

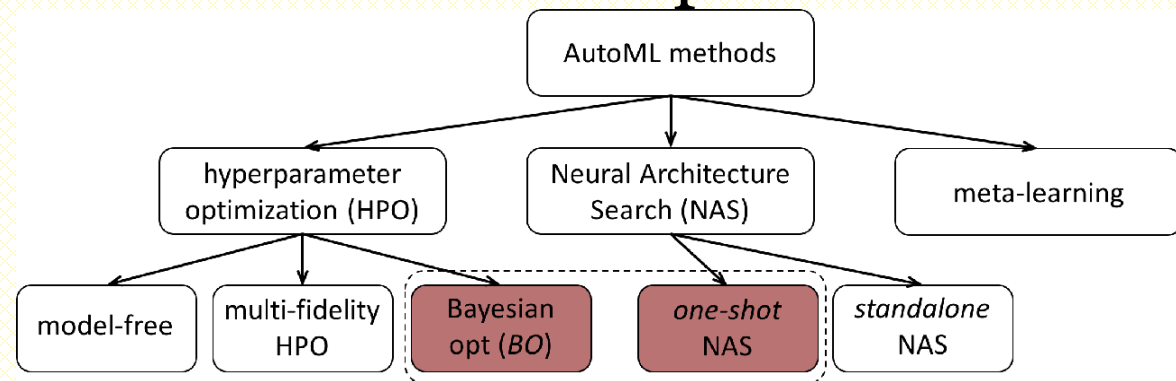
# AutoPytorch

Diego Andrade Canosa

Roberto López Castro

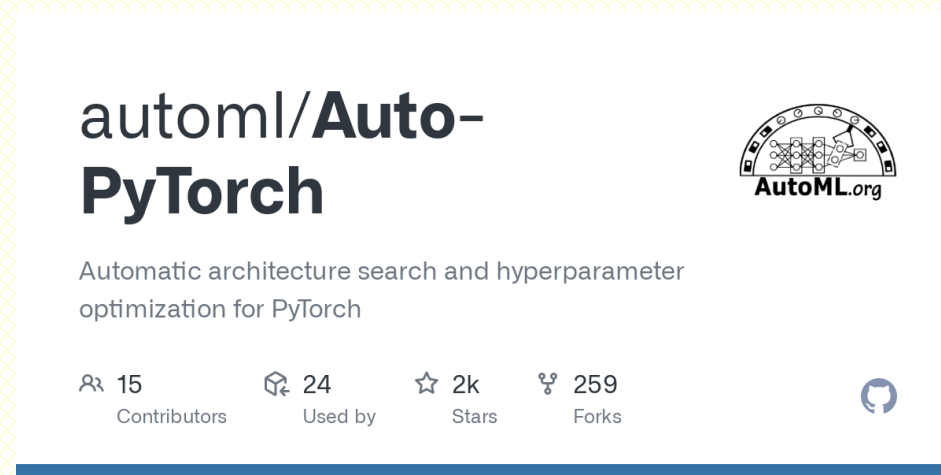
# Introducción a AutoPyTorch

- AutoPyTorch es una biblioteca de ML que ha ganado popularidad en el campo de la automatización del aprendizaje automático (AutoML).
- Utiliza técnicas de búsqueda automática para encontrar el mejor modelo para un conjunto de datos específico.
- Su objetivo es reducir el tiempo y esfuerzo necesarios para encontrar el modelo óptimo, al tiempo que mejora el rendimiento de los modelos mediante la optimización de hiperparámetros.



# Introducción a AutoPyTorch

- AutoPyTorch se basa en la exitosa biblioteca AutoML, AutoSklearn, y proporciona una interfaz intuitiva y flexible que se adapta a diferentes tipos de datos y problemas de aprendizaje automático.
- En esta presentación, exploraremos las características, ventajas y casos de uso de AutoPyTorch, así como su arquitectura y su implementación práctica.



