PyTorch Lightning

1 Imports básicos

```python
import lightning.pytorch as pl
```

2 Definir el modelo

```python
class MyModel(pl.LightningModule):
    def __init__(self):
        super().__init__()
        # Definir capas y funcion de pérdida

    def training_step(self, batch, batch_idx)
        :
        # Calcular salida, pérdida y loguear
            métricas
        # self.log(...)
        # return loss

    def validation_step(self, batch,
        batch_idx):
        # Calcular métricas de validación
        # self.log(...)

    def predict_step(self, batch, batch_idx):
        # Calcular predicciones
        # return pred

    def configure_optimizers(self):
        # Devolver optimizador
        # return optimizer
```

3 Carga de datos

```python
class MyDataModule(pl.LightningDataModule):
    def __init__(self):
        super().__init__()

    def prepare_data(self):
        # Descargar datos

    def setup(self, stage=None):
        # Procesar datos segun la etapa (fit/
            predict)

    def train_dataloader(self):
        return DataLoader(...)

    def val_dataloader(self):
        return DataLoader(...)

    def test_dataloader(self):
        return DataLoader(...)
```

4 Entrenamiento / Predicción

```python
model = MyModel()
trainer = pl.Trainer(max_epochs=...,
    accelerator="auto")
dm = MyDataModule()
trainer.fit(model, datamodule=dm)
preds = trainer.predict(model, datamodule=dm)
```

OPTUNA

1 Imports básicos

```python
import optuna
from optuna.pruners import HyperbandPruner
from optuna.trial import TrialState
```

2 Función objetivo

```python
def objective(trial):
    # Pedir a valores de hiperparámetros
    # Entrenar el modelo y evaluarlo
    # return métrica a optimizar
```

3 Funciones para muestreo de hiperparámetros

```python
# Ejemplos de funciones de muestreo
trial.suggest_float("...", 0.001, 0.1)
trial.suggest_int("...", 1, 5)
trial.suggest_categorical("...", ["...", "...
    "])
```

4 Podando ejecuciones poco prometedoras

```python
# Llamar esta función en cada época
def epoch_callback(loss, epoch):
    trial.report(loss, epoch)
    if trial.should_prune():
        raise optuna.exceptions.TrialPruned()
```

5 Crear y ejecutar el estudio

```python
pruner = HyperbandPruner()
study = optuna.create_study(
    direction="minimize",
    study_name="...",
    storage="sqlite:///
        busqueda_hiperparametros.db",
    load_if_exists=True,
    pruner=pruner,
)
study.optimize(objective, n_trials=10)
```