18 modelos distintos, que se muestran en el cuadro 4.4.2. Notar que el modelo X.7 es equivalente al modelo 9 de 4.4.1

batch size	*optimizer*	*learning rate*	*data aumentation*	**Modelo**
16	Adam	0.0001	Sí	**1**
			No	**2**
		0.001	Sí	**3**
			No	**4**
	SGD	0.0001	Sí	**5**
			No	**6**
		0.001	Sí	**7**
			No	**8**
64	Adam	0.0001	Sí	**9**
			No	**10**
		0.001	Sí	**11**
			No	**12**
	SGD	0.0001	Sí	**13**
			No	**14**
		0.001	Sí	**15**
			No	**16**
256	Adam	0.0001	Sí	**17**
			No	**18**
		0.001	Sí	**19**
			No	**20**
	SGD	0.0001	Sí	**21**
			No	**22**
		0.001	Sí	**23**
			No	**24**

Cuadro 4.4.1: Definición de los modelos y sus hiperparámetros

Fine tuning

Luego con la arquitectura seleccionada (modelo X.Y), se procede a realizar *fine tuning*. Este proceso consiste en descongelar neuronas del modelo *VGG-16* utilizado para extracción de características (habilitar la actualización de la neurona) e incorporarlas en el proceso de entrenamiento. En total se compararon 5 formas de *fine tuning* distintas, las cuales consistían en ir descongelando uno por uno progresivamente los bloques convolucionales, desde el último bloque hasta el primero.

En resumen, se analizaron 5 modelos distintos, que se muestran en el cuadro 4.4.2. Cada uno definido por la cantidad de bloques descongelados a partir de la última capa. Por ejemplo, el modelo X.Y.*i* tiene descongelados los últimos *i* bloques convolucionales de la red *VGG-16*.

Primera capa	Segunda capa	**Modelo**
32	0	**X.1**
	16	**X.2**
	32	**X.3**
64	0	**X.4**
	32	**X.5**
	64	**X.6**
128	0	**X.7**
	64	**X.8**
	128	**X.9**
256	0	**X.10**
	128	**X.11**
	256	**X.12**
512	0	**X.13**
	256	**X.14**
	512	**X.15**
1,024	0	**X.16**
	512	**X.17**
	1,024	**X.18**

Cuadro 4.4.2: Definición de los modelos y sus capas ocultas

Cantidad de bloques descongelados	**Modelo**
1	**X.Y.1**
2	**X.Y.2**
3	**X.Y.3**
4	**X.Y.4**
5	**X.Y.5**

Cuadro 4.4.3: Definición de los modelos y su cantidad de bloques descongelados

Umbral de decisión

A partir del modelo seleccionado en la subsección anterior, se realizó un nuevo proceso de entrenamiento pero guardando el vector de probabilidades, no el de clasificación. Con este vector se realizó una búsqueda del mejor umbral de decisión con un precisión de 10^{-3}. Específicamente, se analizó el umbral de predicción positiva de cada imagen variando la probabilidad de corte desde 0.000 a 1.000 en intervalos de 0.001.

Capítulo 5

Resultados

En este capítulo se muestran y discuten los resultados de los experimentos definidos en las secciones 4.2 y 4.4. En el primero, se muestran las imágenes obtenidas y los subconjuntos de ellas. Luego, se realiza la búsqueda del modelo de clasificación para imágenes, donde se divide en búsqueda de hiperparámetros (ver 5.2.1), búsqueda de arquitectura (ver 5.2.2), *fine tunning* (ver 5.2.3), y búsqueda de umbral de decisión (ver 5.2.4). Finalmente, se compara el modelo seleccionado con el modelo determinado por [Castillo, 2023] (ver 5.2.5).

Todos los resultados de esta sección fueron obtenidos a través de la metodología explicada anteriormente. En resumen, todos los resultados corresponden a la media y desviación estándar de cada métrica, que se estimaron usando *5-fold cross-validation*. Las métricas de desempeño son *F2-score*, *Precision*, *Accuracy*, y *Recall*. Además, para analizar sobreentramiento se utilizó la función de pérdida del modelo de clasificación.

5.1. Set de datos

A partir del procedimiento de extracción de imágenes de los videos obtenidos por SERNAPESCA, definido en la sección 4.2, se extrajeron un total de 40,774 imágenes, donde 19,511 corresponden a imágenes con presencia de *Beggiatoa*, y fueron divididas en los siguientes subconjuntos:

- Subconjunto 1: Corresponde a 7,847 imágenes, donde 3,943 corresponden a imágenes con presencia de *Beggiatoa*.
- Subconjunto 2: Corresponde a 8,394 imágenes, donde 3,763 corresponden a imágenes con presencia de *Beggiatoa*.

- Subconjunto 3: Corresponde a 8,908 imágenes, donde 4,192 corresponden a imágenes con presencia de *Beggiatoa*.

- Subconjunto 4: Corresponde a 7,893 imágenes, donde 3,398 corresponden a imágenes con presencia de *Beggiatoa*.

- Subconjunto 5: Corresponde a 7,732 imágenes, donde 4,216 corresponden a imágenes con presencia de *Beggiatoa*.

5.2. Búsqueda modelo de clasificación

La búsqueda del modelo de clasificación comenzó a partir de la arquitectura utilizada en [Castillo, 2023], a la cual se le realizaron variaciones en cuanto a los hiperparámetros utilizados. Luego, una vez seleccionados los mejores hiperparámetros, se estudio si el uso de una arquitectura distinta con ese set de hiperparámetros mejoraba el desempeño. Después, teniendo seleccionada la mejor arquitectura se agrego la técnica de *fine tunning* para evaluar su impacto en el desempeño. Finalmente, se realizó una búsqueda exhaustiva del umbral de decisión para realizar la clasificación.

5.2.1. Búsqueda hiperparámetros

La búsqueda de hiperparámetros consiste en comparar el desempeño de modelos con distintos valores de hiperparámetros para encontrar el mejor de ellos. Como se mencionó en 4.4 se analizaron los hiperparámetros *batch size*, *optimizer*, *learning rate*, y *data augmentation*, generando 24 modelos distintos (ver cuadro 4.4.1). El proceso de búsqueda se separa en dos partes, determinar la cantidad de épocas de entrenamiento para cada modelo, y comparar y seleccionar el modelo con los mejores hiperparámetros. Para las épocas de entrenamiento se analizó la curva de la función de perdida de aprendizaje a medida que aumentaban las épocas de entrenamiento, donde cada modelo se entrenó con 20 épocas. Para comparar los modelos se utilizaron el promedio y desviación estándar alcanzados por el *F2-score* en los subconjuntos de entrenamiento y prueba.

Para cada modelo se determinó, de acuerdo con lo definido en 4.4, el mejor número de épocas de entrenamiento para evitar sub y sobre entrenamiento. El número de épocas seleccionado para cada caso se muestra en la tabla A.1.1, y se puede observar gráficamente en la figura 5.1 para el caso del modelo 18 (los gráficos de cada uno de los otros modelos se encuentran en la subsección A.1.1).

En la figura 5.1 podemos observar la evolución de la función de perdida para el conjunto de entrenamiento (azul) y prueba (rojo) a medida que avanzan las épocas de entrenamiento del modelo. La linea punteada negra representa la cantidad de épocas seleccionadas, para los hiperparámetros correspondiente, en este caso 4. Como se puede observar hacia la izquierda de la linea negra punteada, el modelo se encuentra subentrenado (el conjunto de prueba tiene un valor significativamente menor que el entrenamiento). Por otro lado, una vez que el modelo supera las 4 épocas de entrenamiento, se observa un aumento en la diferencia entre las funciones de perdida de ambos modelos. Esto corresponde a un sobreajuste del modelo al conjunto de datos de entrenamiento. Por lo mismo, se selecciona el modelo con 4 épocas de entrenamiento.

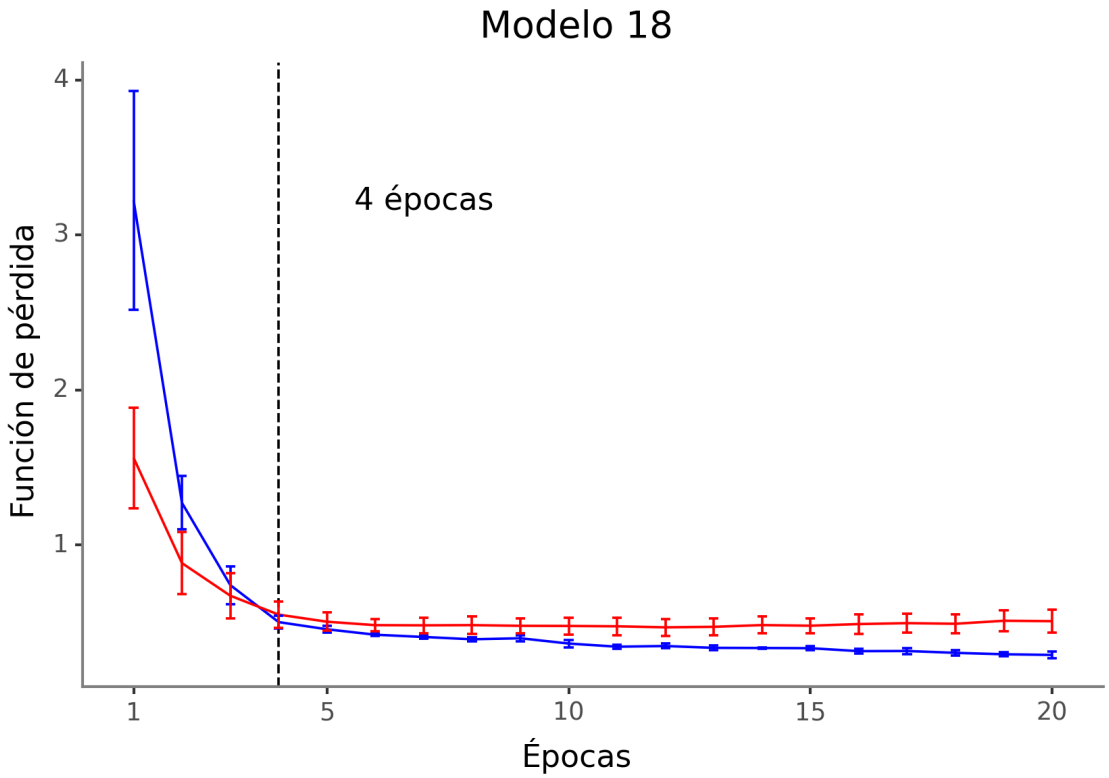


Figura 5.1: Función de perdida modelo 18 a partir de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo), y época seleccionada (línea punteada negra).

Una vez seleccionadas las épocas de entrenamiento para cada modelo (cuadro A.1.1), se procede a analizar el desempeño de cada uno estos según el *F2-score*. Los resultados de cada modelo se presentan en la figura 5.2 (valores exactos en anexo A.2), donde se muestra el promedio (círculo) y desviación estándar (barras) de los datos de entrenamiento (azul) y prueba (rojo) para cada uno de los 24 modelo descritos en el cuadro A.1.1.

Con estos valores en consideración, se puede concluir que, en general, el desempeño

empeora significativamente al usar las técnicas de *data augmentation*. Esto lo podemos observar en la figura 5.2, ya que en la mayoría de los casos, al comparar los modelos impares (con *data augmentation*) con el modelo que le sigue (mismos hiperparámetros que el impar pero sin *data augmentation*) notamos que el segundo tiene un mejor desempeño. Esto se podría explicar debido a que las técnicas de *data augmentation* generan un conjunto con mayor diversidad y variabilidad con el objetivo de robustecer el modelo ante imágenes en distintas circunstancias (punto de vista, escala, luz, etc.); sin embargo, en el caso de la detección de *Beggiatoa* estas técnicas podrían generar circunstancias que no se dan en la realidad, lo que podrían explicar el impacto negativo en el desempeño. Por ejemplo, si una imagen tiene *Beggiatoa* en una de las esquinas, al aplicar algún proceso de data augmentation se podría eliminar la *Beggiatoa*. En este caso, se mantendría la etiqueta como positiva; sin embargo, la verdadera etiqueta debería ser negativa.

En el resto de los hiperparámetros no se encuentra un tendencia generalizada. Además, no se observa que exista un modelo que sea significativamente mejor que todos los otros (en términos estadísticos). Sin embargo, el modelo 20 tiene el mejor valor promedio de *F2-score* tanto para el conjunto de entrenamiento como el de prueba y poseen desviaciones estándar pequeñas. Es por este motivo que el modelo 20 es el seleccionado para la siguiente etapa (batch size 256, optimizer ADAM, learning rate 0.001, y sin data augmentation).

5.2.2. Búsqueda arquitectura

A partir del modelo obtenido de la selección de hiperparámetros, se analizó una búsqueda de arquitectura de la *FNN*, realizando cambios en las cantidad de neuronas de las capas ocultas, resultando en 18 nuevos modelos (cuadro 4.4.2). Tal como en la subsección anterior, se analizó la cantidad de épocas de entrenamiento para cada uno de estos modelos, cuyos resultados se muestran en el cuadro A.1.2.

Una vez seleccionadas las épocas de entrenamiento para cada modelo (cuadro A.1.2), se procede a analizar el desempeño de cada uno estos según el *F2-score*. Los resultados de cada modelo se presentan en la figura 5.3 (valores exactos en cuadros A.2.1 y A.2.2), donde se muestra el promedio (círculo) y desviación estándar (barras) de los datos de entrenamiento (azul) y prueba (rojo) para cada uno de los 18 modelo descritos en el cuadro A.1.1.

Al observar la figura 5.3 podemos notar el nulo desempeño de los modelos 20.2, 20.3, y 20.6. Es más, para los modelos con una primera capa oculta menor a 128 neuronas (20.1-20.6) la incorporación de otra capa oculta disminuye su desempeño. Por otro lado, al observar los modelos restantes, destacan los modelos 20.8 y 20.11 por su alto valor

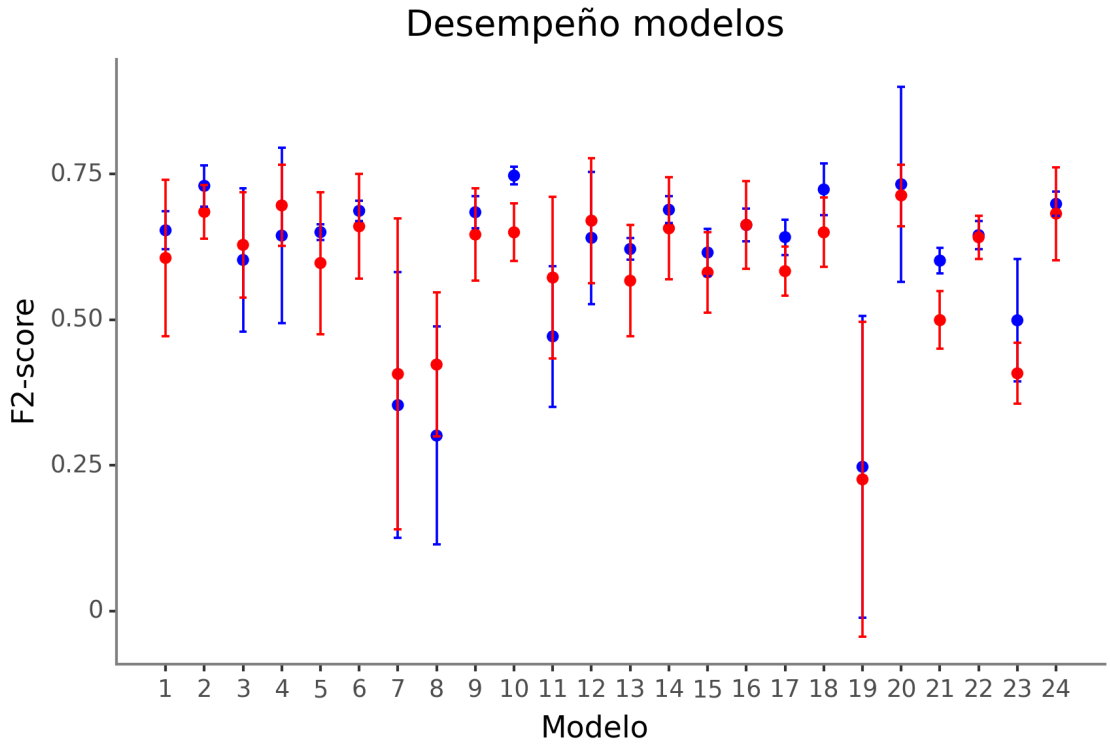


Figura 5.2: Desempeño modelos búsqueda de hiperparámetros. Valor promedio (círculo) y desviación estándar (barras) de *F2-score* para datos de entrenamiento (azul) y prueba (rojo).

promedio y pequeña desviación estándar. Al comparar todos los modelos, no se observa que alguno sea significativamente mejor que otro en términos estadísticos. No obstante, el modelo 20.8 posee un valor promedio de *F2-score* más alto en el conjunto de prueba y entrenamiento. Es por este motivo que el modelo 20.8 es el seleccionado para la siguiente etapa.

5.2.3. *Fine tunning*

En base al modelo seleccionado en la búsqueda de arquitectura, se definieron cinco nuevos modelos que unen la extracción de características (*CNN*) y la clasificación (*FNN*) en el proceso de entrenamiento. Estos son definidos por la cantidad de bloques convolucionales descongelados de la *CNN*.

Tal como en la subsección anterior, se analizó la cantidad de épocas de entrenamiento para cada uno de estos modelos, cuyos resultados se muestran en el cuadro A.1.3.

Una vez seleccionadas las épocas de entrenamiento para cada modelo (cuadro A.1.3), se procede a analizar el desempeño de cada uno estos según el *F2-score*. Los resultados

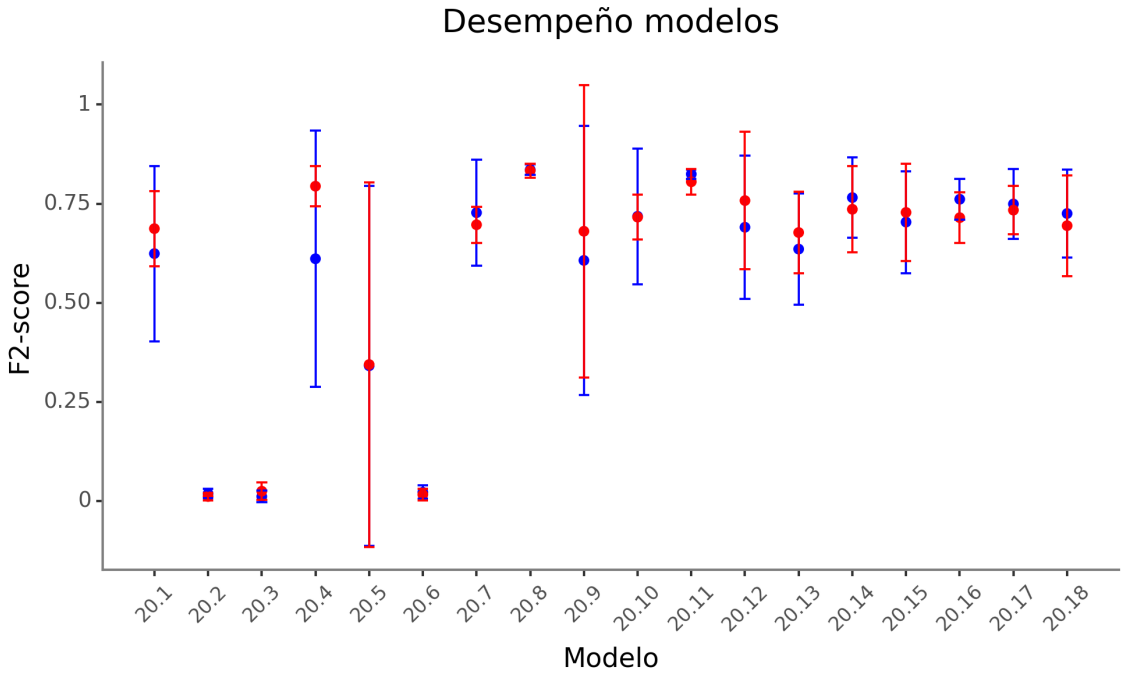


Figura 5.3: Desempeño modelos búsqueda de arquitectura red neural *feed forward*. Valor promedio (círculo) y desviación estándar (barras) de *F2-score* para datos de entrenamiento (azul) y prueba (rojo).

de cada modelo se presentan en el cuadro 5.2.1. A diferencias de los resultados anteriores, se procede a mostrar también el Precision y Recall para poder analizar y entender el comportamiento de los modelos. Como se puede observar, al introducir *fine tuning* los modelos empeoraron drásticamente, es más, tendieron a clasificar las imágenes como falsas (sin *Beggiatoa*). De modo que el modelo 20.8 se mantiene seleccionado para la siguiente etapa.

