# Class Train

Class Train

* Načtení modelu
* Načtení tokenizeru
* Načtení konfigurátoru
* Zvolení typu trenování přes terminál, dle toho se pak načte dataset potřebný pro trenování a následně se spustí trening
* Spuštění funkce set\_dataset

1. \_load\_datasets

* Jako input obdržíme cestu k adresáři a vybereme všechny soubory typu csv
* V terminálu vypíšeme všechny datasety pokud je jich více jak 1
* Po výběru se data z daného csv načtou a vytvoří array, kde hlavička csv bude klíč, pro daný řadek

1. Set\_dataset

* Loading datasetu jako prostou array
* Rozdělení na train a validation dataset
* Náhodné vytažení dat podle velikosti batch
* v terminalu vypsa otázku jestli provest augmentaci
* Vymyslet augmentační kofigurace
* Function augment\_dataset

1. augment\_dataset

- Po každém trenování se data uloži do souboru model.pth

* Po doběhnutí tréninkového cyklu vyskočí možnost pokračovat v dalším kole
* Případně změnit trenovací mod a tím padem i načtení nového datasetu

1. train\_model

- Po každém trenování se data uloži do souboru model.pth

* Po doběhnutí tréninkového cyklu vyskočí možnost pokračovat v dalším kole
* Případně změnit trenovací mod a tím padem i načtení nového datasetu
* Před každým trénováním naskočí v terminálu trenovací konfigurace a zepta se jestli něco nechci změnit

1. fine\_tune\_model

- Po každém trenování se data uloži do souboru model.pth

* Po doběhnutí tréninkového cyklu vyskočí možnost pokračovat v dalším kole
* Případně změnit trenovací mod a tím padem i načtení nového datasetu
* Před každým trénováním naskočí v terminálu trenovací konfigurace a zepta se jestli něco nechci změnit

1. Reinforce\_model

- Po každém trenování se data uloži do souboru model.pth

* Po doběhnutí tréninkového cyklu vyskočí možnost pokračovat v dalším kole
* Případně změnit trenovací mod a tím padem i načtení nového datasetu
* Před každým trénováním naskočí v terminálu trenovací konfigurace a zepta se jestli něco nechci změnit