DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Predicción de notas de estudiantes de secundaria con Redes Neuronales

22 de febrero de 2023 — Redes, Sociedad y Economía

Integrante	LU	Correo electrónico
Matías Waehner	294/17	matiaswaehner@gmail.com

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

Índice

1. Introducción

1.1. Motivación

La educación tiene un rol de vital importancia en todas las sociedades del mundo, y muchos estados demostraron estar interesados en invertir dinero y recursos en mejorar la calidad de la misma. En particular, la educación secundaria juega un papel fundamental en la vida de cualquier persona, ya que es una etapa de transición entre la niñez y la adultez, con su consecuente inserción en la esfera laboral. Si bien a grandes rasgos el nivel educativo de las poblaciones del mundo ha venido mejorando, numerosos países siguen teniendo el grave problema del bajo desempeño escolar. Dentro de estos países podemos lamentablemente situar a Argentina: 19 % de la población entre 25 y 34 años de edad tenía en 2021 algún título terciario, comparado con un promedio de 47 % en otros países [1].

Por otro lado, en los últimos años muchas técnicas de *Machine Learning* han ganado renombre y tenido aplicaciones muy diversas. En particular, el *Deep Learning* ha venido teniendo mejor performance que métodos tradicionales de *Machine Learning* en tareas como clasificación, regresión, segmentación, entre otras [2–5]. Lo que distingue al *Deep Learning* de otras técnicas es que no requiere una extracción manual de características de los datos, sino que las redes neuronales se entrenan para detectar automáticamente los patrones que mejor se adaptan a ellos. Dentro de las aplicaciones más novedosas, encontramos reconocimiento de objetos, traducción por computadora, diagnósticos con imágenes médicas, vehículos autónomos, entre otras. El ámbito educativo es un buen candidato de aplicación para estas técnicas: existen varias bases de datos online con datos demográficos y socio-económicos de los estudiantes, que por su volumen son idóneas para la aplicación de herramientas automatizadas como redes neuronales.

1.2. Objetivos

¿Qué estudiantes van a mantener su escolaridad y seguir yendo a clase? ¿Cuáles tienen más riesgo de abandonar? ¿Qué materias son más atractivas para los estudiantes? ¿Cuáles son los principales factores del desempeño académico? ¿Es posible predecirlo? Estas son algunas de las preguntas que nos gustaría contestar ayudándonos de nuestros modelos. En este trabajo nos enfocaremos en la última.

Nuestro objetivo es reproducir y mejorar los resultados de un experimento realizado en 2008 que buscó predecir las notas alcanzadas por alumnos de secundaria de Portugal a partir de varios atributos utilizando diversos métodos de *Machine Learning* [6]. Portugal, al igual que Argentina, tiene el problema de tener numerosos estudiantes con bajo desempeño escolar, que lo sitúan entre los eslabones más bajos de Europa. Dado que nos va a interesar construir una red neuronal propia, compararemos nuestros resultados con los de aquel método. El objetivo principal será entonces, dada una serie de atributos demográficos y socioeconómicos de estudiantes que atienden a las clases de Portugués y de Matemática, predecir quiénes aprobaron y quiénes no al final de la cursada. Realizar este tipo de predicciones puede ser útil en contextos educativos donde haya estudiantes con escolaridad en riesgo, ya que identificarlos tempranamente permitiría a los docentes y trabajadores educativos tomar medidas para mejorar sus condiciones, su desempeño y evitar el abandono de la escuela.

2. Desarrollo experimental

2.1. Recolección de datos

Los datos del trabajo en que nos basamos corresponden al año escolar 2005-2006 de dos escuelas públicas de la región de Antejo, Portugal. Se recolectaron atributos demográficos y socioeconómicos de los estudiantes y sus familias, junto a las notas alcanzadas por los alumnos para dos materias: matemáticas y portugués. Como al momento de la recolección aún no había un sistema informático funcionando para aquellas escuelas, los datos recolectados vienen de reportes de las escuelas y cuestionarios completados por los estudiantes, ambos en papel.

El cuestionario fue completado por 788 estudiantes, de los cuales se descartaron 111 respuestas que no pudieron identificarse (necesarias para ser mezcladas con los reportes). Luego de unificar los datos recolectados, los atributos finales que fueron usados en el trabajo original, y también lo serán en este, son los del cuadro 1.

Atributo	Descripción
sex	sexo (binario: varón o mujer)
age	edad (numérico: de 15 a 22)
school	escuela (binaria: Gabriel Pereira or Mousinho da Silveira)
address	tipo de dirección (binario: urbana o rural)
Pstatus	Estado de cohabitación de los padres (binario: juntos o separados)
Medu	Educación de la madre (numérico: de 0 a 4[a])
Mjob	Trabajo de la madre (nominal[b])
Fedu	Educación del padre (numérico: de 0 a 4[a])
Fjob	Trabajo del padre (nominal[b])
guardian	Responsable del estudiante (nominal: madre, padre o otro)
famsize	Tamaño de la familia (binario: ≤ 3 o > 3)
famrel	Calidad de las relaciones familiares (numérico: de 1 - muy mal a 5 - excelente)
reason	Razón para elegir esta escuela (nominal: cerca de casa, reputación de la escuela, preferencia de curso u otro)
traveltime	Tiempo de viaje a la escuela (numérico: 1 - $<$ 15 min., 2 - 15 a 30 min., 3 - 30 min. a 1 hora o 4 - $>$ 1 hora)
studytime	Tiempo de estudio semanal (numérico: 1 - $<$ 2 horas, 2 - 2 a 5 horas, 3 - 5 a 10 horas o 4 - $>$ 10 horas)
failures	Número de fracasos en clases pasadas (numérico: n si $1 \leq n < 3$, de lo contrario 4)
schoolsup	Apoyo educativo extra en la escuela (binario: sí o no)
famsup	Apoyo educativo familiar (binario: sí o no)
activities	Actividades extracurriculares (binario: sí o no)
paidclass	Clases particulares extras (binario: sí o no)
internet	Acceso a Internet en el hogar (binario: sí o no)
nursery	Asistió a jardín de infantes (binario: sí o no)
higher	Quiere tener educación superior (binario: sí o no)
romantic	Con una relación romántica (binario: sí o no)
freetime	Tiempo libre después de la escuela (numérico: de 1 - muy bajo a 5 - muy alto)
goout	Salir con amigos (numérico: de 1 - muy bajo a 5 - muy alto)
Walc	Consumo de alcohol en el fin de semana (numérico: de 1 - muy bajo a 5 - muy alto)
Dalc	Consumo de alcohol en días laborables (numérico: de 1 - muy bajo a 5 - muy alto)
health	Estado de salud (numérico: de 1 – muy malo a 5 – muy bueno)
absences	Cantidad de ausencias a la escuela (numérico: de 0 a 93)
G1	Primera nota de la cursada (numérico: de 0 a 20)
G2	Segunda nota de la cursada (numérico: de 0 a 20)
G3	Nota final de la cursada (numérico: de 0 a 20)

Cuadro 1: atributos recolectados de los estudiantes.

a: 0 – ninguno, 1 – educación primaria (4to grado), 2 – 5to a 9no grado, 3 – educación secundaria, 4 – educación superior.

b: docente, personal de salud, servicio civil (e.g. administrativo o policía), responsable del hogar u otro.