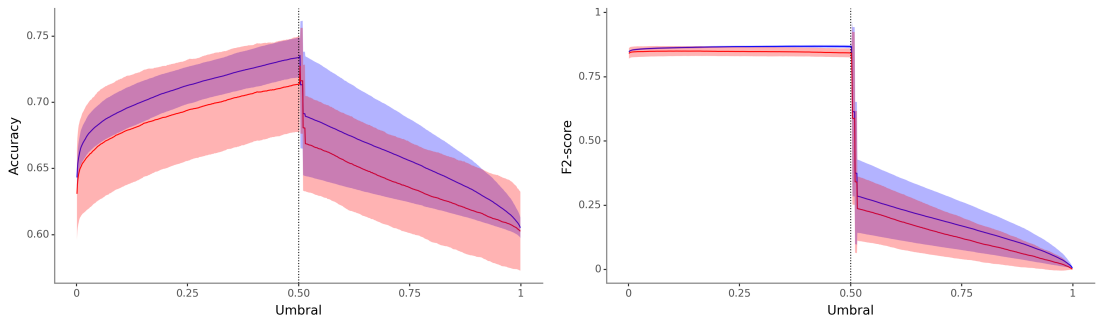
El empeoramiento dado por el uso de *fine tuning* se podría explicar por una falta de datos para poder entrenar al modelo. Al descongelar bloques convolucionales la cantidad de parámetros a entrenar aumenta, en consecuencia se necesitan más imágenes para que el modelo pueda aprender de manera correcta.

5.2.4. Búsqueda umbral de decisión

Finalmente se realizó una búsqueda del mejor umbral de decisión para la clasificación del modelo seleccionado con una precisión de 10^{-3} (modelo 20.8). En el cuadro A.2.4 se encuentra el desempeño para cada valor utilizado basado en una búsqueda binaria. Estos resultados se pueden observar gráficamente en la figura 5.4.Como se puede observar en la imagen de la derecha (figura 5.4b), el *F2-score* no varía de forma significativa para valores

Modelo	Conjunto	*Accuracy*	*Precision*	*Recall*	*F2-score*
20.8.1	entrenamiento	0.6 ± 0.02	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
	prueba	0.6 ± 0.03	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
20.8.2	entrenamiento	0.6 ± 0.01	0.11 ± 0.24	0.02 ± 0.05	0.14 ± 0.0
	prueba	0.6 ± 0.03	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
20.8.3	entrenamiento	0.59 ± 0.01	0.22 ± 0.21	0.03 ± 0.04	0.06 ± 0.04
	prueba	0.6 ± 0.03	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
20.8.4	entrenamiento	0.58 ± 0.02	0.42 ± 0.01	0.17 ± 0.12	0.19 ± 0.12
	prueba	0.6 ± 0.03	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
20.8.5	entrenamiento	0.59 ± 0.02	0.43 ± 0.03	0.1 ± 0.08	0.12 ± 0.08
	prueba	0.6 ± 0.03	0.01 ± 0.02	0.0 ± 0.0	0.0 ± 0.0

Cuadro 5.2.1: Resultados métricas obtenidas por modelos 20.8.1 al 20.8.5



(a) *Accuracy* (b) *F2-score*

Figura 5.4: Desempeño búsqueda de umbral. Valor promedio (linea) y desviación estándar (sombra) para datos de entrenamiento (azul) y prueba (rojo).

menores o igual a 0.50, obteniendo un desempeño estable. Sin embargo, el desempeño disminuye considerablemente cuando se sobrepasa este umbral. En el caso del *Accuracy*, el desempeño alcanza uno de sus mejores valores cuando es cercano a 0.50, y disminuye a medida que se aleja de este umbral. Por tanto, se mantiene el valor del umbral de 0.5 ya que los otros valores no mejoran el desempeño.

5.2.5. Comparación con modelo de Castillo (*baseline*)

Finalmente, comparamos el modelo 20.8 contra el modelo de clasificación obtenido por Castillo. Se consideraron los mismos subconjuntos definidos en 4.2 calculando las métricas de desempeño definidas en 4.3. En el cuadro 5.2.2, a modo comparativo, se encuentra el desempeño de este para cada métrica definida y el de nuestro modelo.

El desempeño alcanzado por el modelo de Castillo, en el conjunto de datos, es considerablemente menor al reportado en su tesis ($\Delta > 30\,\%$), y a su vez es menor al alcanzado

Modelo	*Accuracy*	*Precision*	*Recall*	*F2-score*
Castillo	0.52 ± 0.04	0.49 ± 0.05	0.14 ± 0.06	0.17 ± 0.06
20.8	0.72 ± 0.03	0.60 ± 0.05	0.93 ± 0.05	0.83 ± 0.02

Cuadro 5.2.2: Comparación de desempeño alcanzado entre el modelo de la tesis de Castillo y nuestra selección correspondiente al modelo 20.8

por nuestro modelo 20.8. Esto se puede explicar, de acuerdo a lo expuesto por Castillo, en la posible sensibilidad que tiene su modelo a clasificar segmentos de imágenes que puedan tener características no tan claras. Lo anterior deriva del proceso de extracción de imágenes utilizado, ya que consideraba una selección manual de imágenes nítidas, por lo que al enfrentarse a imágenes con menor nitidez no logra identificar la presencia de *Beggiatoa* correctamente.

La *Precision* muestra que el modelo de Castillo no tiene seguridad de que al detectar *Beggiatoa* esta se encuentre efectivamente en la imagen, siendo aproximadamente un 50 % de los casos falsos positivos. Por otro lado, el bajo *Recall* alcanzado por el modelo implica que es poco sensible a la presencia de *Beggiatoa*, solo logrando detectar aproximadamente un 20 % de los casos con presencia de *Beggiatoa*.

Por lo anteriormente señalado es que el *F2-score* de nuestro modelo 20.8 obtiene un desempeño significativamente mejor al modelo de Castillo, alcanzando una mejora aproximada de 0.60, en comparación al valor obtenido por el modelo de Castillo. Esto debido a que la sensibilidad, que es fundamental de acuerdo con la definición del problema, tiene una gran incidencia en el desempeño de esta métrica.

Capítulo 6

Conclusiones

Los tapetes microbianos de *Beggiatoa* son ecosistemas naturales que se encuentran presentes en los fondos marinos de zonas anóxicas. Su presencia en el fondo marino bajo zonas de producción de salmones en la industria acuícola es un indicador de alta contaminación. Actualmente, la presencia de *Beggiatoa* en salmoneras se realiza de manera manual analizando videos del fondo marino, lo que no es escalable y está sujeto al error humano.

En esta tesis se buscó un modelo capaz de detectar la presencia de tapetes bacterianos de *Beggiatoa* en imágenes del fondo marino bajo zonas de acuicultura. Para ello, se compararon modelos basados en la arquitectura *CNN VGG-16* pre entrenada con el set de datos de *Imagenet*, variando los hiperparámetros de *batch size*, *optimizer*, *learning rate*, y *data augmentation*. Además, se modificó la arquitectura (cantidad de capas y neuronas) de la red *feed forward* (*FNN*) final que realiza la tarea de clasificación, el uso de *fine-tuning*, y el valor del umbral de decisión. Se utilizaron imágenes extraídas de 114 videos de los años 2020 y 2021 de 30 centros de acuicultura ubicados a lo largo Chile, y que fueron etiquetados por expertos de SERNAPESCA.

Se demostró de forma empírica la capacidad del modelo para detectar la presencia de *Beggiatoa* en imágenes, siendo esta respaldada por el uso de *k-folds cross validation*, donde cada fold considera videos distintos para evitar el sobre ajuste del modelo al conjunto de entrenamiento. Específicamente el modelo alcanzó un *f2-score* de 0.83 ± 0.02 y una sensibilidad del 0.93 ± 0.05 en el conjunto de prueba, lo que implica que aproximadamente el modelo no logra detectar solo un 10 % de las imágenes con presencia de *Beggiatoa*.

El trabajo futuro se divide en dos áreas. La primera consiste en la búsqueda de un mejor modelo de capaz de detectar la presencia de tapetes bacterianos de *Beggiatoa*, incorporando nuevas formas de extracción de características y/o otro tipo de modelo para la clasificación. La segunda consiste en la aplicación del modelo actual para elaboración

automática de informes con intervalos de presencia de *Beggiatoa* a partir de videos. Para esta última, actualmente se encuentra en curso la confección de un algoritmo de segmentación temporal de videos a partir de la presencia de *Beggiatoa*, basado en el uso de la técnica *rolling average* en conjunto con el modelo de clasificación de imágenes obtenido.

Bibliografía

[Aguzzi et al., 2011] Aguzzi, J., Costa, C., Robert, K., Matabos, M., Antonucci, F., Juniper, S., and Menesatti, P. (2011). Automated image analysis for the detection of benthic crustaceans and bacterial mat coverage using the venus undersea cabled network. *Sensors*, 11(11):10534–10556.

[Beijbom et al., 2012] Beijbom, O., Edmunds, P. J., Kline, D. I., Mitchell, B. G., and Kriegman, D. (2012). Automated annotation of coral reef survey images. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1170–1177.

[Bewley et al., 2015] Bewley, M., Friedman, A., Ferrari, R., Hill, N., Hovey, R., Barrett, N., Marzinelli, E. M., Pizarro, O., Figueira, W., Meyer, L., et al. (2015). Australian sea-floor survey data, with images and expert annotations. *Scientific data*, 2(1):1–13.

[Castillo, 2023] Castillo, J. (2023). *Detección automática de la bacteria Beggiatoa en fondos marítimos basados en tratamiento digital de videos por medio de Redes Neuronales Convolucionales*. Tesis de magíster, Universidad Adolfo Ibañez.

[Chailloux et al., 2008] Chailloux, C., Allais, A., Simeoni, P., and Olu, K. (2008). Automatic classification of deep benthic habitats: Detection of microbial mats and siboglinid polychaete fields from optical images on the håkon mosby mud volcano. In *Oceans 2008*, pages 1–7. IEEE.

[Chen et al., 2024] Chen, Y., Zhou, Y., Park, M., Tran, S., Hadley, S., and Bai, Q. (2024). A novel adaptive ensemble learning framework for automated beggiatoa spp. coverage estimation. *Expert Systems with Applications*, 237:121416.

[Chen et al., 2022] Chen, Y., Zhou, Y., Tran, S., Park, M., Hadley, S., Lacharite, M., and Bai, Q. (2022). A self-learning approach for beggiatoa coverage estimation in aquaculture. In *Australasian Joint Conference on Artificial Intelligence*, pages 405–416. Springer.

[CIEP, 2010] CIEP (2010). Caracterización de condiciones ambientales, centros de cultivo categoría 4, región de aysén.

[Crawford et al., 2001] Crawford, C., Mitchell, I., and Macleod, C. (2001). Video assessment of environmental impacts of salmon farms. *ICES Journal of Marine Science*, 58(2):445–452.

[Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

[Douglas et al., 2017] Douglas, E., Pilditch, C., Kraan, C., Schipper, L., Lohrer, A., and Thrush, S. (2017). Macrofaunal functional diversity provides resilience to nutrient enrichment in coastal sediments. *Ecosystems*, 20(7):1324–1336.

[Elliott et al., 2006] Elliott, J., Spear, E., and Wyllie-Echeverria, S. (2006). Mats of beggiatoa bacteria reveal that organic pollution from lumber mills inhibits growth of zostera marina. *Marine Ecology*, 27(4):372–380.

[Feelders et al., 2000] Feelders, A., Daniels, H., and Holsheimer, M. (2000). Methodological and practical aspects of data mining. *Information & Management*, 37(5):271–281.

[Gray and Elliott, 2009] Gray, J. and Elliott, M. (2009). *Ecology of marine sediments: from science to management*. Oxford University Press.

[Harrison et al., 2006] Harrison, J., Turner, R., Marques, L., and Ceri, H. (2006). Biopelículas. *Investigación y Ciencia*, page 77.

[Hoeser and Kuenzer, 2020] Hoeser, T. and Kuenzer, C. (2020). Object detection and image segmentation with deep learning on earth observation data: A review-part i: Evolution and recent trends. *Remote Sensing*, 12(10):1667.

[Jerosch et al., 2007] Jerosch, K., Lüdtke, A., Schlüter, M., and Ioannidis, G. (2007). Automatic content-based analysis of georeferenced image data: Detection of beggiatoa mats in seafloor video mosaics from the håkon mosby mud volcano. *Computers & Geosciences*, 33(2):202–218.

[Ji et al., 2012] Ji, S., Xu, W., Yang, M., and Yu, K. (2012). 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231.

[Jiao et al., 2021] Jiao, L., Zhang, R., Liu, F., Yang, S., Hou, B., Li, L., and Tang, X. (2021). New generation deep learning for video object detection: A survey. *IEEE Transactions on Neural Networks and Learning Systems*.

[Kittler and Illingworth, 1985] Kittler, J. and Illingworth, J. (1985). Relaxation labelling algorithms—a review. *Image and vision computing*, 3(4):206–216.

[Knight et al., 2021] Knight, R., Verhoeven, J., Salvo, F., Hamoutene, D., and Dufour, S. (2021). Validation of visual bacterial mat assessment at aquaculture sites through abiotic and biotic indicators. *Ecological Indicators*, 122:107283.

[Langenkämper et al., 2020] Langenkämper, D., van Kevelaer, R., Purser, A., and Nattkemper, T. (2020). Gear-induced concept drift in marine images and its effect on deep learning classification. *Frontiers in Marine Science*, 7:506.

[Livingston et al., 1994] Livingston, D., Förlin, L., and George, S. (1994). Molecular biomarkers and toxic consequences of impact by organic pollution in aquatic organisms.

[Mahmood et al., 2020] Mahmood, A., Bennamoun, M., An, S., Sohel, F., and Boussaid, F. (2020). Resfeats: Residual network based features for underwater image classification. *Image and Vision Computing*, 93:103811.

[Mahmood et al., 2018] Mahmood, A., Bennamoun, M., An, S., Sohel, F., Boussaid, F., Hovey, R., Kendrick, G., and Fisher, R. (2018). Deep image representations for coral image classification. *IEEE Journal of Oceanic Engineering*, 44(1):121–131.

[Mahrishi et al., 2021] Mahrishi, M., Morwal, S., Muzaffar, A. W., Bhatia, S., Dadheech, P., and Rahmani, M. K. I. (2021). Video index point detection and extraction framework using custom yolov4 darknet object detection model. *IEEE Access*, 9:143378–143391.

[Martínez-Alonso and Gaju, 2005] Martínez-Alonso, M. and Gaju, N. (2005). El papel de los tapetes microbianos en la biorrecuperación de zonas litorales sometidas a la contaminación por vertidos de petróleo. *Ecosistemas*, 14(2).

[Matabos et al., 2011] Matabos, M., Aguzzi, J., Robert, K., Costa, C., Menesatti, P., Company, J., and Juniper, S. (2011). Multi-parametric study of behavioural modulation in demersal decapods at the venus cabled observatory in saanich inlet, british columbia, canada. *Journal of Experimental Marine Biology and Ecology*, 401(1-2):89–96.

[Mittal et al., 2022] Mittal, S., Srivastava, S., and Jayanth, J. P. (2022). A survey of deep learning techniques for underwater image classification. *IEEE Transactions on Neural Networks and Learning Systems*.

[Muhammad et al., 2021] Muhammad, K., Ullah, A., Imran, A. S., Sajjad, M., Kiran, M. S., Sannino, G., de Albuquerque, V. H. C., et al. (2021). Human action recognition using attention based lstm network with dilated cnn features. *Future Generation Computer Systems*, 125:820–830.

[Ngiam et al., 2011] Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., and Ng, A. (2011). Multimodal deep learning. In *ICML*.

[Othmani, 2022] Othmani, M. (2022). A vehicle detection and tracking method for traffic video based on faster r-cnn. *Multimedia Tools and Applications*, pages 1–19.

[Otsu, 1979] Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66.

[Pearson, 1975] Pearson, T. (1975). The benthic ecology of loch linnhe and loch eil, a sea-loch system on the west coast of scotland. iv. changes in the benthic fauna attributable to organic enrichment. *Journal of Experimental Marine Biology and Ecology*, 20(1):1–41.

[Pearson, 1978] Pearson, T. (1978). Macrobenthic succession in relation to organic enrichment and pollution of the marine environment. *Oceanogr. Mar. Biol. Ann. Rev.*, 16:229–311.

[Pearson and Stanley, 1979] Pearson, T. and Stanley, S. (1979). Comparative measurement of the redox potential of marine sediments as a rapid means of assessing the effect of organic pollution. *Marine Biology*, 53(4):371–379.

[Politikos et al., 2021] Politikos, D., Fakiris, E., Davvetas, A., Klampanos, I., and Papatheodorou, G. (2021). Automatic detection of seafloor marine litter using towed camera images and deep learning. *Marine Pollution Bulletin*, 164:111974.

[Raffaelli, 1987] Raffaelli, D. (1987). The behaviour of the nematode/copepod ratio in organic pollution studies. *Marine Environmental Research*, 23(2):135–152.

[Rimavicius and Gelzinis, 2017] Rimavicius, T. and Gelzinis, A. (2017). A comparison of the deep learning methods for solving seafloor image classification task. In *International Conference on Information and Software Technologies*, pages 442–453. Springer.

[Roerdink and Meijster, 2000] Roerdink, J. B. and Meijster, A. (2000). The watershed transform: Definitions, algorithms and parallelization strategies. *Fundamenta informaticae*, 41(1-2):187–228.

[Shihavuddin et al., 2013] Shihavuddin, A., Gracias, N., Garcia, R., Gleason, A. C., and Gintert, B. (2013). Image-based coral reef classification and thematic mapping. *Remote Sensing*, 5(4):1809–1841.

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[Szegedy et al., 2017] Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*.

[Tran et al., 2015] Tran, D., Bourdev, L., Fergus, R., Torresani, L., and Paluri, M. (2015). Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497.

[Villon et al., 2018] Villon, S., Mouillot, D., Chaumont, M., Darling, E., Subsol, G., Claverie, T., and Villéger, S. (2018). A deep learning method for accurate and fast identification of coral reef fishes in underwater images. *Ecological Informatics*, 48:238–244.

[Wu et al., 2020] Wu, X., Sahoo, D., and Hoi, S. C. (2020). Recent advances in deep learning for object detection. *Neurocomputing*, 396:39–64.

[Zhao et al., 2019] Zhao, Z.-Q., Zheng, P., Xu, S.-t., and Wu, X. (2019). Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11):3212–3232.

[Zurowietz et al., 2018] Zurowietz, M., Langenkämper, D., Hosking, B., Ruhl, H., and Nattkemper, T. (2018). Maia—a machine learning assisted image annotation method for environmental monitoring and exploration. *PLOS One*, 13(11):e0207498.

Anexo A

Anexos

A.1. Selección cantidad de épocas

A.1.1. Gráficos función de pérdida

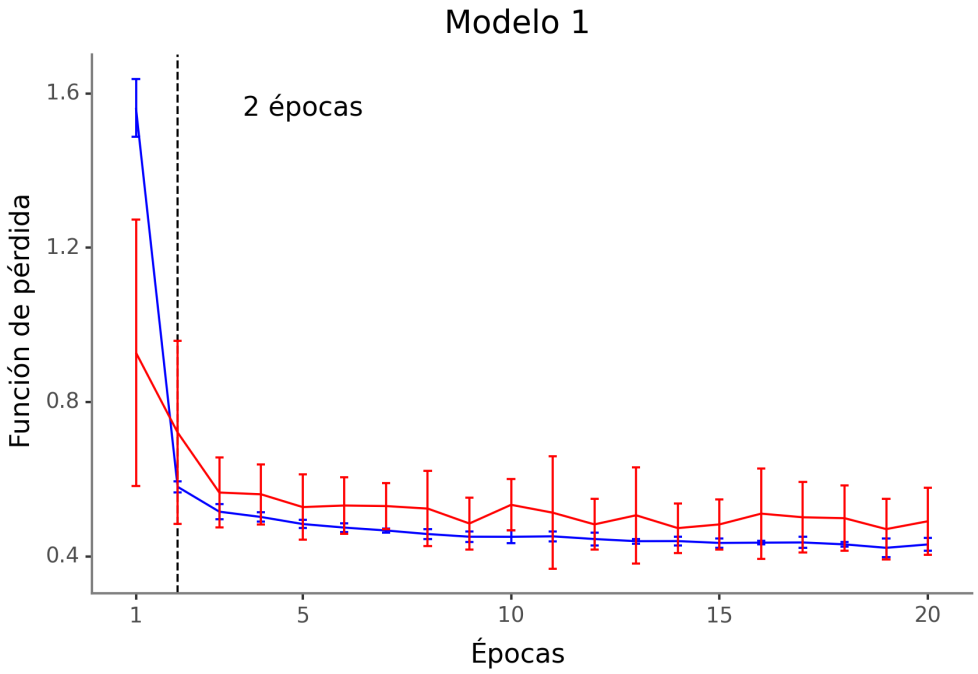


Figura A.1: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 1. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

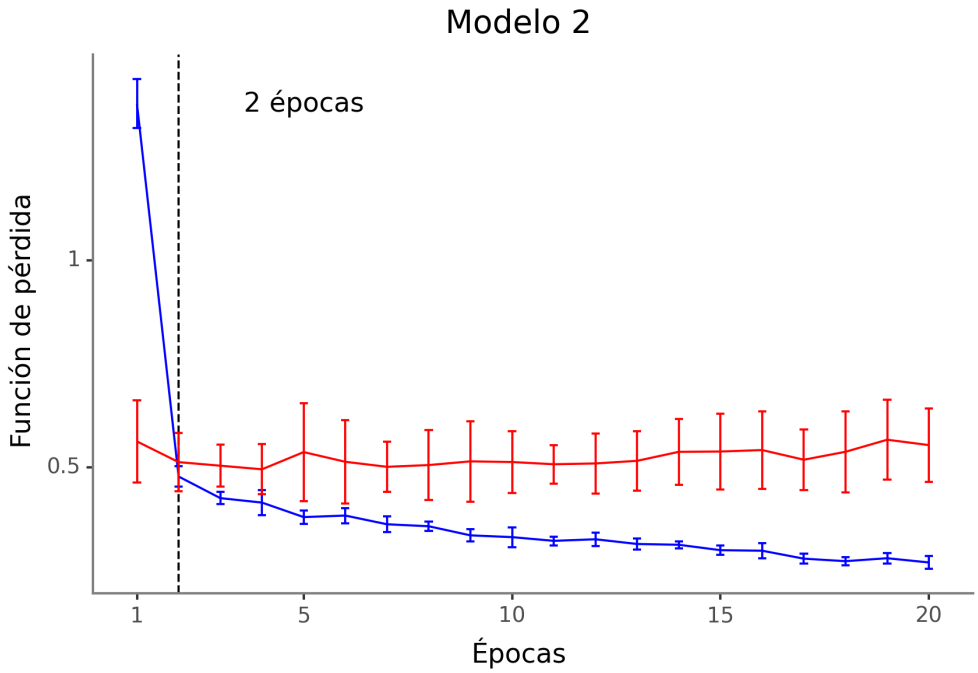


Figura A.2: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 2. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

