ŽILINSKÁ UNIVERZITA V ŽILINE
FAKULTA RIADENIA A INFORMATIKY

DIPLOMOVÁ PRÁCA

BC. ANDREJ BELIANČÍN

Využitie autoenkodéra v klasifikačných úlohách

Vedúci práce: Ing. Peter Tarábek, PhD.
Registračné číslo: 982/2019
Žilina, 2020

ŽILINSKÁ UNIVERZITA V ŽILINE
FAKULTA RIADENIA A INFORMATIKY

DIPLOMOVÁ PRÁCA

ŠTUDIJNÝ ODBOR: INFORMATIKA

BC. ANDREJ BELIANČÍN

Využitie autoenkodéra v klasifikačných úlohách

Žilinská univerzita v Žiline
Fakulta riadenia a informatiky
Školiace pracovisko: Katedra matematických metód a operačnej analýzy
Žilina, 2020

ŽILINSKÁ UNIVERZITA V ŽILINE, FAKULTA RIADENIA A INFORMATIKY.

ZADANIE TÉMY DIPLOMOVEJ PRÁCE.

Študijný program: Informačné systémy

Zameranie: Spracovanie dát

Meno a priezvisko
Andrej Beliančín

Osobné číslo
558108

Názov práce v slovenskom aj anglickom jazyku

Využitie autoenkodéra v klasifikačných úlohách

Use of Autoencoder in Classification Tasks

Zadanie úlohy, ciele, pokyny pre vypracovanie (Ak je málo miesta, použite opačnú stranu)

Cieľ diplomovej práce:
Cieľom práce je preskúmať možnosti využitia autoenkodéra v klasifikačných úlohách založených na hlbokom strojovom učení. V práci bude vyskúšaný prístup súbežného trénovania autoenkodéra a klasifikátora. Tento prístup bude porovnaný s klasickým trénovaním samotného klasifikátora a s prístupom založeným na predtrénovaní siete pomocou autoenkodéra a jej následným dotrénovaním. Porovnávanie bude robené v niekoľkých rovinách: porovnanie výsledkov na základe presnosti, posúdenie robustnosti voči neznámym dátam a posúdenie vplyvu množstva anotovaných dát na výsledky.

Obsah:
1. Analýza súčasného stavu.
2. Výber a oboznámenie sa s vhodným frameworkom pre hlboké strojové učenie.
3. Výber vhodných datasetov a architektúr pre klasifikátor a autoenkóder.
4. Vytvorenie experimentov pre jednotlivé prístupy.
5. Vyhodnotenie experimentov na základe presnosti, robustnosti voči neznámym dátam a vplyvu množstva anotovaných dát na výsledky.

Meno a pracovisko vedúceho DP: Ing. Peter Tarábek, PhD., KMMOA, ŽU
Meno a pracovisko tútora DP:

11.2.2020

vedúci katedry
(dátum a podpis

Čestné vyhlásenie

Čestne prehlasujem, že som prácu vypracoval samostatne s využitím dostupnej literatúry a vlastných vedomostí. Všetky zdroje použité v diplomovej práci som uviedol v súlade s predpismi.

Súhlasím so zverejnením práce a jej výsledkov.

V Jyväskyle, dňa 18.5.2020 Andrej Beliančín

Poďakovanie

Rád by som na tomto mieste poďakoval vedúcemu mojej diplomovej práce Ing. Petrovi Tarábekovi, PhD. za odbornú pomoc, pripomienky, vecnú kritiku a cenné rady pri tvorbe práce. Na záver ďakujem rodine a najbližším za podporu počas celého štúdia.

ABSTRAKT V ŠTÁTNOM JAZYKU

BELIANČÍN, Andrej: *Využitie autoenkodéra v klasifikačných úlohách*. [Diplomová práca]. – Žilinská univerzita v Žiline, Fakulta riadenia a informatiky, Katedra matematických metód a operačnej analýzy. – Školiteľ: Ing. Peter Tarábek, PhD.– Stupeň odbornej kvalifikácie: Inžinier. – Žilina: FRI UNIZA, 2020. – 80 s.

Cieľom diplomovej práce je preskúmať možnosti využitia autoenkódera v klasifikačných úlohách. Na začiatku práce navrhneme vysokoúrovňovú architektúru pre modely autoenkódera a klasifikátora, v ktorej tieto modely zdieľajú spoločnú časť, ktorú vieme predtrénovať pre klasifikátor autoenkóderom. Na základe architektúry sme navrhli viacero prístupov k tréningu autoenkódera a klasifikátora, kde základná myšlienka spočíva v predtrénovaní autoenkódera a následnom dotrénovaní klasifikátora. Užitočnosť použitia autoenkódera vychádza z predpokladu, že autoenkóder dokážeme predtrénovať na neanotovaných dátach, ktoré je zvyčajne oveľa jednoduchšie získať ako anotované dáta.

Na natrénovaných modeloch skúmame, ako použitie autoenkódera ovplyvní presnosť klasifikácie, aký má vplyv množstvo anotovaných dát na presnosť a porovnáme výsledky s tradičným prístupom k tréningu klasifikátora. Z výsledkov sa ukázalo, že použitie autoenkódera má zmysel iba pri extrémne nízkom množstve trénovacích dát - približne 600 vzoriek v trénovacej množine pre nami skúmané datasety. V ostatných prípadoch dosahoval štandardný prístup k tréningu rovnakú ak nie lepšiu presnosť ako navrhnuté prístupy k tréningu.

V poslednej časti práce navrhneme a realizujeme spôsoby pre označenie vstupných dát za neznáme, kde budeme pozorovať, aký má autoenkóder vplyv na robustnosť voči neznámym dátam. Prvý z nich používa rekonštrukčnú chybu autoenkódera a druhý používa výstup z modifikovaného klasifikátora. Všeobecne nemožno odporučiť ani jeden z navrhovaných spôsobov, ale druhý menovaný vykazuje za určitých podmienok veľmi dobré výsledky, kde viac ako 90 % dát z neznámych datasetov označil správne za neznáme.

Kľúčové slová: konvolučné neurónové siete, klasifikácia, autoenkóder, rekonštrukcia obrazu

ABSTRAKT V CUDZOM JAZYKU

BELIANČÍN, Andrej: *Use of Autoencoder in Classification Tasks*. [Master thesis]. – University of Žilina in Žilina, Faculty of Management Science and Informatics, Department of Mathematical Methods and Operations Research; – Supervisor: Ing. Peter Tarábek, PhD.– Degree of qualification: Master. – Žilina: FRI UNIZA, 2020. – 80 p.

The aim of master thesis is to examine the possibilities of using an autoencoder in classification tasks. In the beginning of the thesis, we design a high-level architecture for models of the autoencoder and a classifier, in which models share a common part, which can be pre-trained for classifier by autoencoder. Based on architecture we proposed several approaches to the training of the autoencoder and the classifier, where the basic concept is to pre-train the autoencoder for later training of the classifier. The usefulness of using an autoencoder is based on the assumption that we can pre-train the autoencoder on unannotated data, which is usually much easier to obtain than annotated data.

On trained models, we examine how the use of the autoencoder affects the accuracy of classification, how the amount of annotated data affects the accuracy and we compare the results with the traditional approach to classifier training. The results indicate that the use of the autoencoder makes sense only with an extremely low amount of training data - about 600 samples in a training set for datasets we examine. In other cases, the standard approach to the training achieved the same if not better accuracy than the proposed approaches to the training.

In the last part of the thesis, we will suggest and implement methods for marking input data as unknown, where we observe an effect of the autoencoder on the robustness against unknown data. The first one uses the autoencoder's reconstruction error and the second one uses the output from the modified classifier. In general, none of proposed methods can be recommended, but the latter indicates very good results under certain conditions, where more than 90% of the data from unknown datasets were correctly identified as unknown.

Key words: convolutional neural network, classification, autoencoder, image reconstruction

Obsah

Zoznam obrázkov **10**

Zoznam tabuliek **12**

Zoznam skratiek **15**

Úvod **16**

1 Súčasný stav **18**

1.1 Konvolučné neurónové siete 21

1.2 Framework pre strojové učenie a implementácia 24

2 Návrh architektúry a základných prístupov tréningu **25**

2.1 Vysokoúrovňová architektúra modelov 25

2.1.1 Autoenkóder 25

2.1.2 Klasifikátor 26

2.1.3 Autoklasifikátor 27

2.2 Používané datasety 27

2.3 Architektúra komponentov konvolučnej neurónovej siete 30

2.3.1 Architektúra pre dataset Fashion Mnist 31

2.3.2 Architektúra pre dataset Cifar Gray 33

2.4 Prístupy k tréningu 34

3 Vplyv množstva anotovaných dát na presnosť klasifikácie **38**

3.1 Tréning 38

3.1.1 Parametre, sledované veličiny, priebeh tréningu 38

3.1.2 Množstvo trénovacích dát 39

3.2 Výsledky tréningu 39

3.2.1 Zhodnotenie výsledkov 49

3.2.2 Porovnanie presnosti navrhnutých architektúr s existujúcimi architektúrami na datasete Fashion Mnist 52

3.2.3 Výsledky vybraných prístupov k tréningu na iných datasetoch a architektútrach 52

3.3 Klasifikácia na datasetoch získaných pomocou autoenkódera 55

4 Oddelenie neznámych dát **58**

4.1 Porovnanie rekonštrukčnej chyby autoenkódera na rôznych datasetoch pre označenie obrazu za neznámy 58

4.2 Rekonštrukčná chyba na správne a nesprávne klasifikovaných dátach .. 62

4.3 Vizualizácia rekonštrukcie 64

4.4 Postprocessing výstupu z klasifikátora 68

4.4.1 Zhodnotenie výsledkov postprocessingu 76
Záver 78
Zoznam použitej literatúry 79
Zoznam príloh 80

Zoznam obrázkov

Obrázok 1: Štruktúra umelého neurónu 18
Obrázok 2: Štruktúra úplne prepojene doprednej NS s jednou skrytou vrstvou 20
Obrázok 3: Príklad výpočtu konvolúcie pre *f(1,1)* 21
Obrázok 4: Vzorkovacia vrstva 22
Obrázok 5: Architektúra konvolučnej NS 23
Obrázok 6: Architektúra autoenkódera 23
Obrázok 7: Model autoenkódera 26
Obrázok 8: Model klasifikátora 26
Obrázok 9: Model autoklasifikátora 27
Obrázok 10: Ukážka obrázkov datasetu Mnist[15] 28
Obrázok 11: Ukážka obrázkov z datasetu Fashion Mnist[16] 28
Obrázok 12: Ukážka obrázkov z datasetu Cifar 10[17] 29
Obrázok 13: Porovnanie obrázkov z datasetu Cifar 10 a Cifar Gray 30
Obrázok 14: Graf vývoja rekonštrukčnej chyby prístupu 1-A na FM – BM, LM .. 40
Obrázok 15: Graf vývoja presnosti prístupu 2-KET na FM – BM, LM 41
Obrázok 16: Graf vývoja presnosti prístupu 3-KEN na FM – BM, LM 42
Obrázok 17: Graf vývoja presnosti prístupu 4-AKET na FM – BM, LM 43
Obrázok 18: Graf vývoja presnosti prístupu 5-AKEN na FM – BM, LM 44
Obrázok 19: Graf vývoja presnosti prístupu 6-DKET na FM – BM, LM 45
Obrázok 20: Graf vývoja presnosti prístupu 7-DKEN na FM – BM, LM 46
Obrázok 21: Graf vývoja presnosti prístupu 8-AK na FM – BM, LM 47
Obrázok 22: Graf normálneho rozdelenia[4] 58
Obrázok 23: Rekonštrukčné chyby na datasetoch - autoenkóder natrénovaný na M 60
Obrázok 24: Rekonštrukčné chyby na správne a neprávne klasifikovaných dátach63
Obrázok 25: Rekonštrukčné chyby na správne a neprávne klasifikovaných dátach63
Obrázok 26: Rekonštrukcia na FM, autoenkóder z prístupu 1-A 64
Obrázok 27: Rekonštrukcia na FM, autoenkóder z prístupu 4-AKET 64
Obrázok 28: Rekonštrukcia na FM, autoenkóder z prístup 8-AK 65
Obrázok 29: Rekonštrukcia autoenkóderov z M a CG na na FM 66
Obrázok 30: Rekonštrukcia na Mniste 66

