Proyecto Deep Learning 2021

Marco Zuñiga
mazg95@gmail.com

Abstract

Las Redes Neuronales nos permiten trabajar problemas mas complejos que el enfoque tradicional de ML. Y para probar los diferentes problemas que se pueden resolver con este tipo de Modelos, en este reporte presentamos 3 casos de uso de Deep Learning. Utilizando los 3 modelos de DL mas conocidos, perceptron multicapa (MLP), redes convolucionales (CNN) y redes recurrentes (RNN)

1 Introducción

Para este trabajo se llevo a cabo el diseño y entrenamiento de 3 modelos distintos. Y se aplico a uno de los 3 a los problemas para los que mejor estaba preparado el modelo. Por ejemplo las redes neuronales como el MLP, son buenas con datos tabulares. Para este caso este tipo de red era ideal para el problema de Salud del Feto. En este problema teníamos varias observaciones organizadas en características en una forma de tabla. Por lo que se decidió utilizar una red neuronal sencilla para resolver este problema. Para el segundo problema, este requería un modelo mas especifico pues se tenia como objetivo tratar un problema espacial. Estos tipos de problemas necesitan tomar en cuenta patrones sin importar la localización del mismo. El problema que se decidió resolver con este tipo de modelos fue de Clasificación de Tumores Cerebrales, para ello se consumió un data set de exámenes de MRI.

Para el ultimo problema, ya que se deseaba aplicar una Red Recurrente para resolver algún problema secuencial se decidió utilizar la RNN para clasificar géneros musicales apartar del texto de la canción. Esto para ayudarse de modelos como LSTM que guardan cierta memoria para poder darle un contexto a las palabras que conforman la canción y lograr el objetivo de clasificar la letra de las canciones musicales.

Para resolver estos problemas se utilizo diferentes herramientas, el stack elegido para implementarlas fue Tensorflow GPU 2.5, CUDA 11.2, cuDNN 8.0, SciKit Learn, Numpy, Pandas, Spacy.

2 Parte 1: MLP

El problema seleccionado para tratar con un modelo de perceptrones multicapa fue uno de tipo clasificacion. Se busco un dataset relacionado a la salud para demostrar la capacidad y robustez de estos modelos con dataset estructurados. Para ello se selecciono el dataset de Fetal Health Classification (`https://www.kaggle.com/andrewmvd/fetal-health-classification`).

Los campos que contiene el dataset seleccionado para intentar resolver el problema con DL:

- FHR (BaseLine Value)
- Accelerations
- Fetal Movement
- Uterine Contractions
- Light Decelerations
- Severe Decelerations

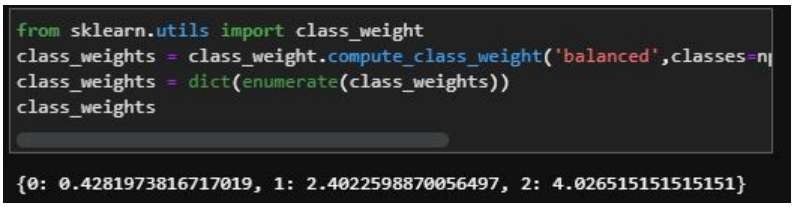

Figure 1: Calculo de pesos para las clases

- Prolongued Decelerations
- Abnoraml Short Term Variability
- Mean Value Short Term Variability
- Percentage of time with abnormal long term variability
- Mean value of long term variability
- Width of the histogram made using all values from a record
- Histogram minimum value
- Histogram maximum value
- Number of peaks in the exam histogram
- Number of zeroes in the exam histogram
- Hist mode
- Hist mean
- Hist Median
- Hist variance
- Histogram trend
- Fetal health

El objetivo de este set de datos es intentar construir un clasificador de la salud del feto para prevenir la mortalidad del niño como de la madre. Se busca reducir en base de los datos de exámenes de CTG. Las Naciones Unidas se han propuesto como expectativa reducir estas muertes de recién nacidos. Herramientas que puedan ayudar a clasificar casos anormales son necesarias, especialmente en lugares de escasos recursos. Por lo que este dataset fue construido por sus autores en base a Cardiogramas (CTGs) debido a su relativo bajo costo y esparcido acceso como una opción para analizar y determinar casos que necesiten ser atendidos para prevenir la mortalidad de bebe y/o madre.

