Introducción a las Redes Neuronales y Aprendizaje Profundo
- Trabajo Final

Profesor:

- Dr. Juan Manuel Rodriguez

Integrantes:

- Cesarano, Ignacio
- Santillán Cooper, Martín

Descripción del problema

El problema que nos proponemos resolver está enmarcado dentro de la predicción del comportamiento sedentario futuro (FSBP, por sus siglas en inglés) y es el de predecir el gasto de energía (METs) que tendrá una persona en el futuro en base a datos recolectados de sensores de Smartphone. El gasto energético es medido en MET (Metabolic Equivalent of Tasks), que es la medida estándar en la comunidad científica que estudia la salud en relación con la actividad física. En el ámbito de la salud, se ha llegado a un acuerdo entre los investigadores en determinar cómo actividad física a toda aquella actividad cuyo MET asociado sea menor o igual a 1.5.

El problema de predicción a resolver es de regresión, ya que, como queremos predecir el gasto de energía, y éste es un valor continuo, debemos usar algoritmos de regresión. Más específicamente, de todos los tipos de redes neuronales vistas durante la cursada, utilizaremos Redes Neuronales Recurrentes (RNN) y Redes Neuronales Convolucionales, que son especialmente aptas para el problema que trataremos porque los datos que usamos están organizados en series de tiempo, por lo que tenemos un problema donde la secuencialita juega un papel importante. Para generar las features que serán utilizadas como entrada para diferentes Redes Neuronales, pre-procesaremos un dataset llamado StudentLife[1].

Es importante notar que el trabajo desarrollado plantea un nuevo enfoque al problema de FSBP ya que la forma de estimar el nivel de sedentarismo utilizada hasta ahora en el estado del arte no está de acuerdo a la propuesto por la comunidad científica ya que no toma en cuenta el valor de MET de cada tipo de actividad física llevada a cabo, a diferencia de nuestro trabajo, en el que si fue llevado a cabo el cálculo de MET.

En este trabajo, trataremos predecir el valor de MET para un usuario únicamente en la hora siguiente. Es decir, dados los datos de las horas *t-n, ..., t-2, t-1, t* utilizaremos NNs para hallar el gasto energético en el tiempo *t+1.* Nuestro objetivo principal es evaluar si es factible utilizar redes neuronales que puedan tratar con problemas de predicción que deban aprender de una secuencia de datos para el problema de FSBP. Además, nos interesa comparar la performance de las RNN y las CNN (mas específicamente, las TCN), con el objetivo de comprobar los resultados empíricos conseguidos en “An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling”, por S. Bai et. al. Para evaluar nuestras hipótesis diseñaremos arquitecturas con RNN, CNN normales, TCN y simple NN.

Trabajos relacionados

A continuación se realiza una revisión del trabajo relacionado en el área de FSBP. Los trabajos relacionados, es decir, aquellos que buscan resolver el problema de predecir el

[1] http://studentlife.cs.dartmouth.edu/

comportamiento sedentario a futuro, son escasos y hacen suposiciones diversas a la hora de tratar de resolverlo, por lo que la comparación entre ellos y con nuestro enfoque propuesto se torna imprecisa. Es interesante resaltar que modelos que usen algoritmos de Deep Learning no han sido publicados para tratar el problema de FSBP.

Q. He and E. Agu son los únicos dos autores que han propuesto, analizado y comparado modelos para FSBP. En todos sus trabajos, determinan el nivel de sedentarismo como el porcentaje de logs estacionarios que se tomaron de los datos de actividad física de cada usuario en cada hora, con respecto a los demás tipos de logs (corriendo, caminando, etc.).

Primero, propusieron un algoritmo de dominio de frecuencia para detectar patrones sedentarios recurrentes a partir de los datos de los usuarios. Este trabajo buscó ajustar funciones periódicas (seno, coseno) al porcentaje de actividad estacionaria de cada hora y de cada usuario, tratando así de identificar los patrones subyacentes en su actividad física.

En un segundo estudio, los mismos autores exploraron la idea de si el contexto de un usuario puede ser usado para predecir su comportamiento sedentario futuro. Usando regresión logística fueron capaces de clasificar las variables de contexto de cada usuario para predecir si iba a ser ¨muy sedentario¨ en la próxima hora, con una precisión de 73.1% y un *recall* de 87.7% (He and Agu 2016b).

Paralelamente, ellos propusieron un enfoque para descubrir automáticamente los patrones de actividad sedentaria de los usuario usando modelos auto-regresivos (He and Agu 2016c).

Finalmente, se enfocaron en detectar los ritmos predominantes de comportamiento sedentario modelando los ritmos cíclicos y ritmos lineares expuestos por la filosofía de Lefebvre (He and Agu 2017).

FSBP fue estudiado por Martín Santillán Cooper y Marcelo Armentano, investigando la posibilidad de utilizar técnicas de Deep Learning. En el trabajo se propusieron diferentes arquitecturas neuronales que utilizaban únicamente capas completamente conectadas. Dicho trabajo fue enviado a la revista International-Journal of Computer Studies y se encuentra en etapa de revisión.

