

# Klasifikátor hodnotenia kvality obrazu sietnice

## Dokumentácia k projektu BIO

6. decembra 2023

Bc. Tomáš Ondrušek(xondru18)

Bc. Peter Ďurica(xduric05)

# **Obsah**

<b>1</b>	<b>Úvod</b>	<b>2</b>
<b>2</b>	<b>Hodnotenie kvality obrazu sietnice</b>	<b>2</b>
2.1	Fotografia sietnice . . . . .	2
2.2	Kategórie kvality snímkov . . . . .	3
<b>3</b>	<b>Implementácia</b>	<b>4</b>
3.1	Členenie implementácie . . . . .	4
3.1.1	Popis funkcionality programu na ohodnotenie snímkov . . . . .	4
3.1.2	Rozdelenie datasetov . . . . .	5
3.1.3	Predspracovanie snímkov . . . . .	5
3.1.4	Klasifikácia . . . . .	6
<b>4</b>	<b>Výsledky práce</b>	<b>9</b>
<b>5</b>	<b>Záver</b>	<b>11</b>
<b>6</b>	<b>Prílohy</b>	<b>13</b>

# 1 Úvod

Táto práca sa zaoberá automatickým hodnotením kvality snímkov sietnice využitím neurónových sietí. Sietnica [1] je tenká, polopriehľadná, viacvrstvová časť neurálneho tkaniva. Pokrýva dve tretiny vnútra každého oka. Má za úlohu konvertovať elektromagnetický signál z vonku do neurónového signálu, ktorý následne odovzdá optickému nervu.

Kvôli architektúre sietnice [2] je možné v jej snímkach zachytiť choroby oka alebo aj choroby mozgu a cirkulácie krvy. Vďaka analýze snímkov je možné tieto choroby správne identifikovať a nariadiť účinnú liečbu pre pacienta.

Zlá kvalita sietnicových obrázkov [1] môže zvýšiť pravdepodobnosť zlej diagnózy alebo sponzorovania choroby. Takisto môže spôsobiť predĺženie stráveného času pri analýze snímkov. Napríklad rozmazané obrázky môžu zakryť poškodené cievy v sietnici a tým pádom sa môže daná sietnica výskumníkom javiť ako zdravá. Automatická kontrola a identifikácia kvality snímkov môže proces analýzy spresniť a urýchliť. Výsledkom bude väčšia bezpečnosť pacientov a zvýšený komfort zdravotníkov.

## 2 Hodnotenie kvality obrazu sietnice

Sietnicu je možné odfotiť pomocou fundus fotoaparátu. Následné fotografie sa vedia analyzovať a podľa tej analýzy vyhodnotiť zdravie sietnice na obrázku. Táto kapitola sa zaoberá štruktúrou sietnice a hodnotením kvality fotografie.

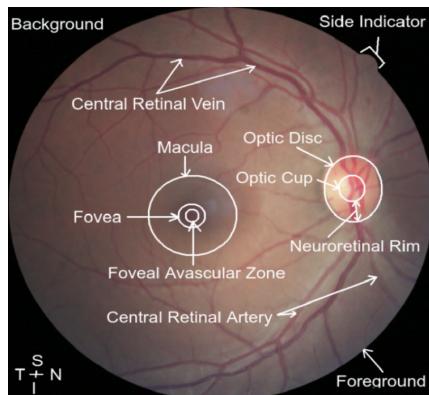
### 2.1 Fotografia sietnice

Na obrázku 1 je zobrazená fotografia sietnice oka. Na snímke je vidno [1] stromovú štruktúru centrálnej sietnicovej tepny (angl. Central Retinal Artery) a centrálnej sietnicovej žily (angl. Central Retinal Vein). Spolu tvoria štruktúru krvných ciev na sietnici.

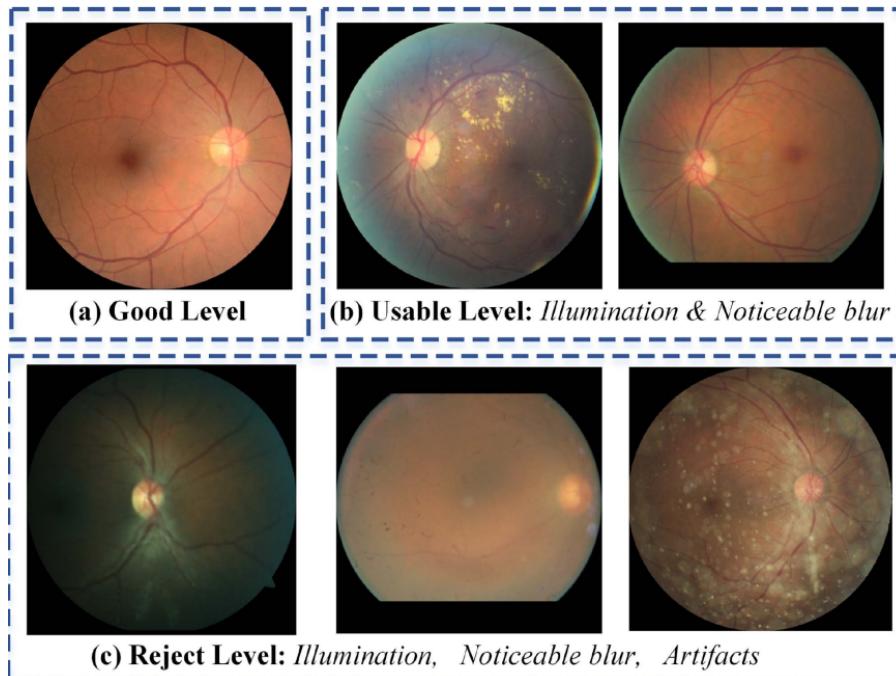
Ďalšie anatomické štruktúry [1], ako žltá škvRNA (angl. Macula), fovea, optický disk (angl. Optic Disk) a optický pohár (angl. Optic Cup) by mali byť rovnako jasne identifikovateľné na dobre zosnímanom obrázku sietnice.

Pozorovaním týchto oblastí sietnice je možné spozorovať niektoré choroby [2] napríklad:

- **Diabetická retinopatia** - Zmeny na cievach sietnice,
- **Makulárna degenerácia** - Zmeny v oblasti žltej škvRNA,
- **Glaukóm** - Zmeny v optickom nerve alebo v cievach.



Obr. 1: Fotografia štruktúr sietnice oka [1]



Obr. 2: Rozdelenie fotografií sietnice oka podľa kvality [3]

## 2.2 Kategórie kvality snímkov

Kvalita fotiek sietnice [3] sa delí do troch kategórií:

- **Good** - Ideálna výsledná kvalita fotografie,
- **Usable** - Stále použiteľná fotografia, ale nie bezchybná,
- **Reject** - Fotografia má výrazné nedostatky a nie je vhodná na analýzu.