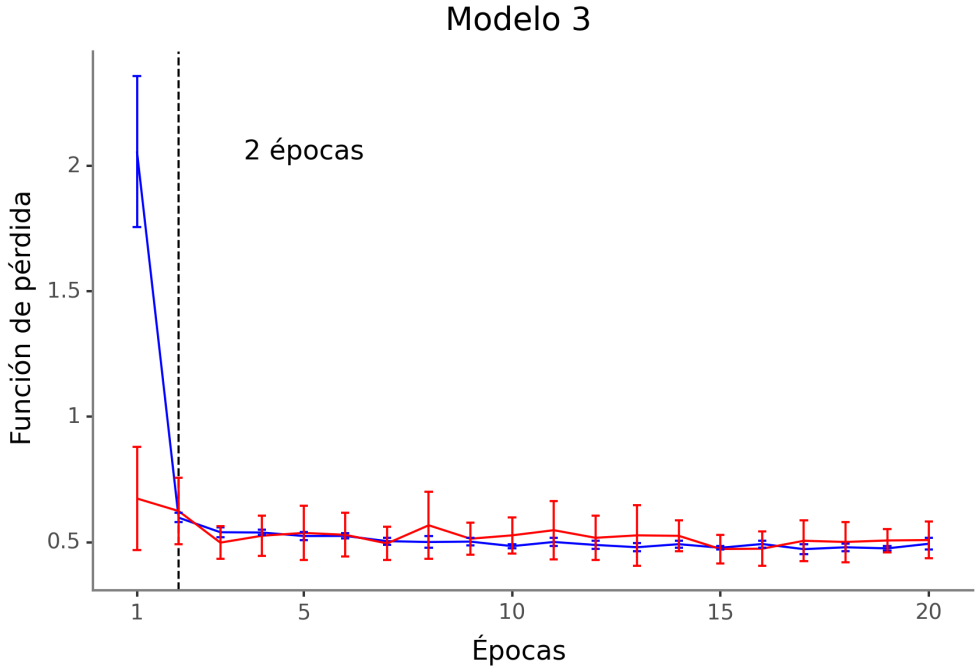


Figura A.3: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 3. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

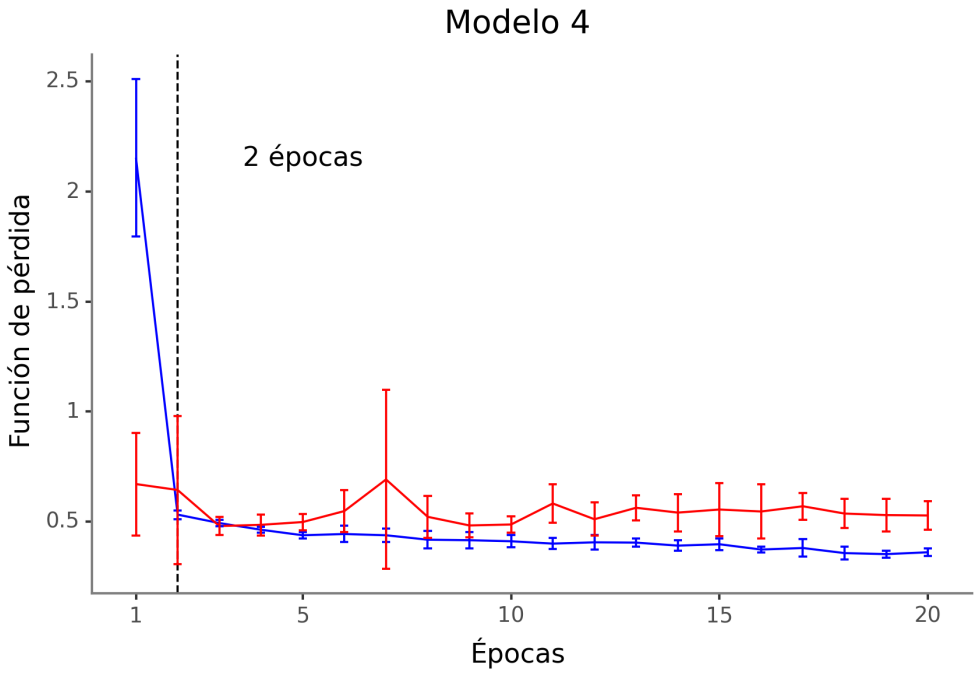


Figura A.4: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 4. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

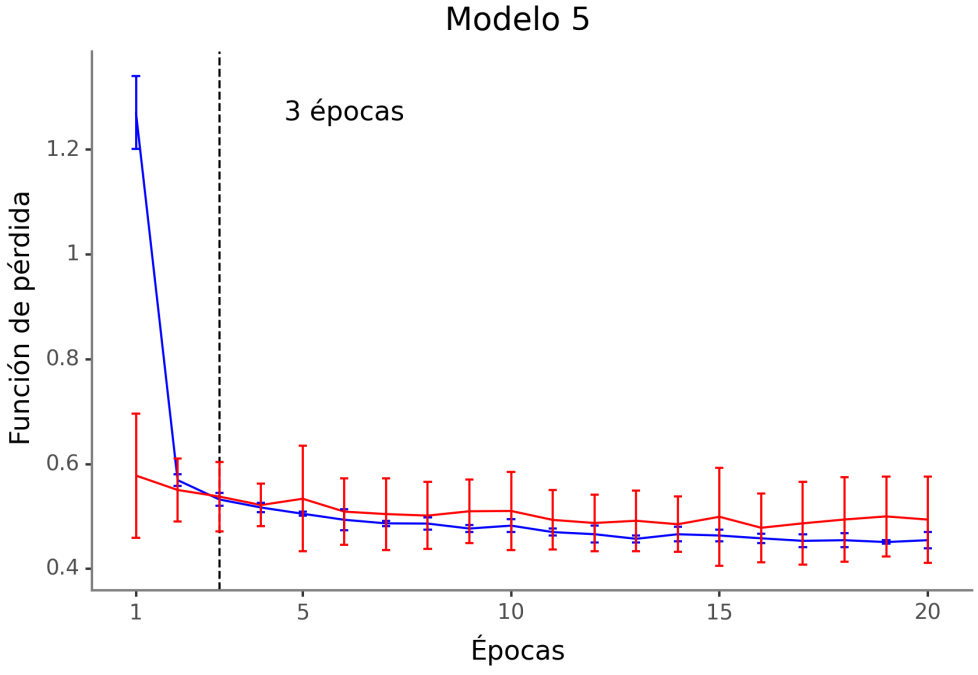


Figura A.5: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 5. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

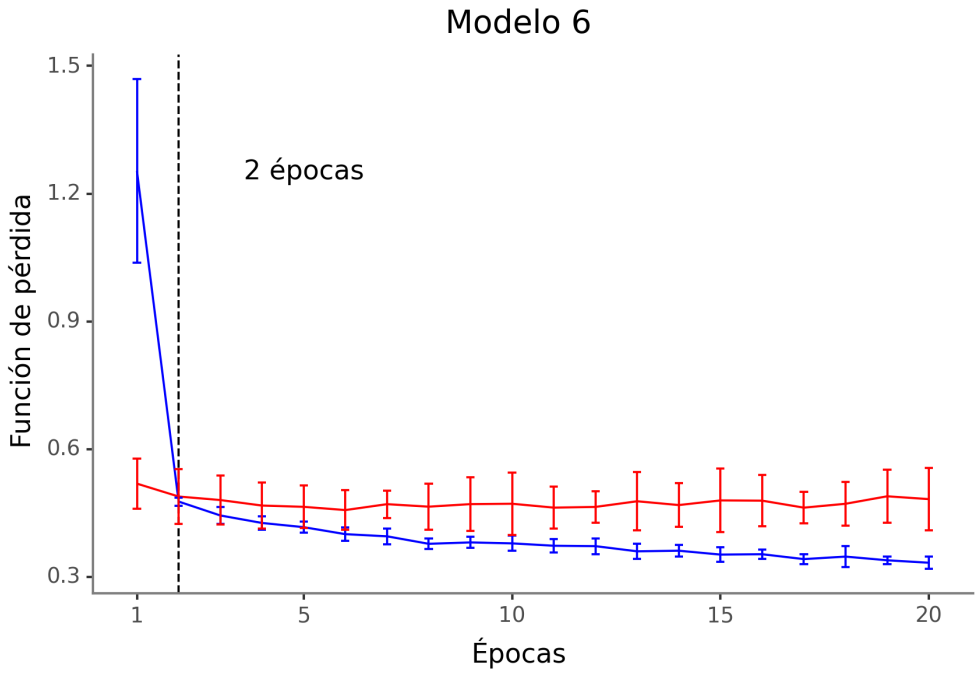


Figura A.6: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 6. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

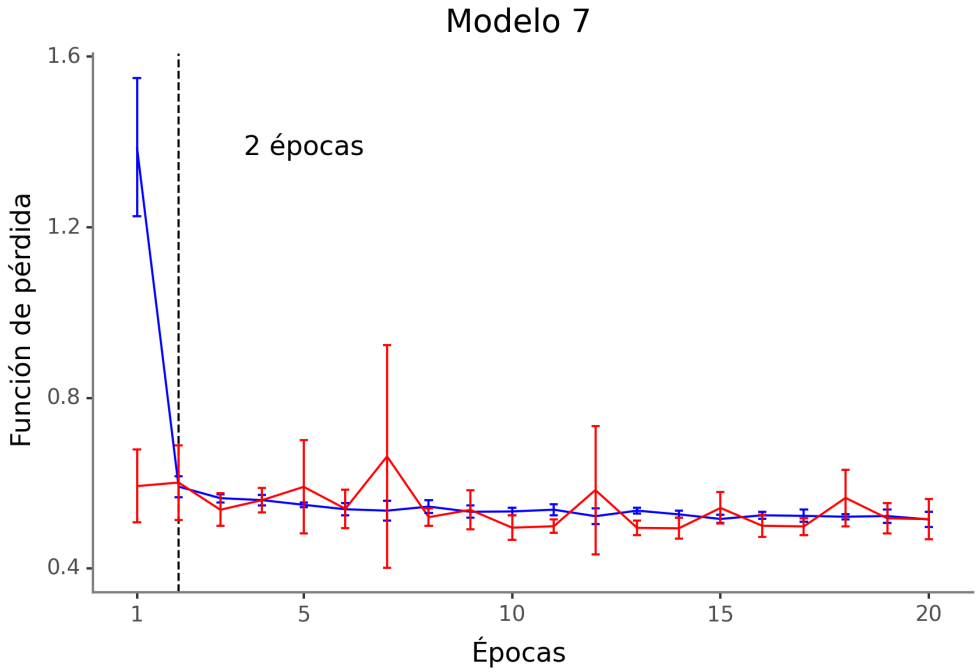


Figura A.7: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 7. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

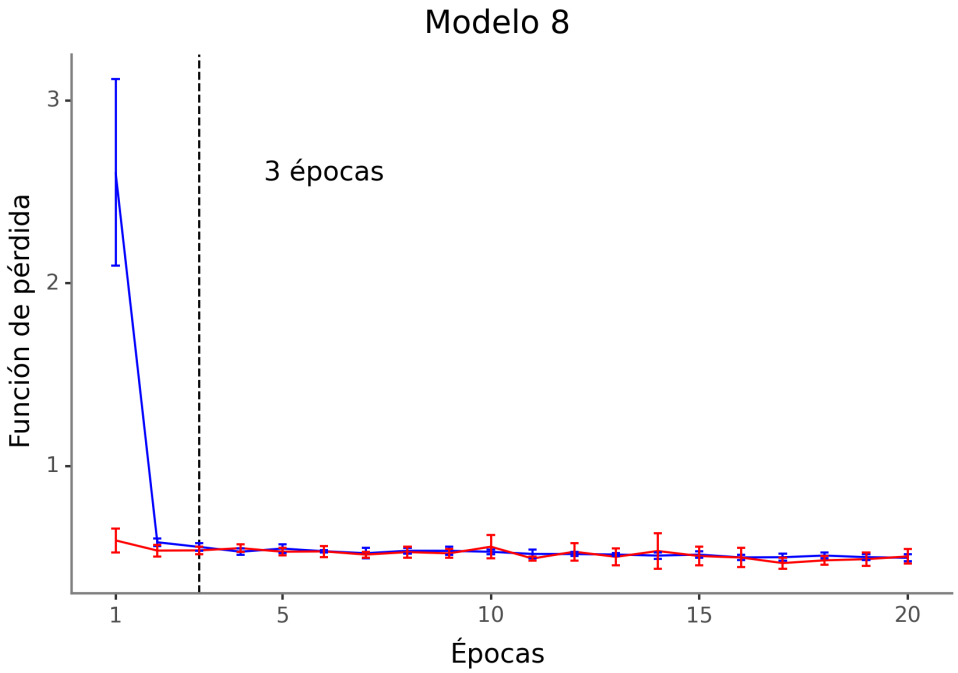


Figura A.8: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 8. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

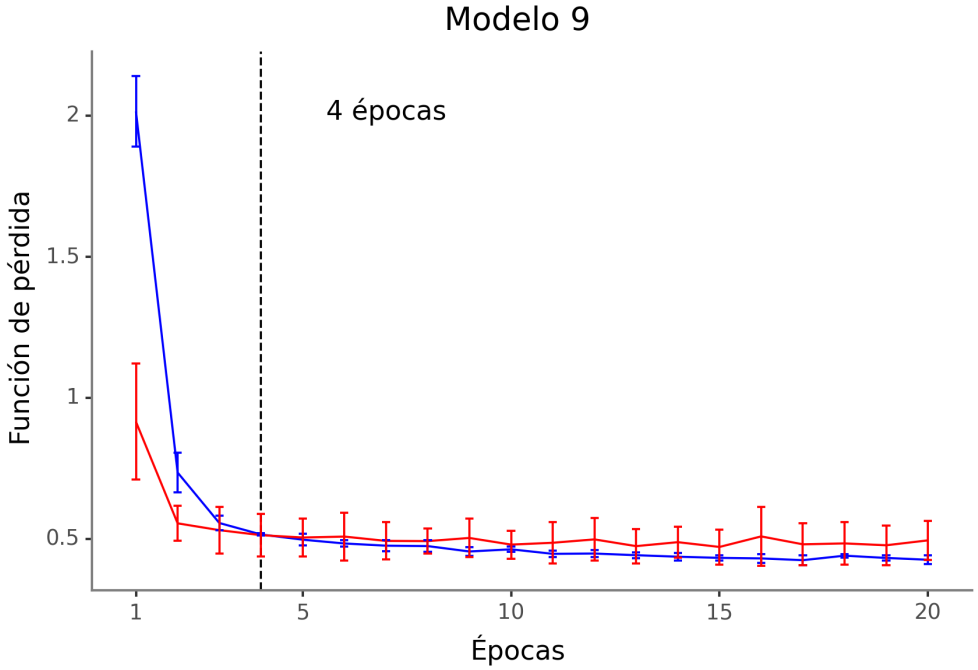


Figura A.9: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 9. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

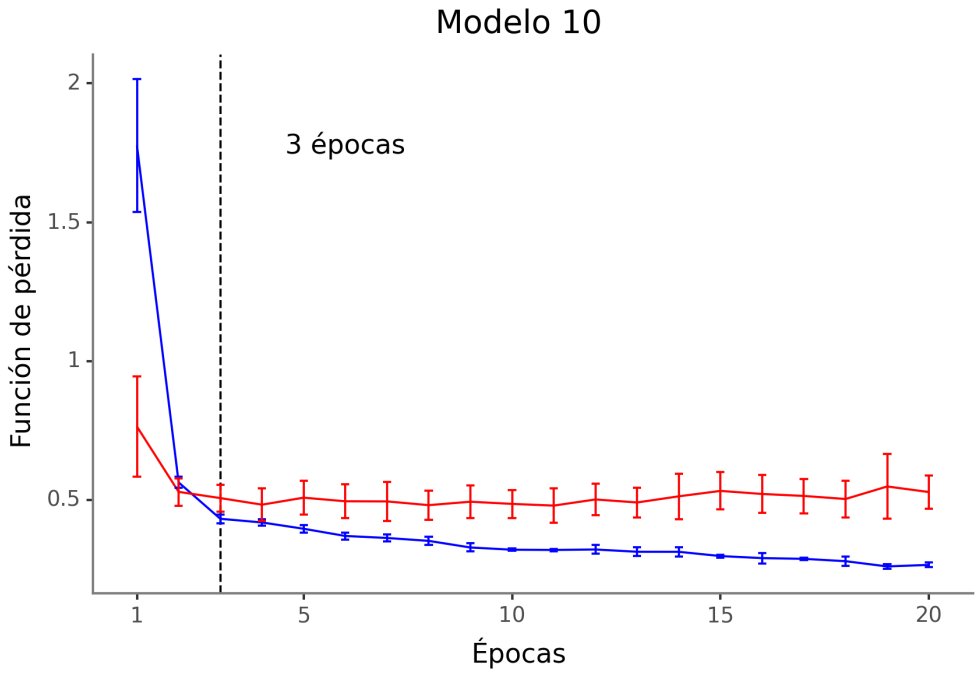


Figura A.10: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 10. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

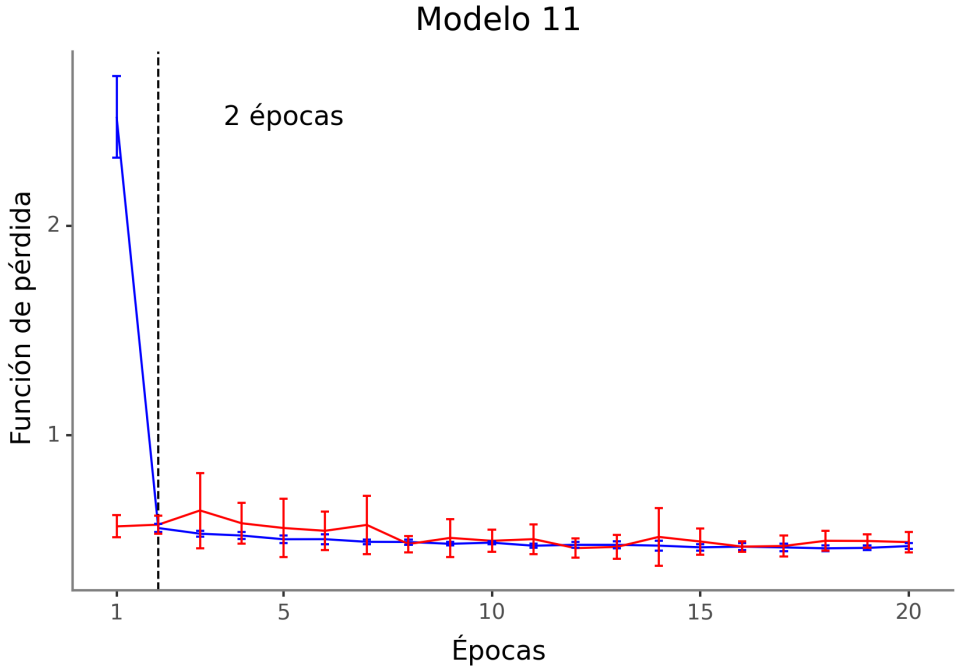


Figura A.11: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 11. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

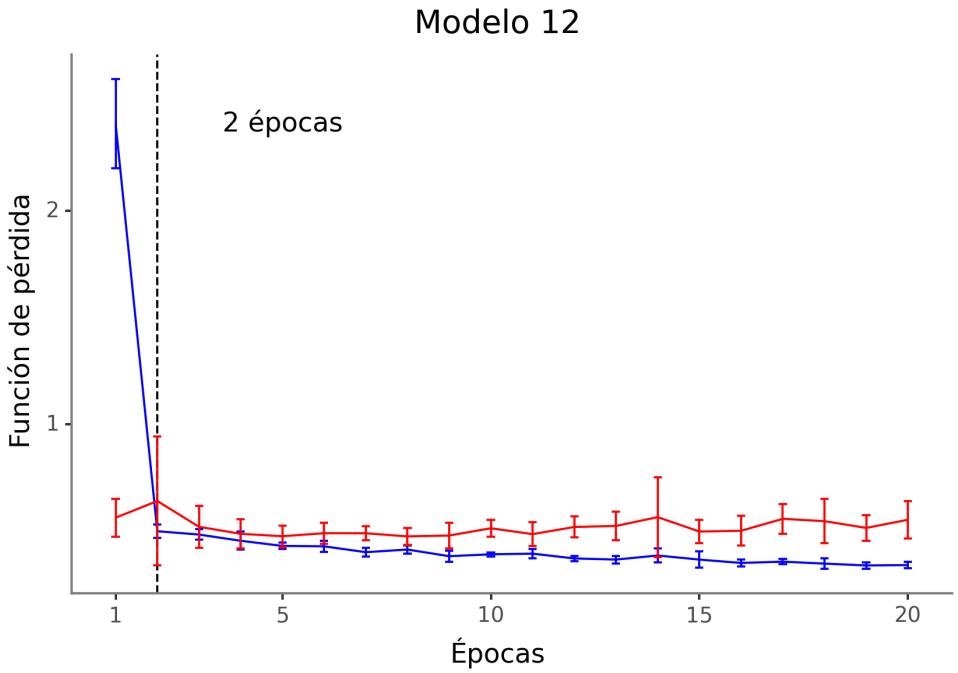


Figura A.12: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 12. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