Descripción del dataset

Analizamos y pre-procesamos el dataset StudentLife[2] (Wang et al. 2014) para probar la validez de nuestros modelos. Este dataset contiene datos de los sensores de los Smartphone de 48 estudiantes a lo largo de 10 semanas en la primavera de 2013.

El dataset se recogió de 30 estudiantes de pregrado y 18 graduados. De todo el grupo de estudiantes, 38 eran varones y 10 mujeres. Del grupo de estudiantes de pregrado, dos eran de primer año, 14 de segundo año, 6 de tercer año y ocho de cuarto año. También hubo 13 estudiantes de maestría de primer año y 1 de segundo año, y 3 estudiantes de doctorado. Los participantes fueron racialmente diversos, con 23 caucásicos, 23 asiáticos y 2 afroamericanos. Los datos disponibles incluyen: datos de actividad física, datos de conversación, datos de ubicación, entre otros.

[2] Link al dataset: http://studentlife.cs.dartmouth.edu/

Diferentes tipos de datos fueron registrados con diferentes frecuencias, dependiendo del tipo de sensor y la carga de trabajo global del Smartphone. Por ejemplo, la actividad física se registró cada 2–3 segundos en 1 de cada 4 minutos, el uso de aplicaciones fue se registró cada 20 minutos, y los registros de ubicación cada 10 minutos.

Pre-procesamiento del dataset

Todos los datos del dataset StudentLife poseen un *timestamp* asociado. Decidimos discretizar las series de tiempo en *buckets* de una hora ya que esta es la granularidad utilizada en la mayor parte del trabajo relacionado para la predicción de sedentarismo. Por lo tanto, todas las *features* que se generaron corresponden a una combinación de usuario / hora en particular y son algún cómputo que resume algún aspecto de los datos de algún usuario / hora particular. Por ejemplo, un *bucket* específico puede corresponder al usuario 10 y la hora 2013-04-24 19: 00–20: 00. Los *buckets* para los cuales no teníamos información sobre la actividad física del usuario fueron eliminados. Además, se eliminó la información de un usuario en particular que presentaba inconsistencias en sus registros de actividad física. En total, obtuvimos 60,819 *buckets* para los 48 usuarios.

Los registros de actividad de cada usuario se están clasificados en el dataset como *estacionario*, *caminando* o *corriendo.* A partir de estos registros se calculó el valor del gasto energético para cada *bucket.* Tal como fue explicado anteriormente, el gasto energético es comúnmente medido de términos de Metabolic Equivalent of Tasks (MET). A cada tipo de actividad se le asignó un valor estático de MET de acuerdo al Compendio de Actividades Físicas[3]. Finalmente se calculó el promedio de MET para cada *bucket.*

Dado que discretizamos todos los datos en *buckets* de una hora, tuvimos que tomar en cuenta que información iba a ser útil para predecir el valor del MET en las horas siguientes. Las *features* usadas fueron elegidas basándonos en previas investigaciones (He and Agu 2016)(Cook and Krishnan 2015), así como también otras fueron agregadas.

A continuación definimos las 20 *features* generadas pre-procesando el dataset StudentLife:

GPS features

- distanceTraveled: la distancia *haversine* recorrida calculada a partir de los registros de GPS;
- locationVariance: utilizado para medir la variabilidad en la ubicación GPS de un participante en cada *bucket*, computado como se describe en (Saeb et al. 2015).

Time features

- hourSine: transformación de seno de la hora;
- hourCosine: transformación de coseno de la hora;
- dayOfWeek: día de la semana;
- pastMinutes: el número de minutos transcurridos desde el comienzo del día; remainingMinutes: el número de minutos que quedan para terminar el día;

[3] https://sites.google.com/site/compendiumofphysicalactivities/

Physical activity features

- Stationary level: el porcentaje de instancias de actividad física clasificadas como estacionario/a' en cada *bucket;*
- Walking level: el porcentaje de instancias de actividad física clasificadas como 'caminando' en cada *bucket;*
- Running level: el porcentaje de instancias de actividad física clasificadas como de 'corriendo' en cada *bucket;*
- activityMajor: el tipo de actividad física con más instancias en cada *bucket*;

Audio features

- SilenceLevel: el porcentaje de instancias de audio clasificadas como 'silencio' en cada bucket;
- voiceLevel: el porcentaje de instancias de audio clasificadas como 'voz' en cada bucket;
- noiseLevel: el porcentaje de instancias de audio clasificadas como 'ruido' en cada bucket;
- numberOfConversations: el número de conversaciones que ese estudiante tuvo en cada bucket;

Other features

- isCharging: si el Smartphone se estaba cargando;
- isLocked: si el Smartphone estaba bloqueado;
- isInDark: si el Smartphone estaba en la oscuridad;
- hasCalendarEvent: si tenía un evento programado en el calendario;
- wifiChanges: el número de cambio de conexiones wifi en cada bucket;
- sLevel: el valor de MET para la hora actual.