V kategórii **Good** sa nachádzajú snímky [3], kde sú jasne viditeľné všetky štruktúry sietnice oka. Obrázok nieje rozmazaný a kontrast ciev oproti prostrediu je dostatočný, aby boli cievy ľahko viditeľné.

V kategórii **Usable** sa nachádzajú snímky [3] z ktorých sa dá stále rozoznať štruktúra sietnice, ale nie všetky atribúty fotografie sú perfektné. Napríklad kontrast ciev nieje úplne výrazný alebo oko nieje správne nasvietené, prípadne je fotografia kúsok rozmazaná. Takisto sa na obrázku môžu vyskytovať nejaké artefakty, ktoré však výrazne neovplyvňujú viditeľnosť objektov na snímke.

V kategórii **Reject** sa nachádzajú snímky [3], ktoré sú nepoužiteľné pre analýzu. Snímka môže byť zle nasvietená, výrazne rozmazaná alebo môže obsahovať rozsiahle artefakty. Na týchto fotografiách by sa len ľahko hľadali relevantné informácie.

Príklady snímkov v jednotlivých kategóriách kvality sú na obrázku 2.

### 3 Implementácia

Projekt je implementovaný v jazyku *Python* a používa viacero dôležitých knižníc a nástrojov na spracovanie dát a prácu s neurónovými sieťami. V rámci implementácie v tomto jazyku boli v projekte použité nasledovné knižnice:

- **PyTorch:** Pre implementáciu neurónových sietí.
- **NumPy:** Na manipuláciu s dátami a prácu s maticami.
- **Pandas:** Pre prácu s dátovými štruktúrami a manipuláciu s údajmi.
- **Dask:** Pre paralelizáciu a rýchlejšie spracovanie dát.
- **Matplotlib a Seaborn:** Na vizualizáciu dát a výsledkov.

#### 3.1 Členenie implementácie

Riešenie projektu sa skladá z troch častí. V prvej časti je za potrebu získať ohodnotenie jednolivých snímkov, či už na trénovanie alebo na testovanie, na to bol v rámci riešenia vytvorený *GUI* program v jazyku *Python* a jeho grafickom frameworku *Tkinter*. Následne je za potrebu prsspracovať obrázky, čo je presnejšie opísané v časti 3.1.3. Ako posledné je samotné trénovanie, testovanie a evaulácia jednolivých neurónových sietí, čo je opísané v časti 3.1.4

##### 3.1.1 Popis funkcionality programu na ohodnotenie snímkov

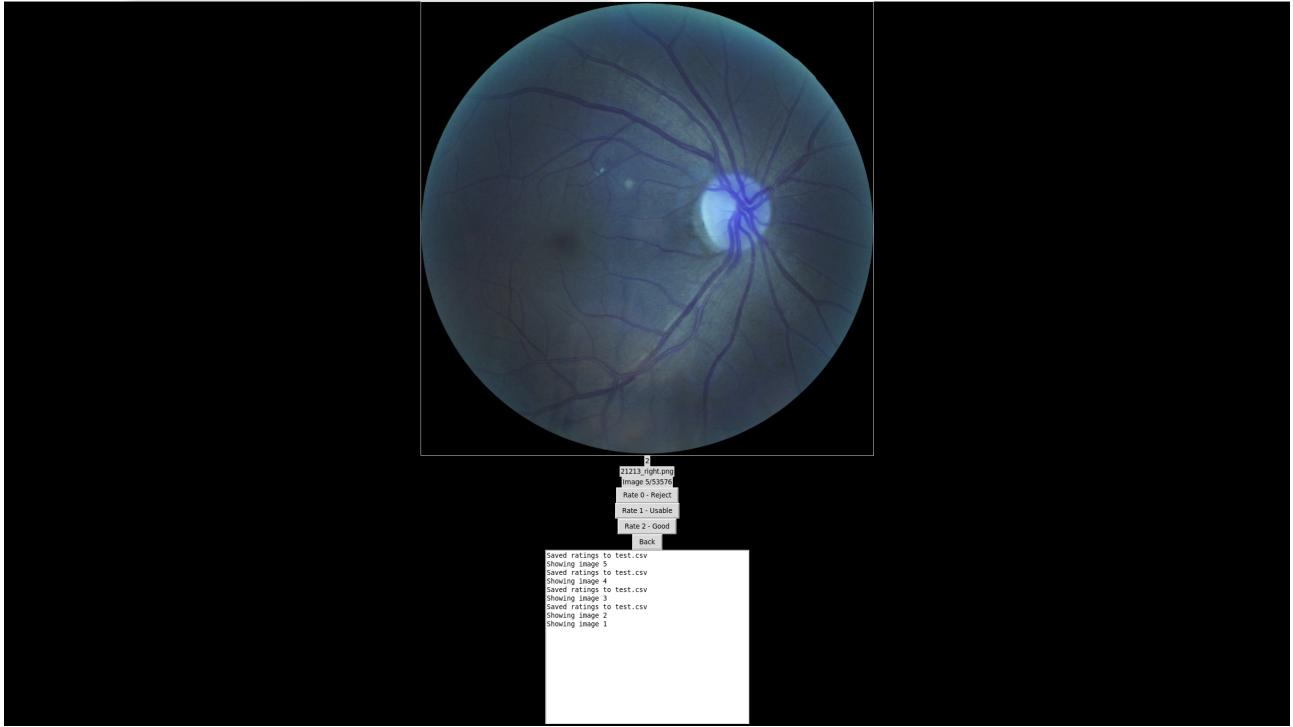
Tento program slúži na hodnotenie kvality obrazu sítnice prostredníctvom grafického užívateľského rozhrania vytvoreného pomocou knižnice Tkinter. Užívateľovi sa zobrazujú snímky sietnice, pričom má možnosť každej priradiť hodnotenie kvality, ako je ukázané na obrázku3.

###### Funkcie programu

- **Zobrazenie obrázkov:** Program načíta zoznam obrázkov zo zadaného adresára a súborového záznamu. Následne zobrazuje obrázky s ich názvami a kvalitou.
- **Hodnotenie kvality:** Užívateľ má možnosť ohodnotiť kvalitu každého obrázku pomocou tlačidiel (0 - Reject, 1 - Usable, 2 - Good). Po ohodnotení sa hodnotenie zapíše do dátového rámca a zobrazí sa nasledujúci obrázok.
- **Navigácia:** Užívateľ môže prechádzať medzi jednolivými obrázkami pomocou tlačidla "Back".
- **Uloženie hodnotení:** Program ukladá hodnotenia kvality do súboru s názvom `...csv`, ktorý je možné zmeniť prepísaním premennej na začiatku kódu a priebežne zobrazuje informácie o procese v logu.