2.2. Procesamiento de los datos

El procesamiento de los datos consiste en transformar los atributos de la tabla 1 en vectores numéricos para ser pasados como *input* de la red neuronal. En este trabajo replicamos el procesamiento hecho por la publicación original [6]. En concreto, los pasos fueron los siguientes:

1. Se binarizó el atributo G3, *aprobado* si G3 $\geq$ 10, *desaprobado* en caso contrario[1]. Este atributo será usado como *target* para entrenar la red.

2. Se descartaron G1 y G2 ya que, como se explica en el paper original, lo realmente interesante es predecir el desempeño del estudiante en la clase sin la "ayuda" extra que esto introduciría (estaríamos prediciendo el desempeño ayudándonos con el desempeño inmediatamente anterior).

3. Se transformaron las variables nominales (por ejemplo, *Mjob*) en su codificación *one-hot* [2].

4. Se normalizaron todos los atributos a media cero y desvío estándar uno. Se usó el método de *StandarScaler* de *scikit learn* [3].

La codificación *one-hot* crea una nueva columna para cada categoría, que será etiquetada de forma binaria. Esta metodología es común en el procesamiento de datos para el aprendizaje automático ya que permite eliminar la cercanía sin semántica en variables categóricas. Por ejemplo, si eligiéramos representar *Mjob* o *Fjob* (trabajo del padre o madre) de forma numérica (es decir, 1 - "docente", 2 - "personal de salud", 3 - "servicio civil", etc.), estaríamos haciendo que "docente" esté más cerca de "personal de salud" que de "servicio civil", introduciendo ruido al entrenamiento.

Por otro lado, la normalización hecha es ampliamente común en el aprendizaje automático, y sirve para facilitar el entrenamiento inicial de la red, ya que llevamos todas las variables a la misma escala.

2.3. Análisis exploratorio de los datos

En la figura 1 se puede visualizar la cantidad de alumnos aprobados y no aprobados en cada clase.

[1] En el sistema de calificación de Portugal, las notas se dan en el rango de 0 a 20, siendo 10 la nota mínima para aprobar.
[2] `https://es.wikipedia.org/wiki/One-hot`
[3] `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html`

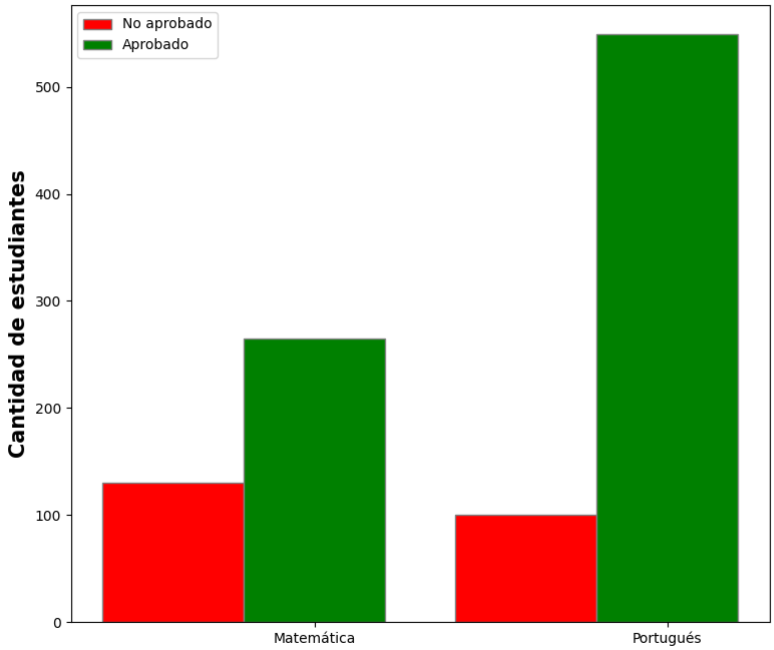


Figura 1: Visualización de los datos para las clases de Matemática y Portugués

Como podemos observar, ambos datasets están desbalanceados: hay más aprobados que desaprobados. Además, el desbalance entre ambos también es muy distinto: para la clase de matemática, la proporción de aprobados contra desaprobados es alrededor de 2 a 1, mientras que en la clase de portugués es más de 5 a 1. Este desbalance creará un sesgo en nuestro método de clasificación, que tendrá mayor impacto en la clase de portugués que en la de matemática. Cuando elijamos métricas para evaluar el desempeño de la clasificación, tendremos que tener en cuenta esta situación.

2.4. Modelo teórico

Una red neuronal es un modelo computacional de *Machine Learning* inspirado en las redes neuronales del cerebro humano. Consiste en un conjunto de nodos interconectados, llamados neuronas, que están distribuidas a lo largo de distintas capas. La conexión entre las neuronas se llama eje, y tanto las neuronas como los ejes tienen un peso que se ajusta a lo largo del entrenamiento del algoritmo. La primera capa de la red es la encargada de procesar el *input*, la última de dar el *output*, y las capas intermedias reciben el nombre de *capas ocultas*. Típicamente, cada neurona realiza su cómputo ayudándose de una función no lineal llamada función de activación[4]. Esta no-linearidad es una de las características más importantes de las redes neuronales, ya que es lo que lo distingue de una simple transformación lineal y le permite modelar sistemas más complejos. Otra cualidad distintiva de las redes neuronales es la de que no haya necesidad de realizar una extracción manual de características de los datos, sino que la red se alimenta directamente de ellos y es capaz de encontrar sus propios patrones según cómo ajuste los pesos. En la figura 2 se puede ver un ejemplo de una red neuronal sencilla de tres capas.

