Predicción de parámetros de líneas espectrales usando redes neuronales convolucionales (CNN) en python con el módulo de Keras

Pablo Olivares Z[a]

[a]*Escuela de Ingeniería Informática, Facultad de Ingeniería, Universidad de Valparaíso, Valparaíso, Chile*

Abstract

El objetivo de este estudio es desarrollar un algoritmo que utilice técnicas de machine learning (ML) y analizar el desempeño del modelo, en específico se utilizaran las redes neuronales profundas (DL). Evaluar los resultados de ejecución de la red neuronal convolucional (CNN) utilizando datos astronómicos de líneas espectrales sintéticas, estas serán utilizadas como datos de entrada para la red CNN y así poder predecir 3 datos parámetros estelares (Teff , log g, y vrot). La variación del modelo estará dada por la integración de una capa extra de convolución y otra de densidad para analizar los resultados.

Keywords: redes neuronales convolucionales, regresión, keras, python, deep learning

1. Introducción

La aplicación de modelos de machine learning (ML) e inteligencia artificial están siendo utilizados constantemente en el mundo astronómico durante esta última décadas [1] , este patrón de comportamiento esta justificado por la cantidad de datos que se están generando en los distintos observatorios tantos espaciales como terrestres. Esto genera como necesidad el análisis de estos datos de alguna manera que sea autónoma o automática. Los modelos de ML basados en redes neuronales profundas (DL) son normalmente utilizados en el mundo de la astronomía para automatizar los análisis, y obtener buenos rendimientos de estimaciones los parámetros estelares. La estimación de parámetros estelares representa una tarea que es fundamental en la astronomía[1].

En este trabajo se analizará el desempeño de una red neuronal convolucional (CNN) en la predicción de 3 parámetros estelares (Teff , log g, y vrot), esto sera determinado mediante el análisis de las distintas funciones de perdida, entre ellas el error absoluto medio (MAE),La diferencia relativa media (MRD) y la precisión (ACC), este ultimo con el objetivo de determinar que tan efectivo es el entrenamiento en x cantidad de epocas para nuestra red CNN. Esta red neuronal se caracteriza por tener un alto desempeño en procesamiento principalmente de imágenes, sonido, voz, entre otras. Para este caso utilizaremos datos correspondientes a espectros sintéticos de modelos de atmosferas estelares, y el analisis estara basado en una linea de tiempo especifica.

Al realizar la modificación de los hiper-parametros se observan mejoras importantes en los resultados de perdida y en la afinidad del entrenamiento obteniendo valor de acc 90% , validación de perdida 0.1171 y error absoluto medio 0.2052.

2. Métodos

Las redes neuronales convolucionas o CNN (del inglés covolutional neural networks) corresponden a un método del área de aprendizaje profundo ampliamente utilizadas para clasificación de imágenes , voz, ondas, entre otro.

Dentro de este modelo se puede aplicar la modificación de en los hiper parámetros de la red donde podremos aplicar distintos métodos de activación, filtros, capas de la red e incluso cantidad de neuronas.

2.1. Datos

Los datos proporcionados para este experimento corresponden a 5425 archivos con lineas espectrales, que contienen 201 puntos de flujos, los datos serán pre-procesados para obtener 2 conjuntos de datos para nuestro análisis, un grupo, es el grupo control que normalmente se determina como nuestros conjunto de pruebas o test y por otro lado tenemos los datos de entrenamiento, los cuales proporcionaran al modelos un aprendizaje continuo según la cantidad de épocas, filtros, activaciones, neuronas y capas densas aplicadas en este caso.

2.2. Arquitectura

Una red neuronal convolucional (CNN) es una red multicapa como dice el nombre, la salida de una capa de convolución es el resultado de operación de convolución, en lugar de la multiplicación de matrices. Normalmente, esta operación de convolución se realiza a través de un conjunto de filtros[1].Las CNN han tenido mucho éxito en tareas de reconocimiento de imágenes[2].

La arquitectura del modelo ser diferente por cada estudio realizado, es decir, que los modelos de arquitectura de una red neuronal serán realizados o determinados ad hoc. En este caso los datos de entrada serán nuestras lineas espectrales sintéticas con el objetivo de poder predecir 3 parámetros clave (Teff , log

g, y vrot). Decidimos hacer todas nuestras pruebas usando la plataforma ML TensorFlow[3] con la interfaz de keras[4].

Este modulo de Keras en el lenguaje de programación python es uno de los elementos mas populares hoy en día al momento de realizar ejecuciones que competen una red neuronal o distintos componentes de ML.

Dentro de estas practicas de programación orientada al análisis de datos Keras destaca como un modulo de alto nivel para "facilitar" la modificación de los hiper-parámetros.