Finalmente, dado que las features dayOfWeek y activityMajor son categóricas, se realizo dummy-encoded. Con esta transformación, obtenemos 7 dummy features representando las categorías para la *feature* dayOfWeek y 3 para la feature activityMajor. Como resultado de esta última transformación, finalizamos con 28 Smartphone *features* para predecir el comportamiento sedentario de los usuarios.

El siguiente paso del pre-procesamiento consistió en generar los time-lags. Para ello, a una instancia del dataset que corresponde a un tiempo *t* se le agregan las features de tiempos anteriores, dependiendo de la cantidad de time-lags que utilice la arquitectura. Este proceso aumenta la cantidad de features tantas veces como time-lags utilice la arquitectura, aumentando así el tamaño del dataset. Luego, se separa el dataset entre *X* e *y,* siendo *X* todas las *features* diferentes al *metLevel* del tiempo t (tiempo presente). Por último, se dividen *X* e *y* entre casos de entrenamiento y de testeo. Para esta última división se usó un proporción de 2 a 1 en todos los casos.

El último paso del pre-procesamiento consiste en normalizar los datos. Para este paso se utilizaron normalizadores proveídos por la librería *Scikit-learn*. Varios normalizadores fueron

tenidos en cuenta y, finalmente, se utilizaron diferentes según las características de cada usuario. Para evitar la fuga de datos, o también llamado *data lickeage,* para preparar los datos de entrenamiento de cada modelo primero realizamos una normalización de los datos de entrenamiento y, con la distribución aprendida por el normalizador, normalizamos los datos de entrenamiento. El *data lickeage* debe ser siempre tenido en cuenta ya que si se normalizan todos los datos al mismo tiempo se estaría perdiendo la posibilidad de evaluar la capacidad generalización de los predictores, ya que les estaríamos dando ya información de los datos que serán luego utilizados para testeo.

1. Arquitecturas propuestas

Las arquitecturas propuestas fueron aquellas aptas para tratar con series de tiempo. Ellas son: Redes Neuronales Recurrentes (RNN) y Redes Neuronales Convolucionales (CNN). Clásicamente, las RNN fueron la elección para cualquier problema en el que la secuencia de los datos fuera importante (por ejemplo: traducción de texto). Recientemente, ciertos tipos de CNN han comenzado a utilizarse en los problemas para los que las RNN conseguían el mejor desempeño. Más específicamente, las CNN mencionadas utilizan operaciones convolucionales de una única dimensión. Entre las CNN propuestas, se encuentran aquellas implementadas con Keras por nosotros y las Temporal Convolutional Networks (TCN)[4].

Las TCN son CNN que aplican la convolución de una dimensión, son causales, utilizan conexiones residuales y dilataciones para aumentar el campo receptivo. Las TCN demostraron superar ampliamente a las clásicas RNN basadas en GRU o LSTM en los problemas típicos como el problema de sumar o de copiar. Además, las TCN han demostrado tener más memoria que redes recurrentes de la misma capacidad. Por otro lado y no menos importante, las TCN y las CNN en general, poseen la característica de ser altamente paralelizables, en contraste con las RNN, por los que el proceso de la búsqueda óptima del valor de los hiper-parametros -proceso también llamado “tuneo”- se realiza de una forma más rápida. La implementación utilizada para las TCN puede hallarse en el siguiente link. En Fig. 1 se puede observar un ejemplo de una sencilla TCN. En dicha figura se ve claramente el carácter causal de la red, ya que nunca se toma información del futuro. La red del ejemplo posee un único bloque residual y dilaciones de 1, 2, 4 y 8.