# Propiedades de AutoPyTorch

- Automatización del proceso de selección y ajuste de modelos: AutoPyTorch busca automáticamente el mejor modelo y los hiperparámetros óptimos para un conjunto de datos dado, eliminando la necesidad de realizar ajustes manuales tediosos.
- Ahorro de tiempo y esfuerzo: Al automatizar tareas repetitivas y complejas, AutoPyTorch reduce significativamente el tiempo y esfuerzo necesarios para desarrollar modelos de aprendizaje automático.
- Mejora del rendimiento del modelo: Al explorar eficientemente diferentes modelos y ajustar los hiperparámetros, AutoPyTorch puede mejorar el rendimiento del modelo y su capacidad para generalizar a nuevos datos.

# Propiedades de AutoPyTorch

- Adaptable a diferentes tipos de datos y problemas: AutoPyTorch es versátil y puede adaptarse a una amplia gama de tipos de datos y problemas de aprendizaje automático, incluyendo clasificación, regresión y más.
- Interfaz intuitiva y fácil de usar: Con una interfaz sencilla y amigable, AutoPyTorch permite a los usuarios trabajar de manera eficiente sin requerir un profundo conocimiento técnico en algoritmos de aprendizaje automático.
- Flexibilidad y personalización: AutoPyTorch ofrece opciones de configuración y ajuste para adaptarse a necesidades específicas, permitiendo a los usuarios personalizar y controlar aspectos clave del proceso automatizado.

- La API está inspirada en auto-sklearn y solo requiere unos pocos inputs para ajustar un pipeline de DL en un conjunto de datos dado:
- ```
>>> autoPyTorch = AutoNetClassification("tiny_cs",  
max_runtime=300, min_budget=30, max_budget=90)
```
- ```
>>> autoPyTorch.fit(X_train, y_train, validation_split=0.3)
```
- ```
>>> y_pred = autoPyTorch.predict(X_test)
```
- ```
>>> print("Puntuación de precisión:",  
sklearn.metrics.accuracy_score(y_test, y_pred))
```

# Componentes de AutoPyTorch

- AutoPyTorch se basa en una arquitectura robusta y eficiente que permite la búsqueda automática de modelos y hiperparámetros óptimos.
- La arquitectura de AutoPyTorch consta de varios componentes clave:
  - Búsqueda automática: Utiliza algoritmos de búsqueda automática para explorar el espacio de búsqueda de modelos y hiperparámetros.
  - Selección automática de modelos: Identifica y selecciona el modelo más prometedor para el conjunto de datos a partir de la exploración automática.
  - Optimización automática de hiperparámetros: Ajusta automáticamente los hiperparámetros del modelo seleccionado para mejorar su rendimiento.
  - Evaluación y comparación automática de modelos: Evalúa y compara automáticamente los modelos generados para seleccionar el mejor modelo final.
- Estos componentes trabajan en conjunto para brindar una solución integral y automatizada en la búsqueda del mejor modelo de aprendizaje automático.



# NAS y HPO


- Extensión de NAS
  - Juntar NAS y Optimización de Hyperparámetros (HPO)
  - Neural Ensemble Search

Frank Hutter:

- "The training hyperparameters are much important than the architectures"
- "Hyperparameter optimization is the hidden champion of AutoML"



# Auto-PyTorch: NAS+HPO y Multi-Fidelity Meta-Learning



The diagram illustrates the Auto-PyTorch workflow. It starts with a database icon on the left, which points via a blue arrow to a large box. Inside this box, there is a sub-box labeled 'Meta-level learning & optimization' which points to a 'PyTorch' box. A feedback loop arrow returns from the 'PyTorch' box to the 'Meta-level learning & optimization' box. A red banner in the top right corner of the diagram area says 'Find me on GitHub'.