[4] `https://en.wikipedia.org/wiki/Activation_function`

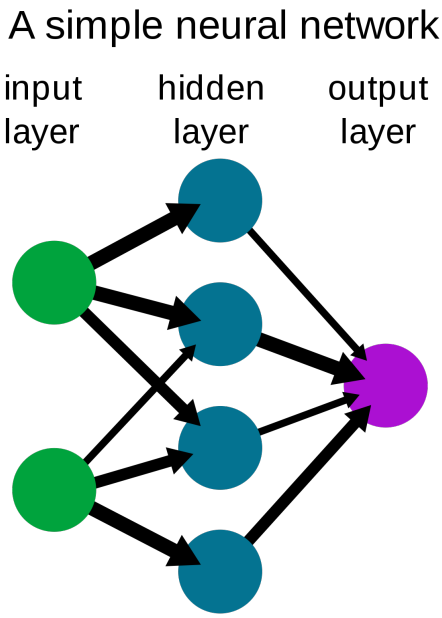


Figura 2: Ilustración de una red neuronal simple

2.4.1. Aprendizaje

La red es capaz de aprender al tener un conjunto de entrenamiento con valores de *output* ya conocidos, lo que se conoce como aprendizaje supervisado. El entrenamiento de la red se da en varios pasos llamados *epochs*. Luego de procesar el *input* se compara el *output* obtenido con la etiqueta original, mediante una función de costos también conocida como *loss*. En este contexto, utilizamos una función conocida como *binary cross entropy*, tradicionalmente usada para problemas de clasificación binaria.

$$L = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)\right) \tag{1}$$

En la ecuación, N es la cantidad de muestras, y_i es la etiqueta original (0 ó 1) e $\hat{y}_i$ es la predicción del modelo, para la muestra i. Esta función mide la diferencia entre las probabilidades predichas y las etiquetas originales, resultando en un valor de *loss* bajo para predicciones cercanas a la etiqueta original, y alto para valores alejados. Intuitivamente, se puede ver cómo para la etiqueta 0, el primer término dentro de la suma desaparece y el segundo se "activa", mientras que sucede lo contrario para la etiqueta 1.

Para maximizar o minimizar el valor de *loss* se usan algoritmos de optimización. En este experimento decidimos usar el popular algoritmo conocido como *adaptive moment estimation* o Adam [7]. En Adam, el impacto del ajuste realizado a los parámetros durante el entrenamiento no es fijo, sino que es variable y se adapta a lo largo del tiempo. Esto permite hacer actualizaciones más grandes para ciertos parámetros y más pequeñas para otros, resultando en una convergencia más rápida. A alto nivel, Adam es un algoritmo computacionalmente eficiente y robusto, ampliamente usado en la práctica y ha mostrado buenos resultados en una variedad de tareas.

2.4.2. Validación

Luego de cada *epoch* hay una etapa de validación, en donde evaluamos la red sobre un conjunto homónimo, distinto al de entrenamiento. La función de *loss* utilizada en el entrenamiento también se usa aquí, para monitorear cualitativamente el resultado obtenido por la red en esta etapa.

Como la tarea de la red en nuestro contexto es una clasificación binaria (estudiante aprobado o no aprobado), utilizamos las métricas de *Accuracy*, *Precision*, *Recall* y *F1-Score* para evaluar la performance.

Accuracy	(TP + TN)/(TN + TP + FN + FP)
Precision	TP/(TP + FP)
Recall	TP/(TP + FN)
F1-Score	$2 * (Precision * Recall)/(Precision + Recall)$

TP = True Positive, FP = False Positive, TN = True Negative, FN = False Negative.

Cuadro 2: métricas de performance.

Intuitivamente, el *recall* puede interpretarse (con nuestros datos) como "cuántos de todos los alumnos aprobados clasifiqué correctamente como aprobados", la precision como "cuántos de todos los que marqué como aprobados efectivamente eran aprobados", la *accuracy* como "cuántos en total clasifiqué bien (tanto aprobados como desaprobados)" , y el *F1-Score* como una combinación entre el recall y el precision.

La *accuracy* es en realidad una métrica engañosa para reportar sobre datasets desbalanceados: si para la clase de portugués reportamos a todos los alumnos como aprobados, tendremos inmediatamente una *accuracy* superior al 80 %. El trabajo original en que nos basamos reporta únicamente esta métrica, de forma que como corrección nuestra fue que decidimos reportar las otras métricas.

2.4.3. Hiperparámetros

Los hiperparámetros de la red nos permitirán experimentar con distintas arquitecturas de la misma e ir mejorando el resultado obtenido. La cantidad de capas ocultas y nodos en cada una de ellas son trivialmente hiperparámetros de la red. El *learning rate* es otro hiperparámetro de la red que define el impacto del ajuste realizado en cada paso. Un learning rate más grande nos permitirá converger más rápido pero probablemente llegando a un resultado más alejado del original; por otro lado, uno chico seguramente tardará más tiempo pero dará un resultado más certero. Los algoritmos de optimización recientes tales como Adam (el usado en este trabajo) adaptan el *learning rate* a lo largo del entrenamiento, logrando balancear el tiempo de convergencia y la precisión, y permitiéndonos evitar preocuparnos por este hiperparámetro.

También contamos con la cantidad de *epochs*, que es el número de veces que pasará el dataset entero por la red, y nos permitirá definir una cota superior para el entrenamiento. Por ejemplo, 100 *epochs* significará que la red entrenará usando la totalidad del dataset 100 veces. Si bien inicialmente la red mejorará mientras más la entrenemos, pasado un punto podrá comenzar a ajustarse excesivamente al conjunto de entrenamiento y empeorar sus resultados en la etapa de validación, fenómeno conocido como *overfitting*.

Vinculado a esto, un último hiperparámetro de la red es la probabilidad de *dropout*, que va a ser utilizado por cada neurona en su cómputo para no tener en cuenta cada uno de sus inputs con dicha probabilidad. Esta característica [8] es famosa en el mundo de las redes neuronales y se usa para evitar que la red *memorice* sus inputs, previniendo también el *overfitting*.

2.4.4. Arquitecturas elegidas

Las arquitecturas tendrán una capa de *input*, una capa oculta, y una capa de output compuesta de un único nodo. La primera arquitectura elegida servirá para experimentar con una capa oculta corta (10 nodos), y la segunda con una larga (30 nodos). La capa de input (nodos igual a la cantidad de atributos) y output (un nodo) se mantendrán igual.

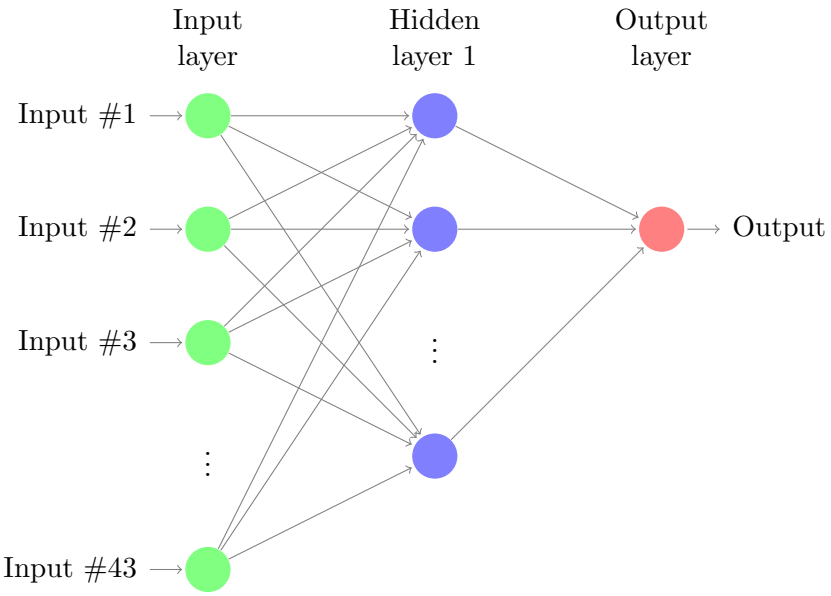


Figura 3: boceto de las arquitecturas elegidas. La cantidad de nodos en la capa oculta será lo que diferencie ambas arquitecturas.