Obrázok 31: Rekonštrukcia autoenkóderov na CG 67
Obrázok 32: Maximálne hodnoty vektora *X* pri aktivačnej funkcii SoftMax na FM 68
Obrázok 33: Maximálne hodnoty vektora *X* pri aktivačnej funkcii Sigmoid na FM 69
Obrázok 34: Maximálne hodnoty vektora *X* bez aktivačnej funkcie na FM 69

Zoznam tabuliek

Tabuľka 1: Enkóder ... 32
Tabuľka 2: Dekóder ... 32
Tabuľka 3: Plne prepojená vrstva – variant 1 ... 32
Tabuľka 4: Plne prepojená vrstva – variant 2 ... 33
Tabuľka 5: Enkóder ... 33
Tabuľka 6: Dekóder ... 34
Tabuľka 7: Plne prepojená vrstva ... 34
Tabuľka 8: Zhrnutie prístupov k tréningu ... 37
Tabuľka 9: Percento trénovacích dát modelov v rámci prístupov k tréningu ... 39
Tabuľka 10: Výsledok prístupu 1-A na FM – BM ... 40
Tabuľka 11: Výsledok prístupu 1-A na FM – LM ... 40
Tabuľka 12: Výsledok prístupu 2-KET na FM - BM ... 41
Tabuľka 13: Výsledok prístupu 2-KET na FM - LM ... 41
Tabuľka 14: Výsledok prístupu 3-KEN na FM - BM ... 42
Tabuľka 15: Výsledok prístupu 3-KEN na FM - LM ... 42
Tabuľka 16: Výsledok prístupu 4-AKET na FM – BM ... 43
Tabuľka 17: Výsledok prístupu 4-AKET na FM - LM ... 43
Tabuľka 18: Výsledok prístupu 5-AKEN na FM - BM ... 44
Tabuľka 19: Výsledok prístupu 5-AKEN na FM - LM ... 44
Tabuľka 20: Výsledok prístupu 6-DKET na FM – BM ... 45
Tabuľka 21: Výsledok prístupu 6-DKET na FM – LM ... 45
Tabuľka 22: Výsledok prístupu 7-DKEN na FM – BM ... 46
Tabuľka 23: Výsledok prístupu 7-DKEN na FM – LM ... 46
Tabuľka 24: Výsledok prístupu 8-AK na FM – BM ... 47
Tabuľka 25: Výsledok prístupu 8-AK na FM – LM ... 47
Tabuľka 26: Presnosť na testovacom datasete FM ... 48
Tabuľka 27: Porovnanie presnosti rôznych architektúr na FM ... 52
Tabuľka 28: Výsledky pre BM - variant 2 - prístup 2-KET ... 53
Tabuľka 29: Výsledky pre BM - variant 2 - prístup 3- KEN ... 53
Tabuľka 30: Výsledky pre BM - variant 2 - prístup 6- DKET ... 53
Tabuľka 31: Výsledky pre BM - variant 2 - prístup 7- DKEN ... 53

Tabuľka 32: Výsledky prístupu 1-A pre LM na CG 54
Tabuľka 33: Výsledky prístupu 2-KET pre LM na CG..... 54
Tabuľka 34: Výsledky prístupu 4-AKET pre LM na CG..... 54
Tabuľka 35: Výsledky prístupu 7-AKEN pre LM na CG 54
Tabuľka 36: Výsledky prístupu 8-AK pre LM na CG..... 54
Tabuľka 37: Výsledky klasifikácie na odvodenom datasete na FM, autoenkóder z 1-A 55
Tabuľka 38: Výsledky klasifikácie na odvodenom datasete na FM, autoenkóder z 4-AKET 56
Tabuľka 39: Výsledky klasifikácie na odvodenom datasete na FM, autoenkóder z AK 56
Tabuľka 40: Výsledky klasifikácie na odvodenom datasete na CG, autoenkóder z A 56
Tabuľka 41: Výsledky klasifikácie na odvodenom datasete na CG, autoenkóder z AKET 57
Tabuľka 42: Výsledky klasifikácie na odvodenom datasete na CG, autoenkóder z AK 57
Tabuľka 43: Rekonštrukčné chyby autoenkódera natrénovaného na M 59
Tabuľka 44: Pomer neznámych vstupov k celkovému počtu vstupov -autoenkóder -M..... 59
Tabuľka 45: Rekonštrukčné chyby autoenkódera natrénovaného na FM 60
Tabuľka 46: Pomer neznámych vstupov k celkovému počtu vstupov -autoenkóder -FM 60
Tabuľka 47: Rekonštrukčné chyby autoenkódera natrénovaného na FMFC 61
Tabuľka 48: Pomer neznámych vstupov k celkovému počtu vstupov -autoenkóder -FMFC 61
Tabuľka 49: Rekonštrukčné chyby autoenkódera natrénovaného na CG 61
Tabuľka 50: Pomer neznámych vstupov k celkovému počtu vstupov -autoenkóder -CG..... 61
Tabuľka 51: Rekonštrukčné chyby na testovacej množine FM 62
Tabuľka 52: Rekonštrukčné chyby na testovacej množine FM 63
Tabuľka 53: Štatistiky pre maximálne hodnoty vektora X pri aktivačnej funkcii SoftMax na FM..... 68
Tabuľka 54: Štatistika pre maximálne hodnoty vektora X pri aktivačnej funkcii Sigmoid na FM 69

Tabuľka 55: Štatistika pre maximálne hodnoty vektora *X* bez aktivačnej funkcie na FM 69

Tabuľka 56: Variant klasifikátora s aktivačnou vrstvou SoftMax na poslednej vrstve 71

Tabuľka 57: Variant klasifikátora s aktivačnou vrstvou Sigmoid na poslednej vrstve 71

Tabuľka 58: Variant klasifikátora bez aktivačnej funkcie na poslednej vrstve..... 72

Tabuľka 59: Výsledky postprocessingu pre prístup 3- KEN..... 72

Tabuľka 60: Výsledky postprocessingu pre prístup 4 -AKET 72

Tabuľka 61: Výsledky postprocessingu pre prístup 5- AKEN..... 73

Tabuľka 62: Výsledky postprocessingu pre prístup 6- DKET 73

Tabuľka 63: Výsledky postprocessingu pre prístup 7- DKEN..... 73

Tabuľka 64: Výsledky postprocessingu pre prístup 8- AK 73

Tabuľka 65: Postprocessing na FMFC, prístup 4-AKET 74

Tabuľka 66: Postprocessing na M, prístup 4-AKET 74

Tabuľka 67: Postprocessing na CG, prístup 2-KET 75

Tabuľka 68: Postprocessing na CG, prístup 4-AKET 75

Tabuľka 69: Postprocessing na CGFC, prístup 4-AKET 76

Zoznam skratiek

NS	Neurónová sieť
BM	Basic Model
CG	Cifar Gray
CGFC	Cifar Gray five classes
CGFOC	Cifar Gray five other classes
FM	Fashion Mnist
FMFC	Fashion Mnist five classes
FMFOC	Fashion Mnist five other classes
LM	Large Model
M	Mnist
A	Autoenkóder
KET	Klasifikátor enkóder trénovateľný
KEN	Klasifikátor enkóder netrénovateľný
AKET	Autoenkóder klasifikátor enkóder trénovateľný
AKEN	Autoenkóder klasifikátor enkóder netrénovateľný
DKET	Dotrénovanie klasifikátora enkóder trénovateľný
DKEN	Dotrénovanie klasifikátora enkóder netrénovateľný
AK	Autoklasifikátor
ADFP	AbsDistanceFromPredicted
SDFP	SquereDistanceFromPredicted
SA	SumAbs
SS	SumSquare
MAP	MaxArgPredictor

Úvod

Konvolučné neurónové siete predstavujú silný nástroj počítačového videnia, hlavne pre rozpoznávanie obrazu a klasifikáciu. Známe sú už od 90-tych rokov, ale až v roku 2012 zaznamenali prelom, vďaka výpočtovej sile grafických procesorov a vysokej úspešnosti pri klasifikácii obrazu. Svoje uplatnenie našli aj v oblastiach spracovania prirodzeného jazyka alebo časových radoch.

V tejto práci sa budeme zaoberať problémami, typickými pre klasifikačné úlohy založených na konvolučných neurónových sieťach. Jeden z nich je veľké množstvo anotovaných dát, ktoré sieť potrebuje, aby sa naučila klasifikovať. Nie vždy je k dispozícii dostatočne reprezentatívny dataset alebo je množstvo dát pre tréning siete nedostačujúce. Pre nové úlohy je často časovo náročné získať dostatočné množstvo reprezentatívnych dát. Prirodzene, budeme pozorovať, ako sa mení presnosť klasifikácie s meniacim sa množstvom anotovaných dát. Druhý problém, ktorý budeme riešiť, je robustnosť siete voči neznámym dátam, kde pod robustnosťou rozumieme schopnosť siete povedať, či vstup, ktorý má sieť klasifikovať, patrí do domény, na ktorom bola sieť natrénovaná. Systém má sklon k tomu pracovať s neznámymi dátami rovnako ako s tými, na ktorých bol natrénovaný, čo vedie k chybám. V práci budeme zisťovať, aký má autoenkóder vplyv v tomto procese, či pomocou neho dokážeme zvýšiť presnosť klasifikácie pri menšom množstve anotovaných dát a či sa nám podarí zvýšiť robustnosť siete voči neznámym dátam. V práci navrhneme a vyhodnotíme viacero prístupov k tréningu a spôsobov pre označenie dát za neznáme s využitím autoenkódera.

