

Malakh Trainer - technická dokumentácia

Generované programom Doxygen 1.10.0



<b>1 Register hierarchie tried</b>	<b>1</b>
1.1 Hierarchia tried	1
<b>2 Register tried</b>	<b>3</b>
2.1 Zoznam tried	3
<b>3 Register súborov</b>	<b>5</b>
3.1 Zoznam súborov	5
<b>4 Dokumentácia tried</b>	<b>7</b>
4.1 Dokumentácia triedy model.CNN	7
4.1.1 Detailný popis	7
4.1.2 Dokumentácia konštruktoru a deštruktoru	8
4.1.2.1 __init__()	8
4.1.3 Dokumentácia k metódam	8
4.1.3.1 forward()	8
4.2 Dokumentácia triedy datamodule.DataModuleBitboards	8
4.2.1 Detailný popis	9
4.2.2 Dokumentácia konštruktoru a deštruktoru	9
4.2.2.1 __init__()	9
4.3 Dokumentácia triedy datamodule.DataModuleImages	9
4.3.1 Detailný popis	10
4.3.2 Dokumentácia konštruktoru a deštruktoru	10
4.3.2.1 __init__()	10
4.4 Dokumentácia triedy trainer.EarlyStopper	10
4.4.1 Detailný popis	11
4.4.2 Dokumentácia konštruktoru a deštruktoru	11
4.4.2.1 __init__()	11
4.4.3 Dokumentácia k metódam	11
4.4.3.1 early_stop()	11
4.5 Dokumentácia triedy model.MLP	12
4.5.1 Detailný popis	12
4.5.2 Dokumentácia konštruktoru a deštruktoru	12
4.5.2.1 __init__()	12
4.5.3 Dokumentácia k metódam	13
4.5.3.1 forward()	13
4.6 Dokumentácia triedy dataset.MvsFSBitboardDataset	13
4.6.1 Detailný popis	14
4.6.2 Dokumentácia konštruktoru a deštruktoru	14
4.6.2.1 __init__()	14
4.6.3 Dokumentácia k metódam	14
4.6.3.1 __getitem__()	14
4.6.3.2 __len__()	14

4.7 Dokumentácia triedy dataset.MvsFSImageDataset . . . . .	15
4.7.1 Detailný popis . . . . .	15
4.7.2 Dokumentácia konštruktoru a deštruktoru . . . . .	15
4.7.2.1 __init__() . . . . .	15
4.7.3 Dokumentácia k metódam . . . . .	16
4.7.3.1 __getitem__() . . . . .	16
4.7.3.2 __len__() . . . . .	16
4.8 Dokumentácia triedy trainer.Trainer . . . . .	16
4.8.1 Detailný popis . . . . .	17
4.8.2 Dokumentácia konštruktoru a deštruktoru . . . . .	18
4.8.2.1 __init__() . . . . .	18
4.8.3 Dokumentácia k metódam . . . . .	18
4.8.3.1 epoch() . . . . .	18
4.8.3.2 fit() . . . . .	19
4.8.3.3 load_checkpoint() . . . . .	19
4.8.3.4 save_checkpoint() . . . . .	19
4.8.3.5 save_model() . . . . .	19
4.8.3.6 save_plot() . . . . .	20
4.8.3.7 test_epoch() . . . . .	20
4.8.3.8 train_epoch() . . . . .	20
4.8.3.9 val_epoch() . . . . .	20
<b>5 Dokumentácia súborov</b>	<b>23</b>
5.1 Dokumentácia súboru datamodule.py . . . . .	23
5.1.1 Detailný popis . . . . .	23
5.2 Dokumentácia súboru dataset.py . . . . .	23
5.2.1 Detailný popis . . . . .	24
5.3 Dokumentácia súboru experiment.py . . . . .	24
5.3.1 Detailný popis . . . . .	24
5.3.2 Dokumentácia funkcií . . . . .	24
5.3.2.1 cnn_experiment() . . . . .	24
5.3.2.2 mlp_experiment() . . . . .	25
5.4 Dokumentácia súboru model.py . . . . .	25
5.4.1 Detailný popis . . . . .	25
5.5 Dokumentácia súboru trainer.py . . . . .	26
5.5.1 Detailný popis . . . . .	26
5.5.2 Dokumentácia funkcií . . . . .	26
5.5.2.1 decide_device() . . . . .	26
<b>Register</b>	<b>27</b>