Decidimos experimentar con el tamaño de la capa oculta ya que es una característica fundamental de las redes neuronales. Cada nodo de las capas ocultas reconocerá un patrón en las neuronas que tiene como *input*. Nuestra hipótesis es que una capa oculta de más nodos será capaz de aprender más patrones, y en consecuencia obtener un mejor resultado (a costa de un mayor consumo de recursos computacionales en el entrenamiento de la red, no medido en el presente trabajo).

Como función de activación se decidió usar ReLU, dada por el gráfico de 4. ReLU es la función de activación más comúnmente usada por su simpleza y eficiencia, manteniendo buenos resultados.

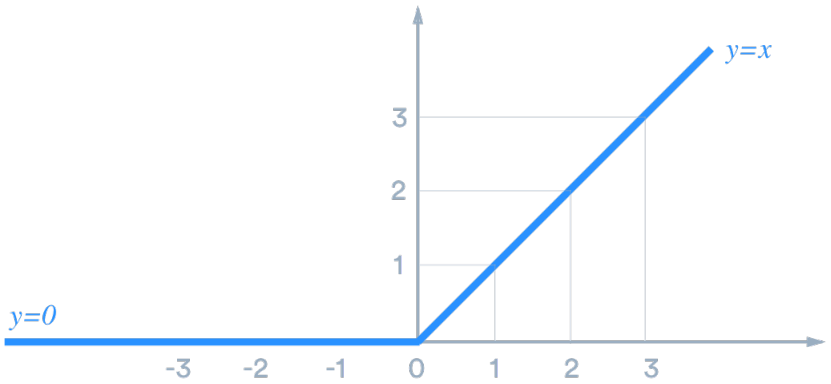


Figura 4: Gráfico de ReLU, f(x) = max(0, x)

Además, como prueba extra, volvimos a correr la arquitectura que arrojó mejores resultados [5] con una probabilidad de *dropout* de 0.5, como se recomienda en la publicación que introdujo la técnica [8].

En el cuadro 3 se exponen los hiperparámetros de las redes a correr en este experimento, junto al nombre que le pondremos a cada una.

Red	**Hidden Units**	**Epochs**	**Dropout**	**Activation**
Hidden_10	10	50	0.0	ReLU
Hidden_30	30	50	0.0	ReLU
Hidden_30_dropout	30	50	0.5	ReLU

Cuadro 3: Hiperparámetros de las redes elegidas.

[5] Presentados en la próxima sección.

2.4.5. Validación cruzada

Para medir la performance de un modelo es común utilizar la técnica de *Cross validation* o validación cruzada. Consiste en dividir los datos en k partes de igual tamaño, conocidas como *folds*. De esta forma, la red se entrena k veces, usando en cada una de ellas *k-1* folds como conjunto de entrenamiento y 1 *fold* como validación. A fin de cuentas, cada elemento del dataset se usa para validación una única vez, y la performance final del modelo se mide promediando la obtenida en cada *fold*. Una visualización de esto se muestra en la figura 5.

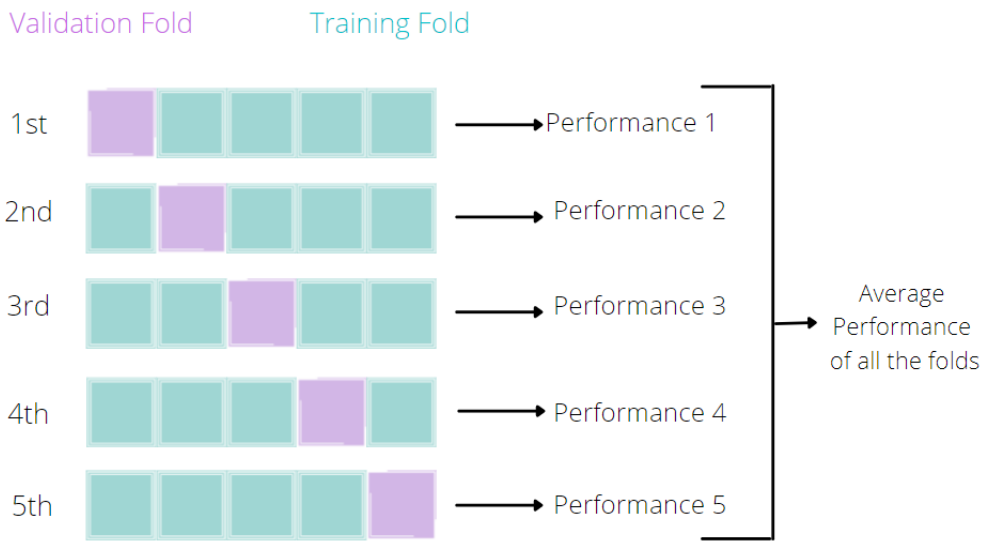


Figura 5: Ilustración de la técnica de validación cruzada para k = 5.

La validación cruzada ayuda a obtener un estimador más confiable de la performance del modelo, ya que reduce la variabilidad introducida por la forma en que se dividen los datos entrenamiento y validación y el consecuente *overfitting* del modelo.

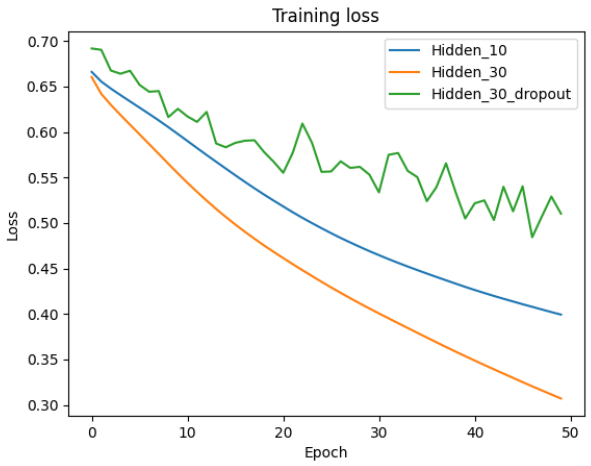
En este trabajo realizamos *10-fold cross-validation*, replicando la metodología del trabajo original. Es decir, cada ronda de entrenamiento tiene un 90 % de los datos destinados al conjunto de entrenamiento, y el resto al de validación. Las métricas de performance reportadas serán un promedio sobre todas estas rondas.