Práca je organizovaná nasledovne do 4 kapitol. V prvej kapitole je stručne popísaný princíp fungovania neurónových a konvolučných neurónových sietí. Zadefinujeme základné pojmy používané v práci ako klasifikátor a autoenkóder. V kapitole venujeme priestor frameworku pre strojové učenie, v ktorom prebiehala implementácia a povieme niečo málo o samotnej implementácii. Čitateľa možno poteší, že v celej práci nie je riadok zdrojového kódu. Zdrojové kódy sú súčasťou prílohy.

V druhej kapitole popisujeme návrh vysokoúrovňovej architektúry pre komponenty, ktoré budeme trénovať, ďalej sa pozrieme na datasety, ktoré budeme používať a potom navrhneme dve konkrétne architektúry sietí, jednu pre dataset Fashion Mnist a druhú pre dataset Cifar Gray. V závere kapitoly sú navrhnuté prístupy k tréningu klasifikátora a autoenkódera.

Kapitola 3 sa venuje tréningu, jeho parametrom, sledovaným veličinám, výsledkom jednotlivých prístupov k tréningu, porovnaniu jednotlivých prístupov k tréningu a zhodnoteniu výsledkov z pohľadu presnosti a vplyvu množstva anotovaných dát. Nájdeme tu aj porovnanie navrhnutých architektúr z pohľadu dosiahnutých výsledkov. V kapitole skúšame vylepšiť presnosť klasifikácie rozšírením datasetu pomocou autoenkódera.

V štvrtej, poslednej kapitole demonštrujeme robustnosť voči neznámym dátam. V podkapitole 4.1 používame rekonštrukčnú chybu autoenkódera na oddelenie neznámych dát. V kapitole ukážeme aj vizualizáciu rekonštrukcie autoenkódera pri rôznych prístupoch k tréningu. V podkapitole 4.4 navrhneme posprocessing výstupu z klasifikátora, na základe

ktorého vieme označiť dáta za neznáme. Ukážeme výsledky posprocessingu pre rôzne prístupe k tréningu a zhodnotíme výsledky postprocessingu a vyvodíme z neho závery.

1 Súčasný stav

V nasledujúcej kapitole bude stručne popísaný princíp fungovania neurónových sietí (NS). Na začiatok zadefinujeme základné pojmy NS ako neurón, aktivačné funkcie, učenie NS, topológie a prejdeme ku konceptom konvolučných NS, ich architektúre a vysvetlíme konkrétne typy neurónových sietí ako klasifikátor a autoenkóder, s ktorými pracujeme v rámci práce. Na záver kapitoly sa venujeme vybranému frameworku pre strojové učenie a implementácii.

Neurónové siete

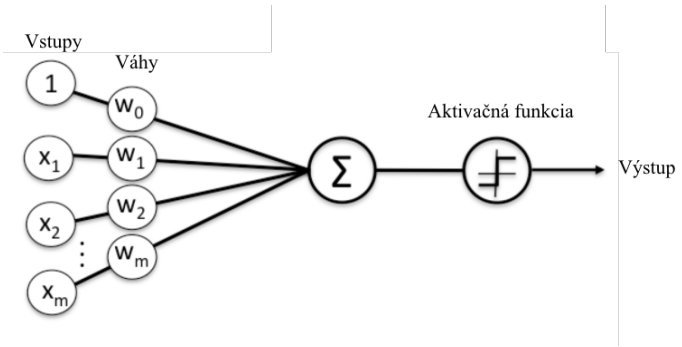
Neurónová sieť je masívne paralelný procesor, ktorý má sklon k uchovávaniu experimentálnych znalostí a ich ďalšieho využívania. Napodobňuje ľudský mozog v dvoch aspektoch:

- poznatky sú zbierané v NS počas učenia,
- medzineurónové spojenia sú využívané na ukladanie znalostí.

Toto je jedna z definícii NS, akceptovaná NS komunitou. Je zrejmé, že inšpirácia ku vzniku NS prišla z biologických systémov. Jednou z veľmi významných vlastností NS je, že svojim spôsobom je tzv. univerzálnym aproximátorom funkcií. Môže sa nám stať, že máme systém, ktorého popis je mimoriadne náročný alebo je systém natoľko zložitý, že jeho popis je skoro nemožný. Máme však dáta, ktoré do systému vstupujú, a k nim odpovedajúce výstupy. V takejto situácii, môžeme použiť vhodnú NS a pokúsiť sa ju naučiť chovať sa ako sledovaný systém pomocou trénovacích údajov. Toto je veľmi dôležitý moment, ktorý determinuje aj aplikačné uplatnenie NS v praxi. Základným elementom NS je neurón.[11]

Umelý neurón

Umelý neurón je matematická funkcia koncipovaná ako model biologického neurónu. Je základnou jednotkou v NS. Prijíma jeden alebo viac vstupov. Zvyčajne má každý vstup váhu a výstup neurónu je suma vstupov s váhami, ktorá prechádza cez nelineárnu funkciu, známu ako aktivačná funkcia, označme *f*.[13]



Obrázok 1: Štruktúra umelého neurónu

Na Obrázok 1 je štruktúra umelého neurónu. Neurónová sieť vzniká prepojením viacerých neurónov. Nech premenné $x_0..x_m$ sú vstupné hodnoty, $w_0..w_m$ sú váhy pre jednotlivé vstupy. Premennej x_0 je priradená hodnota 1, ktorá s váhou w_0 tvorí tzv. bias, $b = w_0$. Potom je počet vstupov do neurónu rovný m. Výstup neurónu potom možno popísať nasledujúcou rovnicou:

$$y = f\left(\sum_{i=1}^{m} w_i x_i + b\right) \tag{1}$$

Aktivačné funkcie

Funkcia, ktorá má na vstupe jedno reálne číslo a počíta na ňom pevne danú matematickú operáciu. Mapuje vstupy neurónu na výstup. Želané vlastnosti aktivačnej funkcie sú:

- nelineárna – zavedenie nelinearity do NS,
- spojitá,
- diferencovateľná –použitie techník používajúcich gradientové metódy,
- monotónna – chybový priestor aktivačnej funkcie je konvexný.

Sú známe aktivačné funkcie rôznych tvarov. Uvedieme prehlaď tých najpoužívanejších.[10]

Sigmoid

Výstup tejto funkcie je reálne číslo z intervalu (0,1). Hlavný problém tejto funkcie je tzv. miznúci gradient, kedy pri hodnotách blízkych 0 a 1 sú hodnoty gradientu veľmi malé, takže počas priebehu učenia nedôjde k zmene váh. Zároveň je výpočtovo náročná, kvôli exponenciálnej operácii.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

Tanh

Výstup funkcie je reálne číslo z intervalu (-1,1). Má rovnaké problémy ako funkcia Sigmoid.

$$f(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{3}$$

Rectified Linear Unit (ReLu)

Jedná sa o najčastejšie používanú aktivačnú funkciu, je jednoduchá na výpočet, nemá problém miznúceho gradientu a hlboké siete konvergujú rýchlejšie ako pri iných aktivačných funkciách.

$$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \tag{4}$$

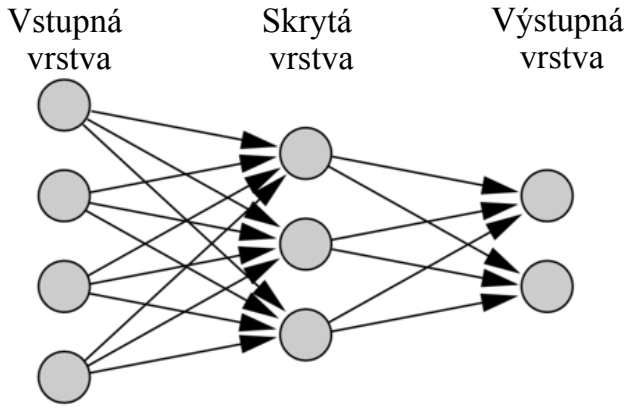
Vrstvy NS

Rozlišujeme nasledujúce vrstvy NS:

- vstupnú vrstvu, v ktorej neuróny dostavajú vstup len z vonkajšieho sveta a výstup obvykle pokračuje k ďalším neurónom NS,
- skrytú vrstvu, v ktorej neuróny dostavajú vstup z ostatných neurónov alebo aj z externého sveta cez bias a ich výstupy pokračujú ďalej do NS,

- výstupná je podobná ako skrytá vrstva, akurát je obvyklé, že výstup z tejto vrstvy vyúsťuje do externého sveta.

V nadväznosti na túto situáciu rozpoznávame aj samotné neuróny ako vstupné, skryté, a výstupné.[11]



Obrázok 2: Štruktúra úplne prepojene doprednej NS s jednou skrytou vrstvou

Topológia NS a spôsoby šírenia signálu

Vo všeobecnosti by NS mohla mať štruktúru opísateľnú ľubovoľným orientovaným grafom pomocou vrcholov (neuróny) a orientovaných hrán (prepojenia). Vlastnosti takýchto všeobecných sietí sa ťažko analyzujú, preto sú študované a analyzované najprv siete s nejakými pravidelnými štruktúrami. Pri návrhu NS vo všeobecnosti rozdeľujeme topológiu NS dvoch základných skupín:

- dopredané NS - pri týchto sa signál šíri po orientovaných prepojeniach len jedným smerom a to dopredu – viď. Obrázok 2. Sieť nazývame úplne prepojenou, ak každý neurón v každej vrstve je prepojený s každým neurónom v predchádzajúcej vrstve,
- rekurentné NS - pri rekurentných sieťach je dosť ťažké rozdelenie vrstiev a neurónov na vstupné, resp. výstupné. Majú aspoň jednu spätnú väzbu.[11]

Paradigmy učenia NS

- supervised learning - učenie s učiteľom – správne odpovede sieti poskytuje externý učiteľ,
- reinforcement learning – známkové učenie - externý učiteľ namiesto dodania správnej odpovede priebežne výsledky učenia siete hodnotí,
- unsupervised learning – samoorganizácia - sieť sa modifikuje reakciou na vstup sama, t.j. nie je prítomný žiadny učiteľ. Sieti sa poskytne iba jediné - meradlo kvality pre reprezentáciu, ktorú sa má sieť naučiť.[13]

Metóda spätného šírenia chyby

Jedná sa o metódu, ktorá sa používa na tréning NS. Táto metóda pre svoju relatívnu jednoduchosť je jednou z najrozšírenejších metód učenia NS. Bude vysvetlený len základný princíp tejto metódy, zjednodušený a bez matematických detailov.