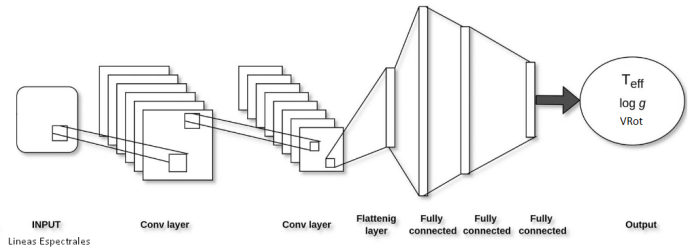

Figure 1: **Arquitectura CNN[1].** Arquitectura CNN utilizada en este trabajo. con las distintas capas de convolución y filtros de salida, donde tenemos como salida final los parámetros estelares (Teff , log g, y vrot).

2.3. Desempeño

El desempeño del modelo estará determinado por el calculo entre **MAE** [5](el error absoluto medio) y **ACC** (accuracy para determinar la precision) del entrenamiento.

El error absoluto medio (**MAE**) es la distancia vertical promedio entre cada uno de los puntos y la recta identidad (y=x), o también la distancia horizontal promedio entre cada punto y la recta identidad.

$$MAE = \frac{\sum_{i=1}^{N} |t_i - \hat{t}_i|}{N}$$

Figure 2: Formula del error absoluto medio.

La precisión (**ACC**)es una métrica para evaluar los modelos de clasificación. De manera informal, la precisión es la fracción de predicciones que nuestro modelo acertó.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

Figure 3: Formula calculo de precisión.

3. Experimentos

Los datos utilizados corresponden a espectros sintéticos de modelos de atmósferas estelares, en particular la Línea de Helio 4471 Å (Angstrom), la que está ensanchada por rotación, por lo que los modelos sintéticos cubre el siguiente rango de longitudes de onda: 4460 - 4480 Å. Los modelos están definidos por tres parámetros estelares, como se indica a continuación:

Parámetros	**Valor**
Temperatura efectiva t_{eff}	$\geq 15000[k]$
log g	$2.0[cm/s^2] \leq \log g \leq 5.0[cm/s^2]$
velocidad de rotación v_{rot}	$\geq 100[km/s^2]$

Table 1: Valores de los parámetros.

A continuación se observa las "alas" de las lineas espectrales capturadas, las cuales son producto de la alta velocidad de rotación que poseen las estrellas de tipo BE [6]

```
df_line = pd.read_csv("filtered_wv_models/f_t24000g50v370.dat", sep=" ", header=None)

# Plot
plt.figure(figsize=(8,6))
plt.plot(df_line[0].values, df_line[1].values)
plt.xlabel("wavelength [A]")
plt.ylabel("normalized flux")
plt.grid()
plt.show()
```

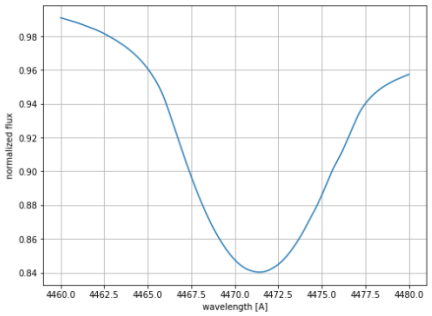

Figure 4: **Lectura de linea espectral.** Ejemplo de lectura de una líneas espectral usando el módulo pandas del lenguaje de programación Python.

Las líneas espectrales están contenidas en 5425 archivos y contienen 201 puntos de flujos. Estas lineas espectrales fueron leídas utilizando python sobre Jupyper notebook en un computador portatil con un procesador i7 de décima generación, 16GB de RAM y Tarj. Graf RTX2060.

Se leen todos los archivos de lineas espectrales sintéticas mediante el ejemplo en python que muestra la siguiente figura.

```
files = glob.glob('filtered_wv_models/*')
m = len(files)
print("Number of files in models", m)
print("5 first files: \n", files[0:5])
```

```
Number of files in models 5425
5 first files:
 ['filtered_wv_models\\f_t15000g20v100.dat', 'filtered_wv_models\\f_t15000g20v110.dat', 'filtered_wv_models\\f_t15000g20v120.dat', 'filtered_wv_models\\f_t15000g20v130.dat', 'filtered_wv_models\\f_t15000g20v140.dat']
```

Figure 5: Lectura de 5425 archivos

Luego mediante el código siguiente generaremos las matrices de X e Y.

```
print("--> X has dimension", m, " x 201")
print("--> y has diemsnion", m, " x 3")

y = np.zeros((m,3))
df = pd.read_csv(files[0], sep=" ", header=None)
x_points = df[0].values
n_points, n_columns = df.shape
display(df[1].values.shape)
print("Num. of points in x axis: ", n_points)
```

Figure 6: Genera matrices para entrenamiento en Python.