[4] https://arxiv.org/abs/1803.01271

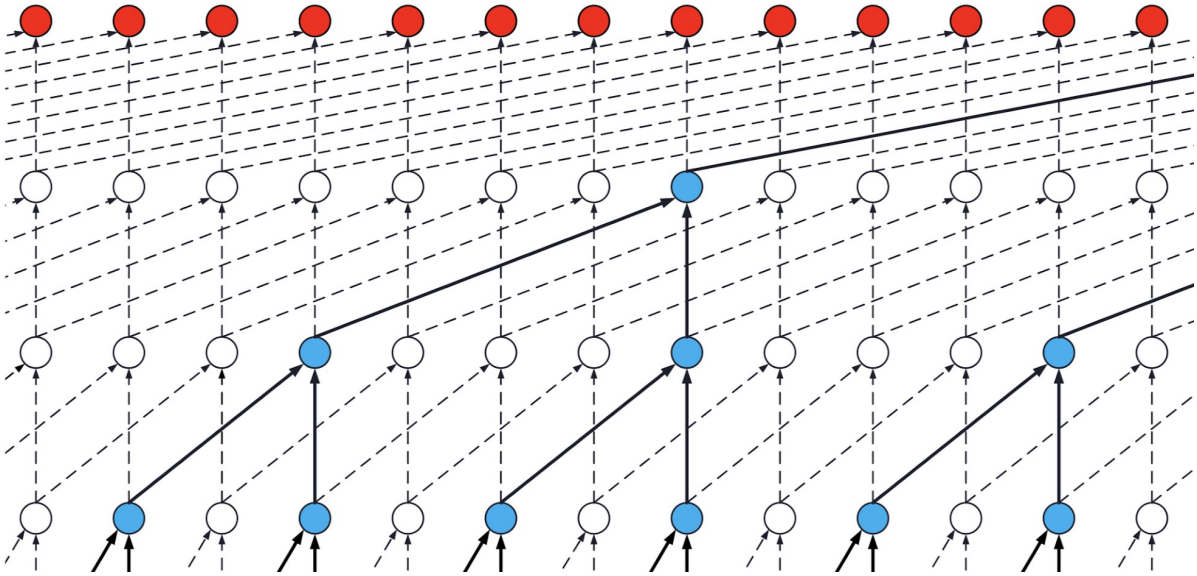

Fig. 1

Uno de los hiper-parámetros que varía en las distintas arquitecturas propuestas es la cantidad de time-lags utilizados. Los time-lags son la cantidad de información que se le da a la arquitectura sobre el pasado. Es decir, si se quiere predecir el gasto de energía de un usuario en un tiempo t, si la cantidad de time-lags es 3, el input de la red neuronal estará formado por las features del tiempo anterior (t-1), así como también las features de los tiempos t-2 y t-3.

En general, los hiper-parámetros a tunear son: tipo de capas utilizadas, cantidad de capas, cantidad de unidades por capa, técnica de regularización a utilizar, time-lags a utilizar, algoritmo de optimización y la función de pérdida. Además las CNN poseen ciertos parámetros que no poseen las demás arquitecturas, como la cantidad de filtros por capa o el tamaño del kernel. Más aun, las TCN poseen hiper-parámetros que no poseen las CNN: como la cantidad de bloques residuales o la lista de dilataciones. Estos últimos hiperparámetros se tunean de una forma menos aleatoria que los demás, ya que lo que se intenta calcular en este caso es que el campo receptivo sea igual o mayor a la cantidad de time-lags.

En resumen, las arquitecturas 1,2 y 3 son RNN, las arquitecturas 4 y 5 son CNN y la arquitectura número 6 es una NN simple. La arquitectura número 6 es una NN simple que utiliza únicamente información de la hora anterior para predecir el valor del MET de la hora siguiente y fue agregada a modo de comparación y para evaluar la utilidad de utilizar datos del pasado para FSBP. Esto quiere decir que si la arquitectura 6 supera a las RNN y CNN que utilizan información del pasado para predecir el valor de MET puede significar que, para este problema, la información del pasado no ayuda a mejorar la calidad de los predictores.

A continuación se muestra un cuadro que describe las arquitecturas propuestas. Las características más detalladas de cada arquitectura pueden ser analizadas en la función *get_model* del script *tp_dl_train.py.*

	Arq. 1	Arq. 2	Arq. 3	Arq. 4	Arq. 5	Arq. 6
Tipo de capas	LSTM - FC x 3	LSTM x 2 - FC x 1	LSTM x 3 - FC x 4	Ver *model.summary()*	Conv1D x 1 - Flatten x 1 - FC x 1	Dense x 3
Neuronas por capa	32-16-8-1	64-32-1	128-64-32-32-16-8-1	Ver *model.summary()*	32 - No aplica - 1	64 - 32 - 1
Técnica de regularización	Dropout	Dropout	Dropout	Dropout	Dropout y Batch Normalization	Dropout
Número de time-lags	8	12	12	8	4	1
Algoritmo de optimización	Adam	Adam	Adam	Adam	Adam	Adam
Función de pérdida	MSE	MSE	MSE	MSE	MSE	MSE

Cantidad de épocas	256	128	128	64	128	256
Tamaño de batch	64	64	64	64	64	128
Relación Train/Test	⅔-⅓	⅔-⅓	⅔-⅓	⅔-⅓`	⅔-⅓	⅔-⅓
Número de bloques residuales	No aplica	No aplica	No aplica	1	No aplica	No aplica
Lista de dilataciones	No aplica	No aplica	No aplica	1-2-4	No aplica	No aplica
Tamaño del kernel	No aplica	No aplica	No aplica	2	2	No aplica
Número de filtros	No aplica	No aplica	No aplica	8	32	No aplica

FC = Fully Connected

Métrica de evaluación seleccionada

La métrica utilizada para evaluar el desempeño de las redes diseñadas es el Error Cuadrático Medio o MSE (Mean Squared Error). El MSE es una medida de desempeño cuantitativa utilizada comúnmente para medir el error que hay entre el valor real y el valor estimado. En este contexto MSE es igual a la sumatoria de los errores cuadráticos. En comparación con el Error Medio Absoluto o MAE, MSE amplifica y penaliza con mayor fuerza aquellos errores de mayor magnitud. La fórmula de cálculo del MSE se muestra a continuación:

$$\mathrm{MSE} = \frac{1}{N} \times \sum_{t=1}^{N} (y_t - p_t)^2$$

Donde:

- y_t es el resultado en el tiempo t.
- p_t es el pronóstico de valor en el tiempo t.

En otras palabras, compara un valor predicho y un valor observado o conocido.