Používa sa s niektorým optimalizačným algoritmom, napríklad metóda gradientu a jej rôzne vylepšené verzie. Táto metóda opakovane vykonáva dva kroky pre trénovanie siete – propagáciu vstupov a úpravu váh.

V prvom kroku je vstupný vektor dopredne propagovaný od vstupnej vrstvy k výstupnej a vyprodukuje výstupný vektor. Pre každý neurón vo výstupnom vektore je vypočítaná hodnota chybovej funkcie, ktorá porovná výstup siete s požadovaným výstupom. Chybová funkcia môže byť akákoľvek diferencovateľná funkcia. Potom sa vypočíta gradient chybovej funkcie vzhľadom na váhy. Spätné šírenie pošle tieto chybové hodnoty späť, začínajúc výstupnou vrstvou a končiac prvou. V druhom kroku optimalizačná metóda prijme hodnotu gradientu, aby upravila váhy tak, aby minimalizovala hodnotu chybovej funkcie. Metóda spätného šírenia chyby je považovaná za techniku učenia sa s učiteľom, pretože potrebuje znalosť želaného výstupu, aby vedela vypočítať gradient chybovej funkcie, aj keď niektoré siete bez učiteľa, ako autoenkóder, ju taktiež dokážu použiť.[1]

Batch normalizácia

Technika, ktorá vychádza z myšlienky, že vstup do siete býva často normalizovaný. Pri batch normalizácii normalizujeme výstup z vrstvy s cieľom odstrániť extrémne hodnoty aktivácií a zlepšiť reakciu siete na rôzne odchýlky vo vstupoch. Cieľom normalizácie je zrýchlenie tréningu siete, zvýšenie stability siete počas tréningu či celkové zlepšenie výkonu.

Dropout

Ide o výpočtovo nenáročnú regularizačnú techniku na náhodné odstraňovanie zvoleného percenta neurónov vo vybranej vrstve počas tréningu siete. Táto technika zabraňuje pretrénovaniu (over-fitting) – javu, keď sieť príliš verne modeluje tréningové dáta.

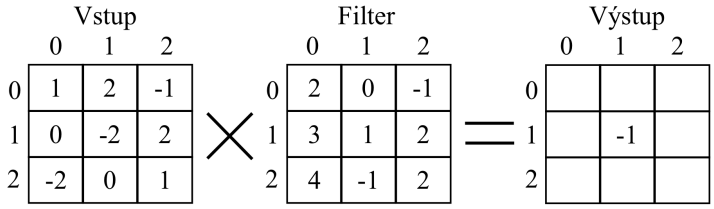
1.1 Konvolučné neurónové siete

Tento typ siete nachádza veľké uplatnenie v oblasti počítačového videnia a analýze obrazových dát. V typickej architektúre konvolučnej NS sa nachádzajú konvolučné vrstvy, vzorkovacie vrstvy a na konci siete plne prepojené vrstvy.

Ako názov napovedá, konvolučná vrstva používa matematickú operáciu nazvanú konvolúcia. Operácia diskrétnej dvojrozmernej konvolúcie má tvar:

$$g(x,y) = \sum_{n,m \in O} \sum h(x-m, y-n) f(m,n), \tag{5}$$

kde *f* je obrazová funkcia pôvodného obrazu, *g* je obrazová funkcia nového obrazu, *h* sa nazýva konvolučné jadro, alebo filter a udáva koeficienty jednotlivých bodov v okolí *O*. Na obrázku nižšie môžeme vidieť príklad výpočtu konvolúcie pre *f(1,1)*. *f(1,1)* vypočítame ako 1*2 + 2*0 + -1*-1 + 0*3 + -2*1 + 2*2 + -2*4 + 0*-1 + 1*2 = -1.

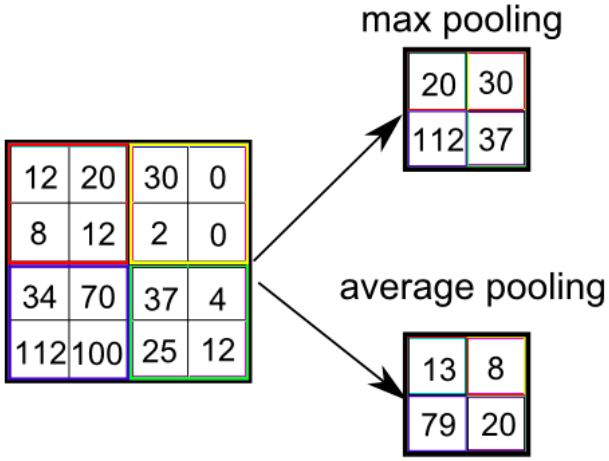


Obrázok 3: Príklad výpočtu konvolúcie pre *f(1,1)*

Nech F je počet filtrov, F určuje hĺbku konvolučnej vrstvy. Tieto filtre aplikujú konvolúciu na vstup a výstupom tejto vrstvy je potom F aktivačných máp (features). Parametre konvolučnej vrstvy sú potom počet filtrov F, rozmery filtra a krok filtra (stride). Zvyčajne sa používajú štvorcové filtre, menších rozmerov - 3x3 alebo 5x5.

Ak potrebujeme, aby bol po konvolúcii zachovaný pôvodný rozmer vstupu, je nutné použiť takzvaný padding – pridanie nulových pixelov okolo vstupu alebo zrkadlenie hraničných pixelov obrázka. Ako aktivačná funkcia je používaná funkcia ReLU, ktorá sa ukázala byť lepšia pri trénovaní hlbokých sietí ako sigmoid.

Vzorkovacia vrstva (pooling layer) je vrstva, ktorá redukuje priestorovú veľkosť vstupu z predchádzajúcich vrstiev. Výhody tejto vrstvy sú, že redukuje množstvo parametrov siete, zabraňuje pretrénovaniu a lepšie generalizuje. Táto vrstva má dva parametre a to sú krok a veľkosť filtra, rovnaké parametre ako mala konvolučná vrstva. Štandardne sa používa pooling s filtrom veľkosti 2 a krokom 2. Táto vrstva neobsahuje trénovateľné váhy, iba transformuje. Podľa spôsobu, akým sa počíta výstup rozoznávame dva druhy filtrov: average pooling a max pooling Prvý menovaný priemeruje hodnoty vo vybranej oblasti a druhý vyberá maximum. Príklad poolingu s filtrom veľkosti 2 a krokom 2 je možné vidieť na Obrázok 4.

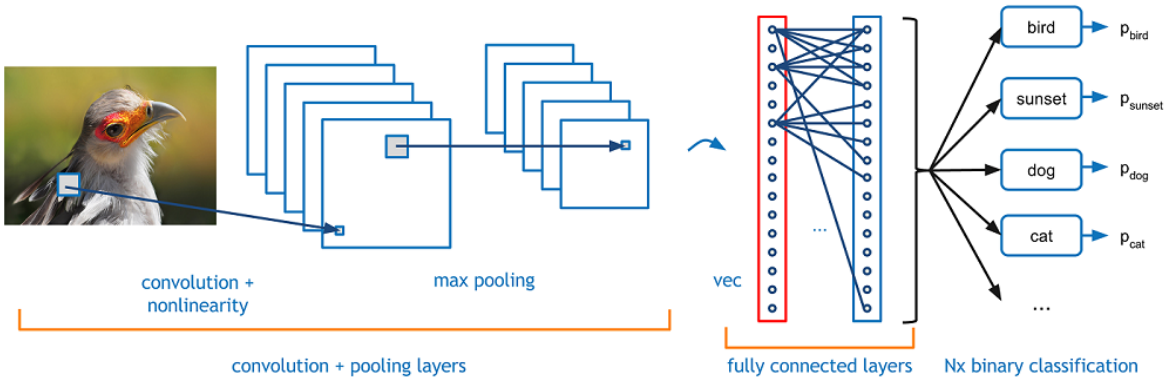


Obrázok 4: Vzorkovacia vrstva

Po konvolučných a vzorkovacích vrstvách nasleduje jedna alebo viac plne prepojených vrstiev, pričom ich výstup je zvyčajne použitý na klasifikáciu vstupných obrázkov. Ak klasifikujeme vstupy do jednej z K tried, potom je aktivačnou funkciou na poslednej vrstve funkcia SoftMax, ktorý predpis je nasledujúci:

$$s(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}, i = 1..K. \quad \textbf{(6)}$$

Funkcia normalizuje výstupy z plne prepojenej vrstvy, aby zodpovedali pravdepodobnostiam zaradenia vstupu do danej triedy. Vstup je potom sieťou zaradený do tej triedy, ktorá nadobúda najväčšiu pravdepodobnosť. Na Obrázok 5 možno vidieť príklad architektúry konvolučnej NS pre klasifikátor.

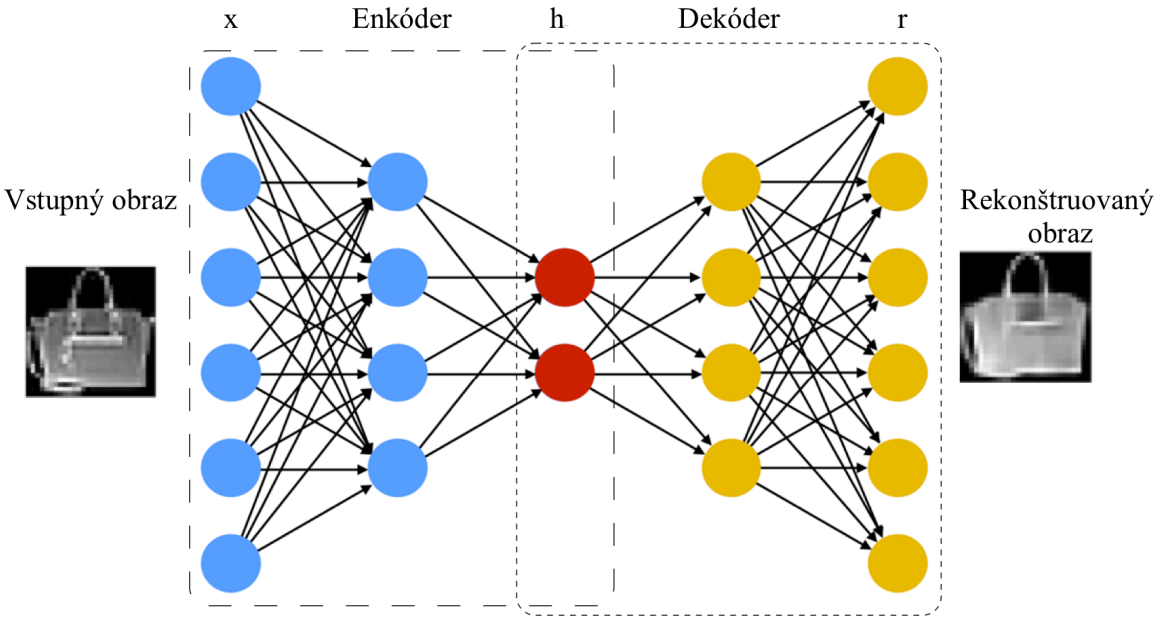


Obrázok 5: Architektúra konvolučnej NS

Autoenkóder

Autoenkóder je špeciálny typ doprednej neurónovej siete, ktorá aproximuje vstup na výstup. Na sieť sa môžeme pozrieť tak, že pozostáva z 2 častí – enkódera-kódovacej časti, ktorá predstavuje kódovaciu funkciu $h = f(x)$, a dekódera- dekódovacej časti, ktorý možno reprezentovať dekódovacou funkciou $r = g(x)$.