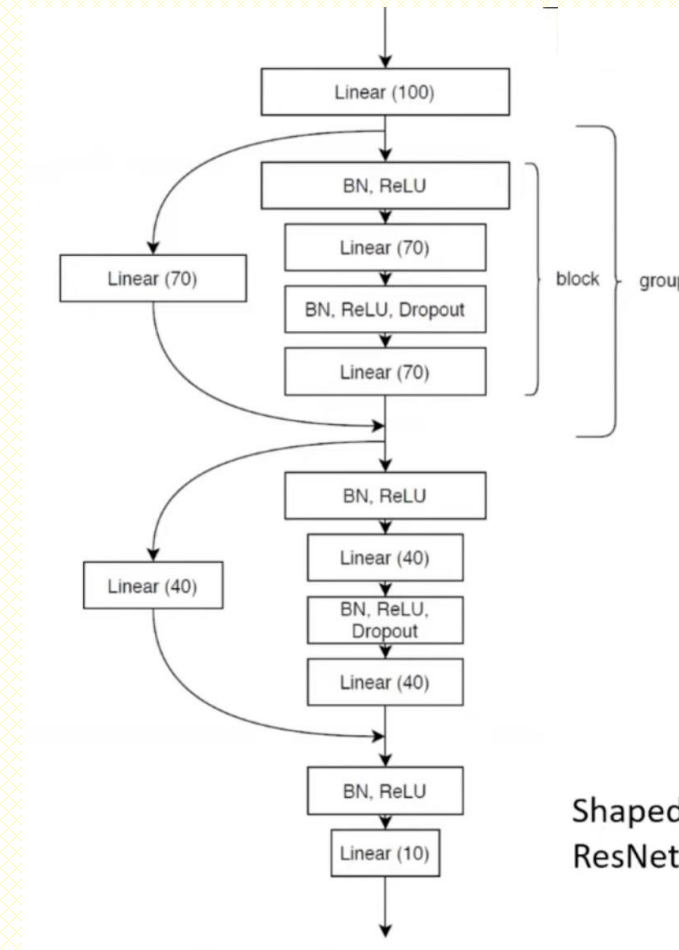
<https://github.com/automl/Auto-PyTorch>

- Joint Architecture & Hyperparameter C
- **Multi-Fidelity Optimization**
- **Meta-Learning to Warmstart Across Datasets**

Watch 38 Star 1.2k Fork 142

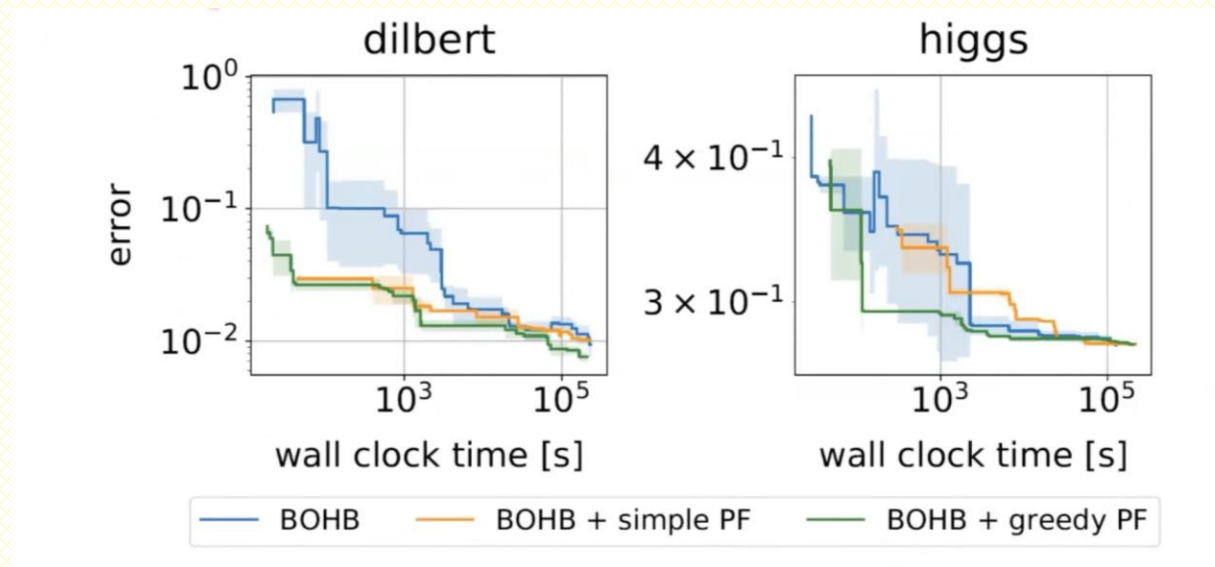
# Auto-PyTorch Tabular: Ejemplo espacio de búsqueda