# Kapitola 1

## Register hierarchie tried

### 1.1 Hierarchia tried

Tu nájdete zoznam, vyjadrujúci vzťah dedičnosti tried. Je zoradený približne (ale nie úplne) podľa abecedy:

datamodule.DataModuleBitboards . . . . .	8
datamodule.DataModuleImages . . . . .	9
trainer.EarlyStopper . . . . .	10
nn.Module	
model.CNN . . . . .	7
model.MLP . . . . .	12
trainer.Trainer . . . . .	16
Dataset	
dataset.MvsFSBitboardDataset . . . . .	13
dataset.MvsFSImageDataset . . . . .	15



## Kapitola 2

# Register tried

### 2.1 Zoznam tried

Nasledujúci zoznam obsahuje predovšetkým identifikáciu tried, ale nachádzajú sa tu i ďalšie netriviálne prvky, ako sú štruktúry (struct), uniony (union) a rozhrania (interface). V zozname sú uvedené ich stručné popisy:

<a href="#">model.CNN</a>	
Architektúra konvolučnej neurónovej siete . . . . .	7
<a href="#">datamodule.DataModuleBitboards</a>	
Dátový modul pre bitboardovú formu MvsFS datasetu . . . . .	8
<a href="#">datamodule.DataModuleImages</a>	
Dátový modul pre obrázkovú formu MvsFS datasetu . . . . .	9
<a href="#">trainer.EarlyStopper</a>	
Trieda zodpovedná za predčasné zastavenie tréningu . . . . .	10
<a href="#">model.MLP</a>	
Architektúra plne prepojenej neurónovej siete . . . . .	12
<a href="#">dataset.MvsFSBitboardDataset</a>	
Datset MvsFS v bitboardovej forme pre MLP architektúru . . . . .	13
<a href="#">dataset.MvsFSImageDataset</a>	
Datset MvsFS v obrázkovej forme pre CNN architektúru . . . . .	15
<a href="#">trainer.Trainer</a>	
Trieda zodpovedná za tréning neurónových sietí . . . . .	16





## Kapitola 3

# Register súborov

### 3.1 Zoznam súborov

Tu nájdete zoznam všetkých dokumentovaných súborov so stručnými popismi:

<a href="#">datamodule.py</a>	Dátové moduly majú na starosť rozdelenie vstupných dát na trénovacie, validačné a testovacie dáta . . . . .	23
<a href="#">dataset.py</a>	Definícia vstupného datasetu MvsFS . . . . .	23
<a href="#">experiment.py</a>	Definícia konfigurovateľných experimentov . . . . .	24
<a href="#">model.py</a>	Definícia architektúr neurónových sietí . . . . .	25
<a href="#">trainer.py</a>	Definícia trénovania neurónovej siete . . . . .	26



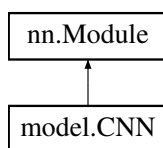
## Kapitola 4

# Dokumentácia tried

### 4.1 Dokumentácia triedy model.CNN

Architektúra konvolučnej neurónovej siete.

Diagram dedičnosti pre triedu model.CNN



#### Verejné metódy

- `__init__` (self, conv\_layers, conv\_norm, conv\_activ, fc\_layers, fc\_activ)  
*Konštruktor konvolučnej neurónovej siete.*
- `forward` (self, x)  
*Predná propagácia cez neurónovú sieť.*

#### Verejné atribúty

- `conv`
- `fc_input`
- `fc`
- `output`

#### 4.1.1 Detailný popis

Architektúra konvolučnej neurónovej siete.

## 4.1.2 Dokumentácia konštruktoru a deštruktoru

### 4.1.2.1 `__init__()`

```
model.CNN.__init__ (
    self,
    conv_layers,
    conv_norm,
    conv_activ,
    fc_layers,
    fc_activ )
```

Konštruktor konvolučnej neurónovej siete.

#### Parametre

<code>conv_layers</code>	Počet konvolučných vrstiev.
<code>conv_activ</code>	Aktivačná vrstva konvolučných vrstiev.
<code>fc_layers</code>	Počet plne prepojených vrstiev.
<code>fc_activ</code>	Aktivačná vrstva plne prepojených vrstiev.