Al dataset se le hizo un análisis exploratorio con el cual se pudo observar que el dataset esta desbalanceado. La mayoría de las observaciones son normales y un porcentaje pequeño son observaciones sospechosos y patológicos. Este tema se tendrá que tratar al momento de entrenar la red. Ya que sino la red aprenderá rápidamente como es un caso normal, pero lo que nos interesa es detectar casos sospechoso como patológicos.

La única operación que se le hizo a los datos de entrenamiento fue la normalización. Y se separo un set de entrenamiento (0.8) y otro de pruebas (0.2).

Para el desbalance de los datos se aplico una solucion que nos proporciona keras, a partir de un análisis de desbalance pudimos determinar el peso de cada clase. Esto con intención de indicarle a la función **fit** a que observaciones debería de darle mas importancia.

Se experimentaron con diferentes modelos para lograr el objetivo. Y el modelo base utilizado fue un modelo MLP con 5 capas. La primera capa de este modelo se utilizo con 1024 unidades ocultas, la segunda con 512 unidades ocultas, la tercera con 256 unidades ocultas y la ultima capa oculta se le asigno 128 neuronas. La capa de salida dado que es un problema de clasificación para un problema multi-clase, se diseño de 3 neuronas de salida.

Mediante experimentación se decidió agregar algunas variaciones que veremos mas adelante. Incluido el optimizador que también fue evaluado entre otros. Pero para este primer modelo se utilizo Adam, ya que este usualmente tiene un buen rendimiento.

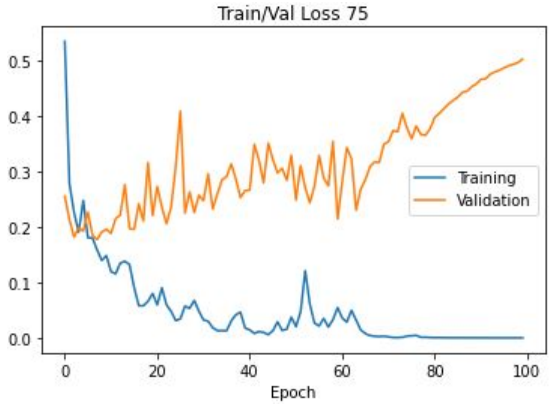

Figure 2: Grafica de Costo vs Epoch de Modelo 1

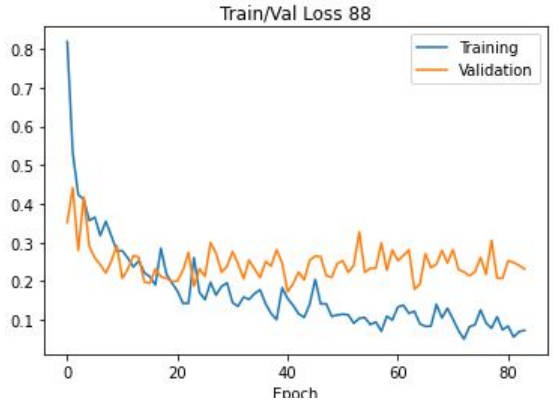

Figure 3: Grafica de Costo vs Epoch de Modelo #88

La función de costo elegida para minimizar fue la de Cross Entropy para encontrar los pesos de nuestras capas para poder resolver este problema. Para este tipo de problema en especifico donde las clases las tenemos representadas como enteros. Utilizamos el tipo de funcion de costo **Sparse Categorical Cross Entropy** y para evaluar el rendimiento de nuestro modelo utilizamos el acurracy especial diseñado para este tipo de problemas que es el **Sparse Categorical Acurracy**.

Podemos observar que para este primer modelo hay un overfitting por parte de nuestro modelo. Por lo que las tecnicas elegidas para intentar mitigar este problema fueron experimentar con el Dropout, el tamaño de nuestra red y se agrego un callback de Early Stopping para evitar que la funcion se sobreentrene.