3. Resultados y análisis

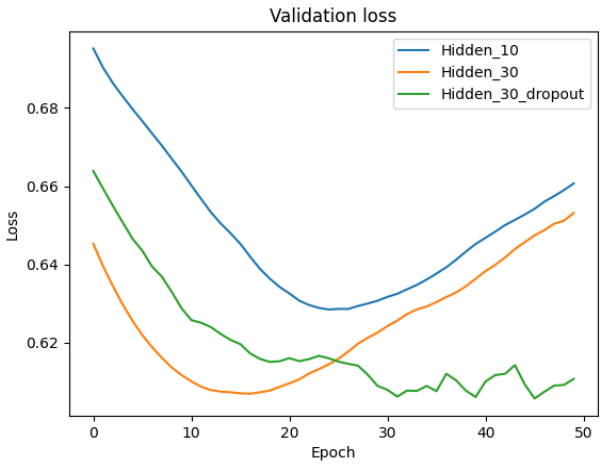
Volcamos en este capítulo los resultados luego de haber corrido la experimentación para las tres arquitecturas propuestas.

3.1. Análisis del entrenamiento

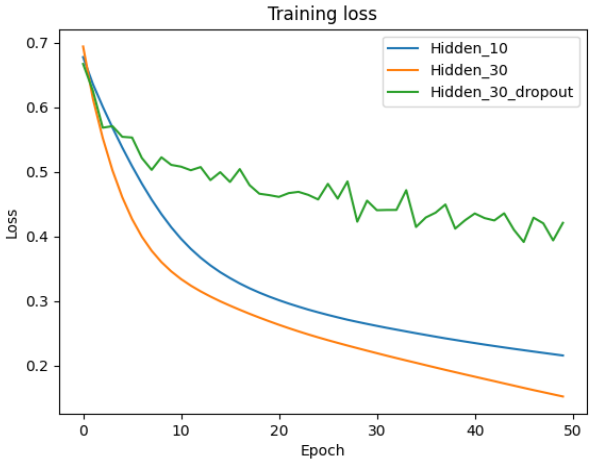
En el gráfico 6 mostramos los valores de *loss* medidos en cada epoch para el primer *fold* de la ejecución [6] para los datos de entrenamiento y los de validación de ambas clases.



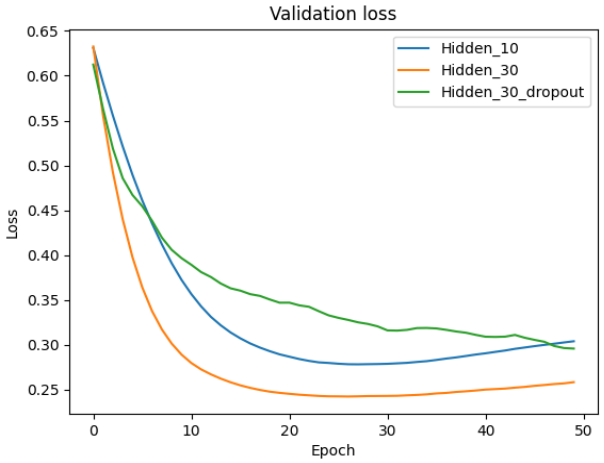
(a) Training loss, clase de matemática



(b) Validation loss, clase de matemática



(c) Training loss, clase de portugués



(d) Validation loss, clase de portugués

Figura 6: *Losses* obtenidas en el primer fold para las tres arquitecturas en sus conjuntos de entrenamiento y validación.

Podemos observar que, para todas las arquitecturas, la training loss (gráficos 6(a) y 6(c)) decrece mientras más epochs entrenamos la red. Esto tiene sentido, ya que justamente aprende a ajustarse a los datos de entrenamiento lo mejor posible. No obstante, la validation loss (gráficos 6(b) y 6(d)) empieza a tomar forma de "U" a medida que nos acercamos a los epochs finales. Esto sucede porque está habiendo *overfitting*, es decir, un sobre-ajuste a los datos de entrenamiento, indicándonos que la mejor performance del modelo seguramente se de en los epochs correspondientes al valle (parte más inferior) de dicha curva. Este *overfitting* es más notorio en la clase de matemática que en la de portugués, ya que como vimos en el gráfico 1, hay muchos menos datos en esa clase que en la otra, haciendo que el sobre-ajuste del modelo sea mayor.

Además, guiándonos por la validation loss, parecería que la mejor performance va a ser alcanzada, para la clase de portugués, por la red *Hidden_30*, y en la clase de matemática por esa o por *Hidden_30_dropout*.

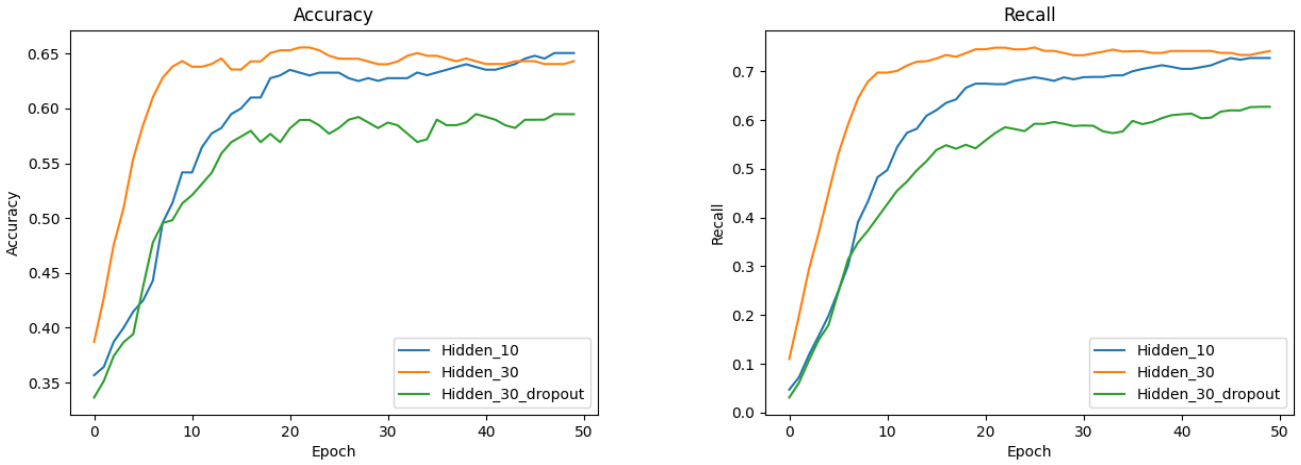
[6] Para simplificar, omitimos los restantes folds por tener resultados muy similares.

Algo interesante para observar es que los valores medidos para la red *Hidden_30_dropout* tienen una curva marcadamente distinta. En los gráficos de *training loss* podemos ver una curva más dentada, que seguramente se esté dando porque el *dropout*, al descartar aleatoriamente *inputs* de algunos de los nodos, introduce una variabilidad al entrenamiento. Por otro lado, las *validation loss* de la misma red parece no tener el mismo aspecto de "U" que las demás, indicando que la red está aún lejos del *overfitting* y tiene más espacio para mejorar, lo cual era el efecto deseado de introducir esta característica a la red. Parecería entonces que para este modelo, la cantidad de *epochs* que elegimos fue insuficiente.