## 4.1.3 Dokumentácia k metódam

### 4.1.3.1 `forward()`

```
model.CNN.forward (
    self,
    x )
```

Predná propagácia cez neurónovú sieť.

#### Parametre

<code>x</code>	Vstup na ktorom vykonávame predikciu.
----------------	---------------------------------------

#### Návratová hodnota

Predikovaná hodnota.

Dokumentácia pre túto triedu bola generovaná z nasledujúceho súboru:

- [model.py](#)

## 4.2 Dokumentácia triedy `datamodule.DataModuleBitboards`

Dátový modul pre bitboardovú formu MvsFS datasetu.

### Verejné metódy

- `__init__` (self, filename, ratio, batch\_size)  
*Konštruktor dátového module bitboardovej formy MvsFS datasetu.*

### Verejné atribúty

- `ratio`
- `train_dataset`
- `val_dataset`
- `test_dataset`
- `batch_size`
- `train_loader`
- `val_loader`
- `test_loader`

## 4.2.1 Detailný popis

Dátový modul pre bitboardovú formu MvsFS datasetu.

## 4.2.2 Dokumentácia konštruktoru a deštruktoru

### 4.2.2.1 `__init__()`

```
datamodule.DataModuleBitboards.__init__ (
    self,
    filename,
    ratio,
    batch_size )
```

Konštruktor dátového module bitboardovej formy MvsFS datasetu.

#### Parametre

<code>filename</code>	Názov vstupného csv súboru.
<code>ratio</code>	Pomer medzi trénovacími a validačnými dátami (validačné a testovacie dáta majú pomer 0.5).
<code>batch_size</code>	Veľkosť dávky pri načítavaní dát.

Dokumentácia pre túto triedu bola generovaná z nasledujúceho súboru:

- [datamodule.py](#)

## 4.3 Dokumentácia triedy datamodule.DataModuleImages

Dátový modul pre obrázkovú formu MvsFS datasetu.

## Verejné metódy

- `__init__` (self, filename, ratio, batch\_size)  
*Konštruktor dátového module obrázkovej formy MvsFS datasetu.*

## Verejné atribúty

- `ratio`
- `train_dataset`
- `val_dataset`
- `test_dataset`
- `batch_size`
- `train_loader`
- `val_loader`
- `test_loader`

### 4.3.1 Detailný popis

Dátový modul pre obrázkovú formu MvsFS datasetu.

### 4.3.2 Dokumentácia konštruktoru a deštruktoru

#### 4.3.2.1 `__init__()`

```
datamodule.DataModuleImages.__init__ (
    self,
    filename,
    ratio,
    batch_size )
```

Konštruktor dátového module obrázkovej formy MvsFS datasetu.

#### Parametre

<code>filename</code>	Názov vstupného csv súboru.
<code>ratio</code>	Pomer medzi tréningovými a validačnými dátami (validačné a testovacie dáta majú pomer 0.5).
<code>batch_size</code>	Veľkosť dávky pri načítavaní dát.

Dokumentácia pre túto triedu bola generovaná z nasledujúceho súboru:

- [datamodule.py](#)

## 4.4 Dokumentácia triedy `trainer.EarlyStopper`

Trieda zodpovedná za predčasné zastavenie tréningovania.

### Verejné metódy

- `__init__` (self, patience=1, min\_delta=0)
- `early_stop` (self, val\_loss)

*V tejto funkcii sa kontroluje, či aktuálna validačná strata je lepšia ako najlepší možný výsledok.*

### Verejné atribúty

- `patience`
- `min_delta`
- `counter`
- `min_val_loss`

## 4.4.1 Detailný popis

Trieda zodpovedná za predčasné zastavenie tréningu.

## 4.4.2 Dokumentácia konštruktoru a deštruktoru

### 4.4.2.1 `__init__()`

```
trainer.EarlyStopper.__init__ (
    self,
    patience = 1,
    min_delta = 0 )
```

Konštruktor predčasného ukončovateľ a tréningu.

@param patience: Počet epoch, ktoré je možné vykonať po sebe bez vylepšenia výsledkov tréningu.