Resultados

Es interesante notar que el dataset que generamos está formado por una gran cantidad de usuarios, donde cada usuario posee su propia rutina y características personales, que no tienen porque parecerse a la de los demás usuarios. Además, algunos usuarios poseen menos datos que otros. Debido a esto, es difícil hallar una arquitectura que funcione correctamente para todos los usuarios por las idiosincrasias de cada uno de ellos. Por este motivo, decidimos realizar un proceso de búsqueda de los hiper-parámetros que conseguían los mejores resultados para ciertos usuarios. Tres usuarios fueron seleccionados para medir la performance de las arquitecturas propuestas, ya que, realizar un análisis tan extenso (48 usuarios) no era el objetivo de la cátedra. Los usuarios seleccionados fueron elegidos específicamente debido a que tenían patrones de gasto energético diferentes. Los 3 usuarios son: 50 (bajo MET), 31 (MET medio), 4 (MET alto). Además los usuarios seleccionados poseen un diferente grado rutinario en su actividad física. Lo anterior puede corroborarse con el cálculo de la correlación entre los gastos energéticos de las mismas horas de los diferentes días de la semana. En Fig. 2, Fig. 3 y Fig. 4 se muestra el porcentaje del valor de MET a lo largo de los días de la semana y las horas del día de los 3 usuarios, respectivamente.

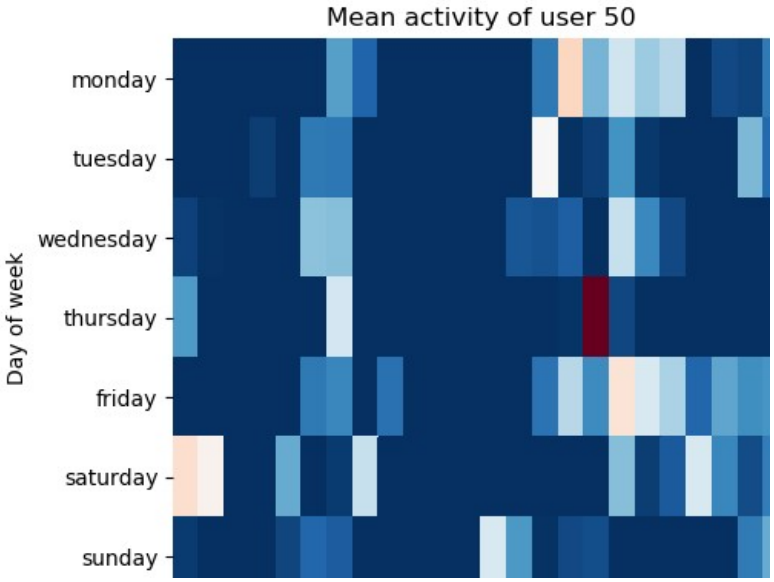
Mean activity of user 50
Day of week
monday
tuesday
wednesday
thursday
friday
saturday
sunday

Fig. 2

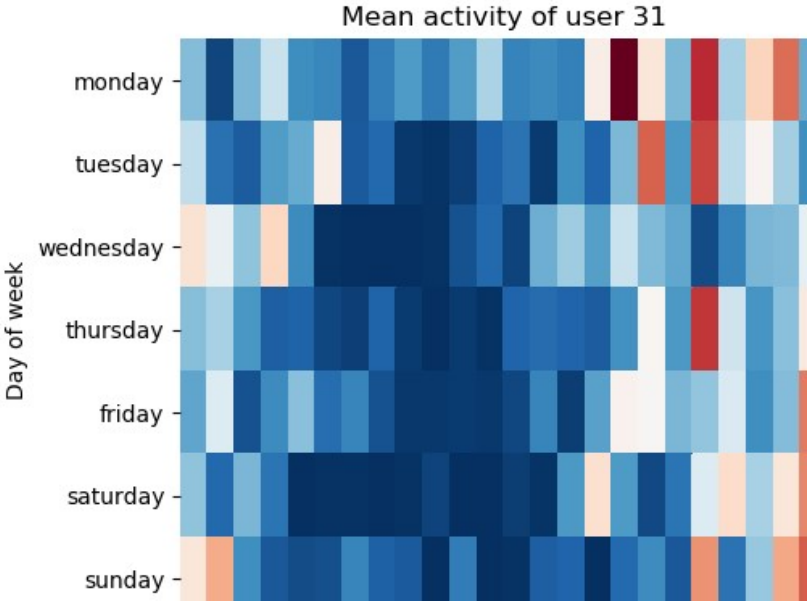
Mean activity of user 31
Day of week
monday
tuesday
wednesday
thursday
friday
saturday
sunday