Si bien la función de *loss* nos sirve para tener una idea del desempeño de nuestro modelo, siempre se deben elegir métricas que se ajusten al dominio del problema que se está resolviendo. En nuestro caso, son las métricas de clasificación dadas en el cuadro 2.

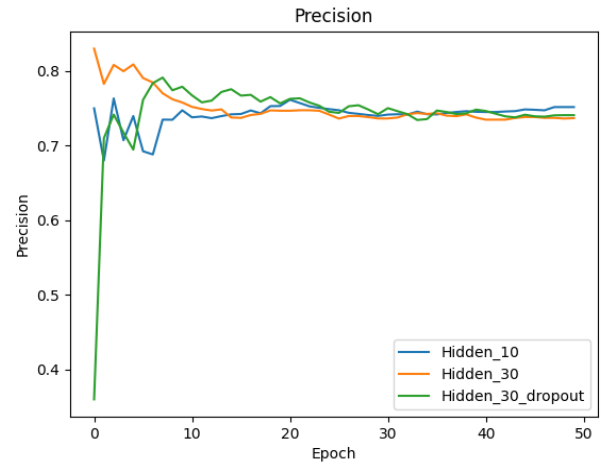
3.2. Métricas obtenidas

Volcamos entonces en los gráficos de la figura 7 las métricas obtenidas en los conjuntos de validación promediadas a través de los 10 *folds* para la clase de matemática, y en los de la figura 8 los mismos para la clase de portugués.

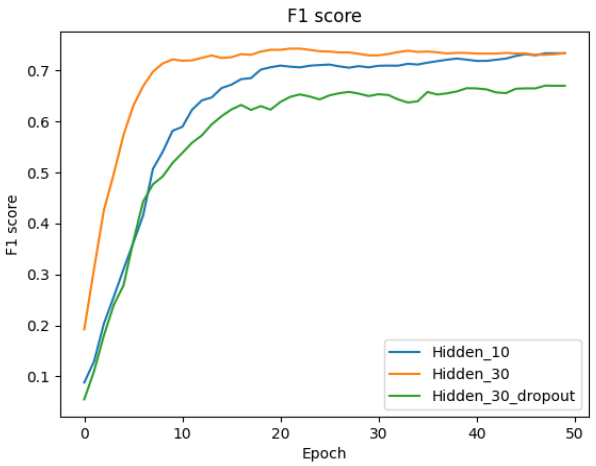


(a) Accuracy

(b) Recall

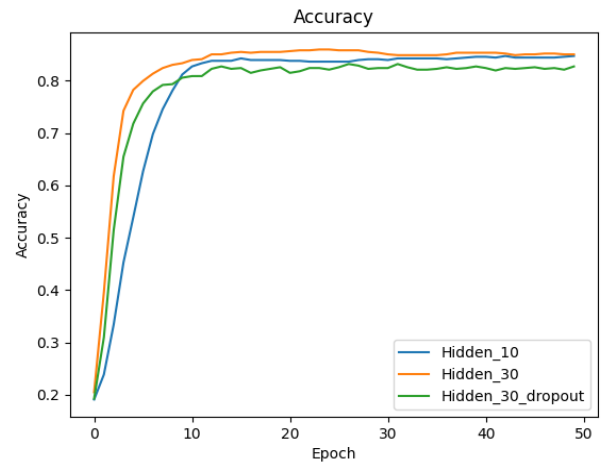


(c) Precision

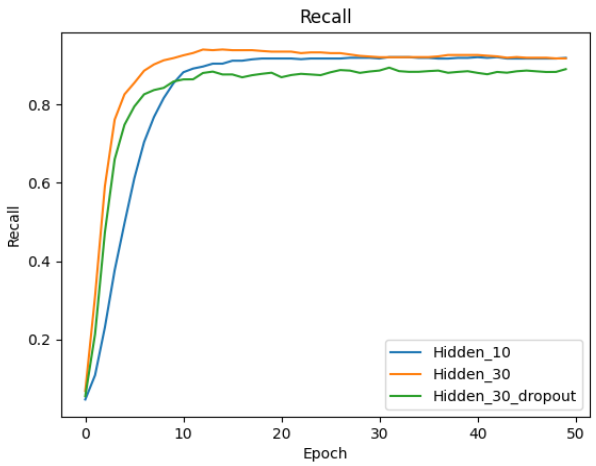


(d) F1 Score

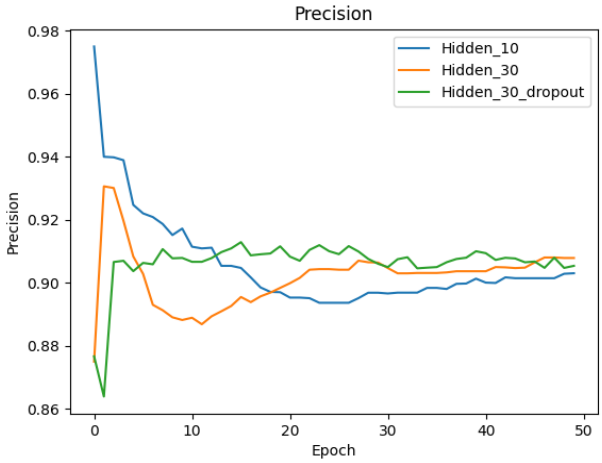
Figura 7: Métricas de performance sobre el conjunto de validación de la clase de matemática, promediadas a través de los 10 *folds*.



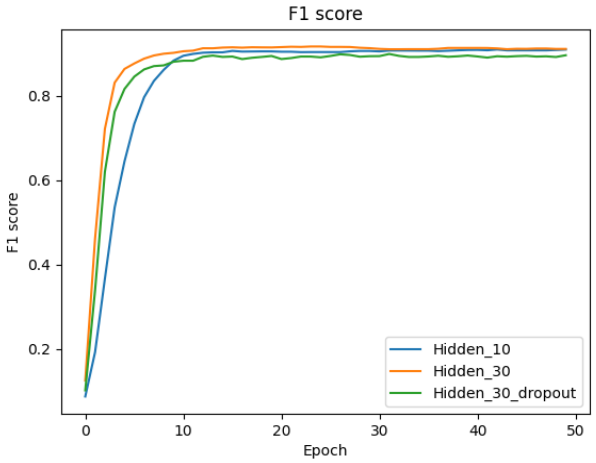
(a) Accuracy



(b) Recall



(c) Precision



(d) F1 Score

Figura 8: Métricas de performance sobre el conjunto de validación de la clase de portugués, promediadas a través de los 10 *folds*.