**Inicializácia a spustenie** Program na začiatku kontroluje existenciu súboru `...csv`, v ktorom ukladá hodnotenia. Tento súbor je možné zmeniť na začiatku kódu prepísaním hodnoty premennej. Ak neexistuje, vytvorí ho tak, že prednáčíta všetky cesty ku obrázkom a inicializuje ich hodnotu na 0. Tkinter je použitý na vytvorenie grafického prostredia, kde sú zobrazované obrázky a tlačidlá pre hodnotenie.

**Záver a ukladanie hodnotení** Po ohodnotení všetkých obrázkov sa program zatvorí. To sa dá korektnie takisto pomocou `Esc` klávesy. Po ukončení sú v `.csv` súbore všetky hodnoty vo formáte, v ktorom môžu byť priamo vložené do hlavného programu, ktorý bude opísaný nižšie.



Obr. 3: quality\_anotator.py

### 3.1.2 Rozdelenie datasetov

`split_data.py` je skript, ktorý slúži na rozdelenie dát zo súboru CSV na trénovacie a testovacie dátá. Používa sa na vytvorenie dvoch nových súborov CSV, jeden obsahujúci trénovacie dátá a druhý s testovacími dátami.

Ked' sa skript spustí, načíta vstupný súbor CSV, potom náhodne rozdelí dátá podľa zadaného prahu (číslo medzi 0 a 1). Čím vyšší prah, tým väčšia časť dát pôjde do trénovacej sady. Po rozdelení dát uloží do výstupných súborov CSV pre trénovanie a testovanie.

Tento skript je potrebný v situáciach, kedy je potrebné rozdeliť dátá na dve časti tak, aby mohli byť náhodne rozdelené, čím zabráním nechceným nenatrénovaním nejakej vzorky dát. Script teda najskôr dátá zamieša a potom ich náhodne rozdelí tak, že na výstupe budú dva CSV súbory, ktoré oba bu' du obsahovať časť snímkov, ktoré budú používať ďalšie programy, ako bude opísané nižšie.

### 3.1.3 Predspracovanie snímkov

**Súbor prepare\_dataset.py** Tento súbor obsahuje triedy a funkcie, ktoré sú zodpovedné za predspracovanie datasetu obrázkov sietnice.

- **Preprocessor:** Trieda s metódami na predspracovanie obrázkov.
  - Obsahuje metódy na získanie binárnej masky obrázka, určenie stredu a polomeru na základe masky, vytvorenie kruhu z masky s využitím ohraničujúceho obdĺžnika a ďalšie úpravy obrázkov na odstránenie pozadia.
  - Slúži na vytvorenie obrázka bez pozadia, získanie masky a jej aplikáciu na pôvodný obrázok.
- **FileHandler:** Trieda obsahujúca metódy na otvorenie, čítanie a zápis obrázkov.
  - Umožňuje manipuláciu s obrázkami vrátane otvorenia, čítania a zápisu do súborového systému.

- Súbory obsahujú dask.delayed anotácie, ktoré sú použité v hlavnom súbore na paralelné spracovanie.

**Súbor preprocess.py** Tento súbor slúži na vykonanieto predspracovania obrázkov v datasete pomocou metód z modulu `prepare_dataset.py`.

- Používa knižnice ako cv2 (OpenCV), numpy a dask pre manipuláciu a predspracovanie obrázkov.
- Definuje triedy `Preprocessor` a `FileHandler`, ktoré poskytujú metódy pre manipuláciu s obrázkami a prácu so súborovým systémom zo súboru `preprocess_dataset.py`
- Obsahuje metódy pre získanie masky, stredu a polomeru z obrázkov a manipuláciu s obrázkami pre odstránenie pozadia.

Slúži na spustenie predspracovania datasetu obrázkov sietnice. K tomu používa, ako bolo spomenuté vyššie paralelné spracovanie za pomoci knižnice *Dask*.

### 3.1.4 Klasifikácia

Tento kód predstavuje hlavný súbor projektu, ktorý sa zaoberá trénovaním, testovaním a vyhodnocovaním modelu pre klasifikáciu obrázkov očí do troch kategórií: "Good", "Usable" a "Reject".

Prvým krokom je nastavenie prostredia a import potrebných knižníc. Ďalej sa spracúvajú argumenty príkazového riadka, kde je možné definovať cesty k dátam, parametre trénovalia a ďalšie. Následne sa inicializuje model neurónovej siete podľa zvolenej architektúry, napríklad `dense121_mcs`, `resnet18_mcs` alebo `resnet50_mcs`.

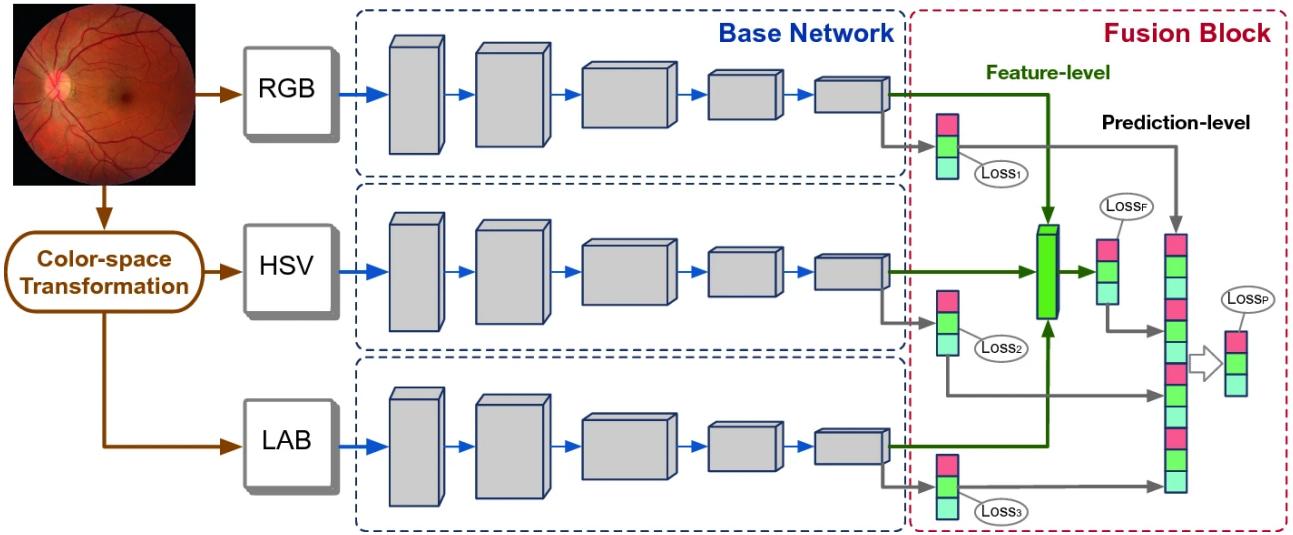
Tieto modely sú inšancované 3 krát, a to pre každú farebnú spektrálnu zložku snímky, konkrétnie RGB, HSV a LAB. Tieto modely sú následne spojené do jedného modelu, ktorý je použitý na klasifikáciu [4]. Tri inštancie daného modelu sú vytvorené pomocou triedy