Una vez generada las matrices definimos los parámetros de nuestra red neuronal convolucional.

```
time_steps = X.shape[1]
input_dimension = 1
```

```
sample_size = X_train.shape[0]
X_train_reshaped = X_train.reshape((sample_size, time_steps, input_dimension))
sample_size = X_val.shape[0]
X_val_reshaped = X_val.reshape((sample_size, time_steps, input_dimension))
sample_size = X_test.shape[0]
X_test_reshaped = X_test.reshape((sample_size, time_steps, input_dimension))
```

Figure 7: Ejemplo código Python de parámetros de CNN.

3.1. Comparación

Se realiza 2 pruebas en un tiempo determinado de 100 épocas, donde podremos a prueba el modelo CNN con los siguientes cambios de en los hiper parámetros:

Descripción	**Capa 1**	**Capa 2**
Conv1D		
Filtro	32	16
Kernel	7	3
Activación	relu	relu
Dense		
Cant Neuronas	32	3
Activación	selu	linear

Table 2: Primera con parámetros iniciales.

Descripción	**Capa 1**	**Capa 2**	**Capa 2**
Conv1D			
Filtro	32	16	64
Kernel	7	3	15
Activación	selu	selu	selu
Dense			
Cant Neuronas	32	16	3
Activación	selu	selu	linear

Table 3: Primera con parámetros modificados.

Como se puede observar en las tablas la diferencia entre la tabla de parámetros iniciales y la de datos modificados es básicamente que se agrega una capa más tanto en la convolución como en la densidad, manteniendo la coherencia de la función de activación, en síntesis se esta agregando una capa mas tanto en la convolución como en la densidad de las neuronas.

```
Model: "model_conv1D"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Conv1D_1 (Conv1D)           (None, 195, 32)           256

 Conv1D_2 (Conv1D)           (None, 193, 16)           1552

 flatten (Flatten)           (None, 3088)              0

 Dense_1 (Dense)             (None, 32)                98848

 Dense_4 (Dense)             (None, 3)                 99

=================================================================
Total params: 100,755
Trainable params: 100,755
Non-trainable params: 0
_________________________________________________________________
```

Figure 8: Resumen de modelo con parámetros iniciales.

```
Model: "model_conv1D"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Conv1D_1 (Conv1D)           (None, 195, 32)           256

 Conv1D_2 (Conv1D)           (None, 193, 16)           1552

 Conv1D_3 (Conv1D)           (None, 179, 64)           15424

 flatten (Flatten)           (None, 11456)             0

 Dense_1 (Dense)             (None, 32)                366624

 Dense_2 (Dense)             (None, 16)                528

 Dense_4 (Dense)             (None, 3)                 51

=================================================================
Total params: 384,435
Trainable params: 384,435
Non-trainable params: 0
_________________________________________________________________
```

Figure 9: Resumen de modelo con parámetros modificados.

3.2. Entrenando la red CNN

Al ejecutar la red con los parámetros iniciales considerando una linea de tiempo de 100 épocas podemos observa que el comportamiento de perdida disminuye por cada ciclo obteniendo un un error absoluto de 253.95 y una perdida de 0.13

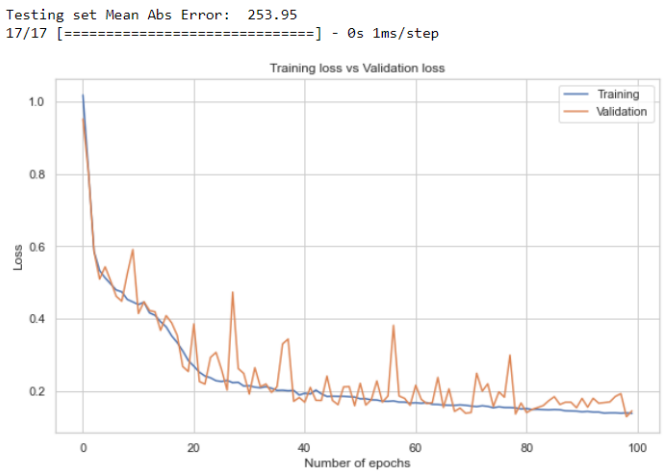

Figure 10: curva de perdida mae.

El gráfico de accuracy nos indica que el entrenamiento tiene una efectividad de cercano al 88%.

Figure 11: curva de perdida mae datos iniciales.