	Choice	Range	log	type	cond.
Architecture	network type	[ResNet, MLPNet]	-	cat	-
	num layers (MLP)	[1, 6]	-	int	✓
	max units (MLP)	[64, 1024]	✓	int	✓
	max dropout (MLP)	[0, 1]	-	float	✓
	num groups (Res)	[1, 5]	-	int	✓
	blocks per group (Res)	[1, 3]	-	int	✓
	max units (Res)	[32, 512]	✓	int	✓
	use dropout (Res)	[F, T]	-	bool	✓
	use shake drop	[F, T]	-	bool	✓
	use shake shake	[F, T]	-	bool	✓
	max dropout (Res)	[0, 1]	-	float	✓
	max shake drop (Res)	[0, 1]	-	float	✓
Hyper-parameters	batch size	[16, 512]	✓	int	-
	optimizer	[SGD, Adam]	-	cat	-
	learning rate (SGD)	[1e-4, 1e-1]	✓	float	✓
	L2 reg. (SGD)	[1e-5, 1e-1]	-	float	✓
	momentum	[0.1, 0.999]	-	float	✓
	learning rate (Adam)	[1e-4, 1e-1]	✓	float	✓
	L2 reg. (Adam)	[1e-5, 1e-1]	-	float	✓
	training technique	[standard, mixup]	-	cat	-
	mixup alpha	[0, 1]	-	float	✓
	preprocessor	[none, trunc. SVD]	-	cat	-
	SVD target dim	[10, 256]	-	int	✓



# Auto-PyTorch Tabular: Meta-Learning

- Inicialización del espacio de búsqueda con arquitecturas complementarias + configuración de hiperparámetros
  - Meta-learning a través de 100 datasets para encontrar buenas inicializaciones
  - Hasta 100 veces más rápido



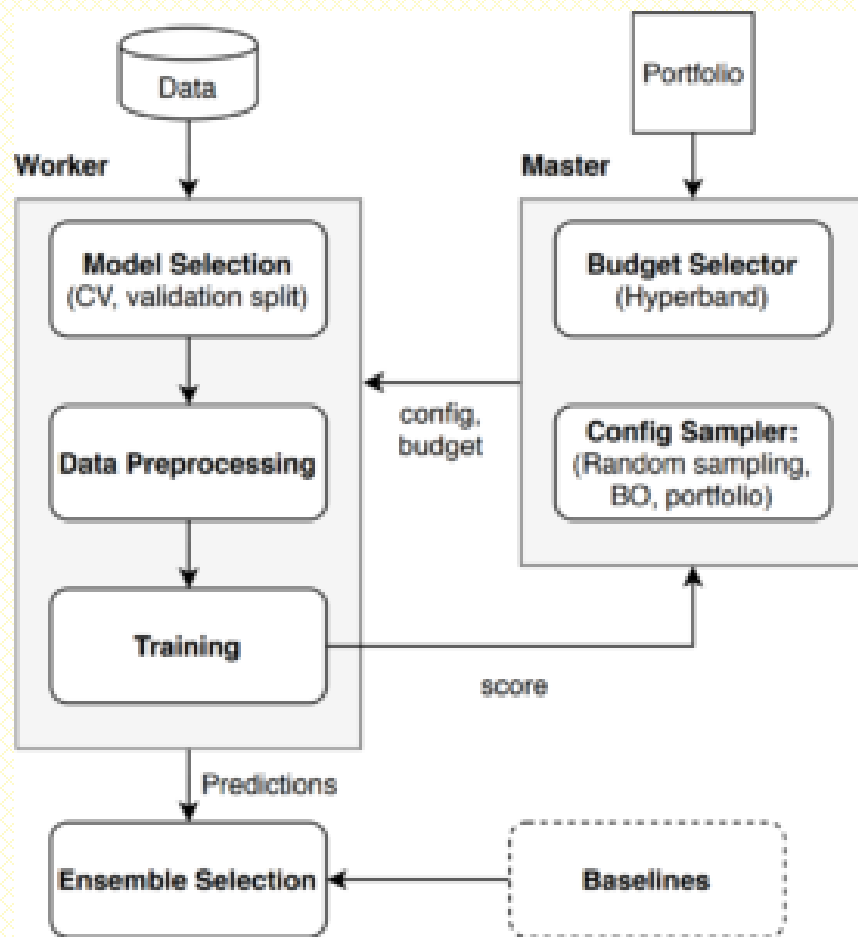
# Auto-Pytorch Tabular vs otros frameworks de AutoML