El mejor modelo luego de varios experimentos fue el experimento numero #88. Ya que tuvo uno de los mejores desempeño en los datos de entrenamiento logrando un 97% y la métrica elegida para evaluar en los datos de prueba el desempeño de los modelos fue el **F1-score Weighted**. Esto porque nos interesaba tomar en cuenta el desbalance de las clases al hacer las predicciones, logrando un 93.75% en esa metrica.

Como conclusión pudimos encontrar en los experimentos que para este tipo de problemas el optimizador Adam es muy eficaz para reducir la función de costo. También pudimos observar que los modelos MLP tienden a sobre-ajustarse a los datos de entrenamiento y la técnica de usar Cross Validation nos permitió detectar, analizar y tunear nuestro modelo para evitar que aprendiera la representación de los datos de entrenamiento y pudiera generalizar una mejor forma para otras observaciones no vistas aun. Estos modelos quitan la necesidad prácticamente de tener que hacer preprocesamiento de datos. Pero nos trae un poco mas de trabajo en la parte de regularización y tratamiento del sobre-ajuste.

Layer (type)	Output Shape	Param #
dense_332 (Dense)	(None, 1024)	22528
dropout_143 (Dropout)	(None, 1024)	0
dense_333 (Dense)	(None, 128)	131200
dropout_144 (Dropout)	(None, 128)	0
dense_334 (Dense)	(None, 1024)	132096
dropout_145 (Dropout)	(None, 1024)	0
dense_335 (Dense)	(None, 128)	131200
dropout_146 (Dropout)	(None, 128)	0
dense_336 (Dense)	(None, 3)	387
softmax_84 (Softmax)	(None, 3)	0

Total params: 417,411
Trainable params: 417,411
Non-trainable params: 0

Figure 4: Arquitectura de Modelo Elegido

88	20210922-11:10:16	0.395657	0.938872	0.937448
51	20210922-10:42:56	0.571759	0.932602	0.933418
6	20210922-01:02:37	0.622141	0.929467	0.928964
76	20210922-11:07:15	0.688373	0.926332	0.926498
43	20210922-01:19:09	0.618986	0.926332	0.925434
27	20210922-01:15:44	0.540176	0.926332	0.924742
16	20210922-01:04:26	0.352599	0.924765	0.924526
45	20210922-01:19:24	0.621478	0.924765	0.923911

Figure 5: Resultados MLP

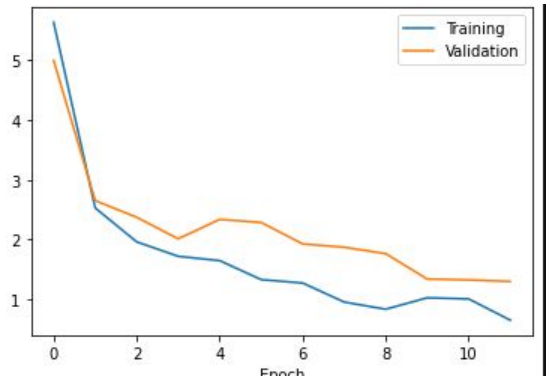

Figure 6: Costo vs Epochs Modelo Base

3 Parte 2: ConvNets

El problema elegido para aplicar ConvNets fue un problema también en el área de aplicaciones Médicas, en este caso se tomo un dataset de imágenes de MRI para detectar Tumores en el cerebro y poder clasificar si un examen de MRI contiene un tumor o no. Una tarea difícil sino imposible para una persona sin entrenamiento especializado. Por lo que se puso a prueba una ConvNets para ver si podía aprender a clasificar con un exactitud arriba del 90%.

El dataset utilizado fue obtenido de `https://www.kaggle.com/sartajbhuvaji/brain-tumor-classification-mri`. El dataset contiene diferentes imágenes ordenado por tipo de tumor contenida en la imagen o sino tiene tumor. Las imágenes tienen una dimension de 512x512x3. Y se utilizaron 2585 fotos de las 4 clases para entrenamiento, 285 para el proceso de Cross Validation y 394 imágenes como set de pruebas.