`{denseNet121, resnet18, resnet50}_v0` a sú následne spojené v triede

`{denseNet121, resnet18, resnet50}_mcs`. Každá inštancia modelu predstavuje jednu vetvu modelu (`featureA`, `featureB`, `featureC`) a obsahuje v sebe architektúru `{denseNet121, resnet18, resnet50}` s modifikovanou klasifikačnou vrstvou na klasifikáciu do `n_class` tried s pomocou sigmoidnej aktivačnej funkcie.

V triede `\{denseNet121, resnet18, resnet50\}_mcs` sú tieto tri vetvy skombinované. Konkrétnie:

1. Vstupné obrázky prechádzajú cez každú vetvu modelu (`x, y, z`) a extrahujú sa príznaky (features) pomocou `featureA`, `featureB`, `featureC`. Tieto extrahované príznaky sú označené ako `x1, y1, z1`.
2. Potom prechádzajú cez klasifikačné vrstvy (`classA, classB, classC`), ktoré sú taktiež súčasťou každej vetvy modelu. Tieto vrstvy generujú výstupné vektory označené ako `x2, y2, z2`.
3. Výstupné vektory z klasifikačných vrstiev pre každú vetvu (`x2, y2, z2`) sú spojené do jedného vektoru s pomocou operácie `torch.cat()`, ktorý je ďalej spracovaný v lineárnej vrstve `combine2` so sigmoidnou aktiváciou.
4. Okrem toho, výstupné príznaky z extrakčných vetiev (`x1, y1, z1`) sú tiež spojené do jedného vektoru pomocou operácie `torch.cat()` a spracované v lineárnej vrstve `combine1` so sigmoidnou aktiváciou.



Obr. 4: Architektúra siete zložená z troch inštancií modelu spojených lineárnow vrstvou so sigmoidnou aktivačnou funkciou. [4]

Takto sú výstupy z jednotlivých vetiev kombinované v jednom výstupnom vektore a výstupy z jednotlivých vrstiev sú spoločne využité pre klasifikáciu obrázkov. Tento komplexný model tak kombinuje informácie zo všetkých troch vetiev pre lepšie výsledky klasifikácie. Je zobrazený na obrázku 4.

V prípade, že je zvolený režim `train`, kód načíta trénovaciu a validačnú sadu obrázkov a vykoná trénovanie modelu. To zahŕňa výpočet straty, optimalizáciu váh pomocou algoritmu spätného šírenia chyby a ukladanie modelu s najlepšími výsledkami.

**Trénovací proces** Trénovací proces je z pohľadu kódu rozdelený do niekoľkých krokov:

### Príprava dát

#### 1. Načítanie dát:

- Načíta trénovaciu a validačnú sadu obrázkov pomocou triedy `DatasetGenerator`, kde sú definované transformácie pre manipuláciu s obrázkami.

### Trénovanie

#### 1. Inicializácia modelu a optimalizátora:

- Vyberie neurónovú sieť (napr. `dense121_mcs`, `resnet18_mcs`, `resnet50_mcs`) na základe argumentov príkazového riadku.
- Definuje stratovú funkciu a algoritmus optimalizácie pre trénovanie modelu.

#### 2. Trénovací cyklus (epoch):

- Prechádza trénovaciu sadu obrázkov pomocou `DataLoader`.
- Načíta vstupné dáta a vykonáva predikcie pomocou modelu.
- Vypočíta stratovú funkciu a získa gradienty pre aktualizácie váh.

- Aktualizuje váhy modelu na základe optimalizačného algoritmu.

### 3. Validácia:

- Hodnotí model na validačných dátach po každom epochu pomocou metódy `validate()`.
- Vypočíta chybu modelu na validačných dátach a porovná ju s najlepším dosiahnutým výsledkom.

### 4. Uloženie modelu:

- Pamäta si najlepší model po každom vylepšení iterácie. Na konci najlepší model uloží ako `.tar` súbor.

**Výstup trénovania** V každom kroku trénovania sa vypisuje priebeh pomocou progress baru, vypíše sa stratová hodnota a prejde sa do ďalšieho epochu.

V režime `test` sa načíta testovacia sada obrázkov a vykonáva výpočet pravdepodobnosti, že daný snímok patrí do jednej z tried pomocou natrénovaného modelu. Výsledky testovania sa ukladajú do súboru CSV.

**Testovací proces** Testovanie v kóde zahŕňa nasledujúce kroky:

#### Načítanie testovacích dát

- Používa sa rovnaká trieda `DatasetGenerator` na načítanie testovacej sady obrázkov s rovnakými transformáciami ako pri trénovaní modelu.

#### Inferencia a vyhodnotenie

##### • Spustenie testovania:

- Načítavajú sa testovacie dáta prostredníctvom `DataLoader` triedy.
- Model sa používa na výpočet pravdepodobnosti patrenia do triedy na testovacích dátach.
- Výstup z modelu sa ukladá a vyhodnocuje.

**Výsledky testovania** Po vykonaní inferencie sa výsledky zaznamenávajú a ukladajú do súborov. Pri testovaní sa tiež vypisujú informácie o dĺžke testovacej sady a spracovanie jednotlivých obrázkov.

**Evaluácia modelu** Po zvolení režimu evaluácie v hlavnom súbore sa vykoná niekoľko krokov:

1. **Načítanie dát:** Načítajú sa označenia testovacej sady z príslušného CSV súboru a predikcie modelu zo súboru s klasifikovanými obrázkami.
2. **Výpočet metrík:** Inštancia triedy `MetricCalculator` je vytvorená na výpočet metrík. Tieto metriky zahŕňajú presnosť (accuracy), F1 skóre, plochu pod ROC krivkou (AUC), senzitivitu, špecifickitu a ďalšie. Vypočítavajú sa na základe pôvodných označení a predikcií modelu.
3. **Výpis metrík:** Vypočítané metriky sú vypísané do konzoly. Tento výpis poskytuje užitočné informácie o výkone modelu pri testovaní na testovacej sade dát.