Fig. 3

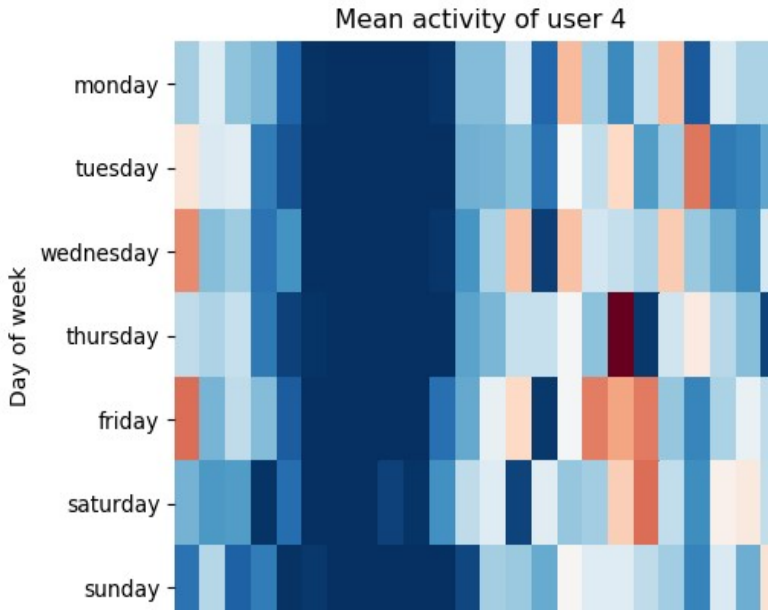

Fig. 4

A continuación, se presenta una tabla que muestra los valores de *mse* conseguidos para cada predictor. Se presenta el *mse* para entrenamiento y para el testeo. A pesar de que para evaluar la capacidad de generalización que posee un predictor debe proveerse su error prediciendo a partir de un conjunto de datos que no fue utilizado para entrenarlo, mostramos en la tabla el *mse* conseguido en el entrenamiento para poder analizar si los predictores produjeron *overfitting*.

MSE Train/ MSE Test	Arquitectura 1	Arquitectura 2	Arquitectura 3	Arquitectura 4	Arquitectura 5	Arquitectura 6
Usuario 50	0.058/0.028	0.039/.0.04	0.043/0.035	0.05/0.028	0.042/0.029	0.034/0.047
Usuario 31	0.143/0.309	0.047/0.0355	0.052/0.382	0.116/0.321	0.108/0.307	0.069/0.342
Usuario 4	0.444/0.608	0.197/1.002	0.168/0.73	0.364/0.619	0.308/0.59	0.281/0.651

Si el entrenamiento se realiza nuevamente, los resultados de cada predictor pueden variar ya que no se consiguió que los resultados sean replicables fijando semillas mediante los generadores de números pseudo-aleatorios de TensorFlow y Numpy. Esto puede deberse a que las librarías que utiliza CudaNN para que los modelos sean entrenados aprovechando la GPU introduzcan cierto grado de aleatoriedad.

Discusión

A continuación pasarán a discutirse los resultados obtenidos. Es interesante notar qué regresor tuvo un mejor desempeño para cada uno de los usuarios y analizar el porqué de dicho resultado.

El usuario más desafiante para los predictores resultó ser el número 50, ya que fue el menos rutinario y con menos casos de comportamiento no sedentario. Por lo tanto, resultó difícil que los NN aprendieran, a partir de las features del pasado, cuándo iba a tener un comportamiento no sedentario. Las arquitecturas 1 y 4 fueron las que mejor desempeño presentaron. Las Fig. 5 y 6 muestran las predicciones de entrenamiento y testeo para esta arquitectura.

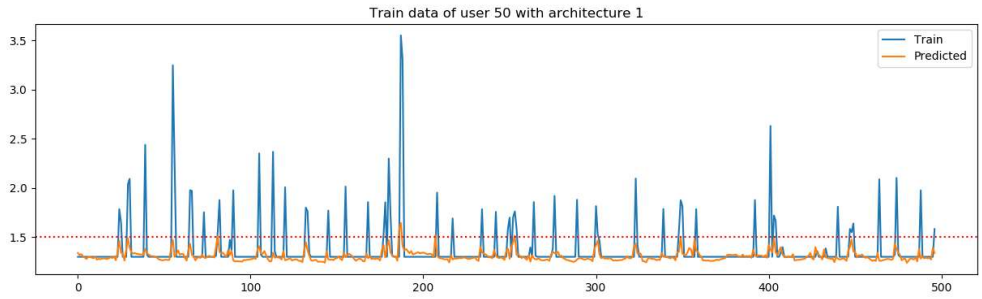

Fig. 5

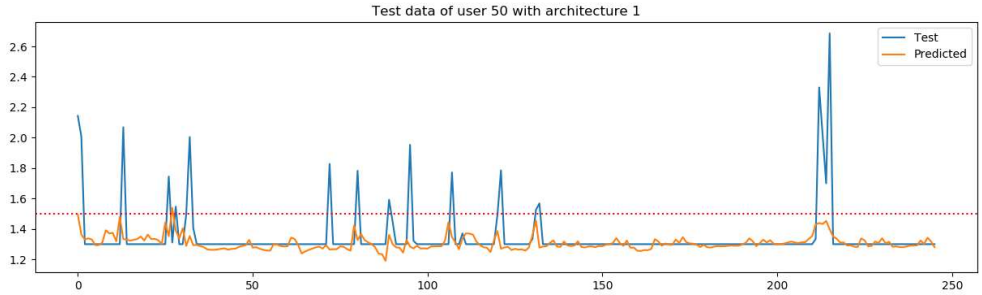

Fig. 6

En el caso del usuario 31, se puede observar un comportamiento más rutinario. Teniendo en cuenta que en el pre-procesamiento se le dio especial atención a las *features* de tiempo, es entendible que los regresores se desempeñen mejor en estos casos. La arquitectura 5 (CNN) fue la que mejor performance tuvo en este caso, con un *mse* de testeo de 0.307. Las Fig. 7 y 8 muestran las predicciones de entrenamiento y testeo para esta arquitectura.