La forma en la que se decidió resolver este problema fue por medio de Transfer Learning. Utilizando modelos Convolucionales ya entrenados en millones de imágenes y que permitieran extraer las características mas importantes de las imágenes. Para ello se decidió utilizar experimentación de varias Redes ya entrenadas y que nos provee el paquete de Keras.

- ResNet50
- InceptionV3
- EfficientNetB0

Al igual que con los modelos MLP, se creo un método de experimentación y se configuraron los callbacks para ayudar al entrenamiento de este tipo de modelos.

Se pudo observar que ResNet50 e InceptionV3 no tuvieron grandes resultados para este tipo de problema, probablemente el entrenamiento previo no se ajusta a imágenes médicas de buena forma. Para estos modelos se tenia que normalizar la entrada en este caso solo era necesario escalar por 1/255 los datos de las imágenes. Por abajo de los esperado, ResNet consiguio un 72% e InceptionV3 un 77%.

El ultimo modelo utilizado para realizar el Transfer Learning fue el de EfficientNetB0 que es un tipo de modelo recientemente publicado y que ha sido entrenado en millones de imágenes en tamaño es el mas grande de los tres elegidos. Con un total de parámetros por los 4 millones. Y según la documentación este modelo incluye una capa de normalización. Y los resultados para realizar la tarea de clasificación supero las expectativas por arriba del 99%. Por lo que se escogió este modelo para realizar las predicciones y clasificar los MRI. En el set de pruebas mantuvo el desempeño.

4 Parte 3: RNN

La aplicación de un red recurrente neuronal requiere un problema que necesite tener en cuenta el contexto de una secuencia. Para ello se eligio un problema de clasificacion de letras de canciones. Con la intencion de intentar determinar el genero de la cancion a partir de su letra.