Tento proces evaluácie poskytuje dôležité informácie o presnosti a výkone modelu, ktoré môžu byť použité na hodnotenie jeho účinnosti pri klasifikácii snímkov sietnice.

## 4 Výsledky práce

Výsledok prvého kroku opísaného v časti 3.1.1 je csv súbor, ktorého štruktúra vyzerá nasledovne

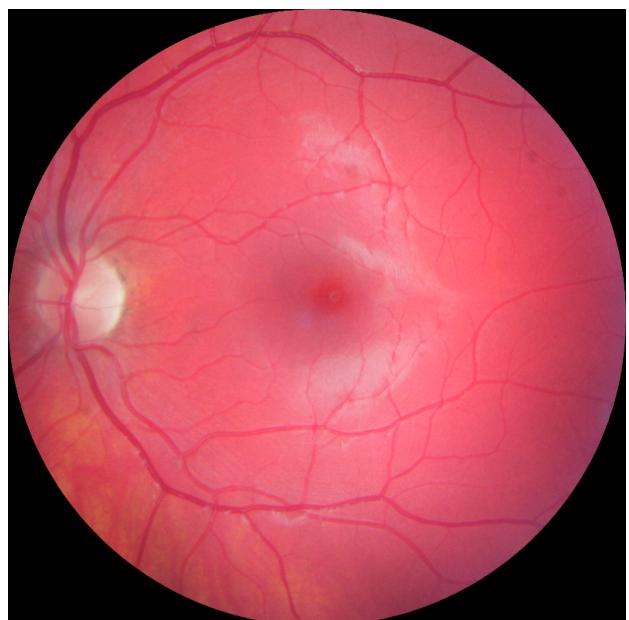
```
,image,quality,DR-grade  
0,1_right.jpeg,1,0  
1,10001_right.jpeg,0,2  
2,10004_right.jpeg,0,0  
3,10008_left.jpeg,0,1  
4,10016_right.jpeg,0,2  
5,10026_left.jpeg,0,0  
.....  
.....
```

Tento formát zahŕňa názov snímku a ohodnotenie kvality v rozsahu od 0 do 2. Zachováva len názov snímku a nie celú cestu k nemu, čo slúži v prípade, že sa snímky budú v budúcnu presúvať alebo sa bude priečinok premenovať.

Ďalším výsledkom je predspracovanie snímku. Vstupný obrázok obsahuje nepotrebné informácie pre trénovanie neurónovej siete, preto je v záujme obmedziť informácie na snímke tak, aby obsahovali len ROI (Region Of Interest). Tento postup je opísaný v časti 3.1.3. Výsledkom je normalizovaný obrázok v rozlíšení 800x800px, teda vo štvorcovom pomere strán, ktorý obsahuje minimum pozadia oproti originálnemu snímku. Tento postup je ukázaný na obrázkoch 5, 6



Obr. 5: Originálny snímok



Obr. 6: Snímok po orezani

Obr. 7: Porovnanie snímku pred a po pred-spracovaní

Trénovanie modelov bolo spustené na počítači s procesorom AMD Ryzen 7 7700x a grafickou kartou NVIDIA GeForce RTX 4070. Z trénovania, testovania a evaluácie boli vytvorené logy, ktoré sú uložené v priečinku logs/. V rámci presnosti jednotlivých modelov bola vytvorená nasledujúca

tabuľka:

Model	Presnosť	Trvanie trénovania	Dataset	Epochy
ResNet18	0.82	206 minút	EyePACS	30
ResNet50	0.67	270 minút	EyePACS	30
DenseNet121	0.75	440 minút	EyePACS	30
ResNet18	0.63	12 minút	STRaDe	30
ResNet50	0.62	17 minút	STRaDe	30
DenseNet121	0.63	28 minút	STRaDe	30

Tabuľka 1: Výsledky trénovania modelov na rôznych datasetoch

Ukážka výstupu trénovanie pre model `densenet\_121` trénovaný 20 epochov je ukázaná prílohe 1. Z tohto výstupu viem postupne vyčítať nasledovné:

- Parametre pri spustení
  - Zariadenie, ktoré bude na trénovanie použité (v tomto prípade grafická karta)
  - Model, ktorý ide byť trénovaný (v tomto prípade model `densenet121`)
  - Vybraný `criterion` a `optimizer`
- Priebeh trénovania
  - Priebeh trénovania cez jeden epoch
  - Priebeh validácie aktuálne natrénovaného modelu
  - Zobrazenie a porovnanie najlepšieho vs aktuálneho epochu
- Priebeh testovania
  - Dĺžka `test_loader-u * 4 = 16252` snímkov na testovanie
  - Priebeh trénovania
  - Oznámenie o uložení výsledkov do súboru `.csv`
- Evaluácia
  - Zobrazenie súborov z ktorých sa ide pracovať a do ktorých sa ide ukladať
  - Zobrazenie metrík, a to konkrétnie:
    - \* **Accuracy:** Percentuálna miera správne klasifikovaných príkladov zo všetkých príkladov.
    - \* **F1-Score:** Harmonický priemer presnosti (precision) a úplnosti (recall). Poskytuje vyváženú predstavu o výkone modelu.
    - \* **AUC (Area Under the Curve):** Plocha pod krivkou ROC (Receiver Operating Characteristic). Miera schopnosti modelu rozlišovať medzi triedami.
    - \* **Sensitivity (Recall):** Percentuálna miera správne identifikovaných príkladov pozitívnej triedy zo všetkých skutočných príkladov tejto triedy.
    - \* **Precision:** Percentuálna miera správne identifikovaných príkladov pozitívnej triedy zo všetkých identifikovaných príkladov tejto triedy.
    - \* **Specificity:** Percentuálna miera správne identifikovaných príkladov negatívnej triedy zo všetkých skutočných príkladov tejto triedy.
    - \* **Micro-Precision:** Vypočítaná ako celková presnosť pre všetky triedy modelu.