En ambos conjuntos de gráficos, vemos que el valor de *Recall* (figuras 7(b) y 8(b)) empieza muy bajo y sube rápidamente, mientras que la *Precision* (figuras 7(c) y 8(c)) tiene un comportamiento distinto, siendo incluso decreciente en varias de las corridas. Estas observaciones tienen sentido teniendo en cuenta la distribución de los datos (más aprobados que desaprobados): dado que el modelo comienza eligiendo uniformemente entre aprobado o no aprobado, la *precision* ("cuántos de todos los que marqué como aprobados efectivamente eran aprobados") será alta, y la *recall* ("cuántos de todos los alumnos aprobados clasifiqué correctamente como aprobados") será baja. En consecuencia, a medida que la red aprende es principalmente el *recall* el que tiene un cambio más abrupto.

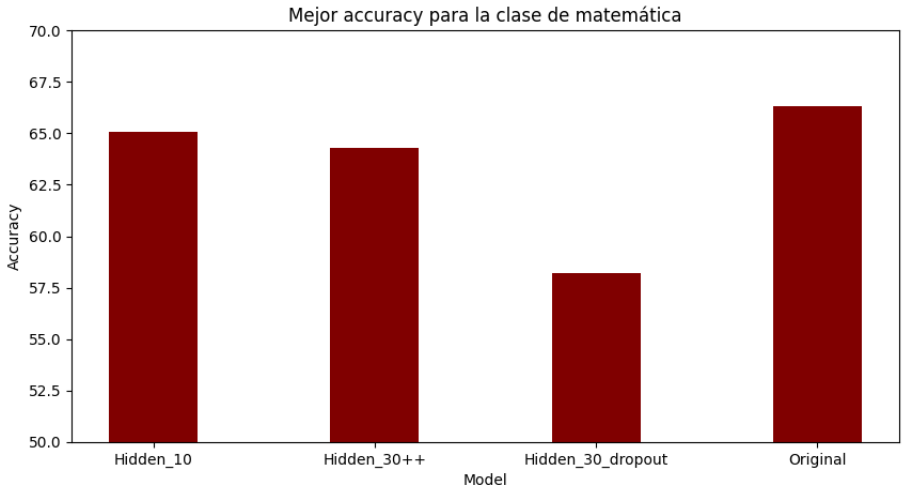
En cuanto a la *Accuracy* (figuras 7(a) y 8(a)), podemos ver que en ambos escenarios mejora mientras más epochs de entrenamiento tengamos. Para el caso de la clase de matemática, podemos ver un avance más ruidoso que para la clase de portugués, probablemente por el menor tamaño del dataset. Los valores más altos alcanzados son también menores para matemática que para portugués, seguramente no solo por contar con un dataset reducido sino también más balanceado (recordemos que la *accuracy* era engañosamente alta para datos desbalanceados).

La métrica de *F1 Score* (figuras 7(d) y 8(d)) nos confirma que el modelo aprende a clasificar independientemente del desbalance de los datos, y con mayor performance en el caso de la clase de portugués que la de matemática. Al tener más datos disponibles para el entrenamiento, es sensato pensar que la red esté aprendendiendo más patrones y alcanzando un mejor desempeño.

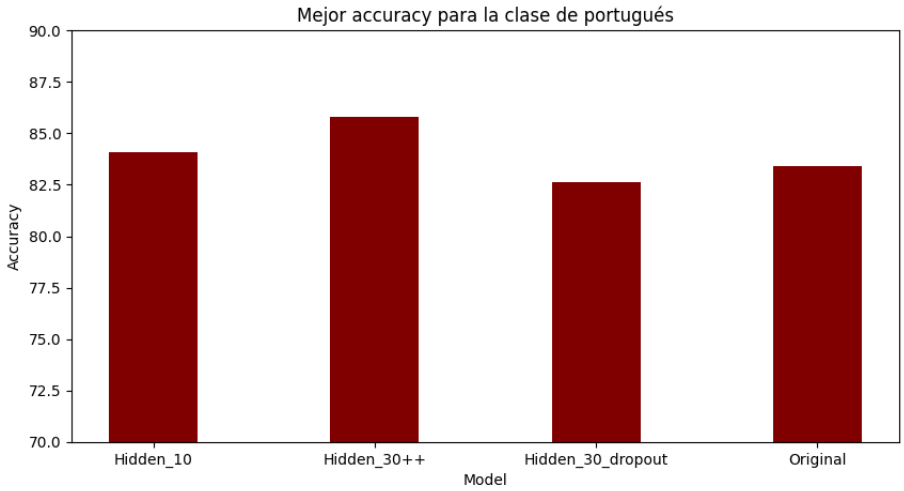
A juzgar por esta última métrica, parecería que el modelo ganador es Hidden_30, teniendo una amplitud mayor en performance en la clase de matemática que en la de portugués. Para verlo con mayor exactitud, volcamos en la siguiente sección una comparación de la performance de nuestras arquitecturas, y también de la reportada por el trabajo original.

3.3. Comparación entre modelos

En el gráfico 9 exponemos una comparación entre las mejores *accuracies* obtenidas en nuestro experimento y la reportada por el trabajo de Cortez y Silva ("Original").



(a) Mejores *accuracies*, matemática



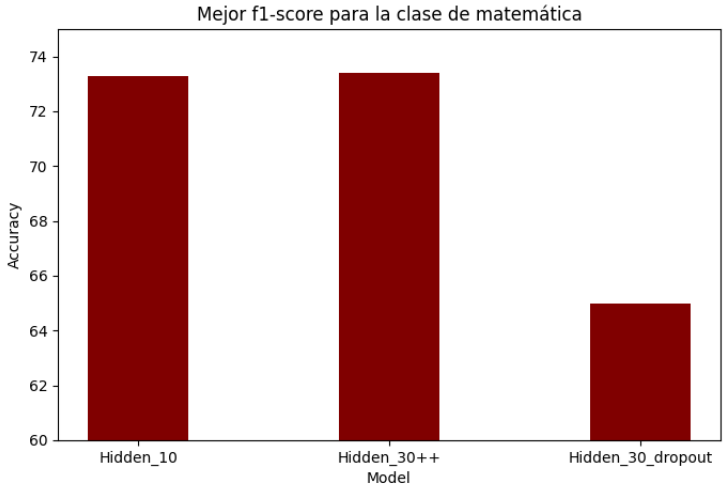
(b) Mejores *accuracies*, portugués

Figura 9: Mejores valores de *accuracy* obtenidos para ambas clases.