Autoenkodér sa trénuje tak, aby rozdiel medzi vstupom x a rekonštruovaným výstupom r bol minimálny. Dôležitý je výstup enkodéra h, ktorý pri vhodnej architektúre dokáže z dát extrahovať podstatné informácie. Táto sieť sa tradične používa pre naučenie príznakov či redukciu dimenzionality. Autoenkóder pracuje s neoznačenými dátami, jedná sa o paradigmu učenia bez učiteľa, pričom pri tréningu používa techniky pre učenie s učiteľom. Na Obrázok 6 možno vidieť štruktúru autoenkódera.[14]



Obrázok 6: Architektúra autoenkódera

1.2 Framework pre strojové učenie a implementácia

Ako framework pre strojové učenie bol vybraný Tensorflow. Tensorflow je open-source framework, vyvinutý a udržiavaný spoločnosťou Google, určený pre numerické výpočty a strojové učenie vo veľkom rozsahu. Je niekoľko frameworkow pre strojové učenie, ale Tensoflow je najpoužívanejší z nich. Dôvodom, prečo bol vybraný pre túto prácu bolo, že sme s ním už mali pozitívne skúsenosti, rozumieme, ako funguje vo vnúti a vieme v ňom implementovať a trénovať modely.

Tensorflow vývojárom umožňuje navrhnúť, postaviť a trénovať modely pre hlboké strojové učenie. Bol navrhnutý tak, aby mohol byť spúšťaný na viacerých CPU a GPU, čo je v prípade hlbokého strojového učenia nevyhnutnosť. Natrénované modely je možné dokonca spúšťať aj na mobilných zariadeniach a webových prehliadačoch. Pre používateľa ponúka prívetivé rozhranie v jazyku Python, zatiaľ čo na pozadí beží vysoko optimalizovaný kód v C++ a CUDA.[6]

Používať Tensorflow priamo môže byť výzvou a taktiež náchylné na programátorské chyby. My budeme používať pre implementáciu modelov a tréning Keras. Keras je vysoko úrovňové API pre neurónové siete, napísané v Pythone a schopné používať ako backend frameworky Tensorflow, CNTK alebo Theano. Keras bol vyvinutý s myšlienkou jednoduchého a rýchleho prototypovania modelov neurónových sietí. Podporuje konvolučné aj rekurentné neurónové siete. Od verzie Tensorflow 2.0, vydanej v septembri 2019, je Tensorflow pevne previazaný s knižnicou Keras, na požiadavku používateľov. To znamená, že Keras je lepšie udržiavaný a má lepšiu integráciu s funkciami frameworku Tensorflow.[7]

Vzhľadom na veľké množstvo tréningov rôznych modelov sme v práci navrhli vlastný aparát postavený na knižnici Keras, ktorý umožňuje flexibilne vytvárať a modifikovať modely, pripraviť trénovacie a testovacie dáta pre tréning modelov, trénovať tieto modely, zbierať údaje z tréningu modelov, vyhodnocovať tréning, vizualizovať priebeh tréningu alebo spraviť komplexný report z tréningu zameraného na konkrétnu vlastnosť.

Ako zaujímavosť môžeme uviesť, že tréning modelov bol vykonávaný na platforme Google Colaboratory. Jedná sa o voľne dostupnú cloudovú službu fungujúcu na báze Jupyter Notebooks pre vzdelávanie a výskum strojového učenia. Google Colaboratory poskytuje virtuálny stroj plne pripravený na hlboké strojové učenie a prístup k robustným grafickým kartám úplne zadarmo. Google Colaboratory používa konkrétne grafické karty Tesla K80 GPU a k nim operačný systém Linux s 12 GB RAM.[8]

2 Návrh architektúry a základných prístupov tréningu

2.1 Vysokoúrovňová architektúra modelov

Pre preskúmanie možnosti využitia autoenkódera pri klasifikácii bola navrhnutá nasledujúca vysokoúrovňová architektúra, ktorá sa skladá z troch komponentov- enkóder, dekóder a plne prepojená vrstva. Z týchto komponentov je možné vyskladať model autoenkódera, klasifikátora a autoklasifikátora, ktorý bude popísaný neskôr. Modely medzi sebou tieto komponenty zdieľajú, čo má za následok, že zdieľajú váhy. Táto vlastnosť je kľúčová pre skúmanie vlastností a bude využívaná pri tréningu týchto modelov.

V teoretickej časti sme popísali enkóder a dekóder ako časti autoenkódera a plne prepojenú vrstvu ako časť klasifikátora. Z toho vyplývajú požiadavky na vstup a výstup týchto komponentov.

Enkóder na vstupe preberie vstupný obraz a na jeho výstupe je enkódovaný obraz. Rozmer enkódovaného obrazu musí byť taký, aby ho bolo možné spätne zrekonštruovať bez paddingu, čo je úloha dekódera. Výstup enkódera je vstupom pre dekóder, ktorý z enkódovaného obrazu rekonštruuje pôvodný obraz. Enkóder a dekóder spolu tvoria autoenkóder.

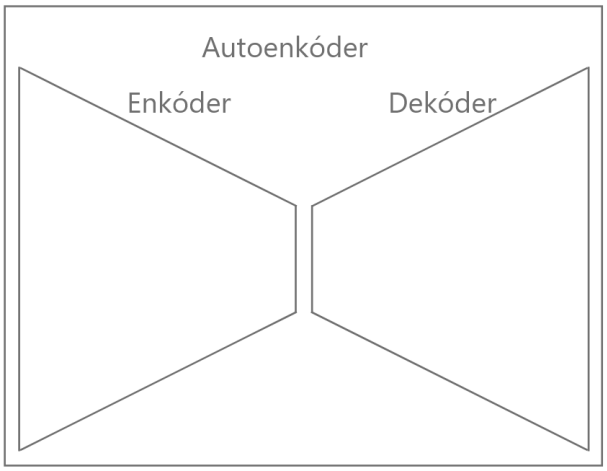
Výstup z enkódera je zároveň vstupom pre plne prepojenú vrstvu, ktorej výstupom je vektor s hodnotami v intervale (0, 1), ktorých súčet je rovný jednej. Rozmer vektora je počet tried, do ktorých obraz zaraďujeme a hodnota je pravdepodobnosť zaradenia obrazu do danej triedy. Ak spojíme enkóder a plne prepojenú vrstvu, dostaneme klasifikátor.

2.1.1 Autoenkóder

Prvý model je autoenkóder, ktorý sa skladá z enkódera a dekódera. Pri trénovaní autoenkódera bude minimalizovaná nasledujúca funkcia s názvom binary cross entropy:

$$\frac{\sum_{i=1}^{w}\sum_{j=1}^{h}(x_{ij} - o_{ij})^2}{\sum_{i=1}^{w}\sum_{j=1}^{h}{x_{ij}}^2}, \tag{7}$$

kde x_{ij} sú hodnoty vstupnej matice a o_{ij} sú hodnoty výstupnej matice, rozmery matíc sú wxh. Hodnotu tejto funkcie môžeme nazvať rekonštrukčnou chybou. Rekonštrukčná chyba je zároveň metrikou, prostredníctvom ktorej budeme posudzovať kvalitu rekonštrukcie obrazu autoenkódera. Smerodajná pre nás samozrejme bude veľkosť rekonštrukčnej chyby na testovacom datasete.



Obrázok 7: Model autoenkódera

2.1.2 Klasifikátor

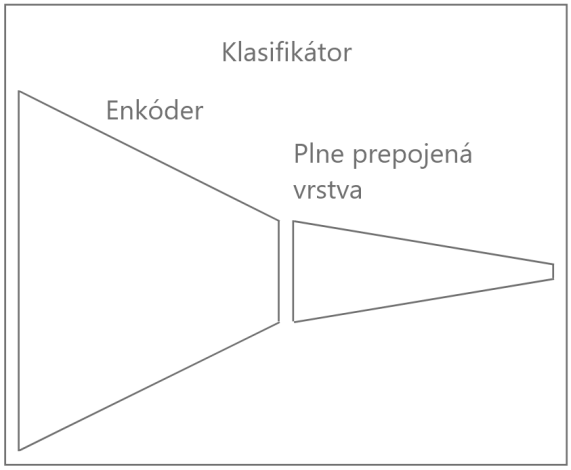
Klasifikátor sa v našej architektúre skladá z enkódera a plne prepojenej vrstvy. Pri trénovaní klasifikátora bude minimalizovaná funkcia s názvom categorical cross entropy:

$$-\sum_{k=1}^{K}(y_k * \log(y'_k)), \quad \textbf{(8)}$$

kde K je počet kategórii, do ktorých klasifikátor klasifikuje vstupy. y_k je vektor veľkosti K, hodnotu 1 nadobúda správna kategória a ostatné hodnotu 0 (tzv. one-hot encoding) a y'_k je vektor veľkosti K a ide o pravdepodobnosť zaradenia vstupu do konkrétnej kategórie, môže nadobúdať hodnoty v intervale <0,1> a vektor y'_k je výstupom plne prepojenej vrstvy. Pri určovaní, do ktorej kategórie vstup patrí sa vyberie tá kategória, ktorá nadobúda najvyššiu pravdepodobnosť vo vektore y'_k. Kvalita natrénovaného klasifikátora bude hodnotená na základe metriky, ktorú nazveme presnosť:

$$presnosť = \frac{Počet\ správnych\ predpovedí}{Celkový\ počet\ predpopovedí} \quad \textbf{(9)}$$

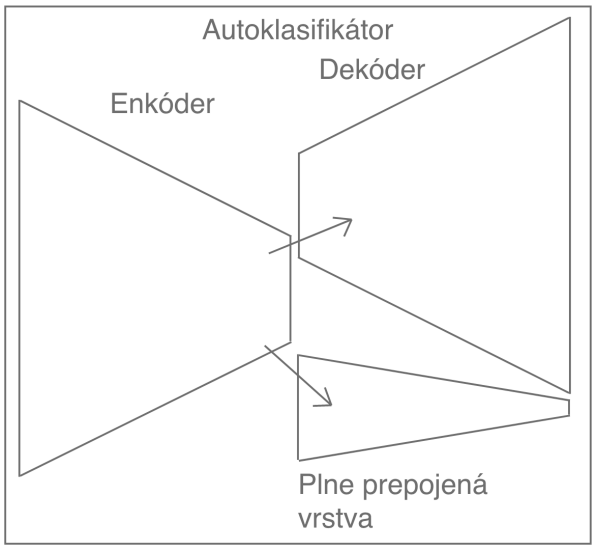
Pre nás bude podstatná presnosť na testovacom datasete. Ako optimalizátor bol zvolený RMSProp, ktorý automaticky prispôsobuje veľkosť učiaceho sa kroku pre každý parameter. Použitý optimalizátor je Adam.