- \* **Micro-Sensitivity:** Vypočítaná ako celková úplnosť pre všetky triedy modelu.
  - \* **Micro-Specificity:** Vypočítaná ako celková špecifita pre všetky triedy modelu.
  - \* **Micro-F1:** Vypočítaná ako celkový F1-skóre pre všetky triedy modelu.
  - \* **TP (True Positives):** Počet správne klasifikovaných pozitívnych príkladov.
  - \* **TN (True Negatives):** Počet správne klasifikovaných negatívnych príkladov.
  - \* **FP (False Positives):** Počet nesprávne klasifikovaných ako pozitívne príklady.
  - \* **FN (False Negatives):** Počet nesprávne klasifikovaných ako negatívne príklady.
- Zobrazenie časových štatistik - doba trvania exekúcie daného módu/módov spustenia programu (bola použitá unixová utilita `time`)

## 5 Záver

Cieľom práce bolo navrhnuť a vytvoriť nástroj, ktorého výsledkom je model neurónovej siete, ktorý analyzuje a následne klasifikuje obrázky sietnice do troch úrovní podľa kvality danej snímky. Táto funkcia môže urýchliť a spresniť kontrolu snímok sietnice pri pacientoch s chorobou, ktorú je možné identifikovať na týchto snímkach. Ako doplnok k práci je aj grafický program na manuálnu klasifikáciu kvality snímkov do generovanej tabuľky v súbore CSV.

Pred samotným trénovaním neurónovej siete bolo potrebné pred-pripraviť obrázky pre korektnú prácu s nimi. Po pripravení obrázkov je možné začať trénovanie modelu v jednotlivých epochách. Po validácii klasifikácie modelu každej epochy sa vyberie model s najväčšou úspešnosťou pri testovaní.

Najviac časovo náročnou časťou práce bolo práve trénovanie modelov, ktoré trvalo hodiny napriek využitiu modernej grafickej karty na akcelerovanie času tréningu. Najúspešnejším modelom neurónovej siete bol model, ktorého výpis z trénovania sa nachádza v prílohe 2. Tento model má úspešnosť správnej klasifikácie približne 82 %. Úspešnosť modelu by mohla byť zvýšená prípadným zväčšením počtu epochov pri trénovaní. Aj napriek tomu je momentálna úspešnosť dostatočne vysoká na výrazné zrýchlenie a spresnenie kontroly snímok sietnice oka.

## Zdroje

- [1] Biswas, S.; Rohdin, J.; Kavetskyi, A.; aj.: Grading Quality of Color Retinal Images to Assist Fundus Camera Operators. In *2020 IEEE 33rd International Symposium on Computer-Based Medical Systems (CBMS)*, 2020, s. 77–82, doi:10.1109/CBMS49503.2020.00022.
- [2] Abràmoff, M. D.; Garvin, M. K.; Sonka, M.: Retinal Imaging and Image Analysis. *IEEE Reviews in Biomedical Engineering*, ročník/vydanie 3, 2010: s. 169–208, doi:10.1109/RBME.2010.2084567.
- [3] Hou, Q.; Cao, P.; Jia, L.; aj.: Image Quality Assessment Guided Collaborative Learning of Image Enhancement and Classification for Diabetic Retinopathy Grading. *IEEE Journal of Biomedical and Health Informatics*, ročník/vydanie 27, č. 3, 2023: s. 1455–1466, doi:10.1109/JBHI.2022.3231276.
- [4] Fu, H.; Wang, B.; Shen, J.; aj.: Evaluation of Retinal Image Quality Assessment Networks in Different Color-Spaces. In *Medical Image Computing and Computer Assisted Intervention – MICCAI 2019*, Cham: Springer International Publishing, 2019, ISBN 978-3-030-32239-7, s. 48–56.

## 6 Prílohy

Listing 1: Výstup z trénovalia a evaluácie modelu pre model densenet121 a 20 epochov

```
device:  
GPU - NVIDIA GeForce RTX 4070  
  
model:  
dense121_mcs  
  
criterion:  
BCELoss()  
  
optimizer:  
SGD (  
Parameter Group 0  
    dampening: 0  
    differentiable: False  
    foreach: None  
    lr: 0.01  
    maximize: False  
    momentum: 0  
    nesterov: False  
    weight_decay: 0  
)  
  
Total params: 28.86M  
  
Training mode selected  
Length of train_loader: 3136  
Length of val_loader: 3136  
Processing train Epoch -> 1 / 20 ##### 3136  
/ 3136 | Time: 0.0 mins | Loss: 0.3288  
Processing validation ##### 3136 / 3136 |  
Time: 0.00 mins  
Current Loss: 0.4153175348967162| Best Loss: inf at epoch: 0  
Processing train Epoch -> 2 / 20 ##### 3136  
/ 3136 | Time: 0.0 mins | Loss: 0.2988  
Processing validation ##### 3136 / 3136 |  
Time: 0.00 mins  
Current Loss: 0.35579348392119364| Best Loss: 0.4153175348967162 at epoch  
: 0  
... (remaining output)  
Saving model with best loss: 0.24337111543673945 at epoch: 17  
Model saved to ./result/densenet121.tar  
  
Testing mode selected  
Length of test_loader: 4063
```

```
Processing inference ##### 4063 / 4063 |
Time: 0.0 min.
Saving results to: data/densenet121.csv
```

```
Evaluating mode selected
Reading test labels from: data/test_labels.csv
Reading results from: data/densenet121.csv
```

```
Values classified by model: dense121_mcs saved in data/densenet121.csv
have following metrics:
```