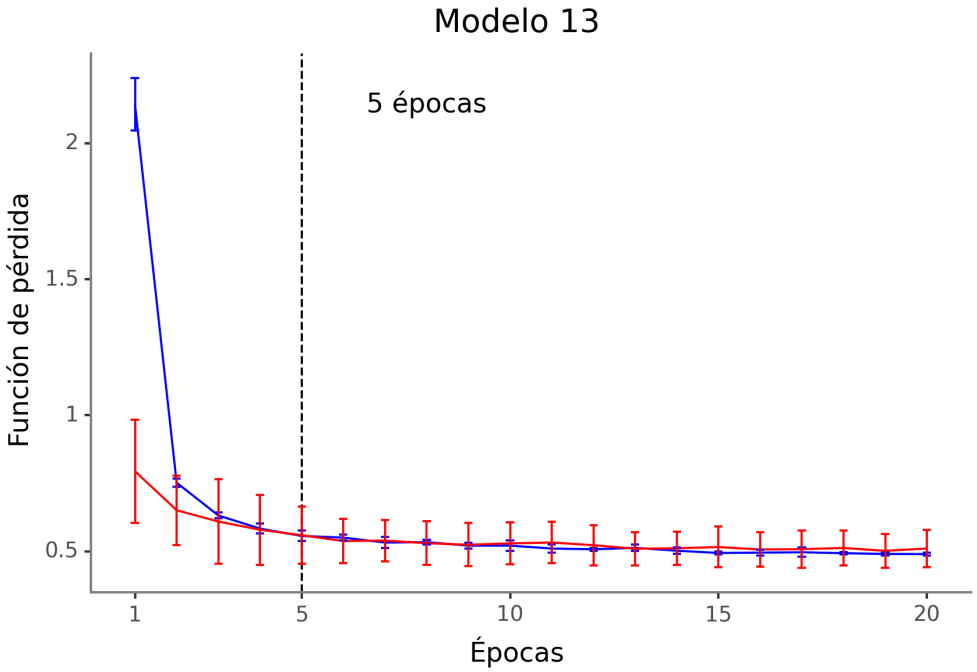


Figura A.13: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 13. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

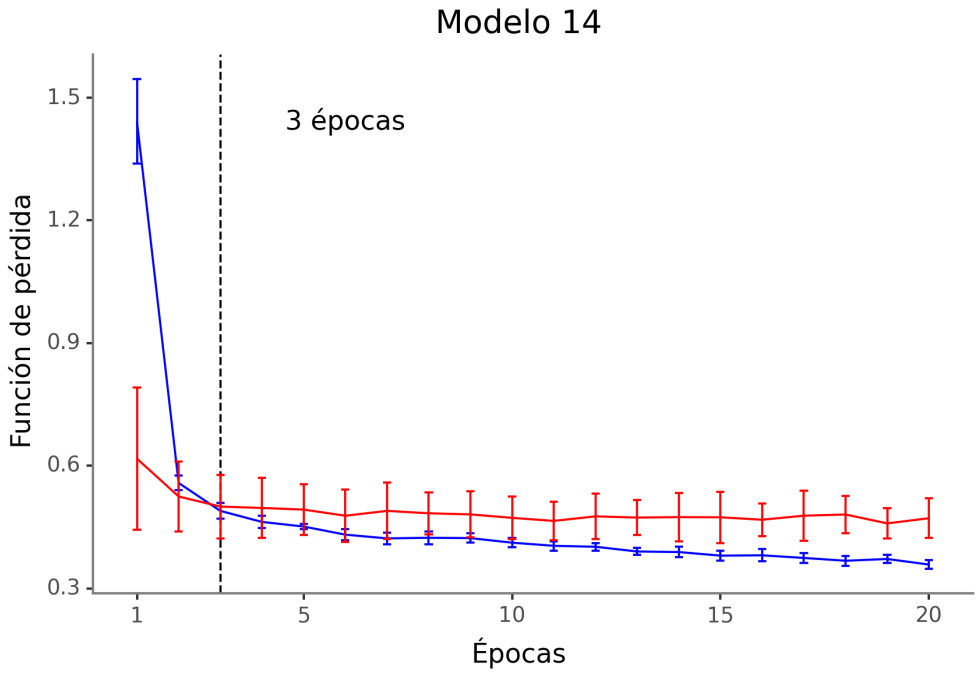


Figura A.14: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 14. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

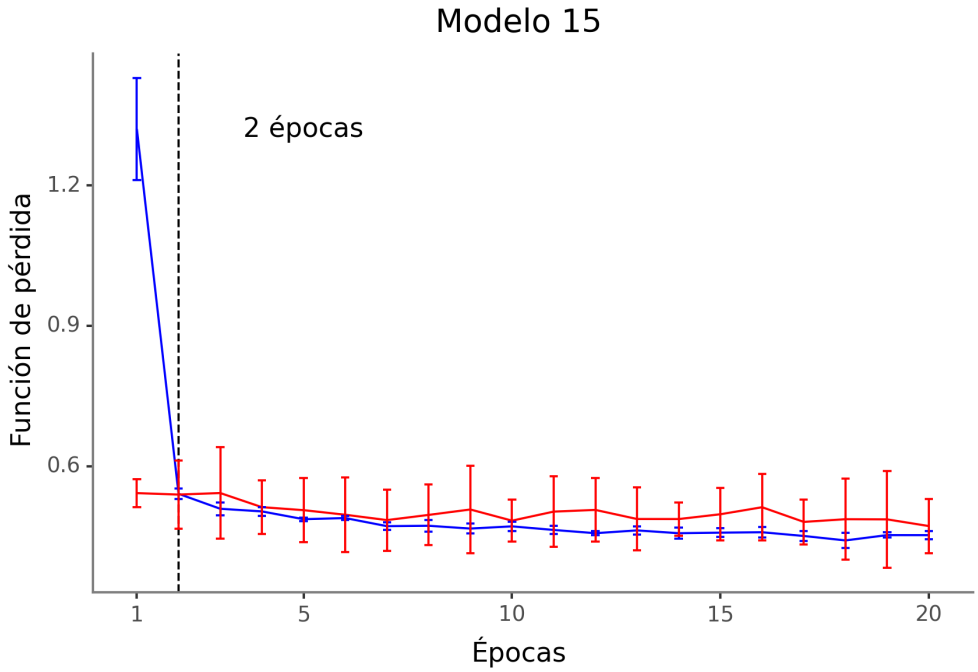


Figura A.15: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 15. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

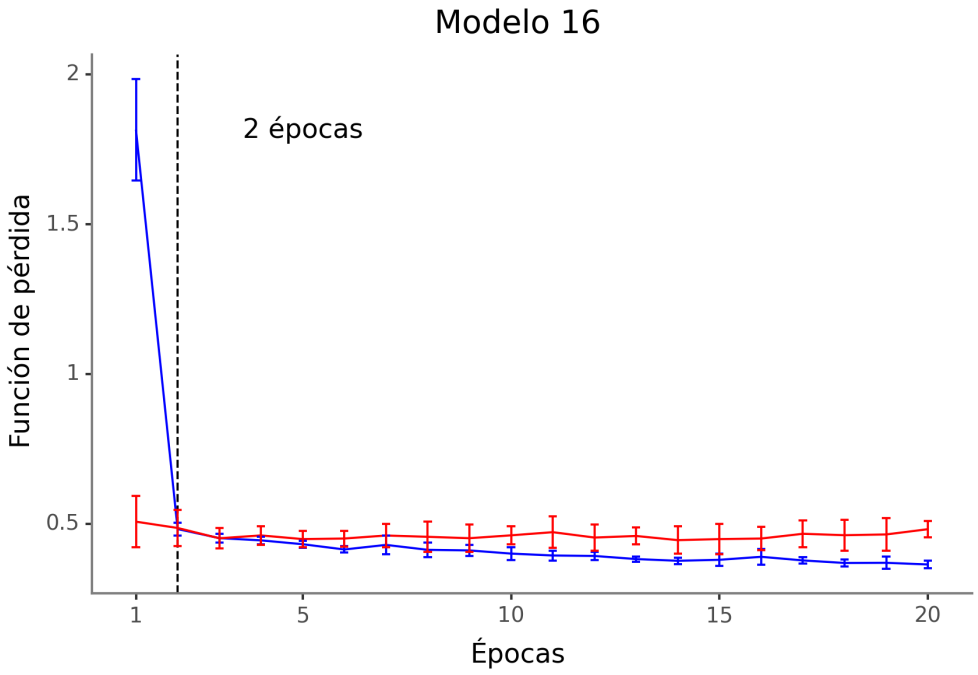


Figura A.16: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 16. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

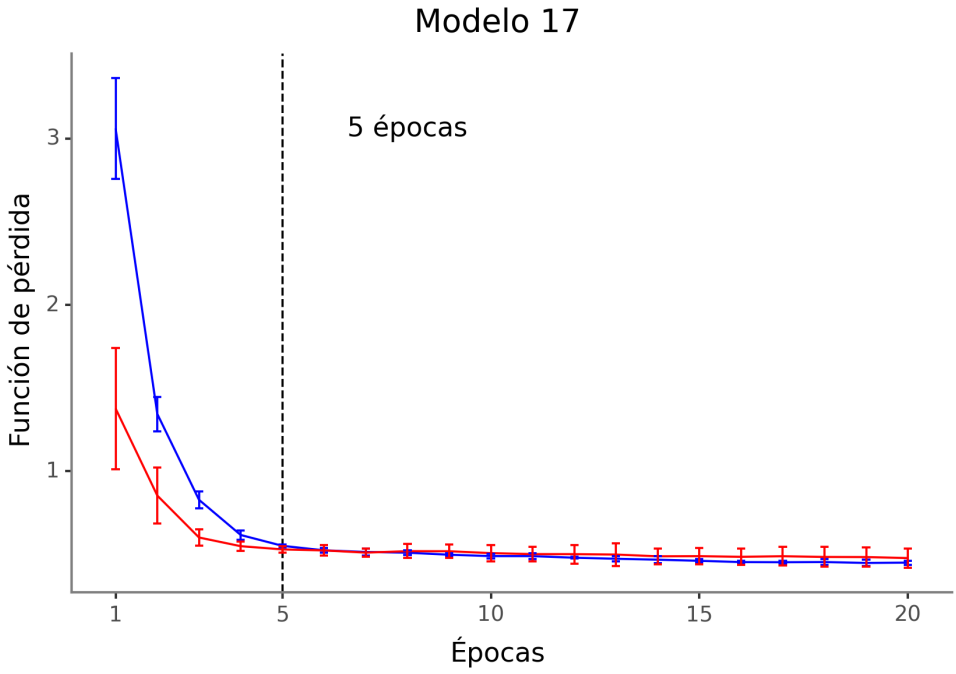


Figura A.17: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 17. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

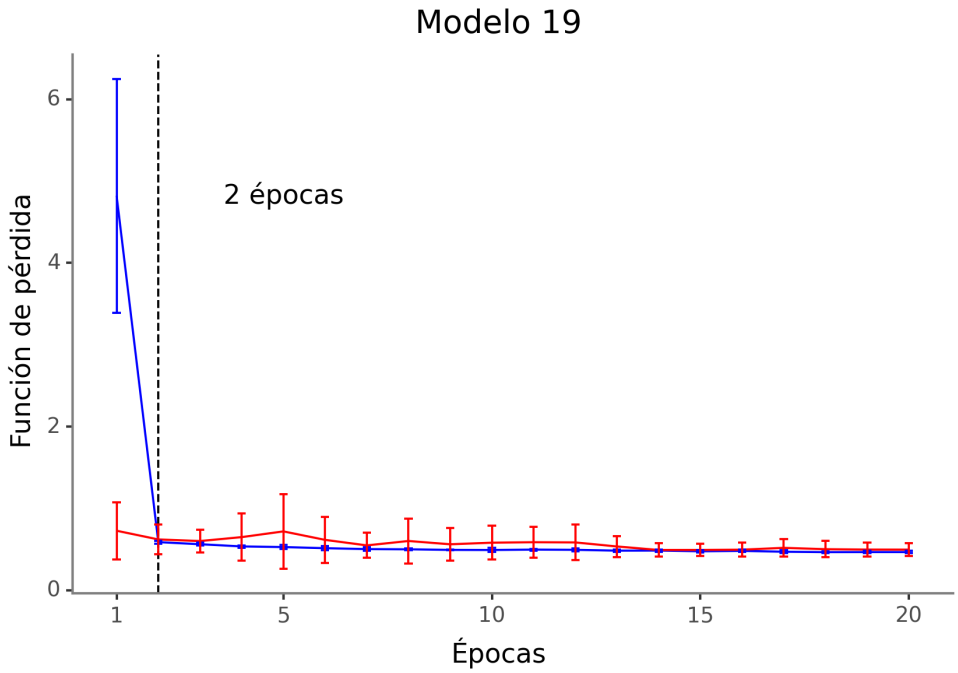


Figura A.18: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 19. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

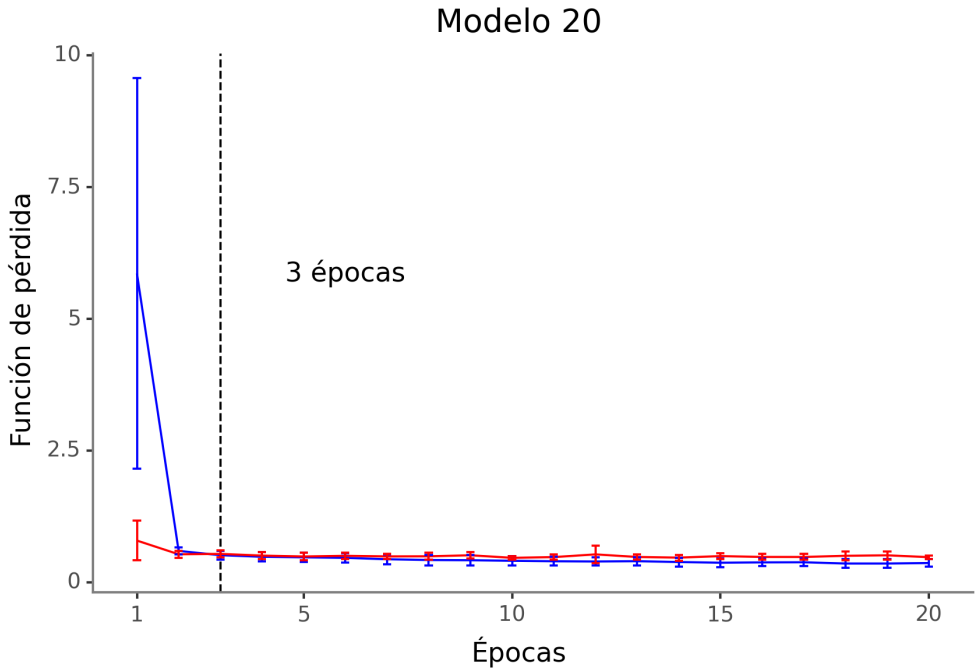


Figura A.19: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 20. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

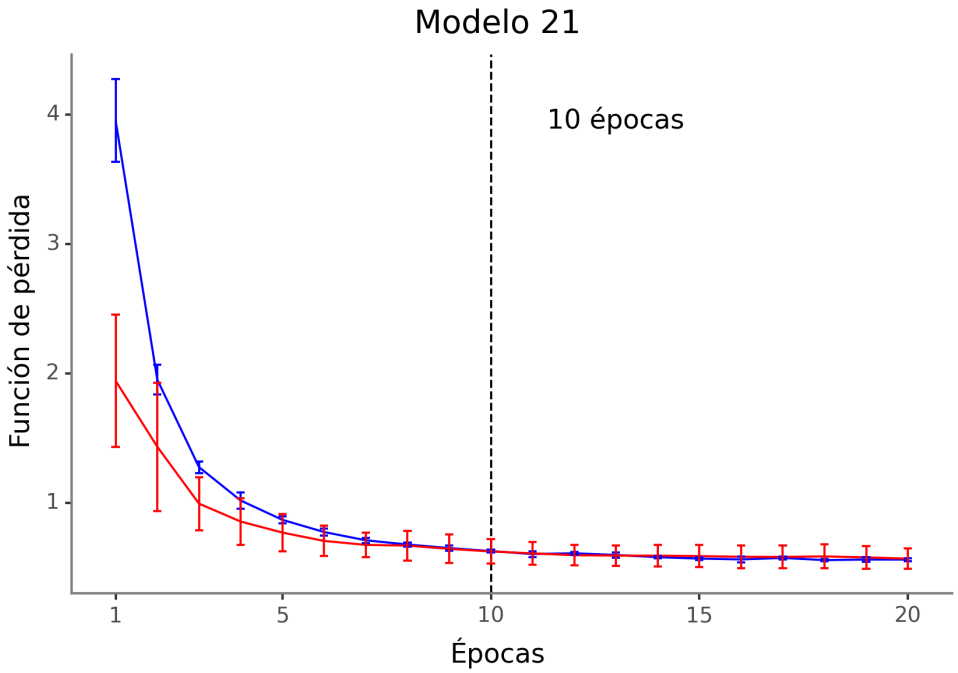


Figura A.20: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 21. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

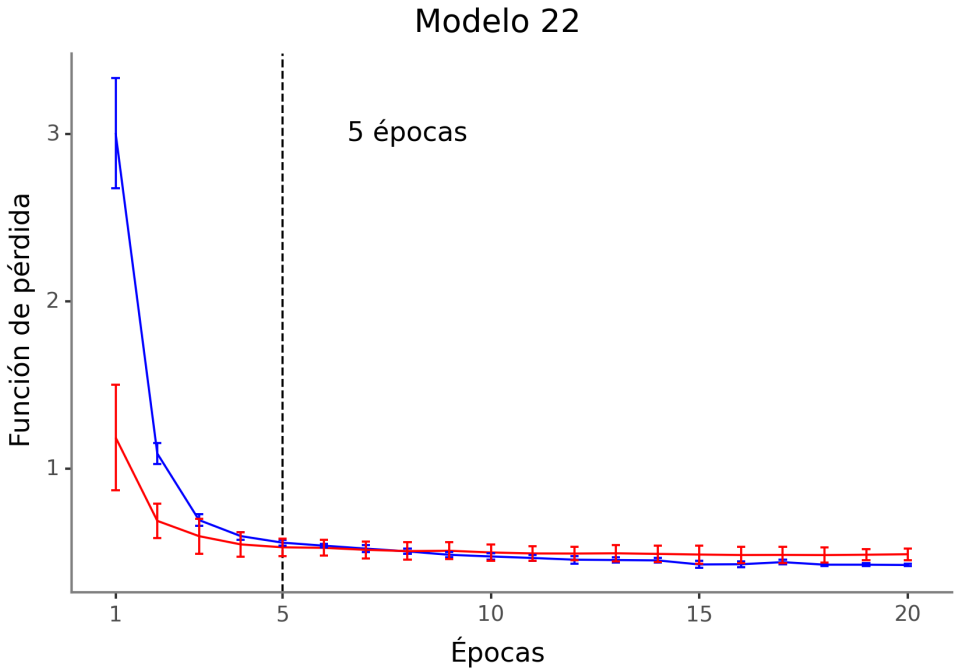


Figura A.21: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 22. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

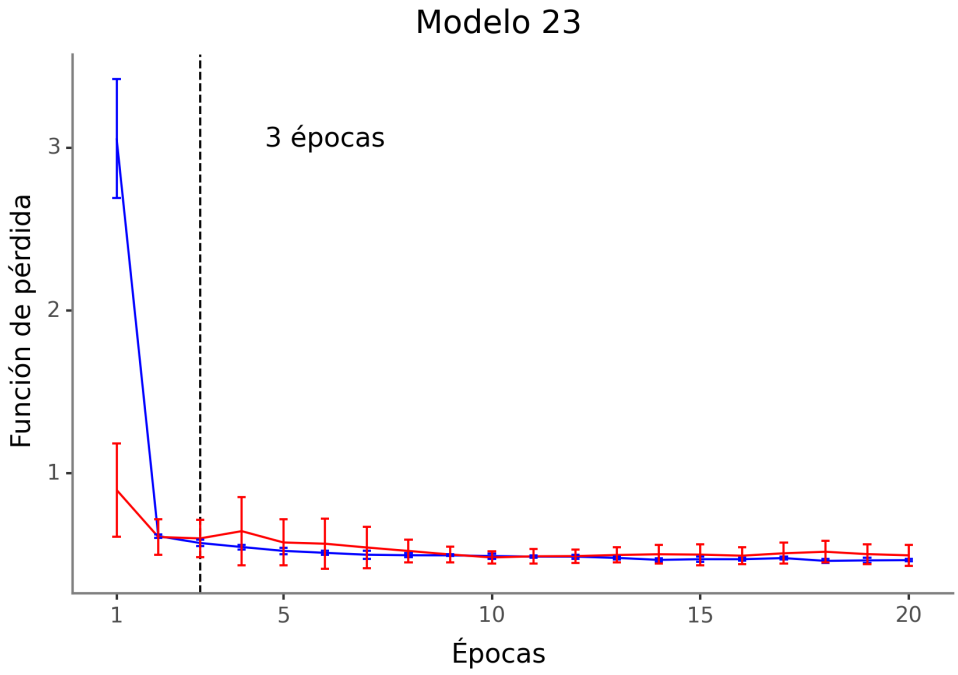


Figura A.22: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 23. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