Al ejecutar con los parámetros modificados tenemos los siguientes resultados un erro absoluto medio de 228.04 y una perdida de 0.10 obteniendo un mejor resultado.

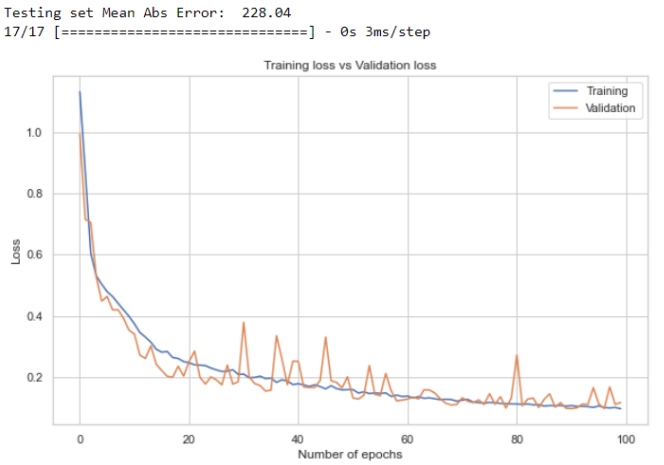

Figure 12: curva de perdida mae con datos modificados.

Por el lado de la precisión se observa que la curva tiene el mismo comportamiento que con las capas anteriores con la pequeña salvedad que tuvo una mejora en el rendimientos con un 90 %.

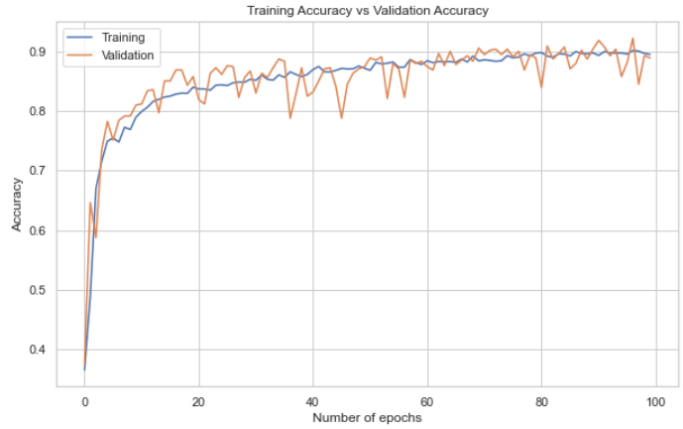

Figure 13: curva de perdida mae con datos modificados.

Descripción	Val normal	Val Modificado	Diferencia
Loss	0.1377	0.106	-0.0314
Val. Loss	0.1448	0.104	-0.0408
Mae	0.2477	0.214	-0.0333
Val. Mae	0.2352	0.205	-0.0298
Acc	0.8825	0.9	0.0175
Val. Acc	0.8821	0.898	0.0166

Table 4: Resumen de diferencias entre los distintos valores de perdida, mae, y acc en función de las modificaciones.

3.3. Utilizando los modelos entrenados

Para realizar la predicción de nuestros parámetros estelares (Teff , log g, y vrot), utilizaremos los datos de estrellas B proporcionados para este ejercicio. Utilizando las redes neuronales con valores iniciales definidos y luego con modificaciones.

Primero debemos realizar la lectura del archivo con Python.

```
df = pd.read_csv("hd127972_2014-01-31_07-43-08_final_corr.txt", sep="\t", header=None)

m1 = df[0] >= 4460
m2 = df[0] <= 4480

df2 = df[m1][m2]

df2
```

Figure 14: Carga de archivos BESOS.

	0	1
2359	4460.025826	1.005874
2360	4460.115200	1.010925
2361	4460.204573	1.014999
2362	4460.293947	1.016962
2363	4460.383320	1.025536
...	...	...
2578	4479.598608	0.994150
2579	4479.687982	0.983137
2580	4479.777355	0.994910
2581	4479.866729	1.008497
2582	4479.956102	0.998500

224 rows × 2 columns

Figure 15: Validación del contenido.

```
plt.figure(figsize=(8,6))
plt.plot(df2[0], df2[1], label="Línea observada")
#plt.plot(df4[0], df4[1], label="Modelo")
plt.grid()
plt.legend(loc="best")
plt.show()
```