@param min\_delta: Minimálny rozdiel medzi najlepším výsledkom a aktuálnym výsledkom potrebný aby sa epocha považovala za vylepšenie.

## 4.4.3 Dokumentácia k metódam

### 4.4.3.1 `early_stop()`

```
trainer.EarlyStopper.early_stop (
    self,
    val_loss )
```

V tejto funkcii sa kontroluje, či aktuálna validačná strata je lepšia ako najlepší možný výsledok.

Ak áno, tak sa počítadlo trpezlivosti rešartuje na 0. Ak nie a chyby je vyššia od najlepšieho výsledku o min\_delta hodnotu, tak sa inkrementuje počítadlo trpezlivosti. Ak počítadlo trpezlivosti dosiahlo maximálnu hodnotu, tak sa ukončí tréning.

#### Parametre

<code>val_loss</code>	Validačnú stratu aktuálnej epochy.
-----------------------	------------------------------------

**Návratová hodnota**

True ak sa tréovanie má predčasne ukončiť, False ak tréovanie má pokračovať.

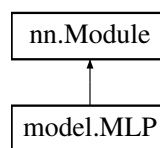
Dokumentácia pre túto triedu bola generovaná z nasledujúceho súboru:

- [trainer.py](#)

## 4.5 Dokumentácia triedy model.MLP

Architektúra plne prepojenej neurónovej siete.

Diagram dedičnosti pre triedu model.MLP

**Verejné metódy**

- `__init__` (self, input\_features, hidden\_features, output\_features, layers, activ)  
*Konštruktor plne prepojenej neurónovej siete.*
- `forward` (self, x)  
*Predná propagácia cez neurónovú sieť.*

**Verejné atribúty**

- `input`
- `hidden`
- `output`

### 4.5.1 Detailný popis

Architektúra plne prepojenej neurónovej siete.

### 4.5.2 Dokumentácia konštruktoru a deštruktoru

#### 4.5.2.1 \_\_init\_\_()

```
model.MLP.__init__(  
    self,  
    input_features,  
    hidden_features,  
    output_features,  
    layers,  
    activ )
```

Konštruktor plne prepojenej neurónovej siete.



## Parametre

<i>input_features</i>	Počet neurónov vstupnej vrstvy neurónovej siete.
<i>hidden_features</i>	Počet neurónov skrytej vrstvy neurónovej siete.
<i>output_features</i>	Počet neurónov výstupnej vrstvy neurónovej siete.
<i>layers</i>	Počet skrytých vrstiev.
<i>activ</i>	Aktivačná funkcia neurónovej siete.

## 4.5.3 Dokumentácia k metódam

## 4.5.3.1 forward()

```
model.MLP.forward (
    self,
    x )
```

Predná propagácia cez neurónovú sieť.

## Parametre

<i>x</i>	Vstup na ktorom vykonávame predikciu.
----------	---------------------------------------

## Návratová hodnota

Predikovaná hodnota.

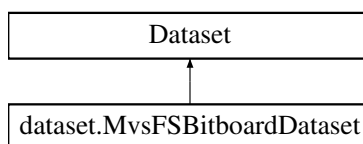
Dokumentácia pre túto triedu bola generovaná z nasledujúceho súboru:

- [model.py](#)

## 4.6 Dokumentácia triedy dataset.MvsFSBitboardDataset

Dataset MvsFS v bitboardovej forme pre MLP architektúru.

Diagram dedičnosti pre triedu dataset.MvsFSBitboardDataset



## Verejné metódy

- `__init__` (self, data)  
*Konštruktor bitboardovej formy MvsFS datasetu.*
- `__len__` (self)  
*Interná funkcia na výpočet veľkosti datasetu.*
- `__getitem__` (self, idx)  
*Interná funkcia na výber položky datasetu.*

## Verejné atribúty

- `data`

### 4.6.1 Detailný popis

Dataset MvsFS v bitboardovej forme pre MLP architektúru.

### 4.6.2 Dokumentácia konštruktoru a deštruktoru

#### 4.6.2.1 `__init__()`

```
dataset.MvsFSBitboardDataset.__init__ (
    self,
    data )
```

Konštruktor bitboardovej formy MvsFS datasetu.