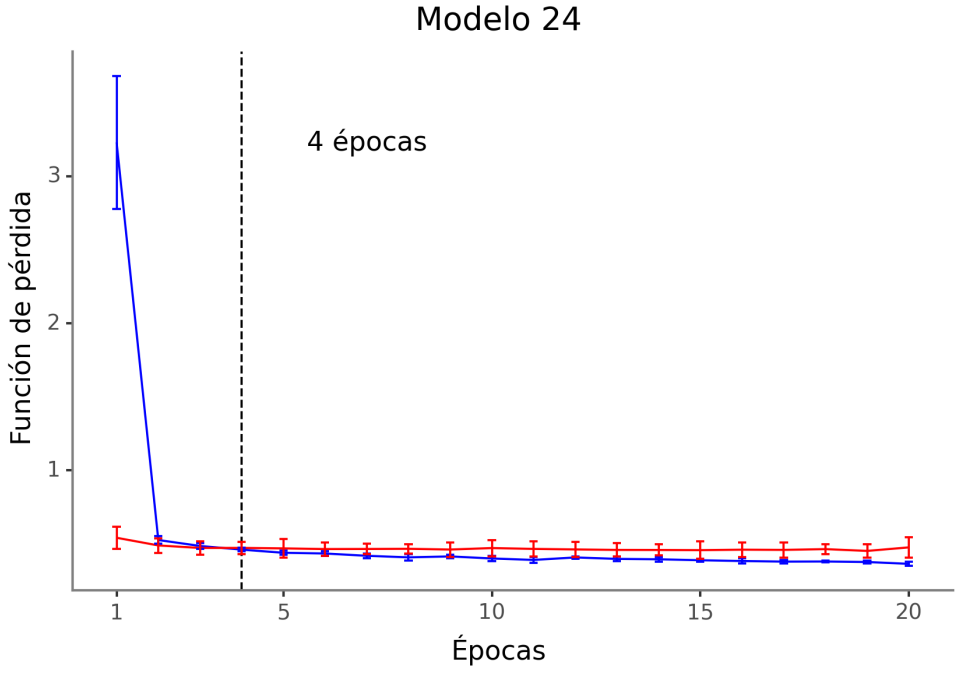


Figura A.23: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 24. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

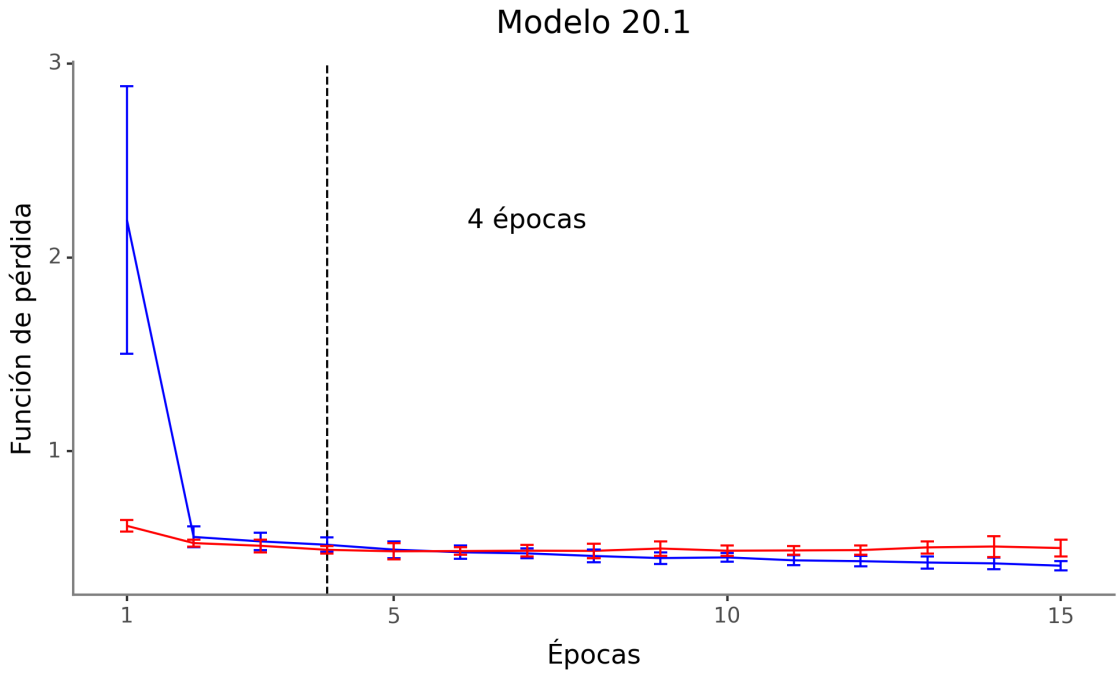


Figura A.24: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 20.1. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

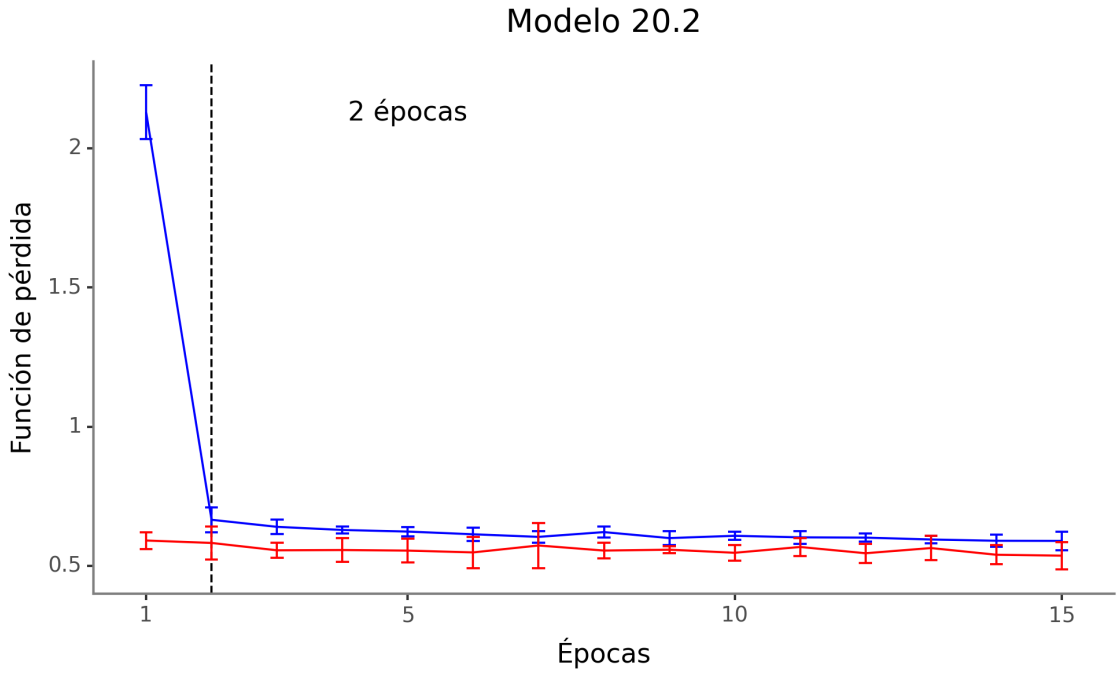


Figura A.25: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 20.2. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

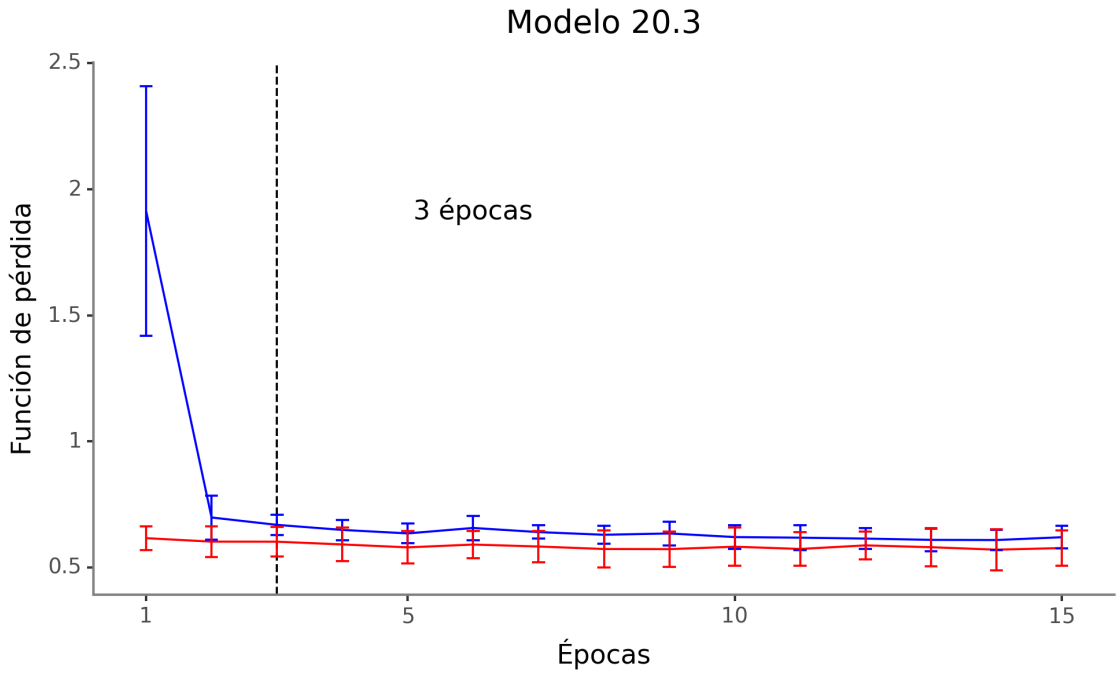


Figura A.26: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 20.3. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

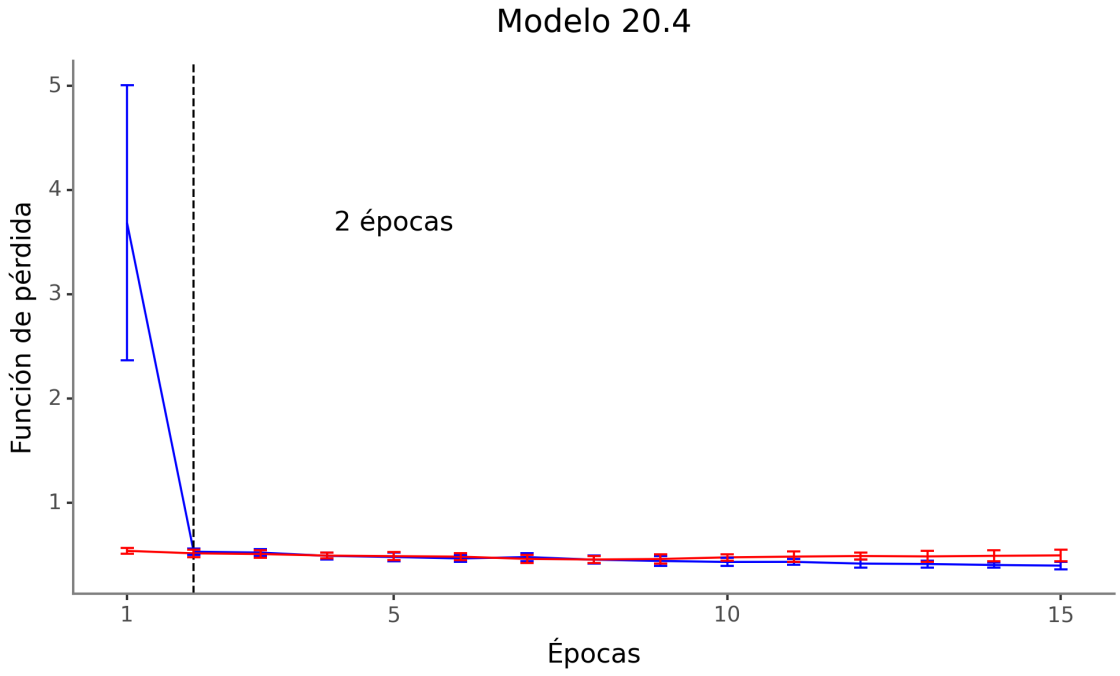


Figura A.27: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 20.4. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

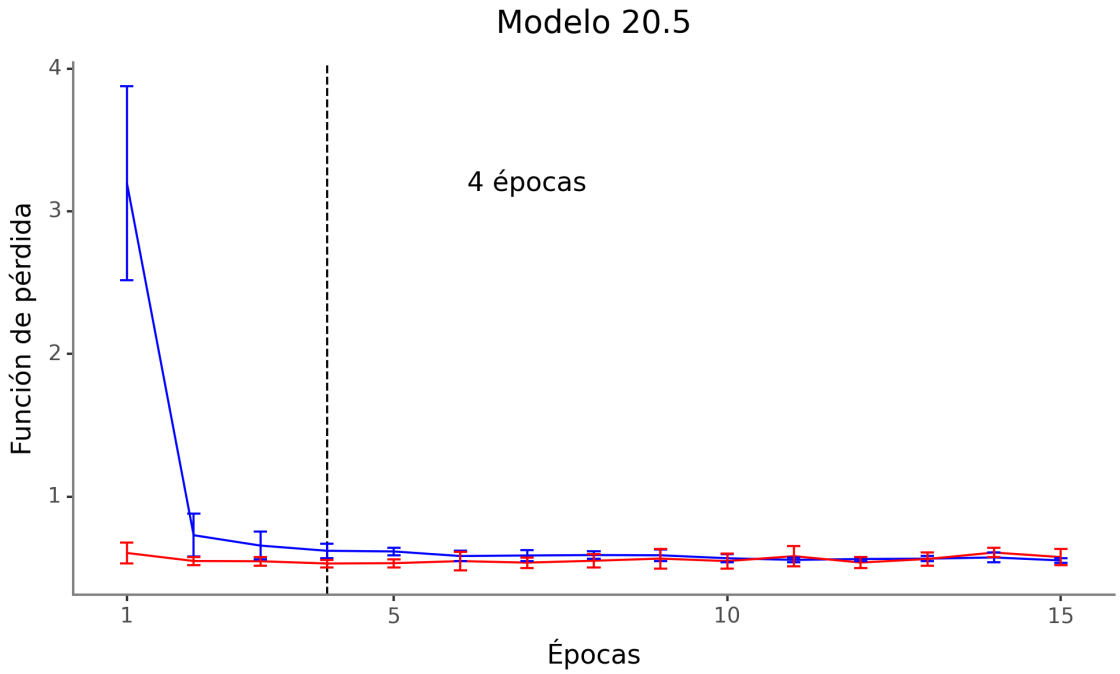


Figura A.28: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 20.5. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

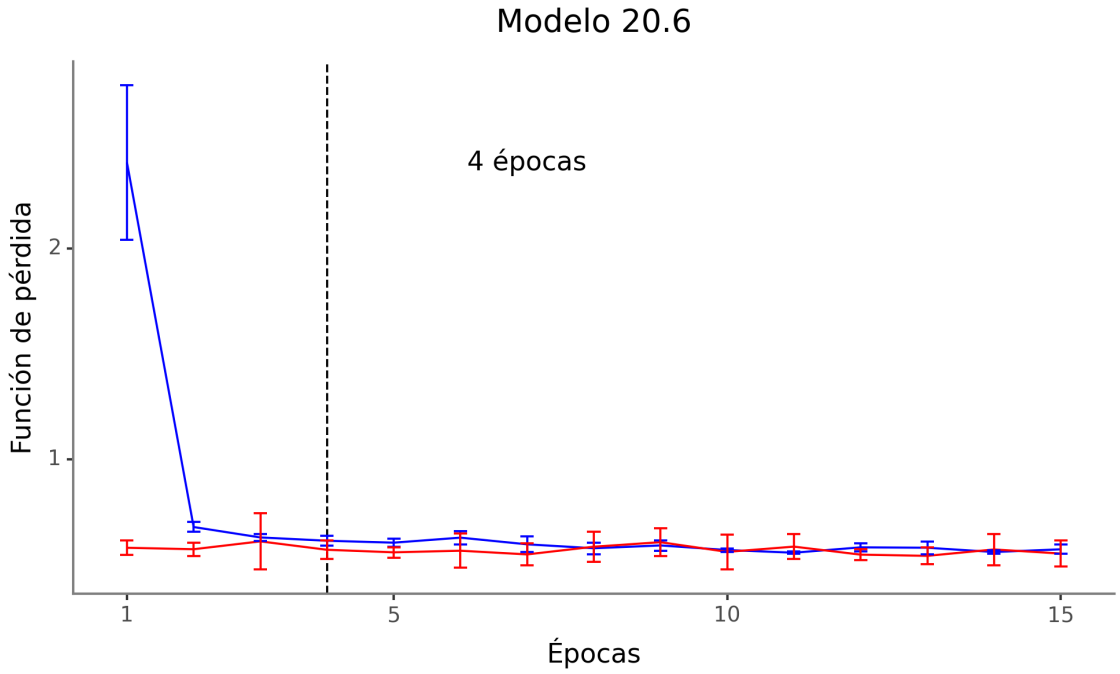


Figura A.29: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 20.6. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

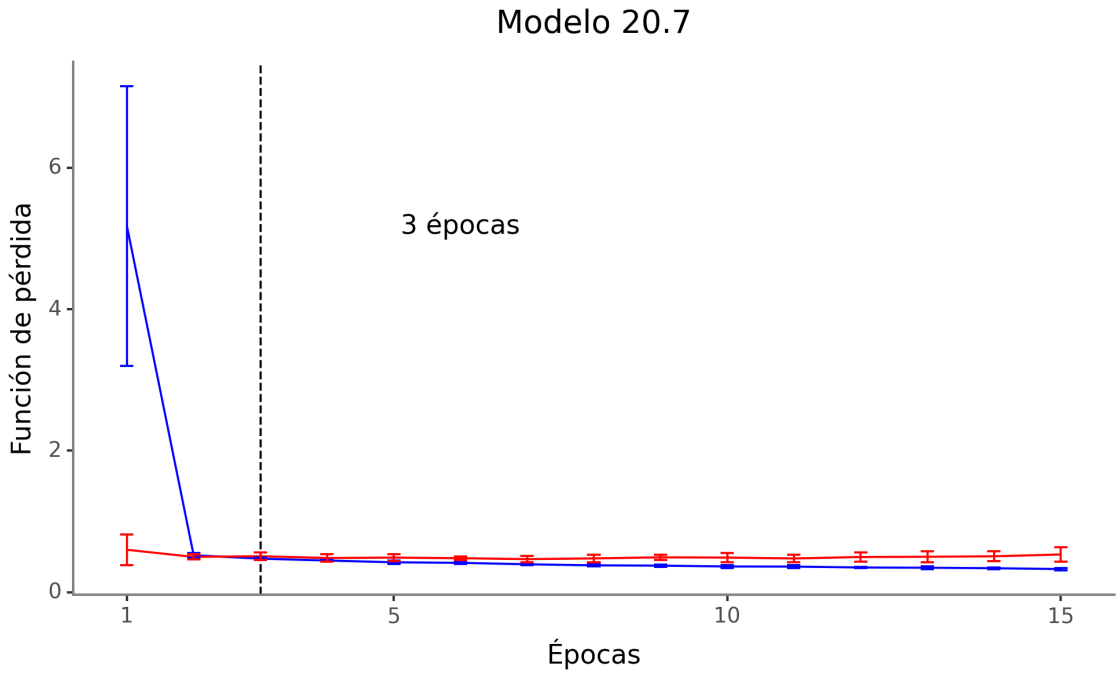


Figura A.30: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 20.7. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

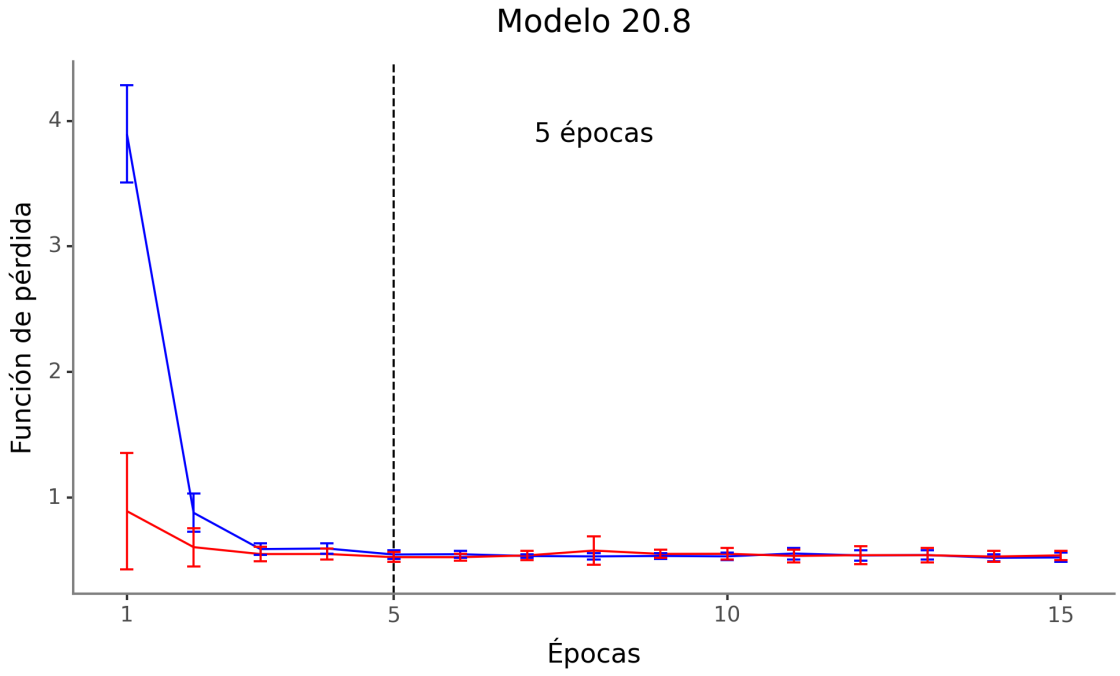


Figura A.31: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 20.8. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