	Auto-PyTorch	AutoKeras	Auto-Sklearn	hyperopt-sklearn
covertypes	<b>96.86 <math>\pm</math> 0.41</b>	61.61 $\pm$ 3.52	-	-
volkert	<b>79.46 <math>\pm</math> 0.43</b>	44.25 $\pm$ 2.38	67.32 $\pm$ 0.46	-
higgs	<b>73.01 <math>\pm</math> 0.09</b>	71.25 $\pm$ 0.29	72.03 $\pm$ 0.33	-
car	<b>99.22 <math>\pm</math> 0.02</b>	93.39 $\pm$ 2.82	98.42 $\pm$ 0.62	98.95 $\pm$ 0.96
mfeat-factors	<b>99.10 <math>\pm</math> 0.18</b>	97.73 $\pm$ 0.23	98.64 $\pm$ 0.39	97.88 $\pm$ 38.48
apsfailure	99.32 $\pm$ 0.01	-	<b>99.43 <math>\pm</math> 0.04</b>	-
phoneme	<b>90.59 <math>\pm</math> 0.13</b>	86.76 $\pm$ 0.12	89.26 $\pm$ 0.14	89.79 $\pm$ 4.54
dibert	<b>99.04 <math>\pm</math> 0.15</b>	96.51 $\pm$ 0.62	98.14 $\pm$ 0.47	-

# Motivación

- Mientras que los primeros frameworks de AutoML se enfocaron en optimizar los pipelines de ML tradicionales y sus hiperparámetros, otra tendencia en AutoML es centrarse en la búsqueda de arquitecturas neuronales. Para combinar lo mejor de estos dos mundos, Auto-PyTorch optimiza conjunta y robustamente la arquitectura de la red y los hiperparámetros de entrenamiento para permitir el aprendizaje profundo completamente automatizado (AutoDL). Auto-PyTorch logró un rendimiento de vanguardia en varios benchmarks tabulares al combinar la optimización multi-fidelidad con la construcción de portafolios para el inicio rápido y la combinación de redes neuronales profundas (DNNs) y baselines comunes para datos tabulares.

# Arquitectura



En la figura, los datos son proporcionados por el usuario y el Portafolio es un conjunto de configuraciones de redes neuronales que funcionan bien en diversos conjuntos de datos. La versión actual solo admite el portafolio descrito en el artículo "Auto-PyTorch Tabular: Multi-Fidelity MetaLearning for Efficient and Robust AutoDL". Este portafolio se utiliza para iniciar rápidamente la optimización de SMAC. En otras palabras, se evalúa el portafolio en los datos proporcionados como configuraciones iniciales.

# API

La API inicia los siguientes procedimientos:

- Validar los datos de entrada: Procesar cada tipo de dato, por ejemplo, codificar datos categóricos para que puedan ser manejados por Auto-PyTorch.
- Crear conjunto de datos: Crear un conjunto de datos que pueda ser manejado por esta API, con la opción de divisiones de validación cruzada o retención (holdout).
- Evaluar baselines:
  - a. Conjunto de datos tabulares \*1: Entrenar cada algoritmo en el conjunto predefinido con una configuración fija de hiperparámetros y un modelo dummy de sklearn.dummy que representa el peor rendimiento posible.
  - b. Conjunto de datos de pronóstico de series de tiempo: Entrenar un predictor dummy que repite el último valor observado en cada serie.
- \*1: Los baselines son un conjunto predefinido de algoritmos de aprendizaje automático, como LightGBM y máquinas de soporte vectorial, utilizados para resolver tareas de regresión o clasificación en el conjunto de datos proporcionado.