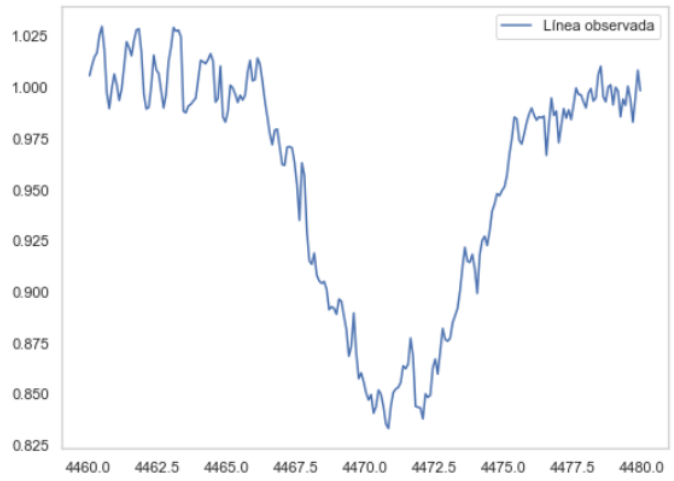

Figure 16: Linea observada de datos espectrales besos en Python.

Se ejecuta el modelo entrenado con los datos recién cargado de Besos.

```
pred_obs =model_conv1D.predict(obs_flux_reshaped)
```

```
print("t_eff:", pred_obs[0][0])
print("log g:", pred_obs[0][1])
print("v_rot:", pred_obs[0][2])
```

Figure 17: Ejecución de modelo entrenado en python y obtención de los 3 parámetros estelares que estamos prediciendo.

Descripción	Val normal	Val Modificado	Diferencia
Temp efectiva	34954.7	27237	-7717.7
log g	4.4	4.39	-0.01
Vel.Rotación	269.7	284.4	14.7

Table 5: Resumen de diferencias entre los distintos valores obtenidos en la predicción.

4. Conclusiones

En el experimento realizado en este trabajo se observo que al realizar modificaciones en los hiper-parámetros de una red convolucional se puede obtener mejoras, en la este caso en especifico las mejoras no fueron significativas al momento de entrenar, la comparación entre la precisión del entrenamiento en la red con los parámetros normales fue de un 88% con 100 épocas de entrenamiento, mientras que la red que tenia los parámetros modificados agregando 1 capas extra tanto en la convolución como en la densidad de neuronas fue de un 90% con 100 épocas de entrenamiento. Otra diferencia observada en el entrenamiento es en los valores de perdida donde los valores iniciales muestra una perdida de 0.13 con respecto al entrenamiento con valores modificados este entrego una perdida de 0.10, estos resultados tampoco son significativos. Por otro lado al momento de utilizar este modelos entrenado utilizando la lineas espectrales de besos observamos diferencias en los resultados, donde la diferencia significativa la podemos ver en la temperatura efectiva la cual tiene una variación -7717.7 respecto a los valores iniciales mientras que la gravedad es casi idéntica y la velocidad de rotación también. Esto nos demuestra que los resultados obtenidos en la fase de entrenamiento pueden ser mejorados de manera poco significativa agregando 1 capa tanto de convolución como de densidad , afectando directamente los resultados de la predicción de los parámetros Teff , log g, y vrot.

References

[1] M. Gebran, K. Connick, H. Farhat, F. Paletou, I. Bentley, Deep learning application for stellar parameters determination: I-constraining the hyperparameters, Open Astronomy 31 (1) (2022) 38–57. doi:10.1515/astro-2022-0007.
URL https://doi.org/10.1515/astro-2022-0007

[2] J. Yim, J. Ju, H. Jung, J. Kim, Image classification using convolutional neural networks with multi-stage feature, in: Advances in Intelligent Systems and Computing, Springer International Publishing, 2015, pp. 587–594. doi:10.1007/978-3-319-16841-8_52.
URL https://doi.org/10.1007/978-3-319-16841-8_52

[3] A. Ye, A deep dive into keras, in: Modern Deep Learning Design and Application Development, Springer, 2022, pp. 1–48.
[4] N. Mohammad, A. M. Muad, R. Ahmad, M. Y. P. M. Yusof, Accuracy of advanced deep learning with tensorflow and keras for classifying teeth developmental stages in digital panoramic imaging, BMC Medical Imaging 22 (1) (2022) 1–13.
[5] A. Agga, A. Abbou, M. Labbadi, Y. El Houm, I. H. O. Ali, Cnn-lstm: An efficient hybrid deep learning architecture for predicting short-term photovoltaic power production, Electric Power Systems Research 208 (2022) 107908.
[6] J. Carson, C. Thalmann, M. Janson, T. Kozakis, M. Bonnefoy, B. Biller, J. Schlieder, T. Currie, M. McElwain, M. Goto, et al., Direct imaging discovery of a "super-jupiter" around the late b-type star κ and, The Astrophysical Journal Letters 763 (2) (2013) L32.