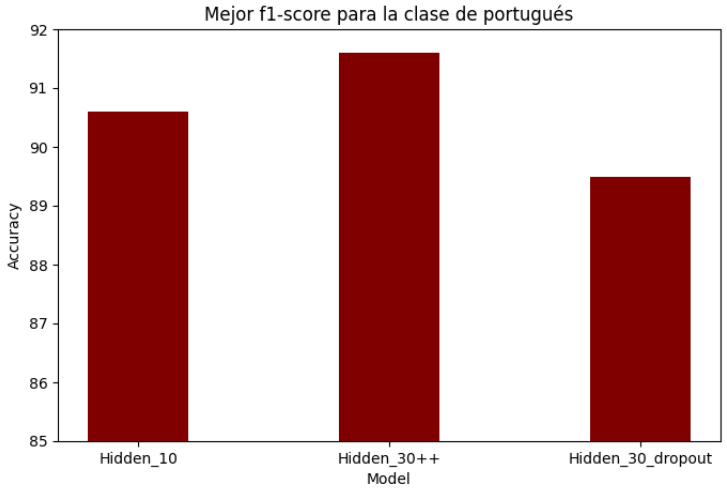
En la clase de matemática ninguno de nuestros modelos alcanzó la performance del original, sino que estuvieron levemente por debajo. En la clase de portugués, el modelo *Hidden_30* superó por unos pocos puntos de *accuracy* al original.

Con estos resultados no podemos decir haber mejorado el modelo propuesto originalmente, ya que solo sucedió en uno de los dos escenarios y con un margen pequeño. Sin embargo, nuestro entrenamiento fue reducido (50 *epochs*), por lo cual nos podemos quedar conformes con haber replicado satisfactoriamente los resultados obtenidos en la publicación. Es probable que un entrenamiento más prolongado alcance resultados un poco más altos, ya que en esta experimentación nos vimos limitados de tiempo y de recursos computacionales disponibles.

En el gráfico 10 mostramos una comparación de los f1-scores obtenidos únicamente por nuestros modelos, ya que el trabajo en que nos basamos no reporta esa métrica. Sin embargo, como ya dijimos, el f1-score es una métrica más confiable para medir el desempeño de tareas de clasificación con datos desbalanceados.



(a) Mejores *F1-Score*, matemática



(b) Mejores *F1-Score*, portugués

Figura 10: Mejores valores de *F1-Score* obtenidos para ambas clases.

A partir de estos últimos gráficos, podemos ver que el modelo *Hidden_30* tuvo un desempeño mayor o igual al de otros modelos en ambos escenarios, por lo que podríamos caratularlo como ganador. En particular, que tenga una brecha mayor en el escenario con más datos (clase de portugués) nos hace pensar que tener más nodos en la capa oculta le otorga más robustez a la hora de lidiar con una mayor variedad de datos, lo cual posiblemente no haya sido necesario con los reducidos datos de la clase de matemática, y razón por la cual la red *Hidden_10* tuvo un resultado similar con menos nodos.

La red *Hidden_30_dropout* merece un comentario especial en relación a los resultados que obtuvo. Si bien en todos los gráficos podemos apreciar que tuvo una performance menor, tal vez la probabilidad que se usó (0.5) fue muy alta y obligó al modelo a descartar muchos datos, con lo cual parece razonable pensar que se debería probar un valor más bajo. Por otro lado, como habíamos mencionado anteriormente, los gráficos 6(b) y 6(d) no muestran una tendencia a que la red haya comenzado a *overfittear* (no se ve la "U" que mencionamos), lo que nos hace pensar que tal vez al introducir *dropout* habría tenido sentido extender los *epochs* del entrenamiento para obtener así un resultado mejor.

Por último y para concluir la discusión, podemos afirmar que sí fue posible predecir el desempeño académico de los estudiantes a partir de los atributos recolectados en el trabajo en que nos basamos. Vimos en los gráficos 7(d) y 8(d) valores de F1 score altos que indican que el clasificador está distinguiendo bien a los estudiantes aprobados de los no aprobados. De todas formas, como vimos, la performance de la clasificación fue muy variable entre ambas clases, por lo que es necesario seguir experimentando con nuevos conjuntos de datos para obtener resultados conclusivos.

4. Conclusiones

A partir de las experimentaciones realizadas y los resultados encontrados concluimos:

- Logramos reproducir y alcanzar los resultados obtenidos en el trabajo de Silva y Cortez [6], ya que los que obtuvimos son muy parecidos.
- Logramos mejorar los resultados solo en uno de los dos dataset que ellos usaron, con lo cual no podemos decir que cumplimos nuestro objetivo de haber superado sus resultados. De todas formas, remontándonos a nuestro objetivo original ("¿Es posible predecir el desempeño académico?"), al haber replicado exitosamente los resultados previos podemos afirmar que sí es posible, con el *disclaimer* de que ambas experimentaciones fueron con un único conjunto de datos y debería replicarse la metodología en trabajos futuros con datos provenientes de otros contextos.
- Pudimos observar diferencias en el comportamiento del entrenamiento de nuestros modelos y explicar las mismas en relación a sus distintas características (por ejemplo, los gráficos "dentados" para la red de *dropout*).
- Realizamos una corrección sobre el trabajo original que consite en reportar una métrica que contemple el desbalance de los datos, como lo es el *F1-score*.
- La exploración de hiperparámetros de este trabajo fue algo escueta dada nuestra limitación de tiempo y recursos computacionales. Proponemos, como trabajo futuro, extender la cantidad de *epochs* a entrenar los modelos, en particular el modelo *Hidden_30_dropout*, ya que como vimos parecería ser el más probable de tener margen para seguir mejorando. También sería interesante explorar distintos valores usados como probabilidad para el *dropout*, dado que en este trabajo simplemente tomamos el valor recomendado.

Tanto el trabajo de Silva y Cortez como este son pruebas de concepto para abordar problemáticas educativas utilizando herramientas del estado del arte de la computación como lo son las redes neuronales, y sería interesante seguir replicando estas metodologías en otros conjuntos de datos. En concreto, la situación educativa de Argentina parece ser un buen disparador para experimentos de esta índole, con lo que proponemos replicar total o parcialmente este trabajo con datos de escuelas de nuestro país.

Referencias

[1] OECD, *Education at a Glance 2022*. 2022.

[2] S. Vesal, N. Ravikumar, A. Davari, S. Ellmann, and A. Maier, "Classification of breast cancer histology images using transfer learning," 2018.

[3] M. Aubreville, C. Knipfer, N. Oetter, C. Jaremenko, E. Rodner, J. Denzler, C. Bohr, H. Neumann, F. Stelzle, and A. Maier, "Automatic classification of cancerous tissue in laserendomicroscopy images of the oral cavity using deep learning," *Scientific Reports*, vol. 7, sep 2017.

[4] S. Chen, X. Zhong, S. Hu, S. Dorn, M. Kachelriess, M. M. Lell, and A. K. Maier, "Automatic multi-organ segmentation in dual energy ct using 3d fully convolutional network," 2018.

[5] T. Würfl, F. C. Ghesu, V. Christlein, and A. Maier, "Deep learning computed tomography," in *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2016* (S. Ourselin, L. Joskowicz, M. R. Sabuncu, G. Unal, and W. Wells, eds.), (Cham), pp. 432–440, Springer International Publishing, 2016.

[6] P. Cortez and A. Silva, *Using Data Mining to Predict Secondary School Student Performance*. EUROSIS, 2008.

[7] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[8] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.