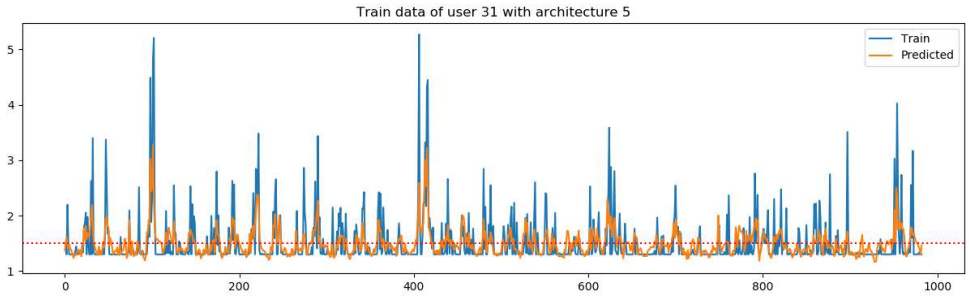

Fig. 7

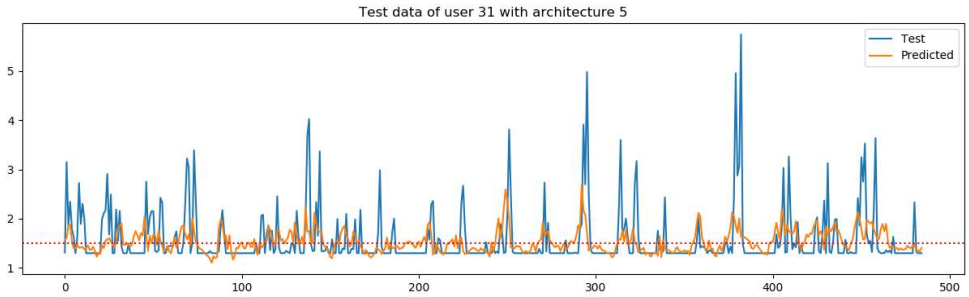

Fig. 8

Por último, los modelos sobre el usuario 4 fueron los que mejor desempeño tuvieron, a pesar de tener un mayor mse que los modelos sobre los otros usuarios. Esto se debe a que este usuario posee muchos mas casos de comportamiento no sedentario, por lo que para que el regresor se desempeñe bien debe aprender los patrones de comportamiento del usuario a partir de las features provistas. En el caso del usuario 50, el predictor podría solo aprender a predecir un valor cercano a 1.3 METs para obtener un bajo *mse.* La arquitectura que mejor se desempeñó en este caso fue la número 4 (TCN). Esto, y el hecho de que para todos los modelos las CNN tuvieron, en general, una mejor performance que las RNN, confirma nuestra hipótesis sobre que las CNN, y en particular las TCN, son más adecuadas para problemas de predicción que deban aprender de una secuencia de datos. Las Fig. 9 y 10 muestran las predicciones de entrenamiento y testeo para esta arquitectura.

