4.3 Instrucciones previas

Descarga e instalación DashAI usando docker

Como se tarda algunos minutos, te pedimos que dejes corriendo la instalación mientras avanzas con los siguientes pasos. Primero, descarga DashAI desde aqui. Luego, dentro del directorio “DashAI-software” ejecuta “docker compose up”. Esto instalará y levantará DashAI en la dirección “*localhost:8000”.* Prosigue con el siguiente paso mientras se instala.

Cómo funciona DashAI

A modo de preparación para las tareas que tendrás que desarrollar a continuación, te presentamos DashAI y te explicamos con más detalle su funcionamiento.

Qué es DashAI

DashAI es un software de código abierto que permite entrenar modelos de aprendizaje automático (ML) mediante una interfaz gráfica interactiva, cubriendo desde la carga de datos hasta la evaluación de resultados.

El software está diseñado para poder integrar y ser compatible con diversas bibliotecas científicas de ML populares, aprovechando así los recursos existentes en el ecosistema de IA. Dicho de otra forma, DashAI apunta a agrupar distintas herramientas como scikit-learn, pytorch, tensorflow o huggingface en un solo software y que la gente pueda utilizarlas para realizar experimentos y entrenamientos de manera sencilla a través de su interfaz gráfica.

Arquitectura

El sistema de extensiones de DashAI se basa en una arquitectura de plugins. Esto consta de, por un lado, un ***core*** que contiene todo lo necesario para orquestar las etapas del flujo de ML y, por otro lado, existen módulos instalables llamados ***plugins*** que empaquetan distintos componentes previamente definidos que se pueden extender: modelos, tareas, dataloaders y métricas de evaluación.

Para que veas cómo funciona DashAI y la instalación de plugins, ve el siguiente video: https://youtu.be/4G0CrX2D8yg

Se espera que personas con un conocimiento avanzado en ML puedan crear *plugins* que extiendan el software y permitan que usuarios finales puedan entrenar distintos modelos de ML, realizando experimentos, configurando sus parámetros y evaluando su desempeño. Para que el proceso de crear una extensión sea sencillo, se diseñaron los mecanismos necesarios para que los plugins se integren fácilmente al sistema y que se puedan desplegar correctamente en la interfaz gráfica: incluso para habilitar la configuración de hiperparámetros desde la interfaz gráfica solamente se debe declarar en el plugin mismo, sin necesidad de hacer modificaciones en el frontend del software.

Componentes y el pipeline

Un componente es una **clase de python que pertenece a uno de los posibles tipos: modelo, tarea, dataloader o métrica de evaluación,** y que cumplen con implementar ciertos métodos (y sus respectivas firmas) y declarar atributos propios de cada uno de los tipos de componente. Los distintos componentes son usados por el software para articular y permitir al usuario completar el flujo de ML desde la carga de datos hasta la evaluación de resultados.

El pipeline de ML supervisado que sigue el software es el siguiente: primero, se carga un dataset (e.g. archivo CSV) mediante un componente de tipo **dataloader** y se guarda como un Dataset de la librería Huggingface[1], que es el formato de dataset estándar que DashAI utiliza de manera interna. Luego, el usuario configura su experimento y realiza el entrenamiento (llamada al método *fit)* de una o más instancias de las clases de tipo **model** con los parámetros ingresados por el usuario. Posteriomente, sus predicciones (llamadas al método *predict*) son evaluadas usando las métricas disponibles para la tarea, definidas en los componentes de tipo **metric**. Tanto el método *fit* como *predict* reciben su data respectiva en el formato dataset de Huggingface[2] guardado anteriormente.

Tanto componentes de tipo **model** y **metric,** necesitan señalar explícitamente los nombres de la o las *task* a las que están asociados dichos componentes[3]. De esta manera, DashAI sabe qué componentes están disponibles para la tarea que el usuario seleccionó.

Por último, aquellos componentes que son configurables por el usuario mediante la interfaz gráfica son denominados “Objetos Configurables” y deben heredar de la clase ConfigObject, así como también deben tener asociado un esquema de *pydantic* que declare el tipo de dato que espera y permita su despliegue y visualización en la interfaz gráfica como un campo que recibe un input por parte del usuario.

Plugins

Un plugin de DashAI corresponde a un **paquete de python que contiene uno o más componentes extensibles** y que es reconocible por el software mediante el uso de *entrypoints*[4]. Esto implica que el empaquetamiento de los componentes en un plugin es totalmente libre: puede contener todos los componentes necesarios para una tarea en particular o un plugin puede ser simplemente un modelo específico o un dataloader para algún tipo de dataset determinado.

Los plugins están pensados para ser compartidos mediante el repositorio de paquetes de python (*pypi*) y a la vez, seleccionables e instalables desde la propia interfaz gráfica de DashAI.

---

[1] https://huggingface.co/docs/datasets/index
[2] Notar que parte de lo que el modelo debe implementar, es adaptar este tipo de datos a un formato compatible con la librería que el modelo esté utilizando.
[3] Para ello deben contener el atributo “COMPATIBLE_COMPONENTS”.
[4] Un entrypoint es un tipo de metadata que puede ser expuesta por los paquetes de python en su instalación. Resultan útiles si un programa busca la personalización a través de *plugins*.