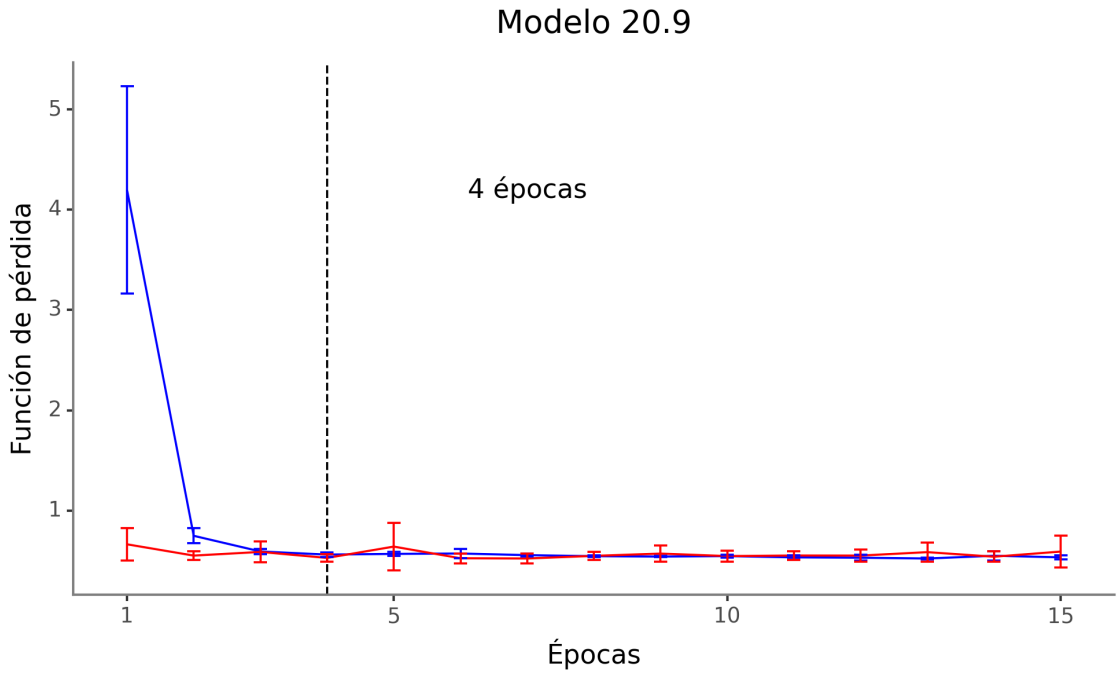


Figura A.32: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 20.9. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

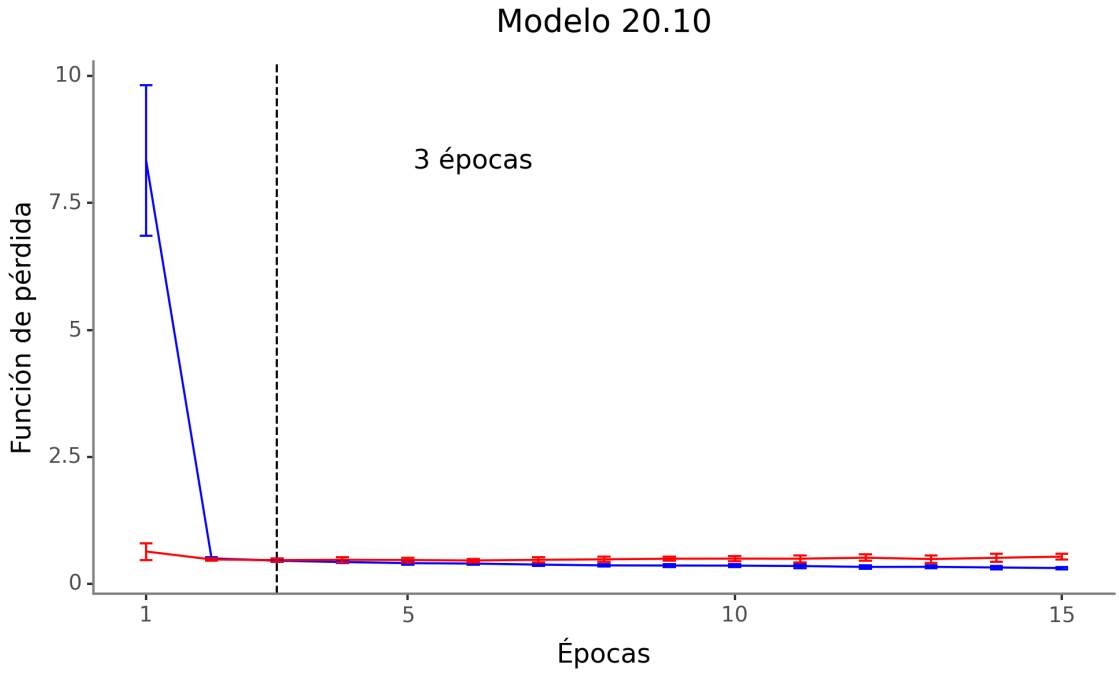


Figura A.33: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 20.10. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

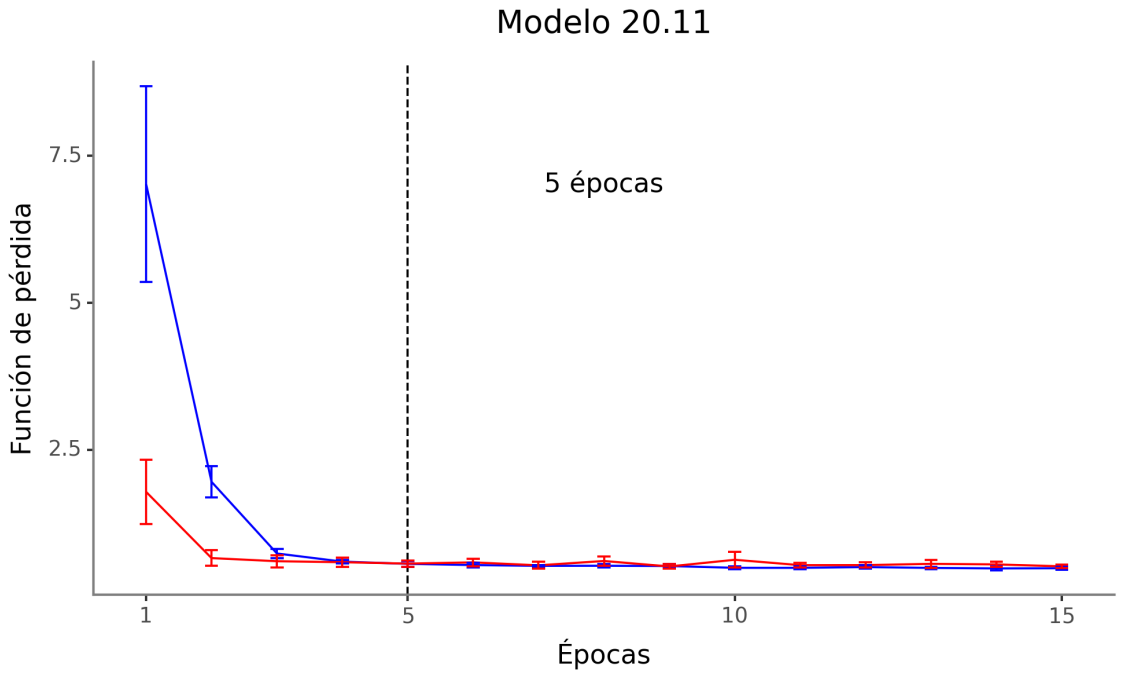


Figura A.34: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 20.11. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

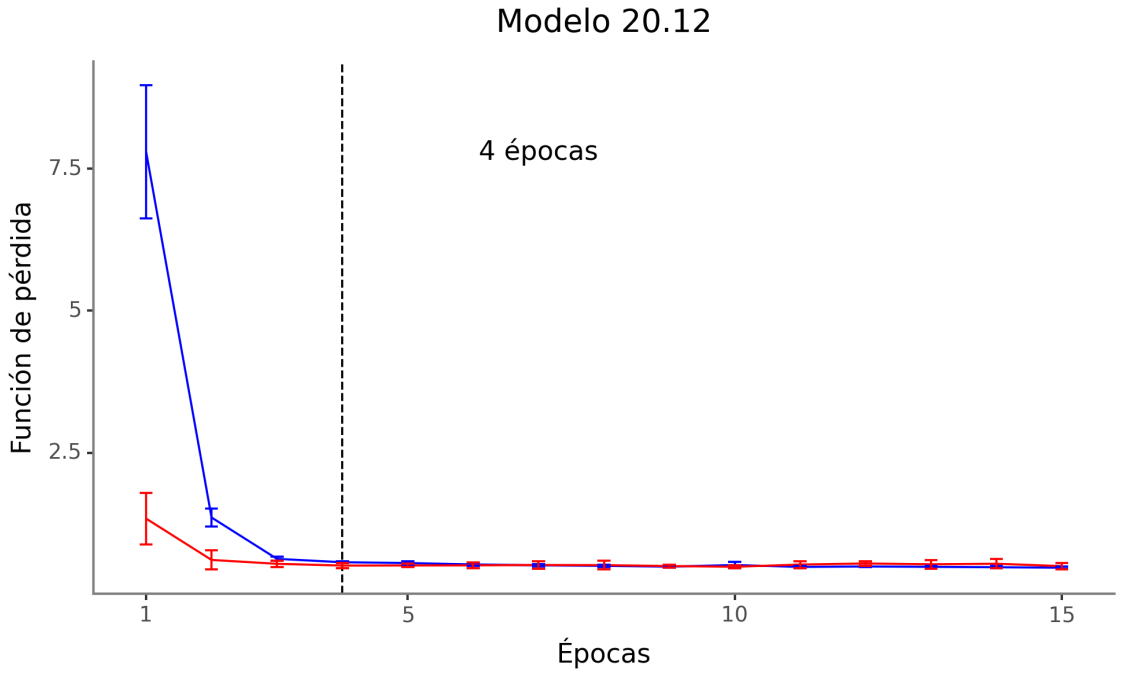


Figura A.35: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 20.12. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

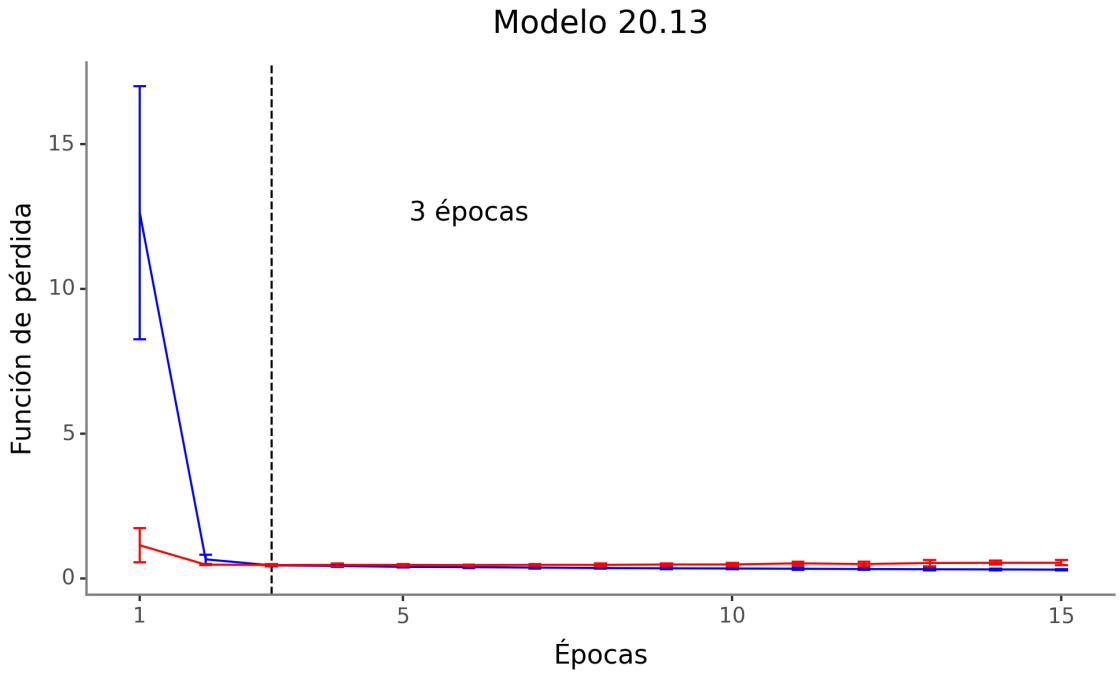


Figura A.36: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 20.13. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

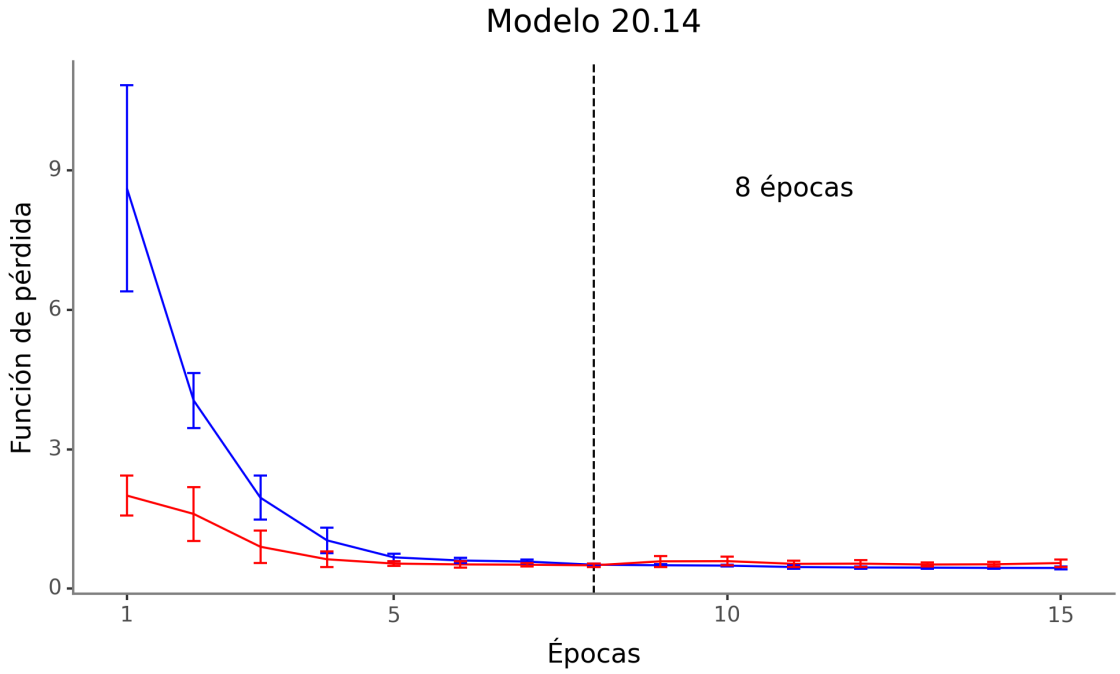


Figura A.37: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 20.14. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

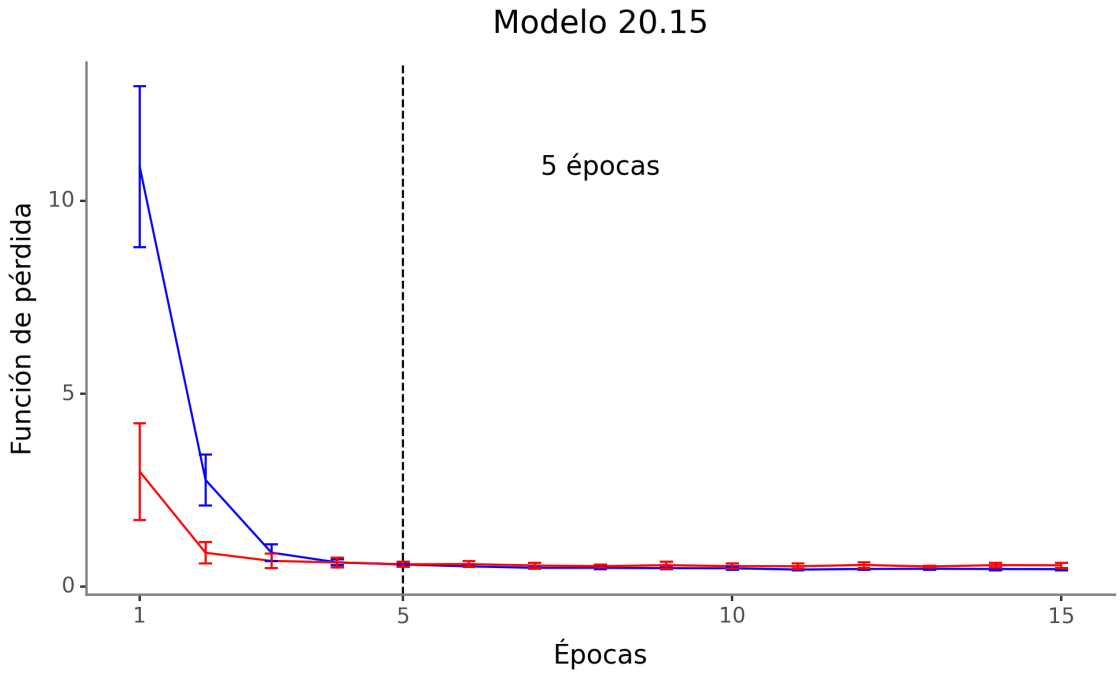


Figura A.38: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 20.15. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

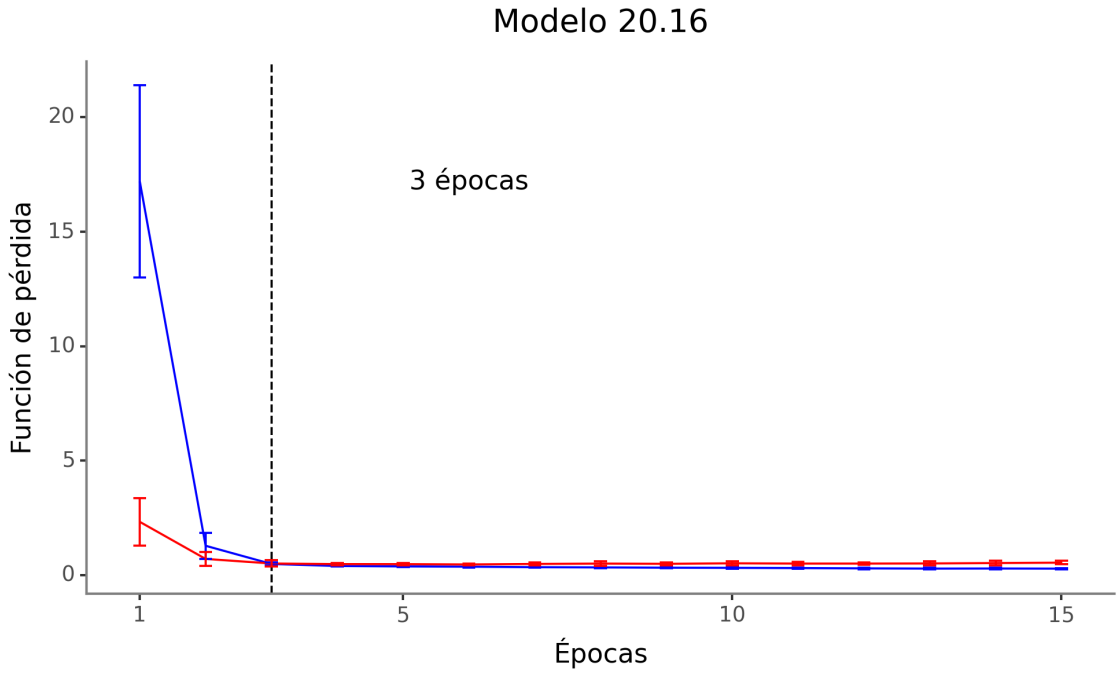


Figura A.39: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 20.16. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