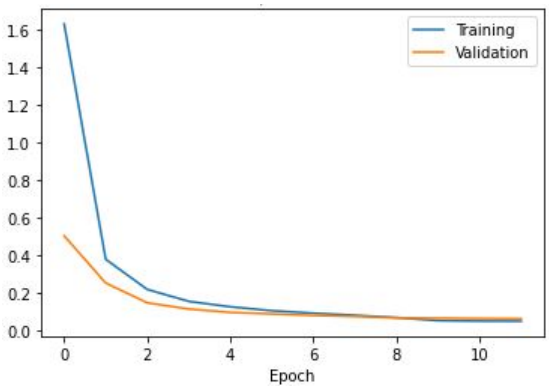

Figure 7: Costo vs Epochs TL EfficientNetB0

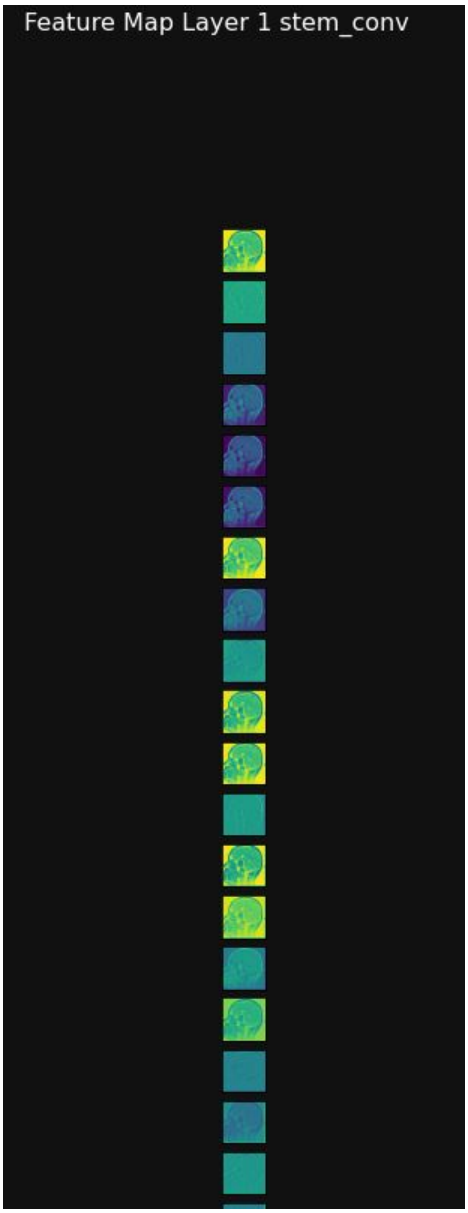

Figure 8: Representaciones Intermedias

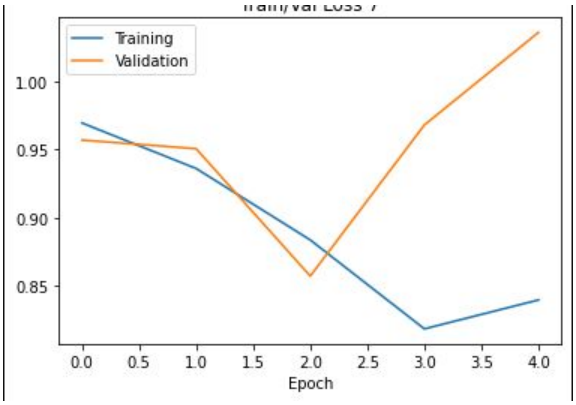

Figure 9: Costo vs Epoch Base RNN

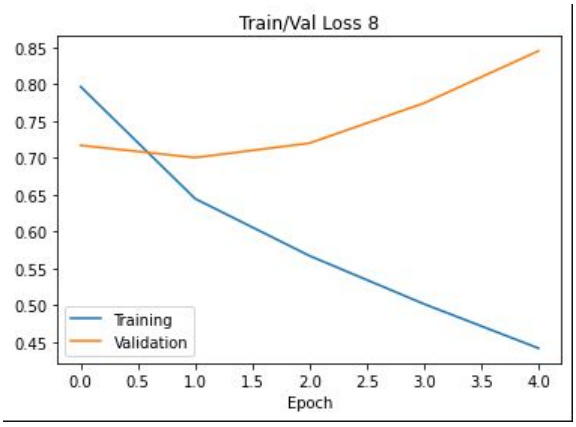

Figure 10: Costo vs Epoch RNN con Spacy

Para resolver este problema se utilizo el siguiente dataset: `https://www.kaggle.com/neisse/scrapped-lyrics-from-6-genres`. Este dataset presento un reto en la parte de limpieza, ya que contenía canciones en distintos idiomas, diferentes géneros para una misma canción o artista. Y era también necesario crear un embedding para alimentar nuestro modelo y entrenarlo.

El primer modelo entrenado con un preprocesamiento solo dejando las palabras mas comunes y quitando los signos de puntacion. Lo que significo un desempeño del modelo poco alentador.

Por lo que se decidio hacer un script para preprocesar las canciones con la libreria Spacy. Remover stopwords, palabras que no agregan ninguna informacion al contexto, y las puntuaciones.

Durante las pruebas tambien se pudo observar que un vocabulario muy grande al utilizar el Tokenizer tambien reduce la efectividad del modelo. Por lo que por medio de Cross Validation ese fue otro hyperparemtro para encontrar un tamaño de vocabulario que no afectara al entrenamiento del modelo.

Luego de preprocesada la data, se exporto a archivos npy. Para improtarlos en el script de experimentacion de las redes RNN.

Con estos cambios pudimos encontrar un modelo que en los datos de entrenamiento alcanzara un buena exactitud. Pero sufrimos de sobreajuste. Por lo que en los datos de entreno alcanzaba un 80% de accuracy pero en el set de entrenamiento no alcanazba un valor mayor al 65%.

Se puede observar después de los experimentos que estos modelos tienden a sobre-ajustarse. Por lo que es necesario aplicar la regularización a la parte Recurrente del Modelo.

Figure 11: Costo vs Epoch RNN con Spacy

References

[1] Ayres de Campos et al. (2000) *SisPorto 2.0 A Program for Automated Analysis of Cardiotocograms.* J Matern Fetal Med 5:311-318