Obrázok 8: Model klasifikátora

2.1.3 Autoklasifikátor

Posledný model je autoklasifikátor, ktorý sa skladá z enkódera, dekódera a plne prepojenej vrstvy. Výstup z enkódera je rovnako vstupom pre dekóder ako aj pre plne prepojenú vrstvu. To znamená, že dokážeme súčasne trénovať pri jednom prechode obrazu sieťou autoenkóder aj klasifikátor. Pri tréningu sa minimalizujú rovnaké funkcie ako pri autoenkódery a klasifikátore, s tým, že je možné určiť váhu pre minimalizačnú funkciu klasifikátora a autoenkóder pri spätnom šírení chyby. Váhy budú uvedené pri konkrétnych výsledkoch. Optimalizátory zostali zachované z autoenkódera a klasifikátora.



Obrázok 9: Model autoklasifikátora

2.2 Používané datasety

Pre úlohu klasifikácie bol už pri písaní zadania práce zvolený dataset Fashion Mnist (FM). Dôvodov, prečo bol zvolený tento dataset namiesto štandardného datasetu Mnist je viacero. Najpodstatnejší je, že Mnist je pre klasifikačné úlohy príliš jednoduchý a nezodpovedá aktuálnym klasifikačným úlohám. Už „plytké“ konvolučné neurónové siete vedia dosiahnuť na datasete Mnist presnosť nad 99%. Pri datasete Fashion mnist tomu tak nie je.[2]

Dataset Mnist (M)

Charakteristiky:

- počet trénovacích dát – 60000,
- počet testovacích dát – 10000,
- 1 vzorka má jeden čiernobiely kanál veľkosti 28 x 28,
- počet kategórii – 10,
- kategórie:
- 0 – nula, 1 – jedna, 2 – dva, 3 – tri, 4 – štyri, 5 – päť, 6 –šesť, 7 – sedem, 8 – osem, 9 – deväť,
- príprava datasetu: Normalizácia hodnôt vstupu na intervale <0,1>.

Obrázok 10: Ukážka obrázkov datasetu Mnist[15]

Dataset Fashion Mnist (FM)

Charakteristiky:

- počet trénovacích dát – 60000,
- počet testovacích dát – 10000,
- 1 vzorka má jeden čiernobiely kanál veľkosti 28 x 28,
- počet kategórii – 10,
- 0 – Tričko, 1 – nohavice, 2 – pulóver, 3 – Šaty, 4 – Kabát, 5 – Sandále, 6 – Košeľa, 7 – tenisky, 8 – Taška, 9 – Čižmy,
- príprava datasetu: Normalizácia hodnôt vstupu na intervale <0,1>.

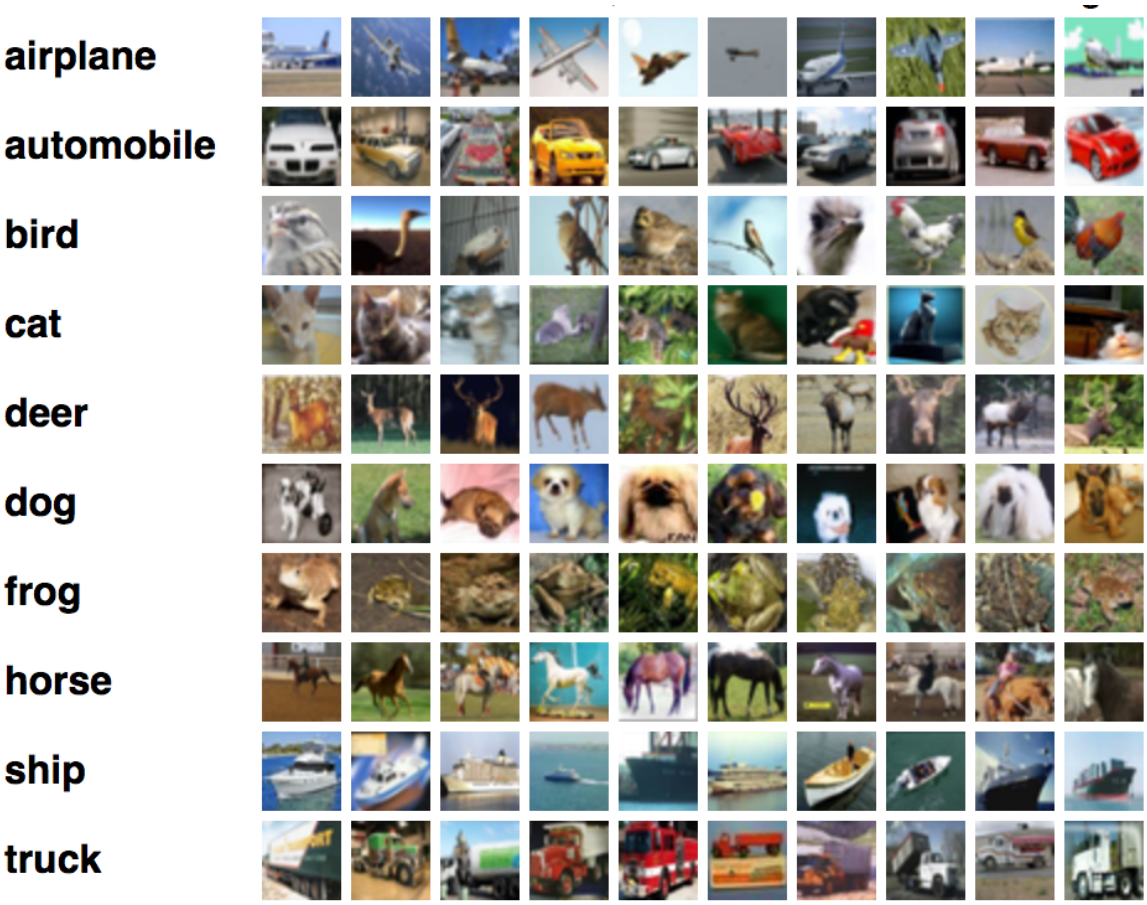
Obrázok 11: Ukážka obrázkov z datasetu Fashion Mnist[16]

Z datasetu Fashion Mnist sme vytvorili selekciou 2 datasety Fashion Mnist five classes a Fashion Mnist five other classes. Prvý menovaný obsahuje prvých 5 kategórii, druhý zvyšných 5. Počet trénovacích dát je 30000 a počet testovacích 5000. Budú použité neskôr pri jednom z navrhnutých experimentov.

Dataset Cifar 10

Charakteristiky:

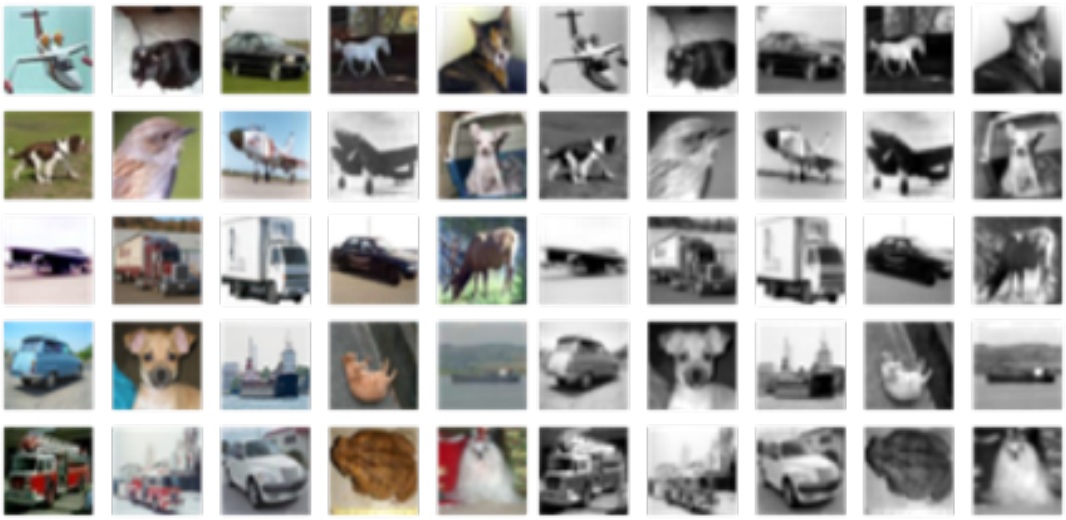
- počet trénovacích dát – 50000,
- počet testovacích dát – 10000,
- 1 vzorka má 3 RGB kanály veľkosti 32 x 32,
- počet kategórii – 10,
- 0 – lietadlo, 1 – auto, 2 – vták, 3 – mačka, 4 – jeleň, 5 – pes, 6 – žaba, 7 – kôň, 8 – loď, 9 – kamión.



Obrázok 12: Ukážka obrázkov z datasetu Cifar 10[17]

Dataset Cifar Gray (CG)

Pre potreby tejto práce bolo potrebné, aby sme získali dataset, ktorého rozmery sú 28 x 28, má iba jeden kanál a aby bol tento dataset náročnejší pre klasifikáciu ako dataset Fashion Mnist. Vybrali sme dataset Cifar 10, ktorý je náročnejší ako Fashion Mnist, ale jeho rozmery sú 32x32x3 . Zmenšili sme preto každý obraz na rozmer 28 x 28 x 3 a potom previedli na obraz s 1 kanálom. Tým sme vytvorili z datasetu Cifar 10 dataset, ktorý budeme označovať Cifar Gray. Na Obrázok 13 je možné vidieť obrázky z datasetu Cifar 10 konvertované na Cifar Gray.



Obrázok 13: Porovnanie obrázkov z datasetu Cifar 10 a Cifar Gray

Z datasetu Cifar Gray sme vytvorili selekciou 2 datasety Cifar Gray five classes a Cifar Gray five other classes. Prvý menovaný obsahuje prvých 5 kategórii, druhý zvyšných 5. Počet trénovacích dát je 25000 a počet testovacích 5000. Budú použité neskôr pri jednom z navrhnutých experimentov.