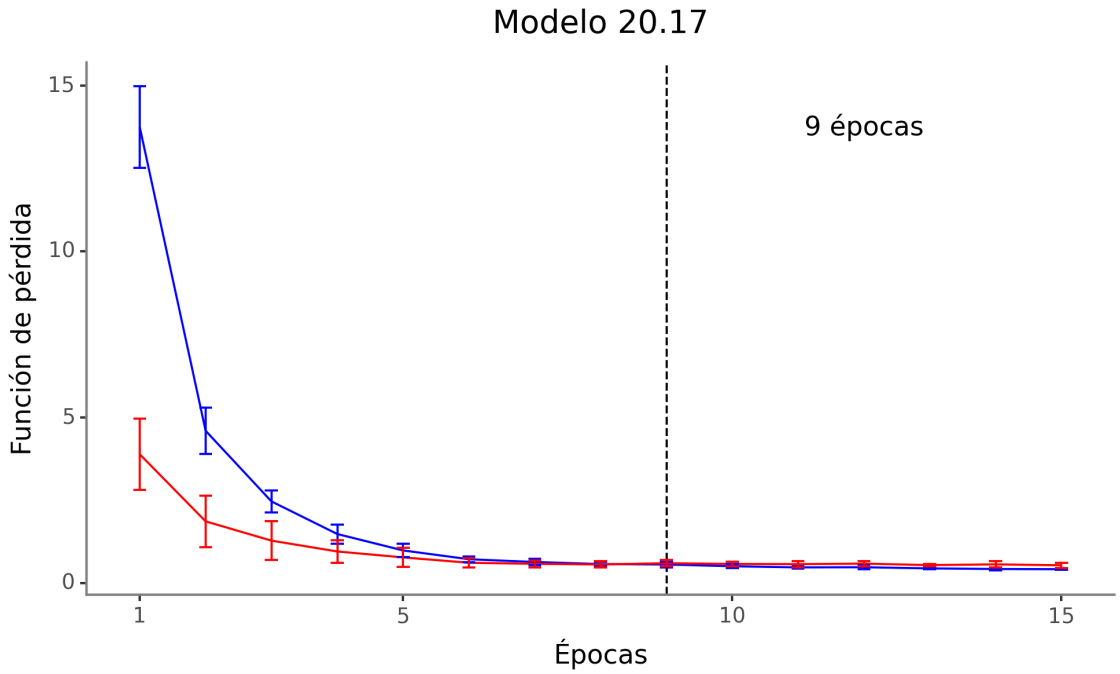


Figura A.40: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 20.17. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

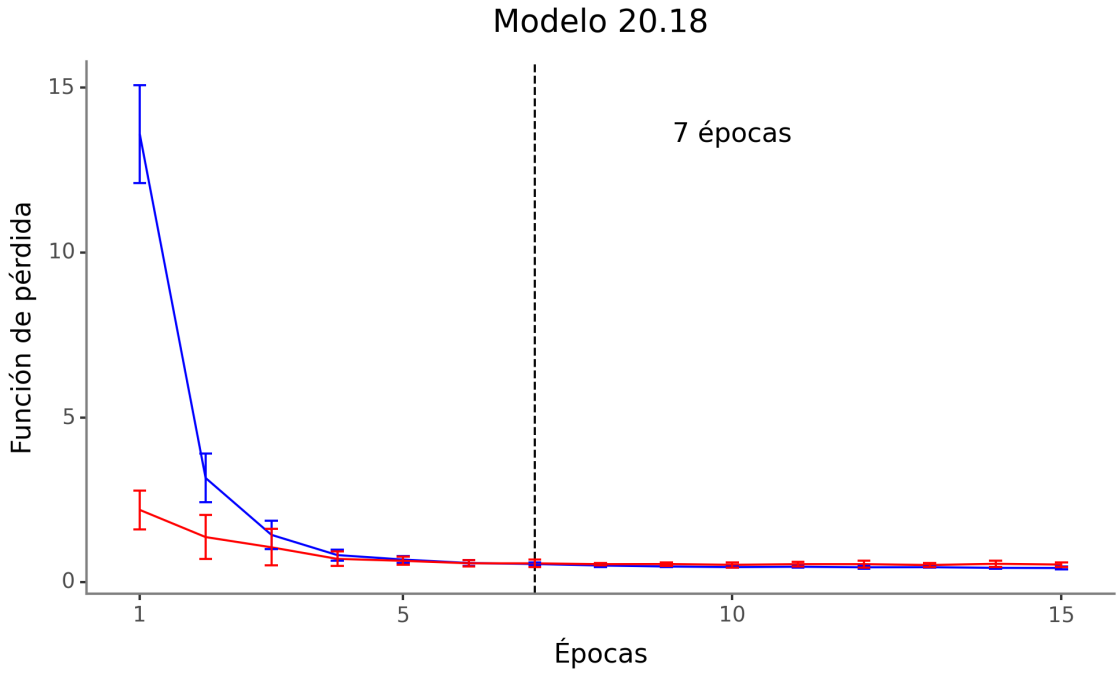


Figura A.41: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 20.18. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

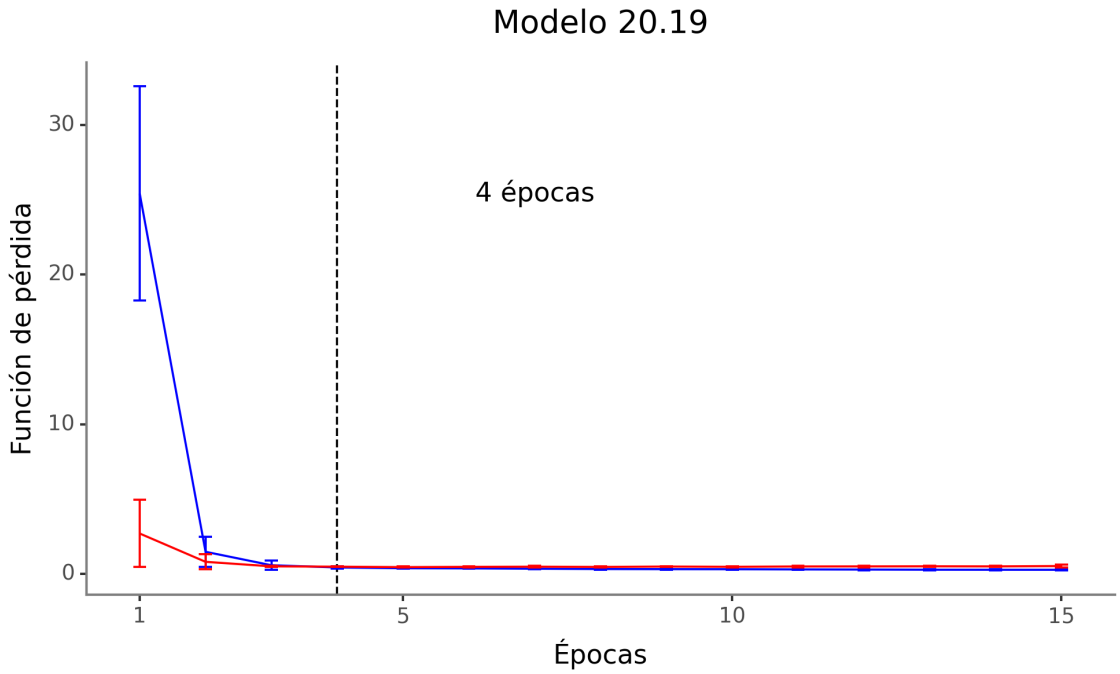


Figura A.42: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 20.19. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

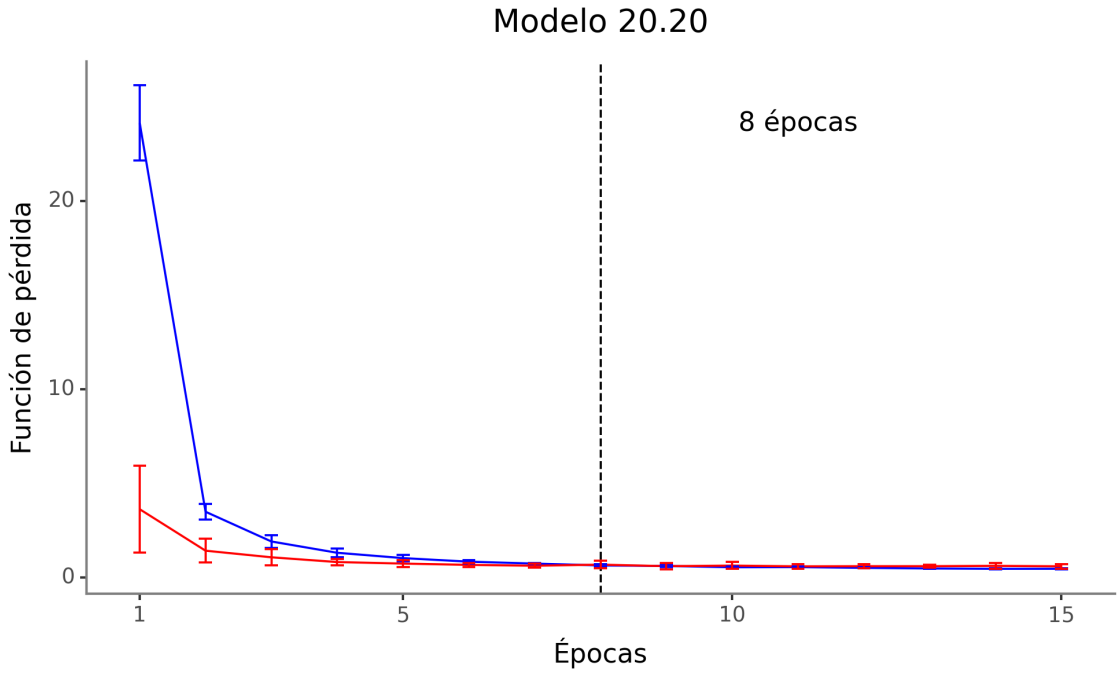


Figura A.43: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 20.20. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

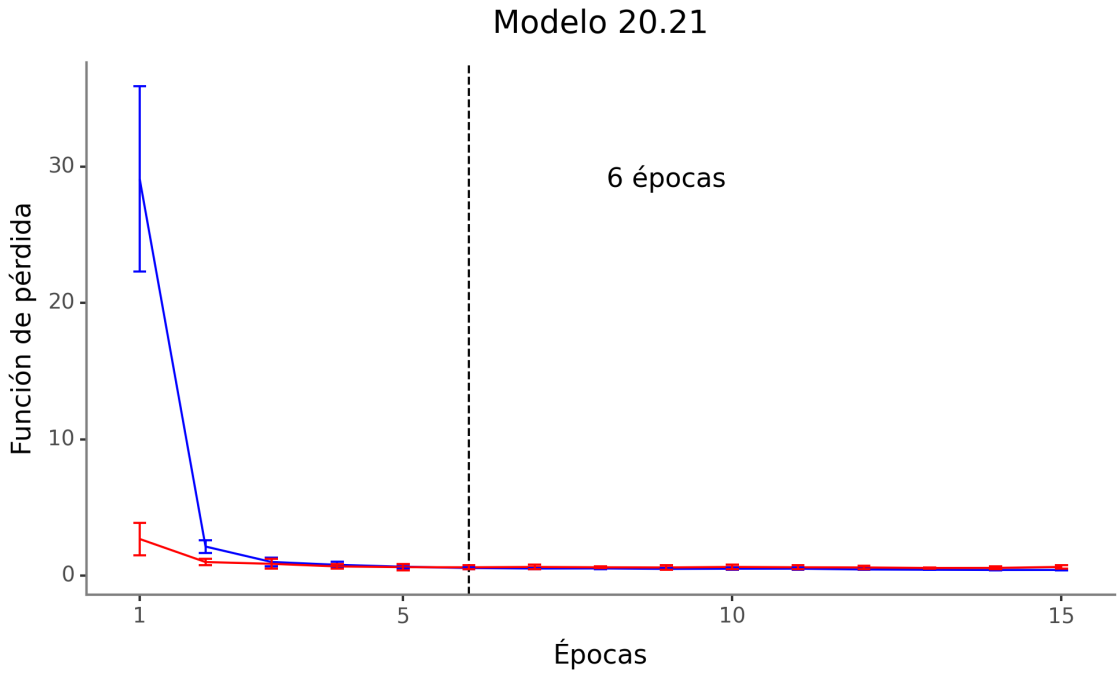


Figura A.44: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 20.21. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

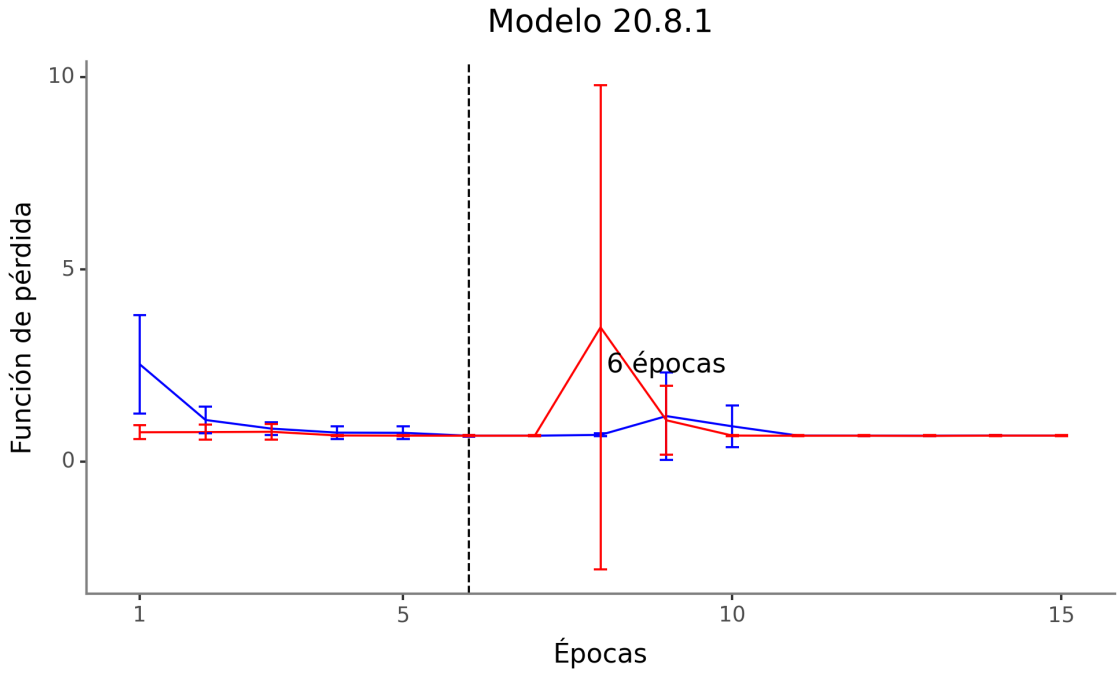


Figura A.45: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 20.8.1. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

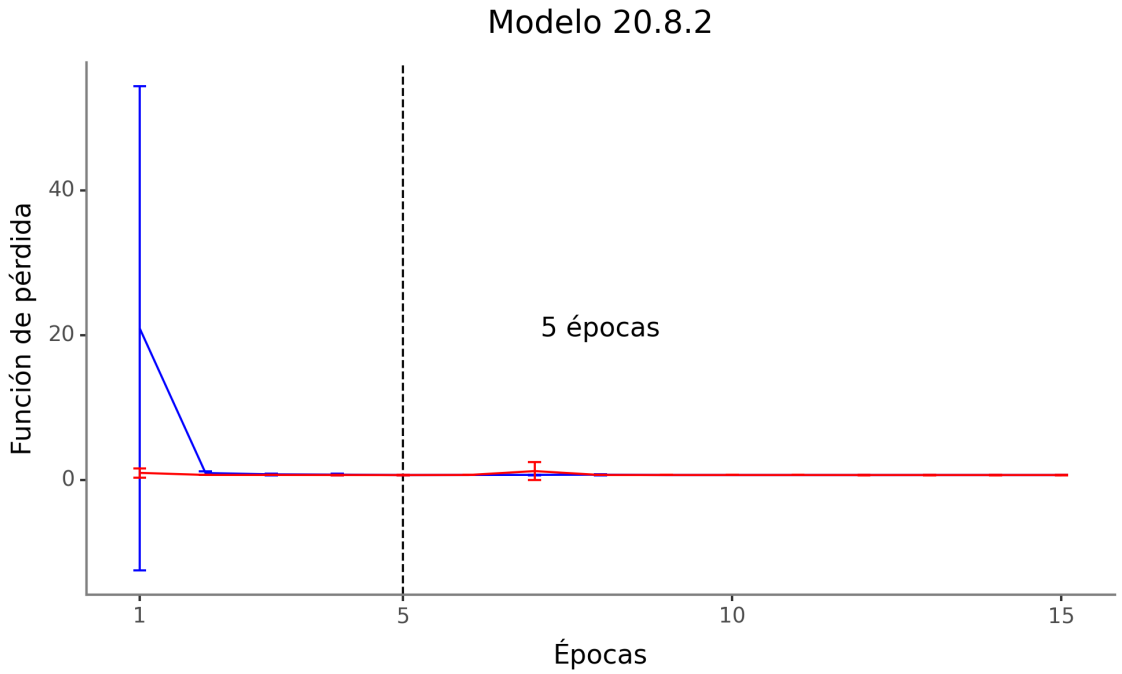


Figura A.46: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 20.8.2. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

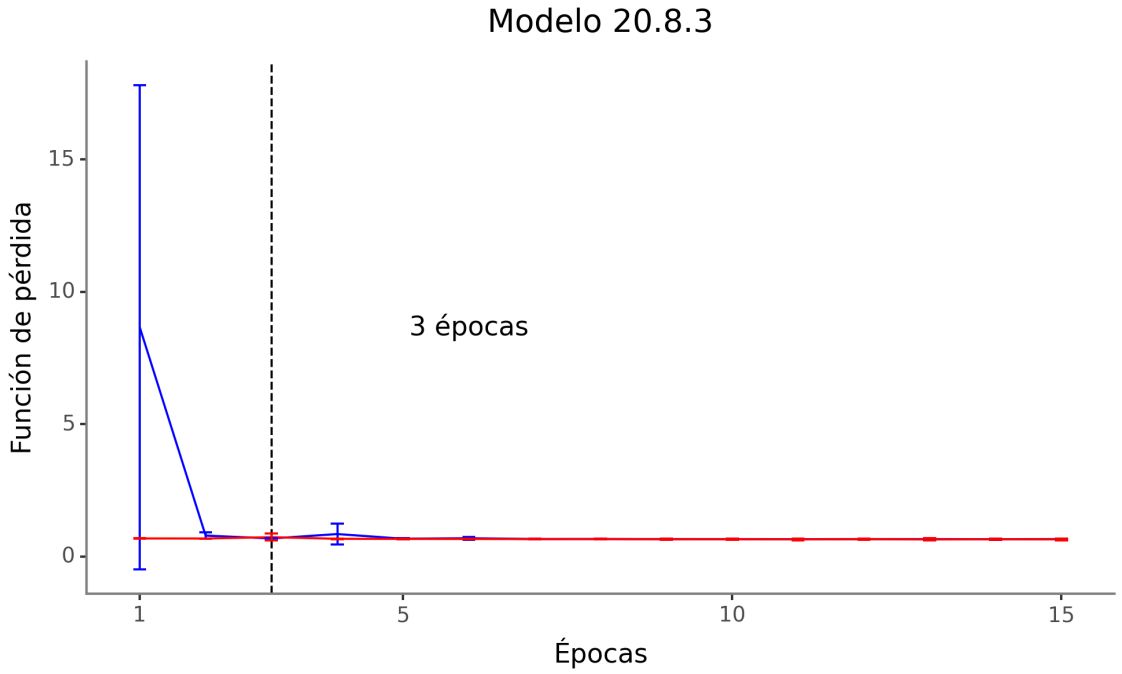


Figura A.47: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 20.8.3. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

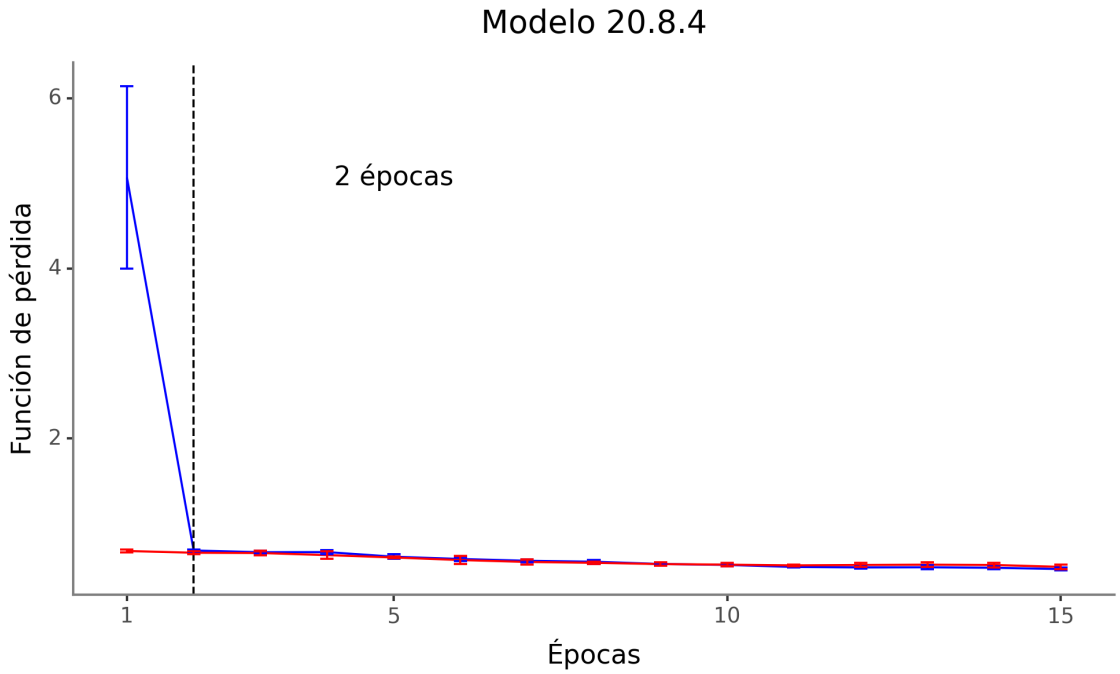


Figura A.48: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 20.8.4. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