4.4 Tareas

Imagina que trabajas como ingeniera o ingeniero en ML en una empresa que usa DashAI para realizar experimentos y entrenar modelos, debido a que el resto de tus colegas no son expertos programadores. Notas que constantemente se deben entrenar modelos sencillos de ML para clasificar imágenes sobre sets de datos muy específicos y variados, pero que DashAI aún no contiene el *plugin* para clasificación de imágenes usando redes neuronales.

Por lo tanto te propones extender el software para que tus compañeros de trabajo puedan entrenar este tipo de modelos y realizar experimentos para poder dar con una configuración del modelo que cumpla con el desempeño deseado.

Descarga los archivos necesarios para esta prueba desde aquí. A continuación se detalla cada archivo y carpeta:

- **dashai_plugin_template:** template para creación del plugin. Contiene plantillas para las clases de los diferentes tipos de componentes.
- **Reduced MNIST Data.zip:** Dataset de prueba.
- **MLP_pytorch_model.ipynb**: Notebook en Colab que contiene la implementación de un modelo de clasificación de imágenes usando redes neuronales.
- **.pypirc:** Archivo con token para subir plugin a pypi.
- **CómoExtenderDashAI.pdf:** Archivo con instrucciones de cómo extender DashAI.

En este momento, DashAI debería estar corriendo en la dirección "*localhost:8000"*. Ábrelo y ejecuta las siguientes tareas:

Tarea 1. Instala un plugin

En la pestaña plugins, instala el plugin de clasificación de imágenes. Si no aparece, refresca la lista y espera unos segundos.

Tarea 2. Carga un dataset

Carga el dataset "Reduced MNIST Data.zip" en el módulo datasets.

Tarea 3. Crea un experimento

Crea un experimento que permita resolver una tarea de clasificación de imágenes. Trata de llegar lo más lejos posible. Comenta en voz alta el paso en que no puedas seguir.

Crear plugin con modelo MLP Classifier

Como te habrás dado cuenta, el plugin de clasificación de imágenes que instalaste define la tarea dentro del software y permite cargar un dataset de imágenes (es decir, contiene los componentes task y dataloader para clasificación de imágenes), pero no contiene modelos que entrenar. Es por esto que ahora crearás un plugin que contenga tan solo un componente: el modelo de clasificación de imágenes usando un Multi Layer Perceptron (MLP) para la tarea de clasificación de imágenes. Este debe ser compatible con DashAI y sus parámetros se deben poder configurar mediante su interfaz gráfica.

En el notebook llamado **MLP_pytorch_model.ipynb,** encontrarás una implementación del clasificador **que ya adaptaste para cumplir con la interfaz de los modelos de DashAI**. Revísalo y comprueba que funciona. Puedes entrenar el modelo con el dataset "Reduced MNIST Data.zip**"**.

El documento auxiliar llamado CómoExtenderDashAI.pdf es el que contiene las instrucciones necesarias para completar el proceso de crear y subir un plugin de manera exitosa. **Te pedimos que lo abras y tengas a mano**.

Tarea 4. Estructura del plugin y nombres

La carpeta "*dashai_plugin_template"*, te servirá como plantilla para poder crear tu *plugin*. Además, siguiendo la parte "a) Estructura de un paquete de python" del documento auxiliar, comprueba la correctitud, y modifica si es necesario, el template para adaptarlo al plugin que crearemos.

Por favor, incluye en el nombre del paquete algo que lo identifique como tuyo.

Tarea 5. Crear clase del componente

En base al punto del documento auxiliar "b) Clase del componente", usando el notebook y el template entregado, crea la clase para el modelo de clasificación de imágenes usando MLP de pytorch en el plugin.

Tarea 6. Crear esquema

En base al punto "c) Esquemas de pydantic" del documento auxiliar, crea la clase que definirá el esquema para los parámetros configurables "epochs", "learning_rate" y "hidden_dims", de tipo int, float y list respectivamente. La lista de hidden_dims corresponden a una lista (elementos separados por comas) donde sus componentes son de tipo "int" mayores que 0.

Tarea 7. Archivo de configuración

En base al apartado del documento auxiliar "4) Archivo de configuración (*pyproject.toml*)", rellena el archivo de configuración con lo necesario para poder instalar el plugin y disponibilizarlo en pypi.

Tarea 8. Subir paquete a pypi

Sigue las instrucciones del punto “II. Subir un plugin a pypi” del documento auxiliar para poder cargar el plugin que creaste a pypi. Pero para que no tengas que crear una cuenta y generar el token, te damos el archivo *.pypirc* con el token ya generado. Cópialo en la carpeta $HOME ejecutando en la terminal “*cp .pypirc $HOME*”.

Tarea 9. Instalar plugin con modelo

Instala el plugin que creaste en el módulo “Plugins”. Si no aparece, prueba apretando el botón “Refresh” y espera unos segundos.

Tarea 10. Completa el experimento de la Tarea 2

Comprueba que puedes modificar los parámetros que añadiste en el esquema y que cumplan con las restricciones