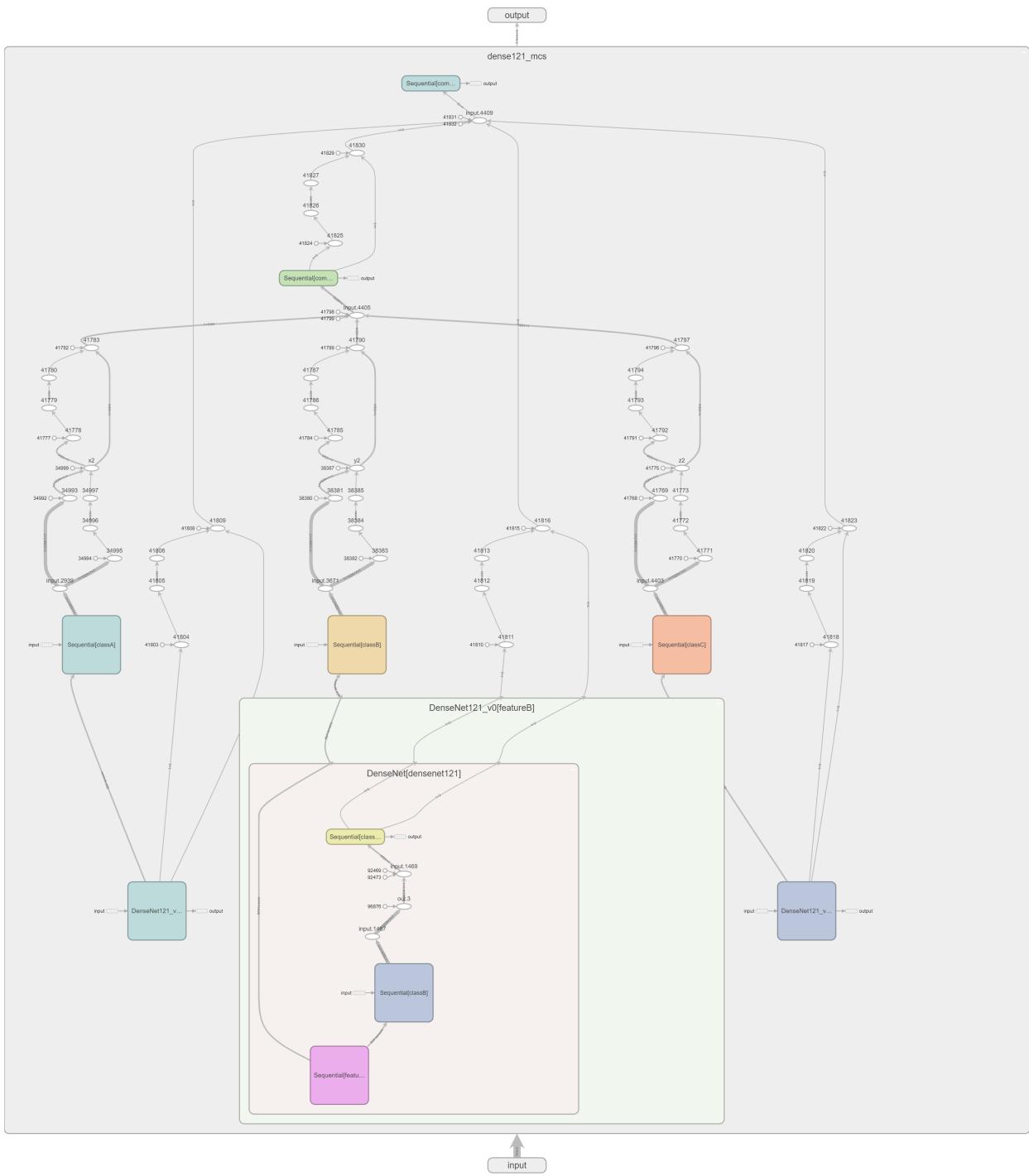
```
Accuracy: 0.6749
F1: 0.5758
AUC: 0.8462
Sensitivity: 0.5943
Precision: 0.6371
Specificity: 0.7973
micro-Precision: 0.6749
micro-Sensitivity: 0.6749
micro-Specificity: 0.8375
micro-F1: 0.6749
tp: 3655.6667
tn: 9072.0000
fp: 1760.6667
fn: 1760.6667
```

```
-----
Executed in 239.94 mins      fish          external
    usr time 298.86 mins     0.00  micros   298.86  mins
    sys time  39.89 mins   430.00  micros   39.89  mins
```

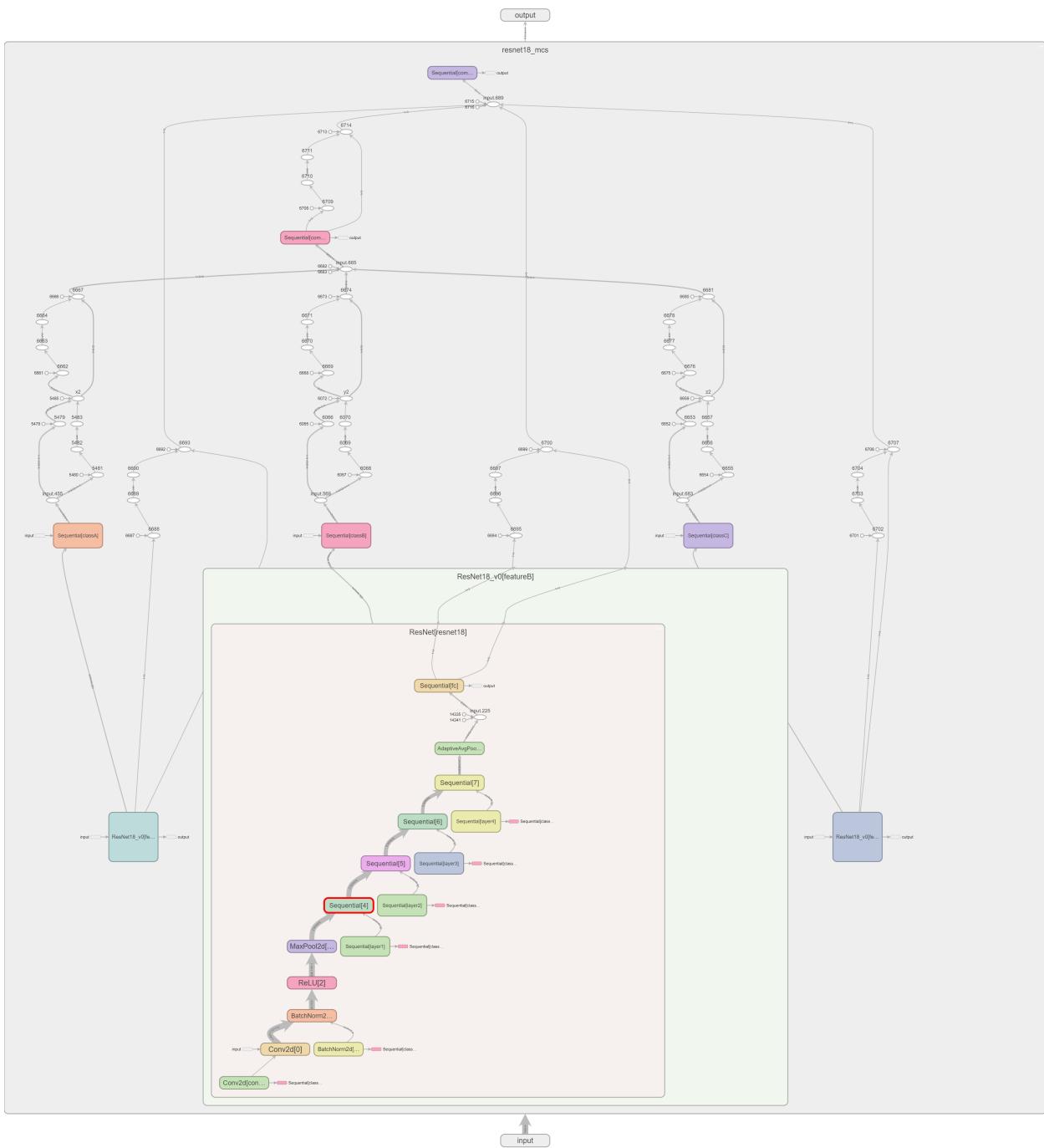
Listing 2: Výstup z trénovania a evaluácie modelu s najlepšími výsledkami pre model resnet18 a 30 epochov

```
device:  
GPU — NVIDIA GeForce RTX 4070  
  
model:  
resnet18_mcs  
  
criterion:  
BCELoss()  
  
optimizer:  
SGD (  
Parameter Group 0  
    dampening: 0  
    differentiable: False  
    foreach: None  
    lr: 0.01  
    maximize: False  
    momentum: 0  
    nesterov: False  
    weight_decay: 0  
)  
  
Total params: 33.54M  
  
Training mode selected  
Length of train_loader: 3136  
Length of val_loader: 3136  
Processing train Epoch -> 1 / 30 ##### 3136 / 3136 | Time: 0.0 mins | Loss: 0.5841  
Processing validation ##### 3136 / 3136 | Time: 0.00 mins  
Current Loss: 0.3967082689668299| Best Loss: inf at epoch: 0  
Processing train Epoch -> 2 / 30 ##### 3136 / 3136 | Time: 0.0 mins | Loss: 0.5701  
Processing validation ##### 3136 / 3136 | Time: 0.00 mins  
Current Loss: 0.3577176400469806| Best Loss: 0.3967082689668299 at epoch: 0  
... (remaining output)  
Saving model with best loss: 0.17737707616262405 at epoch: 29  
Model saved to ./result/resnet_18.tar  
  
Testing mode selected  
Length of test_loader: 4063  
Processing inference ##### 4063 / 4063 |
```