#### Parametre

<code>data</code>	Dáta MvsFS datasetu.
-------------------	----------------------

### 4.6.3 Dokumentácia k metódam

#### 4.6.3.1 `__getitem__()`

```
dataset.MvsFSBitboardDataset.__getitem__ (
    self,
    idx )
```

Interná funkcia na výber položky datasetu.

#### Parametre

<code>idx</code>	Identifikátor položky.
------------------	------------------------

#### Návratová hodnota

Vstupné vlastnosti a ich label.

#### 4.6.3.2 `__len__()`

```
dataset.MvsFSBitboardDataset.__len__ (
    self )
```

Interná funkcia na výpočet veľkosti datasetu.

Návratová hodnota

Veľkosť datasetu

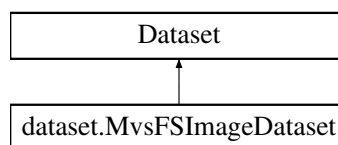
Dokumentácia pre túto triedu bola generovaná z nasledujúceho súboru:

- [dataset.py](#)

## 4.7 Dokumentácia triedy dataset.MvsFSImageDataset

Dataset MvsFS v obrázkovej forme pre CNN architektúru-.

Diagram dedičnosti pre triedu dataset.MvsFSImageDataset



### Verejné metódy

- `__init__` (self, data)  
*Konštruktor obrázkovej formy MvsFS datasetu.*
- `__len__` (self)  
*Interná funkcia na výpočet veľkosti datasetu.*
- `__getitem__` (self, idx)  
*Interná funkcia na výber položky datasetu.*

### Verejné atribúty

- `data`

### 4.7.1 Detailný popis

Dataset MvsFS v obrázkovej forme pre CNN architektúru-.

### 4.7.2 Dokumentácia konštruktoru a deštruktoru

#### 4.7.2.1 `__init__`()

```
dataset.MvsFSImageDataset.__init__(  
    self,  
    data )
```

Konštruktor obrázkovej formy MvsFS datasetu.

## Parametre

<i>data</i>	Dáta MvsFS datasetu.
-------------	----------------------

## 4.7.3 Dokumentácia k metódam

### 4.7.3.1 `__getitem__()`

```
dataset.MvsFSImageDataset.__getitem__ (
    self,
    idx )
```

Interná funkcia na výber položky datasetu.

## Parametre

<i>idx</i>	Identifikátor položky.
------------	------------------------

## Návratová hodnota

Vstupné vlastnosti a ich label.

### 4.7.3.2 `__len__()`

```
dataset.MvsFSImageDataset.__len__ (
    self )
```

Interná funkcia na výpočet veľkosti datasetu.

## Návratová hodnota

Veľkosť datasetu

Dokumentácia pre túto triedu bola generovaná z nasledujúceho súboru:

- [dataset.py](#)

## 4.8 Dokumentácia triedy `trainer.Trainer`

Trieda zodpovedná za tréning neurónových sietí.

### Verejné metódy

- `__init__` (self, datamodule, model, input\_shape, criterion, optimizer, patience, min\_delta, max\_epoch, output\_dir)  
*Konštruktor konfigurovateľného trénera neurónových sietí.*
- `fit` (self, checkpoint=None)  
*Táto funkcia spustí tréning neurónovej siete.*
- `train_epoch` (self, epoch)  
*Spustenie tréningovej epochy.*
- `val_epoch` (self, epoch)  
*Spustenie validačnej epochy.*
- `test_epoch` (self)  
*Spustenie testovacej epochy.*
- `epoch` (self, dataloader, training, caption)  
*Generická metóda pre všetky epochy.*
- `load_checkpoint` (self, filename)  
*Načítanie priebežných výsledkov tréningu z checkpointu.*
- `save_checkpoint` (self, filename)  
*Uloženie priebežných výsledkov tréningu do checkpointu.*
- `save_model` (self)  
*Uloženie výsledného modelu do jit formátu.*
- `save_plot` (self, filename, caption, metric\_name, train\_values, val\_values)  
*Uloženie tréningových a validačných metrík do grafu po ukončení tréningu.*

### Verejné atribúty

- `device`
- `datamodule`
- `model`
- `input_shape`
- `criterion`
- `optimizer`
- `early_stopper`
- `max_epoch`
- `output_dir`
- `train_losses`
- `val_losses`
- `cur_epoch`

#### 4.8.1 Detailný popis

Trieda zodpovedná za tréning neurónových sietí.