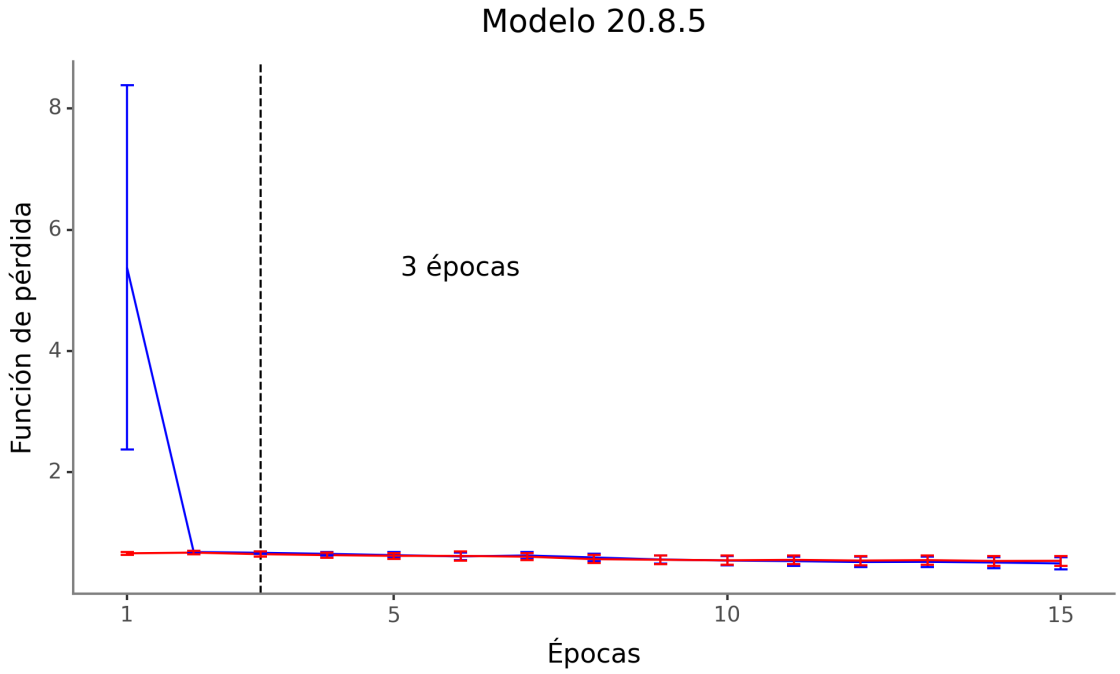


Figura A.49: Cantidad de épocas de entrenamiento seleccionadas (línea punteada negra) para modelo 20.8.5. Función de pérdida dado la cantidad de épocas de entrenamiento para entrenamiento (azul) y prueba (rojo)

A.1.2. Tablas

Modelo	*Batch size*	*Optimizer*	*Learning rate*	*Data augmentation*	**Épocas**
1	16	Adam	0.0001	Sí	**2**
2				No	**2**
3			0.001	Sí	**2**
4				No	**2**
5		SGD	0.0001	Sí	**3**
6				No	**2**
7			0.001	Sí	**2**
8				No	**3**
9	64	Adam	0.0001	Sí	**4**
10				No	**3**
11			0.001	Sí	**2**
12				No	**2**
13		SGD	0.0001	Sí	**5**
14				No	**3**
15			0.001	Sí	**2**
16				No	**2**
17	256	Adam	0.0001	Sí	**5**
18				No	**4**
19			0.001	Sí	**2**
20				No	**3**
21		SGD	0.0001	Sí	**10**
22				No	**5**
23			0.001	Sí	**3**
24				No	**4**

Cuadro A.1.1: Cantidad de épocas seleccionadas para cada combinación de hiperparámetros

Modelo	Primera capa	Segunda capa	**épocas**
20.1	32	0	**3**
20.2		16	**3**
20.3		32	**3**
20.4	64	0	**4**
20.5		32	**5**
20.6		64	**4**
20.7	128	0	**4**
20.8		64	**4**
20.9		128	**3**
20.10	256	0	**3**
20.11		128	**6**
20.12		256	**5**
20.13	512	0	**3**
20.14		256	**6**
20.15		512	**4**
20.16	1,024	0	**3**
20.17		512	**7**
20.18		1,024	**7**

Cuadro A.1.2: Cantidad de épocas seleccionadas para cada combinación de neuronas

Modelo	**épocas**
20.8.1	**6**
20.8.2	**5**
20.8.3	**3**
20.8.4	**2**
20.8.5	**3**

Cuadro A.1.3: Cantidad de épocas seleccionadas para cada modelo de *fine tuning*

A.2. Desempeño modelos

Modelo	Conjunto	*Accuracy*	*Precision*	*Recall*	*F2-score*
1	entrenamiento	0.72 ± 0.01	0.66 ± 0.02	0.65 ± 0.04	0.65 ± 0.03
	prueba	0.73 ± 0.04	0.68 ± 0.05	0.6 ± 0.16	0.61 ± 0.13
2	entrenamiento	0.79 ± 0.01	0.75 ± 0.02	0.73 ± 0.04	0.73 ± 0.04
	prueba	0.77 ± 0.03	0.72 ± 0.07	0.68 ± 0.05	0.69 ± 0.05
3	entrenamiento	0.7 ± 0.02	0.63 ± 0.04	0.6 ± 0.16	0.6 ± 0.12
	prueba	0.75 ± 0.03	0.73 ± 0.1	0.61 ± 0.11	0.63 ± 0.09
4	entrenamiento	0.76 ± 0.02	0.73 ± 0.05	0.64 ± 0.19	0.64 ± 0.15
	prueba	0.79 ± 0.03	0.75 ± 0.08	0.69 ± 0.07	0.7 ± 0.07
5	entrenamiento	0.72 ± 0.01	0.66 ± 0.03	0.65 ± 0.01	0.65 ± 0.01
	prueba	0.75 ± 0.03	0.75 ± 0.08	0.58 ± 0.15	0.6 ± 0.12
6	entrenamiento	0.77 ± 0.01	0.74 ± 0.03	0.68 ± 0.02	0.69 ± 0.02
	prueba	0.77 ± 0.04	0.74 ± 0.09	0.64 ± 0.09	0.66 ± 0.09
7	entrenamiento	0.65 ± 0.01	0.63 ± 0.06	0.34 ± 0.27	0.35 ± 0.23
	prueba	0.67 ± 0.06	0.7 ± 0.12	0.4 ± 0.29	0.41 ± 0.27
8	entrenamiento	0.65 ± 0.02	0.73 ± 0.11	0.28 ± 0.21	0.3 ± 0.19
	prueba	0.71 ± 0.04	0.76 ± 0.08	0.38 ± 0.13	0.42 ± 0.12
9	entrenamiento	0.75 ± 0.01	0.69 ± 0.01	0.68 ± 0.03	0.68 ± 0.03
	prueba	0.77 ± 0.03	0.75 ± 0.07	0.63 ± 0.08	0.65 ± 0.08
10	entrenamiento	0.8 ± 0.01	0.75 ± 0.02	0.75 ± 0.02	0.75 ± 0.01
	prueba	0.77 ± 0.03	0.75 ± 0.07	0.63 ± 0.05	0.65 ± 0.05
11	entrenamiento	0.67 ± 0.03	0.63 ± 0.05	0.45 ± 0.14	0.47 ± 0.12
	prueba	0.73 ± 0.05	0.72 ± 0.04	0.55 ± 0.15	0.57 ± 0.14
12	entrenamiento	0.75 ± 0.01	0.73 ± 0.07	0.63 ± 0.15	0.64 ± 0.11
	prueba	0.77 ± 0.05	0.73 ± 0.07	0.66 ± 0.12	0.67 ± 0.11

Cuadro A.2.1: Resultados métricas obtenidas en la búsqueda de hiperparámetros por los modelos 1 al 12

Modelo	Conjunto	*Accuracy*	*Precision*	*Recall*	*F2-score*
13	entrenamiento	0.72 ± 0.01	0.65 ± 0.03	0.61 ± 0.02	0.62 ± 0.02
	prueba	0.74 ± 0.03	0.73 ± 0.06	0.54 ± 0.1	0.57 ± 0.1
14	entrenamiento	0.78 ± 0.01	0.73 ± 0.02	0.68 ± 0.03	0.69 ± 0.02
	prueba	0.77 ± 0.04	0.73 ± 0.08	0.64 ± 0.09	0.66 ± 0.09
15	entrenamiento	0.72 ± 0.02	0.65 ± 0.02	0.61 ± 0.05	0.62 ± 0.04
	prueba	0.74 ± 0.03	0.73 ± 0.08	0.55 ± 0.07	0.58 ± 0.07
16	entrenamiento	0.76 ± 0.02	0.73 ± 0.05	0.65 ± 0.04	0.66 ± 0.03
	prueba	0.77 ± 0.04	0.74 ± 0.07	0.65 ± 0.08	0.66 ± 0.08
17	entrenamiento	0.72 ± 0.01	0.65 ± 0.02	0.64 ± 0.03	0.64 ± 0.03
	prueba	0.74 ± 0.02	0.73 ± 0.08	0.56 ± 0.04	0.58 ± 0.04
18	entrenamiento	0.79 ± 0.02	0.73 ± 0.03	0.72 ± 0.05	0.72 ± 0.04
	prueba	0.77 ± 0.02	0.73 ± 0.06	0.63 ± 0.06	0.65 ± 0.06
19	entrenamiento	0.63 ± 0.02	0.74 ± 0.12	0.24 ± 0.29	0.25 ± 0.26
	prueba	0.66 ± 0.07	0.74 ± 0.09	0.21 ± 0.27	0.23 ± 0.27
20	entrenamiento	0.69 ± 0.13	0.63 ± 0.13	0.8 ± 0.23	0.73 ± 0.17
	prueba	0.71 ± 0.15	0.67 ± 0.17	0.76 ± 0.15	0.71 ± 0.05
21	entrenamiento	0.69 ± 0.01	0.62 ± 0.02	0.6 ± 0.02	0.6 ± 0.02
	prueba	0.71 ± 0.01	0.71 ± 0.07	0.47 ± 0.05	0.5 ± 0.05
22	entrenamiento	0.75 ± 0.01	0.72 ± 0.03	0.63 ± 0.03	0.65 ± 0.02
	prueba	0.76 ± 0.02	0.74 ± 0.07	0.62 ± 0.03	0.64 ± 0.04
23	entrenamiento	0.68 ± 0.02	0.63 ± 0.03	0.48 ± 0.12	0.5 ± 0.11
	prueba	0.7 ± 0.02	0.76 ± 0.06	0.37 ± 0.06	0.41 ± 0.05
24	entrenamiento	0.78 ± 0.01	0.74 ± 0.01	0.69 ± 0.02	0.7 ± 0.02
	prueba	0.78 ± 0.04	0.74 ± 0.08	0.67 ± 0.08	0.68 ± 0.08

Cuadro A.2.2: Resultados métricas obtenidas en la búsqueda de hiperparámetros por los modelos 13 al 24

Modelo	Conjunto	*Accuracy*	*Precision*	*Recall*	*F2-score*
20.1	entrenamiento	0.72 ± 0.04	0.69 ± 0.11	0.64 ± 0.27	0.62 ± 0.22
	prueba	0.77 ± 0.03	0.73 ± 0.06	0.68 ± 0.11	0.69 ± 0.1
20.2	entrenamiento	0.6 ± 0.01	0.34 ± 0.15	0.02 ± 0.01	0.02 ± 0.01
	prueba	0.6 ± 0.03	0.66 ± 0.13	0.01 ± 0.01	0.01 ± 0.01
20.3	entrenamiento	0.6 ± 0.01	0.19 ± 0.28	0.0 ± 0.01	0.01 ± 0.01
	prueba	0.6 ± 0.03	0.35 ± 0.49	0.01 ± 0.01	0.03 ± 0.02
20.4	entrenamiento	0.65 ± 0.06	0.57 ± 0.06	0.68 ± 0.39	0.61 ± 0.32
	prueba	0.74 ± 0.09	0.66 ± 0.15	0.86 ± 0.13	0.79 ± 0.05
20.5	entrenamiento	0.61 ± 0.02	0.32 ± 0.19	0.4 ± 0.54	0.34 ± 0.45
	prueba	0.63 ± 0.05	0.73 ± 0.19	0.4 ± 0.53	0.34 ± 0.46
20.6	entrenamiento	0.6 ± 0.01	0.48 ± 0.19	0.02 ± 0.01	0.02 ± 0.02
	prueba	0.6 ± 0.03	0.69 ± 0.19	0.01 ± 0.01	0.02 ± 0.01
20.7	entrenamiento	0.76 ± 0.03	0.7 ± 0.06	0.75 ± 0.18	0.73 ± 0.13
	prueba	0.78 ± 0.03	0.74 ± 0.08	0.69 ± 0.05	0.7 ± 0.05
20.8	entrenamiento	0.67 ± 0.04	0.55 ± 0.03	0.96 ± 0.02	0.84 ± 0.01
	prueba	0.72 ± 0.03	0.6 ± 0.05	0.93 ± 0.05	0.83 ± 0.02
20.9	entrenamiento	0.64 ± 0.02	0.53 ± 0.05	0.68 ± 0.4	0.61 ± 0.34
	prueba	0.67 ± 0.07	0.64 ± 0.18	0.78 ± 0.42	0.68 ± 0.37
20.10	entrenamiento	0.76 ± 0.03	0.69 ± 0.05	0.74 ± 0.21	0.72 ± 0.17
	prueba	0.78 ± 0.04	0.73 ± 0.09	0.71 ± 0.06	0.72 ± 0.06
20.11	entrenamiento	0.7 ± 0.01	0.57 ± 0.01	0.93 ± 0.02	0.82 ± 0.01
	prueba	0.72 ± 0.04	0.61 ± 0.07	0.88 ± 0.05	0.81 ± 0.03
20.12	entrenamiento	0.67 ± 0.02	0.58 ± 0.06	0.75 ± 0.25	0.69 ± 0.18
	prueba	0.73 ± 0.04	0.64 ± 0.09	0.82 ± 0.23	0.76 ± 0.17
20.13	entrenamiento	0.76 ± 0.04	0.74 ± 0.03	0.62 ± 0.17	0.64 ± 0.14
	prueba	0.78 ± 0.04	0.75 ± 0.07	0.66 ± 0.11	0.68 ± 0.1
20.14	entrenamiento	0.77 ± 0.02	0.69 ± 0.06	0.8 ± 0.14	0.77 ± 0.1
	prueba	0.76 ± 0.03	0.69 ± 0.06	0.76 ± 0.14	0.74 ± 0.11
20.15	entrenamiento	0.74 ± 0.02	0.68 ± 0.09	0.73 ± 0.18	0.7 ± 0.13
	prueba	0.75 ± 0.02	0.66 ± 0.03	0.75 ± 0.16	0.73 ± 0.12
20.16	entrenamiento	0.78 ± 0.02	0.71 ± 0.04	0.78 ± 0.07	0.76 ± 0.05
	prueba	0.78 ± 0.03	0.73 ± 0.06	0.71 ± 0.07	0.71 ± 0.06
20.17	entrenamiento	0.78 ± 0.02	0.72 ± 0.05	0.76 ± 0.12	0.75 ± 0.09
	prueba	0.77 ± 0.03	0.71 ± 0.1	0.75 ± 0.09	0.73 ± 0.06
20.18	entrenamiento	0.78 ± 0.01	0.73 ± 0.06	0.73 ± 0.15	0.73 ± 0.11
	prueba	0.77 ± 0.04	0.72 ± 0.07	0.69 ± 0.15	0.69 ± 0.13
20.19	entrenamiento	0.81 ± 0.03	0.76 ± 0.02	0.76 ± 0.12	0.76 ± 0.09
	prueba	0.77 ± 0.03	0.72 ± 0.07	0.68 ± 0.09	0.68 ± 0.08
20.20	entrenamiento	0.81 ± 0.01	0.77 ± 0.01	0.74 ± 0.03	0.75 ± 0.03
	prueba	0.77 ± 0.03	0.74 ± 0.07	0.63 ± 0.08	0.65 ± 0.07
20.21	entrenamiento	0.8 ± 0.01	0.73 ± 0.03	0.79 ± 0.08	0.78 ± 0.05
	prueba	0.77 ± 0.03	0.7 ± 0.07	0.71 ± 0.05	0.71 ± 0.05

Cuadro A.2.3: Resultados métricas obtenidas en la búsqueda de arquitectura por los modelos 20.1 al 20.18

Umbral	Conjunto	*Accuracy*	*Precision*	*Recall*	*F2-score*
0.062	entrenamiento	0.69 ± 0.02	0.56 ± 0.01	1.0 ± 0.0	0.86 ± 0.0
	prueba	0.67 ± 0.04	0.55 ± 0.05	0.99 ± 0.01	0.85 ± 0.02
0.094	entrenamiento	0.69 ± 0.02	0.56 ± 0.01	0.99 ± 0.0	0.86 ± 0.0
	prueba	0.68 ± 0.04	0.55 ± 0.05	0.98 ± 0.01	0.85 ± 0.02
0.125	entrenamiento	0.7 ± 0.02	0.57 ± 0.01	0.99 ± 0.0	0.86 ± 0.0
	prueba	0.68 ± 0.04	0.56 ± 0.05	0.98 ± 0.01	0.85 ± 0.02
0.140	entrenamiento	0.7 ± 0.02	0.57 ± 0.01	0.99 ± 0.0	0.86 ± 0.0
	prueba	0.68 ± 0.04	0.56 ± 0.05	0.98 ± 0.01	0.85 ± 0.02
0.148	entrenamiento	0.7 ± 0.02	0.57 ± 0.01	0.99 ± 0.0	0.86 ± 0.0
	prueba	0.68 ± 0.04	0.56 ± 0.05	0.98 ± 0.01	0.85 ± 0.02
0.150	entrenamiento	0.7 ± 0.02	0.57 ± 0.01	0.99 ± 0.0	0.86 ± 0.0
	prueba	0.68 ± 0.04	0.56 ± 0.05	0.98 ± 0.01	0.85 ± 0.02
0.152	entrenamiento	0.7 ± 0.02	0.57 ± 0.01	0.99 ± 0.0	0.86 ± 0.0
	prueba	0.68 ± 0.04	0.56 ± 0.05	0.98 ± 0.01	0.85 ± 0.02
0.154	entrenamiento	0.7 ± 0.02	0.57 ± 0.01	0.99 ± 0.0	0.86 ± 0.0
	prueba	0.68 ± 0.04	0.56 ± 0.05	0.98 ± 0.01	0.85 ± 0.02
0.156	entrenamiento	0.7 ± 0.02	0.57 ± 0.01	0.99 ± 0.0	0.86 ± 0.0
	prueba	0.68 ± 0.04	0.56 ± 0.05	0.98 ± 0.01	0.85 ± 0.02
0.160	entrenamiento	0.7 ± 0.02	0.57 ± 0.01	0.99 ± 0.0	0.86 ± 0.0
	prueba	0.68 ± 0.04	0.56 ± 0.05	0.98 ± 0.01	0.85 ± 0.02
0.164	entrenamiento	0.7 ± 0.02	0.57 ± 0.01	0.99 ± 0.0	0.87 ± 0.0
	prueba	0.69 ± 0.04	0.56 ± 0.05	0.98 ± 0.01	0.85 ± 0.02
0.172	entrenamiento	0.7 ± 0.01	0.57 ± 0.01	0.99 ± 0.0	0.87 ± 0.0
	prueba	0.69 ± 0.04	0.56 ± 0.05	0.98 ± 0.01	0.85 ± 0.02
0.188	entrenamiento	0.71 ± 0.01	0.58 ± 0.01	0.99 ± 0.0	0.87 ± 0.0
	prueba	0.69 ± 0.04	0.56 ± 0.05	0.98 ± 0.01	0.85 ± 0.02
0.250	entrenamiento	0.71 ± 0.01	0.58 ± 0.01	0.99 ± 0.0	0.87 ± 0.0
	prueba	0.69 ± 0.04	0.57 ± 0.05	0.97 ± 0.01	0.85 ± 0.02
0.375	entrenamiento	0.72 ± 0.01	0.59 ± 0.01	0.98 ± 0.0	0.87 ± 0.0
	prueba	0.7 ± 0.04	0.58 ± 0.05	0.96 ± 0.02	0.85 ± 0.02
0.500	entrenamiento	0.73 ± 0.02	0.6 ± 0.01	0.97 ± 0.01	0.87 ± 0.01
	prueba	0.71 ± 0.04	0.59 ± 0.05	0.95 ± 0.02	0.84 ± 0.02
0.750	entrenamiento	0.66 ± 0.03	0.94 ± 0.01	0.14 ± 0.08	0.17 ± 0.09
	prueba	0.64 ± 0.03	0.83 ± 0.09	0.1 ± 0.07	0.12 ± 0.08

Cuadro A.2.4: Resultados métricas obtenidas en la búsqueda de umbral