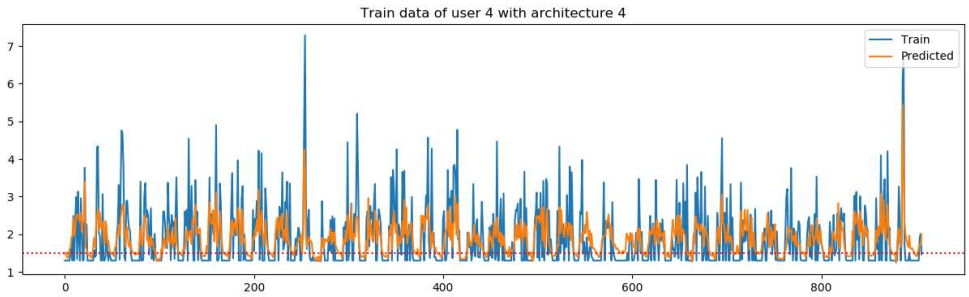

Fig. 9

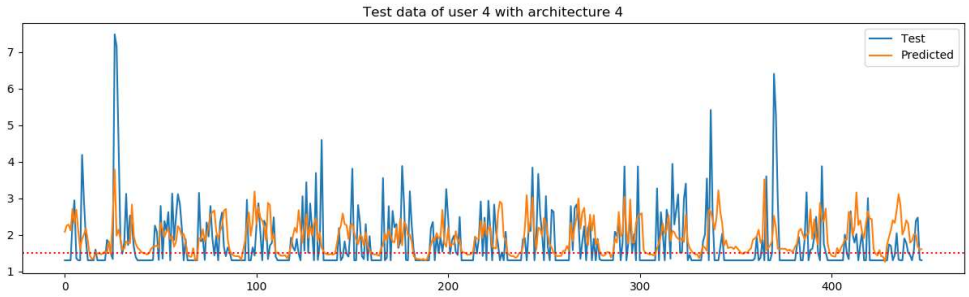

Fig. 10

Conclusiones

En este trabajo nos propusimos comparar diferentes tipos de redes neuronales para evaluar su performance en el problema de la predicción del comportamiento sedentario. Este trabajo es el primero en evaluar el rendimiento de redes neuronales que intenten sacar provecho al carácter secuencial del problema tratado. Se presentaron usuarios con diferentes niveles de gasto energético, lo que hace que muy difícil diseñar una arquitectura que tenga una buena performance para todos ellos.

A pesar de haber utilizado un alto grado de regularización, las NN más extensas son propensas a producir overfitting. En consecuencia, tendrán un mal rendimiento en la etapa de *testing*. Este es un *trade-off* que fue considerado fue el nivel de la complejidad de las redes. Al momento de diseñar cada red, se buscó que tuvieran, cada una, una performance aceptable para cada uno de los usuarios seleccionados. Por lo tanto, es importante que la complejidad de cada red sea la menor posible para que posean la capacidad necesaria de generalización.

Por otra parte, como solo se toman los datos de un usuario en particular para entrenar las redes, la cantidad de datos es muy baja y no permite llegar a entrenar apropiadamente a redes más complejas y con mayor cantidad de parámetros. Este fue un *trade-off* al momento de diseñar las arquitecturas propuestas y podría llegar a ser resuelto aumentando la granularidad en el pre-procesamiento y tomar, por ejemplo, *buckets* de tiempo de 30 minutos o 15 minutos, resultando así en más casos de entrenamiento.

Otro de los trade-off que tuvimos que enfrentar fue la cantidad de time-lags a tomar. Es lógico pensar que cuanta más información del pasado se le dé a la arquitectura, más información tendrá y así podrá hacer mejores predicciones. Esto no es cierto por dos razones. La primera de ellas es que el MET que tendrá un usuario en el tiempo *t+1* puede no estar relacionado con las features que poseemos del mismo usuario en el tiempo *t-n,* siento n un entero arbitrario. Esto, al tener pocos casos de entrenamiento, puede generar ruido en el modelo. La segunda razón es que, como en muchos casos no se poseen datos sobre todas las horas que duró el experimento, cuantos más time-lags se tengan, mayor es la probabilidad de que un caso de entrenamiento sea descartado. Por ejemplo, si la cantidad de time-lags es *n* y si se tienen los datos del usuario *x* desde el tiempo *t* al *t-n+1* pero no del tiempo *t-n,* el caso de entrenamiento se desecha.

Un punto importante a destacar es que, como los estudiantes mantuvieron, por lo general, un comportamiento totalmente sedentario, los clasificadores que intentaban siempre

predecir valores cercanos a cero MET fueron descartados, ya que conseguían un MSE bajo pero no eran útiles. Este problema fue tratado en el trabajo realizado por M. Santillán Cooper y M. Armentano con las técnicas de *over-sampling* y asignación de peso a las clases.

Con respecto a las hipótesis planteadas al comienzo del informe, todas ellas pudieron ser evaluadas y comprobadas. Se lograron diseñar arquitecturas con un mejor desempeño que aquellas arquitecturas que no utilizan información del pasado. Además, se comprobó que las CNN pueden llegar a desempeñarse de una mejor manera que las RNN.

Bibliografía

- Cook, D. J., & Krishnan, N. C. (2015). Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data. John Wiley & Sons.
- He, Q., & Agu, E. O. (2016a). Smartphone usage contexts and sensable patterns as predictors of future sedentary behaviors. In 2016 IEEE Healthcare Innovation Point-Of-Care Technologies Conference (HI-POCT). https://doi.org/10.1109/hic.2016.7797695
- He, Q., & Agu, E. O. (2016b). Towards sedentary lifestyle prevention: An autoregressive model for predicting sedentary behaviors. In 2016 10th International Symposium on Medical Information and Communication Technology (ISMICT). https://doi.org/10.1109/ismict.2016.7498879
- He, Q., & Agu, E. O. (2017). A Rhythm Analysis-Based Model to Predict Sedentary Behaviors. In 2017 IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE). https://doi.org/10.1109/chase.2017.122
- Saeb, S., Zhang, M., Karr, C. J., Schueller, S. M., Corden, M. E., Kording, K. P., & Mohr, D. C. (2015). Mobile Phone Sensor Correlates of Depressive Symptom Severity in Daily-Life Behavior: An Exploratory Study. Journal of Medical Internet Research, 17(7), e175.
- Wang, R., Chen, F., Chen, Z., Li, T., Harari, G., Tignor, S., … Campbell, A. T. (2014). StudentLife. In Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing - UbiComp ’14 Adjunct. https://doi.org/10.1145/2632048.2632054