## 4.8.2 Dokumentácia konštruktoru a deštruktoru

### 4.8.2.1 \_\_init\_\_()

```
trainer.Trainer.__init__ (
    self,
    datamodule,
    model,
    input_shape,
    criterion,
    optimizer,
    patience,
    min_delta,
    max_epoch,
    output_dir )
```

Konštruktor konfigurovateľného trénera neurónových sietí.

#### Parametre

<i>datamodule</i>	Dátový modul obsahujúci MvsFS dataset.
<i>model</i>	Architektúra neurónovej siete.
<i>input_shape</i>	Tvar vstupu do neurónovej siete.
<i>criterion</i>	Loss funkcia.
<i>optimizer</i>	Optimalizačný algoritmus trénovania.
<i>patience</i>	Počet epoch, ktoré je možné vykonať po sebe bez vylepšenia výsledkov trénovania.
<i>min_delta</i>	Minimálny rozdiel medzi najlepším výsledkom a aktuálnym výsledkom potrebný aby sa epocha považovala za neúspešnú.
<i>max_epoch</i>	Maximálny počet vykonaných epoch.
<i>output_dir</i>	Priečinok kde sa uložia výsledky trénovania.

## 4.8.3 Dokumentácia k metódam

### 4.8.3.1 epoch()

```
trainer.Trainer.epoch (
    self,
    dataloader,
    training,
    caption )
```

Generická metóda pre všetky epochy.

#### Parametre

<i>dataloader</i>	Trieda pomocou ktorej periodicky čítame dávky nášho datasetu.
<i>training</i>	Ak je hodnota True, tak vykonávame spätnú propagáciu. Ak je hodnota False, tak spätnú propagáciu nevykonávame.
<i>caption</i>	Popis aktuálnej epochy.

#### 4.8.3.2 `fit()`

```
trainer.Trainer.fit (
    self,
    checkpoint = None )
```

Táto funkcia spustí tréning neurónovej siete.

##### Parametre

<i>checkpoint</i>	Názov súboru s checkpointom tréningu. Ak nechceme použiť checkpoint a chceme začať tréning od začiatku, tak zadáme null hodnotu.
-------------------	--

#### 4.8.3.3 `load_checkpoint()`

```
trainer.Trainer.load_checkpoint (
    self,
    filename )
```

Načítanie priebežných výsledkov tréningu z checkpointu.

##### Parametre

<i>filename</i>	Súbor z ktorého načítame checkpoint.
-----------------	--------------------------------------

#### 4.8.3.4 `save_checkpoint()`

```
trainer.Trainer.save_checkpoint (
    self,
    filename )
```

Uloženie priebežných výsledkov tréningu do checkpointu.

##### Parametre

<i>filename</i>	Súbor do ktorého uložíme checkpoint.
-----------------	--------------------------------------

#### 4.8.3.5 `save_model()`

```
trainer.Trainer.save_model (
    self )
```

Uloženie výsledného modelu do jit formátu.

Tento model používame v šachovom engine Malakh.

#### 4.8.3.6 save\_plot()

```
trainer.Trainer.save_plot (
    self,
    filename,
    caption,
    metric_name,
    train_values,
    val_values )
```

Uloženie trérovacích a validačných metrík do grafu po ukončení trénovania.

Pomocou týchto grafov môžeme vyhodnotiť priebeh trénovania a odpozorovať underfitting alebo overfitting.

##### Parametre

<i>filename</i>	Súbor do ktorého uložíme graf.
<i>caption</i>	Názov uloženého grafu.
<i>metric_name</i>	Názov zobrazenej metriky.
<i>train_values</i>	Hodnoty trérovacích epoch trénovania.
<i>val_values</i>	Hodnoty validačných epoch trénovania.

#### 4.8.3.7 test\_epoch()

```
trainer.Trainer.test_epoch (
    self )
```

Spustenie testovacej epochy.

V tejto epoche nevykonávame spätnú propagáciu. Táto epocha je spustená po ukončení trénovanie pre vyhodnotenie efektivity výsledného modelu.