Augmentácia dát

Jedná sa o techniku, ktorá umelo vytvorí nové trénovacie dáta z existujúcich trénovacích dát. Toto sa dosahuje aplikáciou doménovo-špecifikých techník na vstupoch z trénovacích dát. Augmentácia obrazu je jedna z najznámejších druhov augmentácie dát a zahŕňa vytváranie transformovanej verzie obrazu, ktorá patrí do rovnakej kategórie ako pôvodný obraz. Transformácie, ktoré možno vykonať na obraze sú nasledovné: posuny (horizontálne alebo vertikálne), preklopenie, priblíženie, zmena mierky alebo orezanie obrazu. Zámerom je rozšíriť trénovacie dáta o nové, hodnoverné vstupy. Táto technika bude použitá neskôr pre rozšírenie datasetu.[5]

2.3 Architektúra komponentov konvolučnej neurónovej siete

Hneď na úvod by sme si dovolili upozorniť čitateľa, že našim cieľom nie je vybrať alebo vytvoriť takú architektúru, ktorá dosahuje najlepšie výsledky na danej úlohe, či už pri rekonštrukcii obrazu alebo pri klasifikácii. Architektúra bude vybraná tak, aby nebola príliš veľká- aby čas tréningu nebol dlhý a zároveň dosahovala primeranú presnosť klasifikácie na vybranom datasete v porovnaní s inými, existujúcimi architektúrami.

Pri výbere vhodnej architektúry neurónovej siete sme sa mohli vydať 2 rôznymi cestami. Mohli sme použiť existujúcu architektúru, ktorá zaznamenala úspechy pri klasifikačných úlohách, alebo vytvoriť vlastnú architektúru, ktorá je prispôsobená problému, ktorý riešime. Rozhodli sme sa pre druhý spôsob. Umožní nám to experimentovanie s hyperparametrami siete ako je počet skrytých vrstiev, veľkosť konvolučných filtrov, použité aktivačné funkcie a v neposlednom rade radikálne tým dokážeme znížiť počet váh. Pri vytváraní vlastnej architektúry sme sa inšpirovali aktuálnymi trendami v oblasti konvolučných NS, čo znamená, že používame filtre o rozmeroch 3x3, batch normalizáciu a dropout.

Pri vytváraní architektúry je naším cieľom mať čo najlepší klasifikátor pri malom počte váh. To znamená, že chceme, aby mal čo najväčšiu presnosť na testovacom datasete. Vyššie sme definovali požiadavky na vstupy a výstupy komponentov siete. Enkóder na vstupe preberá vstupný obraz, a na výstupe bude mať maticu rozmerov 7x7x počet features. Rozmer 7x7 je zvolený preto, aby sme spätne mohli rekonštruovať obraz.

Výstup enkódera je vstupom pre dekóder a plne prepojenú vrstvu. Výstupom dekódera musí byť pôvodný obraz, z toho vyplýva požiadavka na použitú aktivačnú funkciu poslednej vrstvy dekódera, musíme použiť sigmoid. Predpokladáme, že vstup do siete je normovaný na interval <0,1>. Výstup musí mať rovnaký rozmer ako vstup enkódera – jedná sa o rekonštruovaný vstup. Je nutné poznamenať, že dekóder vykonáva inverznú operáciu k enkódovaniu, preto bude architektúra a veľkosť dekódera silne závislá na architektúre enkódera.

Výstup plne prepojenej vrstvy je pravdepodobnosť zaradenia vstupu do konkrétnej kategórie, ide o vektor veľkosti rovnému počtu kategórii, ktorého suma je rovná 1. Aby sme dostali požadovaný výstup, použijeme na poslednej vrstve vrstvu, ktorá má počet neurónov rovný počtu kategórii s aktivačnou funkciou SoftMax.

Po definovaní požiadaviek na vstupy a výstupy komponentov sme mohli začať vytvárať architektúru. Postup bol nasledovný. Vytvorili sme architektúru s veľkým počtom skrytých vrstiev, ktorá dosahovala vysokú presnosť klasifikácie na testovacom datasete. Postupne sme zmenšovali počet filtrov v skrytých vrstvách a aj skryté vrstvy samotné, až kým sa nezačala zmenšovať presnosť klasifikácie. Takto dostaneme architektúru, ktorá dosahuje primeranú presnosť a počet trénovateľných váh nie je až tak veľký s porovnaním s existujúcimi sieťami, čo ušetrí čas počas tréningu.

Pôvodne bola navrhnutá iba jedna architektúra konvolučnej NS na základe datasetu Fashion Mnist, ale ako výsledky tréningu ukázali, nemala dostatočnú kapacitu pre dataset Cifar Gray. Neskôr musela byť navrhnutá architektúra na základe datasetu Cifar Gray.

2.3.1 Architektúra pre dataset Fashion Mnist

Na základe vyššie popísaného postupu sme vytvorili architektúru, ktorá dosahuje viac než priemernú presnosť na datasete Fashion Mnist pri danom počte váh. Autoenkóder má ~170000 váh a klasifikátor ~270000 váh. Na testovacej množine datasetu Fashion Mnist dosahuje architektúra presnosť okolo 94 %. Neskôr bude porovnaná klasifikačná schopnosť našej architektúry s inými, existujúcimi na datasete Fashion Mnist, kde sa ukáže efektivita našej architektúry. V tabuľkách nižšie je možné vidieť vrstvy jednotlivých komponentov- enkódera, dekódera. Vytvorili sme dva varianty plne prepojenej vrstvy. Variant 1 s veľkým počtom váh a variant 2 s extrémne nízkym počtom váh. Variant 2 vznikol viac menej pre zaujímavosť a v práci bude používaný variant 1. Variant 2 má 650 váh v porovnaní s ~200000 váhami z variantu 1. Okresať počet váh sa podarilo nahradením Dense vrstvy AveragePoolingom. Architektúru s variantom 1 pre plne prepojenú vrstvu označme potom skratkou– BM- Basic Model.

Konvolučná vrstva bude udaná v tvare: počet filtrov, veľkosť 1 filtra, aktivačná funkcia, padding.

Tabuľka 1: Enkóder

Vrstva	Výstup	Počet parametrov
Vstup	28x28x1	0
Konvolúcia- 32, 3x3, relu, pôvodná veľkosť	28x28x32	320
Batch normalizácia	28x28x32	128
Konvolúcia- 32, 3x3, relu, pôvodná veľkosť	28x28x32	9248
Batch normalizácia	28x28x32	128
MaxPooling2D (2x2)	14x14x32	0
Dropout (pravdepodobnosť dropoutu – 0.1)	14x14x32	0
Konvolúcia- 64, 3x3, relu, pôvodná veľkosť	14x14x64	18496
Batch normalizácia	14x14x64	256
MaxPooling2D (2x2)	7x7x64	0
Dropout (pravdepodobnosť dropoutu – 0.1)	7x7x64	0
Konvolúcia- 64, 3x3, relu, pôvodná veľkosť	7x7x64	36928
Batch normalizácia	7x7x64	256
		65760

Tabuľka 2: Dekóder

Vrstva	Výstup	Počet parametrov
Vstup	7x7x64	0
Konvolúcia- 64, 3x3, relu, pôvodná veľkosť	7x7x64	36928
Batch normalizácia	7x7x64	256
UpSampling2D (2x2)	14x14x64	0
Konvolúcia- 64, 3x3, relu, pôvodná veľkosť	14x14x64	36928
Batch normalizácia	14x14x64	256
Konvolúcia- 32, 3x3, relu, pôvodná veľkosť	14x14x32	18464
Batch normalizácia	14x14x32	128
UpSampling2D (2x2)	28x28x32	0
Konvolúcia- 32, 3x3, relu, pôvodná veľkosť	28x28x32	9248
Batch normalizácia	28x28x32	128
Konvolúcia –1, 3x3, sigmoid, pôvodná veľkosť	28x28x1	289
		102625

Tabuľka 3: Plne prepojená vrstva – variant 1

Vrstva	Výstup	Počet parametrov
Vstup	7x7x64	0
Flatten	3136	0
Dense (počet neurónov: 64, aktivačná f.: relu)	64	200768
Dropout (pravdepodobnosť dropoutu – 0.5)	64	0
Dense (počet neurónov: 10, aktivačná f.: softmax)	10	650
		201418

Tabuľka 4: Plne prepojená vrstva – variant 2

Vrstva	Výstup	Počet parametrov
Vstup	7x7x64	0
AveragePooling2D (7x7)	1x1x64	0
Flatten	64	0
Dropout (pravdepodobnosť dropoutu – 0.5)	64	0
Dense (počet neurónov: 10, aktivačná f.: softmax)	10	650
		650

2.3.2 Architektúra pre dataset Cifar Gray

Architektúra navrhnutá pre Fashion Mnist má nedostatočnú kapacitu pre úlohu klasifikácie na datasete Cifar 10, kde dosahuje presnosť približne 81,6 % a na datasete Cifar Gray je presnosť približne 77 % na testovacom datasete, čo považujeme za nízku presnosť. Navrhli sme teda architektúru, ktorá je kapacitne väčšia a dáva o niečo lepšie výsledky ako architektúra navrhnutá pre dataset Fashion Mnist. Na datasete Cifar 10 dokážeme s novo navrhnutou architektúrou s augmentáciou vstupov dosiahnuť 89 %, na datasete Cifar Gray bez augmentácie 84 %, s augmentáciou 85 %. Autoenkóder má ~720000 váh a klasifikátor má ~490000, čo je približne dvojnásobok s porovnaním s architektúrou pre dataset FM. Architektúru označíme skratkou LM- large model. V tabuľkách nižšie je možné vidieť vrstvy jednotlivých komponentov- enkódera, dekódera a plne prepojenej vrstvy.