# API

## 4. Búsqueda por SMAC:

- a. Determinar el presupuesto y las reglas de corte mediante Hyperband.
- b. Muestrear una configuración de hiperparámetros del pipeline \*2 mediante SMAC.
- c. Actualizar las observaciones con los resultados obtenidos.
- d. Repetir los pasos a. - c. hasta que se agote el presupuesto.

## 5. Construir el mejor conjunto para el conjunto de datos proporcionado a partir de las observaciones y la selección del modelo del conjunto.

\*2: Una configuración de hiperparámetros del pipeline especifica la elección de componentes, por ejemplo, el algoritmo objetivo y la estructura de las redes neuronales, en cada paso y sus correspondientes hiperparámetros.



# Ejemplo

```
"""
=====
Tabular Classification with n parallel jobs
=====

The following example shows how to fit a sample classification model parallelly on 2 cores
with AutoPyTorch

"""
import os
import tempfile as tmp
import warnings

os.environ['JOBLIB_TEMP_FOLDER'] = tmp.gettempdir()
os.environ['OMP_NUM_THREADS'] = '1'
os.environ['OPENBLAS_NUM_THREADS'] = '1'
os.environ['MKL_NUM_THREADS'] = '1'

warnings.simplefilter(action='ignore', category=UserWarning)
warnings.simplefilter(action='ignore', category=FutureWarning)

import sklearn.datasets
import sklearn.model_selection

from autoPyTorch.api.tabular_classification import TabularClassificationTask
```

# Ejemplo

```
if __name__ == '__main__':  
    #####  
    # Data Loading  
    # =====  
    X, y = sklearn.datasets.fetch_openml(data_id=40981, return_X_y=True, as_frame=True)  
    X_train, X_test, y_train, y_test = sklearn.model_selection.train_test_split(  
        X,  
        y,  
        random_state=1,  
    )  
  
    #####  
    # Build and fit a classifier  
    # =====  
    api = TabularClassificationTask(  
        n_jobs=2,  
        seed=42,  
    )
```

# Ejemplo

```
#####  
# Search for an ensemble of machine learning algorithms  
# =====  
api.search(  
    X_train=X_train,  
    y_train=y_train,  
    X_test=X_test.copy(),  
    y_test=y_test.copy(),  
    optimize_metric='accuracy',  
    total_walltime_limit=300,  
    func_eval_time_limit_secs=50,  
    # Each one of the 2 jobs is allocated 3GB  
    memory_limit=3072,  
)  
  
#####  
# Print the final ensemble performance  
# =====  
y_pred = api.predict(X_test)  
score = api.score(y_pred, y_test)  
print(score)  
# Print the final ensemble built by AutoPyTorch  
print(api.sprint_statistics())
```

# Casos de uso de AutoPyTorch

- AutoPyTorch puede aplicarse con éxito en una amplia variedad de casos de uso en el campo del aprendizaje automático.
- Algunos ejemplos de casos de uso comunes incluyen:
  1. Clasificación de imágenes: AutoPyTorch puede ser utilizado para entrenar modelos de clasificación de imágenes automatizando el proceso de selección del modelo y optimización de hiperparámetros.
  2. Regresión: Para problemas de regresión, AutoPyTorch puede ayudar a encontrar el modelo y los hiperparámetros óptimos para ajustar los datos y predecir valores continuos.

# Casos de uso de AutoPyTorch

3. Aprendizaje automático no supervisado: AutoPyTorch también puede ser utilizado en problemas de aprendizaje automático no supervisado, como clustering y reducción de dimensionalidad.

- Estos son solo algunos ejemplos, pero AutoPyTorch es flexible y adaptable, lo que significa que puede aplicarse a una amplia gama de problemas y tipos de datos.
- Con AutoPyTorch, los investigadores y profesionales del aprendizaje automático pueden obtener resultados precisos y eficientes en diferentes dominios y aplicaciones.

# Lab: Ejemplos

- <https://github.com/automl/Auto-PyTorch/tree/master/examples>