#### 4.8.3.8 train\_epoch()

```
trainer.Trainer.train_epoch (
    self,
    epoch )
```

Spustenie trérovacej epochy.

V tejto epoche vykonávame spätnú propagáciu.

##### Parametre

<i>epoch</i>	Poradie epochy.
--------------	-----------------

#### 4.8.3.9 val\_epoch()

```
trainer.Trainer.val_epoch (
```



```
self,  
epoch )
```

Spustenie validačnej epochy.

V tejto epoche nevykonávame spätnú propagáciu.

#### Parametre

<code>epoch</code>	Poradie epochy.
--------------------	-----------------

Dokumentácia pre túto triedu bola generovaná z nasledujúceho súboru:

- [trainer.py](#)



## Kapitola 5

# Dokumentácia súborov

### 5.1 Dokumentácia súboru datamodule.py

Dátové moduly majú na starosť rozdelenie vstupných dát na trénovacie, validačné a testovacie dáta.

#### Triedy

- class [datamodule.DataModuleBitboards](#)  
*Dátový modul pre bitboardovú formu MvsFS datasetu.*
- class [datamodule.DataModuleImages](#)  
*Dátový modul pre obrázkovú formu MvsFS datasetu.*

#### 5.1.1 Detailný popis

Dátové moduly majú na starosť rozdelenie vstupných dát na trénovacie, validačné a testovacie dáta.

#### Autor

Martin Šváb

#### Dátum

Máj 2024

### 5.2 Dokumentácia súboru dataset.py

Definícia vstupného datasetu MvsFS.

#### Triedy

- class [dataset.MvsFSBitboardDataset](#)  
*Dataset MvsFS v bitboardovej forme pre MLP architektúru.*
- class [dataset.MvsFSImageDataset](#)  
*Dataset MvsFS v obrázkovej forme pre CNN architektúru.*

### 5.2.1 Detailný popis

Definícia vstupného datasetu MvsFS.

Autor

Martin Šváb

Dátum

Máj 2024

## 5.3 Dokumentácia súboru experiment.py

Definícia konfigurovateľných experimentov.

### Funkcie

- `experiment.mlp_experiment` (input, output\_dir, hidden\_features, layers)  
*Experiment na testovanie efektivity MLP architektúry a bitboardovej formy MvsFS datasetu.*
- `experiment.cnn_experiment` (input, output\_dir, fc\_layers)  
*Experiment na testovanie efektivity CNN architektúry a obrázkovej formy MvsFS datasetu.*

### 5.3.1 Detailný popis

Definícia konfigurovateľných experimentov.

Autor

Martin Šváb

Dátum

Máj 2024

### 5.3.2 Dokumentácia funkcií

#### 5.3.2.1 `cnn_experiment()`

```
experiment.cnn_experiment (
    input,
    output_dir,
    fc_layers )
```

Experiment na testovanie efektivity CNN architektúry a obrázkovej formy MvsFS datasetu.

## Parametre

<i>input</i>	Názov vstupného csv súboru.
<i>output_dir</i>	Priečinok kde sa uložia výsledky experimentu.
<i>fc_layers</i>	Počet plne prepojených vrstiev CNN architektúry.

## 5.3.2.2 mlp\_experiment()

```
experiment.mlp_experiment (
    input,
    output_dir,
    hidden_features,
    layers )
```

Experiment na testovanie efektivity MLP architektúry a bitboardovej formy MvsFS datasetu.

## Parametre

<i>input</i>	Názov vstupného csv súboru.
<i>output_dir</i>	Priečinok kde sa uložia výsledky experimentu.
<i>hidden_features</i>	Počet neurónov skrytých vrstiev MLP architektúry.
<i>layers</i>	Počet skrytých vrstiev MLP architektúry.

## 5.4 Dokumentácia súboru model.py

Definícia architektúr neurónových sietí.

## Triedy

- class [model.MLP](#)  
*Architektúra plne prepojenej neurónovej siete.*
- class [model.CNN](#)  
*Architektúra konvolučnej neurónovej siete.*

### 5.4.1 Detailný popis

Definícia architektúr neurónových sietí.

## Autor

Martin Šváb

## Dátum

Máj 2024

## 5.5 Dokumentácia súboru trainer.py

Definícia trénovania neurónovej siete.