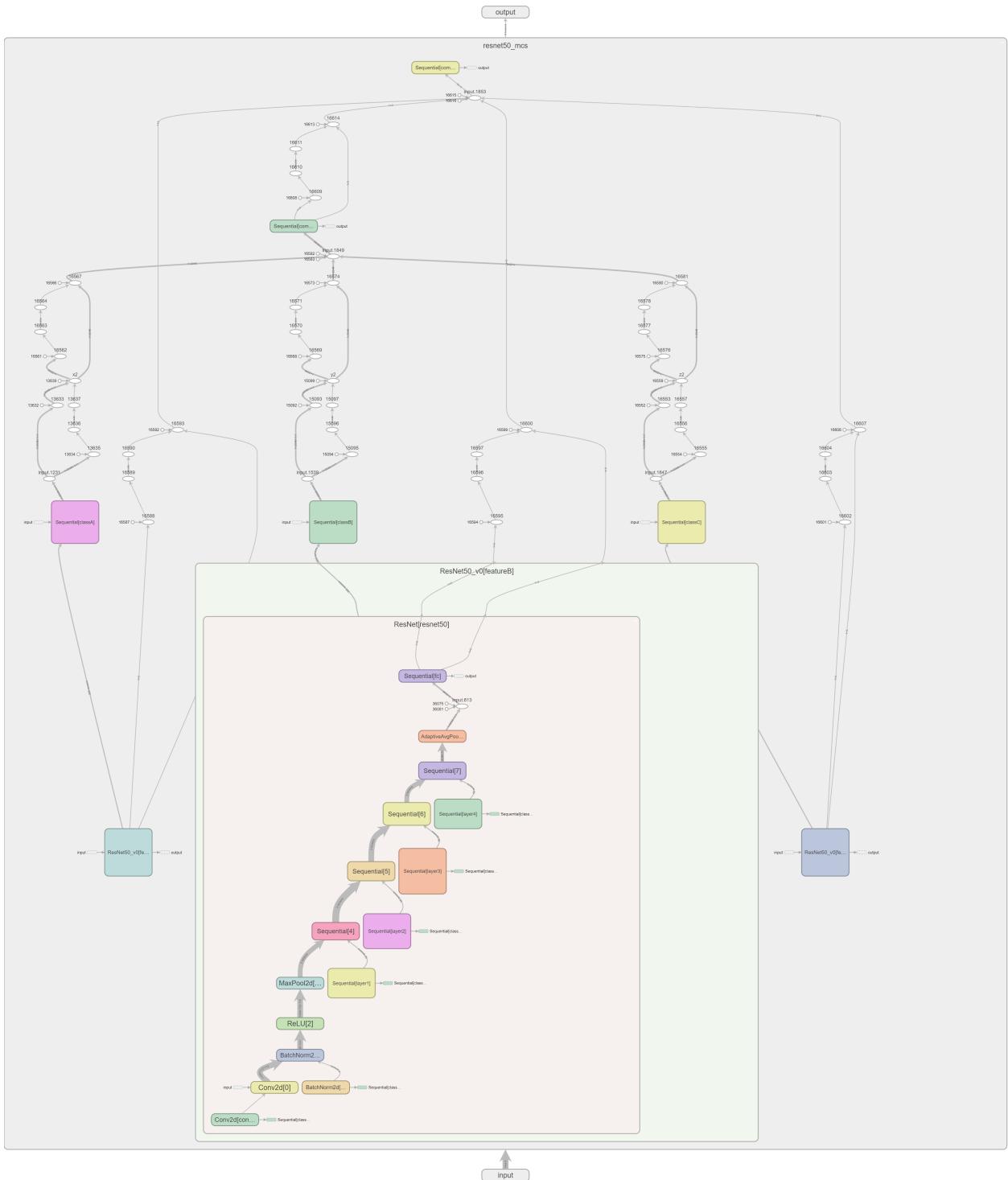
```
Time: 0.0 min.  
Saving results to: data/resnet18.csv  
  
Evaluating mode selected  
Reading test labels from: data/test_labels.csv  
Reading results from: data/resnet18.csv  
  
Values classified by model: resnet18_mcs saved in data/resnet18.csv  
have following metrics:  
  
Accuracy: 0.8178  
F1: 0.7784  
AUC: 0.9358  
Sensitivity: 0.7733  
Precision: 0.7991  
Specificity: 0.8964  
micro-Precision: 0.8178  
micro-Sensitivity: 0.8178  
micro-Specificity: 0.9089  
micro-F1: 0.8178  
tp: 4429.6667  
tn: 9846.0000  
fp: 986.6667  
fn: 986.6667  
  
-----  
Executed in    72.04 mins      fish          external  
    usr time   206.33 mins     0.00  micros  206.33  mins  
    sys time    19.43 mins   430.00  micros   19.43  mins
```



Obr. 8: Architektúra DenseNet121 - vizualizovaná pomocou TensorBoard



Obr. 9: Architektúra ResNet18 - vizualizovaná pomocou TensorBoard



Obr. 10: Architektúra ResNet50 - vizualizovaná pomocou TensorBoard