### Triedy

- class [trainer.EarlyStopper](#)  
*Trieda zodpovedná za predčasné zastavenie trénovania.*
- class [trainer.Trainer](#)  
*Trieda zodpovedná za trénovanie neurónových sietí.*

### Funkcie

- [trainer.decide\\_device](#) ()  
*Pomocná funkcia na rozhodnutie použitia CPU alebo cuda režimu.*

### 5.5.1 Detailný popis

Definícia trénovania neurónovej siete.

#### Autor

Martin Šváb

#### Dátum

Máj 2024

### 5.5.2 Dokumentácia funkcií

#### 5.5.2.1 `decide_device()`

```
trainer.decide_device ( )
```

Pomocná funkcia na rozhodnutie použitia CPU alebo cuda režimu.

#### Návratová hodnota

Ak je cuda dostupné tak hodnota cuda, ak nie je dostupné tak hodnota cpu.

# Register

- `__getitem__`
    - `dataset.MvsFSBitboardDataset`, [14](#)
    - `dataset.MvsFSImageDataset`, [16](#)
  - `__init__`
    - `datamodule.DataModuleBitboards`, [9](#)
    - `datamodule.DataModuleImages`, [10](#)
    - `dataset.MvsFSBitboardDataset`, [14](#)
    - `dataset.MvsFSImageDataset`, [15](#)
    - `model.CNN`, [8](#)
    - `model.MLP`, [12](#)
    - `trainer.EarlyStopper`, [11](#)
    - `trainer.Trainer`, [18](#)
  - `__len__`
    - `dataset.MvsFSBitboardDataset`, [14](#)
    - `dataset.MvsFSImageDataset`, [16](#)
- `cnn_experiment`
  - `experiment.py`, [24](#)
- `datamodule.DataModuleBitboards`, [8](#)
  - `__init__`, [9](#)
- `datamodule.DataModuleImages`, [9](#)
  - `__init__`, [10](#)
- `datamodule.py`, [23](#)
- `dataset.MvsFSBitboardDataset`, [13](#)
  - `__getitem__`, [14](#)
  - `__init__`, [14](#)
  - `__len__`, [14](#)
- `dataset.MvsFSImageDataset`, [15](#)
  - `__getitem__`, [16](#)
  - `__init__`, [15](#)
  - `__len__`, [16](#)
- `dataset.py`, [23](#)
- `decide_device`
  - `trainer.py`, [26](#)
- `early_stop`
  - `trainer.EarlyStopper`, [11](#)
- `epoch`
  - `trainer.Trainer`, [18](#)
- `experiment.py`, [24](#)
  - `cnn_experiment`, [24](#)
  - `mlp_experiment`, [25](#)
- `fit`
  - `trainer.Trainer`, [18](#)
- `forward`
  - `model.CNN`, [8](#)
  - `model.MLP`, [13](#)
- `load_checkpoint`
  - `trainer.Trainer`, [19](#)
- `mlp_experiment`
  - `experiment.py`, [25](#)
- `model.CNN`, [7](#)
  - `__init__`, [8](#)
  - `forward`, [8](#)
- `model.MLP`, [12](#)
  - `__init__`, [12](#)
  - `forward`, [13](#)
- `model.py`, [25](#)
- `save_checkpoint`
  - `trainer.Trainer`, [19](#)
- `save_model`
  - `trainer.Trainer`, [19](#)
- `save_plot`
  - `trainer.Trainer`, [19](#)
- `test_epoch`
  - `trainer.Trainer`, [20](#)
- `train_epoch`
  - `trainer.Trainer`, [20](#)
- `trainer.EarlyStopper`, [10](#)
  - `__init__`, [11](#)
  - `early_stop`, [11](#)
- `trainer.py`, [26](#)
  - `decide_device`, [26](#)
- `trainer.Trainer`, [16](#)
  - `__init__`, [18](#)
  - `epoch`, [18](#)
  - `fit`, [18](#)
  - `load_checkpoint`, [19](#)
  - `save_checkpoint`, [19](#)
  - `save_model`, [19](#)
  - `save_plot`, [19](#)
  - `test_epoch`, [20](#)
  - `train_epoch`, [20](#)
  - `val_epoch`, [20](#)
- `val_epoch`
  - `trainer.Trainer`, [20](#)