Tabuľka 5: Enkóder

Vrstva	Výstup	Počet parametrov
Vstup	28x28x1	0
Konvolúcia- 32, 3x3, relu, pôvodná veľkosť	28x28x32	320
Batch normalizácia	28x28x32	128
Konvolúcia- 32, 3x3, relu, pôvodná veľkosť	28x28x32	9248
Batch normalizácia	28x28x32	128
MaxPooling2D (2x2)	14x14x32	0
Dropout (pravdepodobnosť dropoutu – 0.2)	14x14x32	0
Konvolúcia- 64, 3x3, relu, pôvodná veľkosť	14x14x64	18496
Batch normalizácia	14x14x64	256
Konvolúcia- 64, 3x3, relu, pôvodná veľkosť	14x14x64	36928
Batch normalizácia	14x14x64	256
MaxPooling2D (2x2)	7x7x64	0
Dropout (pravdepodobnosť dropoutu – 0.3)	7x7x64	0
Konvolúcia- 128, 3x3, relu, pôvodná veľkosť	7x7x128	73856
Batch normalizácia	7x7x128	512
Konvolúcia- 128, 3x3, relu, pôvodná veľkosť	7x7x128	147584
Batch normalizácia	7x7x128	512
Dropout (pravdepodobnosť dropoutu – 0.4)	7x7x128	0
		288224

Tabuľka 6: Dekóder

Vrstva	Výstup	Počet parametrov
Vstup	7x7x128	0
Konvolúcia- 128, 3x3, relu, pôvodná veľkosť	7x7x128	147584
Batch normalizácia	7x7x128	512
Dropout (pravdepodobnosť dropoutu – 0.2)	7x7x128	0
Konvolúcia- 128, 3x3, relu, pôvodná veľkosť	7x7x128	147584
Batch normalizácia	7x7x128	512
Konvolúcia- 64, 3x3, relu, pôvodná veľkosť	7x7x64	73792
Batch normalizácia	7x7x64	256
UpSampling2D (2x2)	14x14x64	0
Dropout (pravdepodobnosť dropoutu – 0.3)	14x14x64	0
Konvolúcia- 64, 3x3, relu, pôvodná veľkosť	14x14x64	36928
Batch normalizácia	14x14x64	256
UpSampling2D (2x2)	28x28x64	0
Dropout (pravdepodobnosť dropoutu – 0.3)	28x28x64	0
Konvolúcia- 32, 3x3, relu, pôvodná veľkosť	28x28x32	18464
Batch normalizácia	28x28x32	128
Konvolúcia- 32, 3x3, relu, pôvodná veľkosť	28x28x32	9248
Batch normalizácia	28x28x32	128
Konvolúcia –1, 3x3, sigmoid, pôvodná veľkosť	28x28x1	289
		435681

Tabuľka 7: Plne prepojená vrstva

Vrstva	Výstup	Počet parametrov
Vstup	7x7x128	0
Flatten	6272	0
Dense(počet neurónov: 64, aktivačná f: relu)	32	200736
Dropout(pravdepodobnosť dropoutu – 0.4)	32	0
Dense(počet neurónov: 10, aktivačná f: softmax)	10	330
		201066

2.4 Prístupy k tréningu

Existujú základné, štandardné prístupy k tréningu, ako môžeme natrénovať modely autoenkódera, klasifikátora a autoklasifikátora. Pod pojmom štandardný tréning rozumieme, že váhy jednotlivých komponentov (enkóder, dekóder, plne prepojená vrstva) sú trénovateľné, na začiatku tréningu náhodne inicializované a počas tréningu ich hodnotu nemení žiadny externý proces a jedine trénovací proces môže počas spätnej propagácie chyby upraviť váhy jednotlivých komponentov. Prístup 1 zodpovedá štandardnému tréningu autoenkódera, prístup 2 štandardnému tréningu klasifikátora a prístup 8 štandardnému tréningu autoklasifikátora. Pri ďalších prístupoch k tréningu môžu byť váhy komponentov inicializované nie náhodne, alebo predtrénovaním komponentu na inom prístupe, alebo sa váhy niektorého komponentu počas tréningu nemenia, komponent je potom netrénovateľný.

Pri návrhu ďalších prístupoch k tréningu vychádzame z predpokladu, že autoenkóder dokážeme predtrénovať na neanotovaných dátach, uvažujeme zatiaľ iba o predtrénovaní na

rovnakej dómene, na ktorej bude trénovaný klasifikátor. Klasifikátor spolu s autoenkóderom zdieľajú spoločný komponent enkóder, vďaka navrhnutej vysokoúrovňovej architektúre nebude náročné dotrénovať klasifikátor po natrénovaní autoenkódera.

Predpokladáme, že predtrénovaním enkódera pomocou autoenkódera urýchlime tréning klasifikátora, teda bude potrebný menší počet epoch na tréning klasifikátora, teda konvergencia klasifikátora bude rýchlejšia. Ďalej predpokladáme, že predtrénovaním budeme pri tréningu klasifikátora potrebovať menšie množstvo anotovaných dát pri zachovaní rovnakej presnosti. Z toho vyplýva, že pri tréningu bude potrebné meniť množstvo trénovacích dát klasifikátora. Na túto problematiku sa bližšie pozrieme v kapitole Vplyv množstva anotovaných dát na presnosť klasifikácie.

Pre overenie vyššie spomenutých predpokladov sme navrhli niekoľko prístupov, kde skúšame rôzne varianty tréningu autoenkódera a klasifikátora. Jednotlivé prístupy spolu so štandardnými prístupmi sú popísané nižšie. Pri každom prístupe je popísané, ktorý komponent sa bude trénovať a ako a čo daným prístupom sledujeme. Jednotlivé prístupy budú uvedené v tvare – Prístup číslo prístupu – textový popis prístupu – skratka prístupu – rozpísaná skratka – popis.

Prístup 1 – Autoenkóder (A – autoenkóder)

V prístupe 1 budeme štandardne trénovať autoenkóder s rôznym množstvom trénovacích dát. Tento prístup nám ukáže, ako nami navrhnutá sieť dokáže rekonštruovať obraz a získané hodnoty rekonštrukčných chýb budú slúžiť ako referenčné hodnoty pre porovnanie s ostatnými prístupmi.

Prístup 2 - klasifikátor s trénovateľnými vrstvami enkódera počas tréningu (KET – klasifikátor enkóder trénovateľný)

Jedná sa o klasický tréning klasifikátora. V tomto prístupe sú trénované všetky komponenty klasifikátora (enkóder aj plne prepojená vrstva) a tréning nie je kombinovaný s tréningom autoenkódera, s ktorým klasifikátor zdieľa spoločnú časť enkóder. Váhy enkódera sú teda trénované iba na klasifikáciu a nie sú ničím ovplyvnené (v ďalších prístupoch tomu tak nemusí byť). Z týchto dôvodov by mal tento prístup ukázať, akú maximálnu presnosť dokáže klasifikátor dosiahnuť na testovacom datasete pri 100 % trénovacích dát.

Prístup 3– klasifikátor s netrénovateľnými vrstvami enkódera počas tréningu (KEN – klasifikátor enkóder netrénovateľný)

V tomto prístupe budú váhy enkódera inicializované náhodne a budú počas tréningu netrénovateľné (zmrazené). Potom sa bude trénovať iba plne prepojená vrstva na výstupe z enkódera. Tento prístup má poukázať na to, akú vysokú presnosť dokáže dosiahnuť plne prepojená vrstva bez pomoci enkódera a o koľko sa zníži presnosť klasifikácie, ak zmrazíme váhy enkódera počas tréningu.

Prístup 4– striedavý tréning autoenkódera a klasifikátora s trénovateľným enkóderom počas tréningu klasifikátora (AKET – autoenkóder klasifikátor enkóder trénovateľný)

V tomto prístupe sledujeme, ako pomáha natrénovanie enkódera autoenkóderom pri klasifikácii. Tréning prebieha striedavo, najskôr trénujeme autoenkóder 1 epochu so 100 %

trénovacích dát a následne 1 epochu klasifikátor s rôznym percentom trénovacích dát. Pod týmto jedným cyklom rozumieme 1 epochu tréningu. Tým, že autoenkóder a klasifikátor zdieľajú spoločnú časť enkóder, tak pri tréningu si budú autoenkóder a klasifikátor tieto váhy navzájom modifikovať, každý model pre svoju úlohu.

Prístup 5– striedavý tréning autoenkódera a klasifikátora s netrénovateľným enkóderom počas tréningu klasifikátora (AKEN – autoenkóder klasifikátor enkóder netrénovateľný)

Prístup 5 je podobný prístupu 4, s tým rozdielom, že počas tréningu klasifikátora budú vrstvy enkódera netrénovateľné (zmrazené) a trénovať sa bude iba plne prepojená vrstva. Počas tréningu autoenkódera bude ale enkóder trénovateľný, z toho vyplýva, že klasifikátor bude pracovať v každej epoche s inými váhami enkódera, ktoré nebude vedieť v rámci tréningu zmeniť.

Prístup 6– dotrénovanie klasifikátora na natrénovanom autoenkóderi s trénovateľným enkóderom počas tréningu (DKET – dotrénovanie klasifikátora enkóder trénovateľný)

Zoberieme model autoenkódera z prístupu 1 natrénovanom na 100% trénovacích dát s najmenšou rekonštrukčnou chybou a ďalej trénujeme klasifikátor. Enkóder je predtrénovaný na autoenkóderi, pracujeme s predpokladom, že by malo byť v tomto prípade jednoduchšie dotrénovať klasifikátor, lebo enkóder už pozná obraz z domény, na ktorej bude klasifikátor trénovaný.

Prístup 7– dotrénovanie klasifikátora na natrénovanom autoenkóderi s netrénovateľným enkóderom počas tréningu (DKEN– dotrénovanie klasifikátora enkóder netrénovateľný)

Zoberieme model autoenkódera z prístupu 1 natrénovanom na 100% trénovacích dát s najmenšou rekonštrukčnou chybou a ďalej trénujeme klasifikátor. Enkóder je predtrénovaný na autoenkóderi a jeho váhy sa počas tréningu nebudú meniť. Trénovaná bude iba plne prepojená vrstva. Tento prístup je podobný prístupu 3, kde sme mali enkóder s náhodne inicializovanými váhami a dotrénovavala sa plne prepojená vrstva. Teraz je enkóder predtrénovaný. Tu uvidíme, ako dokáže predtrénovanie enkódera vylepšiť klasifikáciu v porovnaní s náhodne inicializovanými váhami.

Prístup 8–Autoklasifikátor (AK – autoklasifikátor)

Tento prístup tréningu je špecifický tým, že súčasne dokážeme trénovať model klasifikátora a autoenkódera na tých istých dátach. Počítame s tým, že všetky dáta sú anotované. Pred tréningom máme možnosť určiť váhu príspevku funkcie, ktorá sa minimalizuje na jednotlivých modeloch. Autoklasifikátor bude počas tréningu minimalizovať hodnotu, ktorá je váženou sumou hodnôt individuálnych minimalizačných funkcii klasifikátora a autoenkódera. To znamená, že ak určíme väčšiu váhu modelu klasifikátora, potom by mal byť model autoklasifikátora lepšie natrénovaný na úlohu klasifikácie.

V Tabuľka 8 sa nachádza prehľadné zhrnutie prístupov k tréningu. V každom riadku sa nachádza informácia o prístupe, o tom, či je komponent trénovateľný, spôsobe inicializácie váh komponentu, a model, ktorý sa bude trénovať. Ak pri informáciou, či je

komponent trénovateľný, je označenie „-“, potom je tento komponent v danom prístupe tréningu pre nás nepodstatný.

Tabuľka 8: Zhrnutie prístupov k tréningu

Komponent / Prístup	Enkóder trénovateľný / spôsob inicializácie váh	Dekóder trénovateľný / spôsob inicializácie váh	Plne prepojená vrstva trénovateľná / spôsob inicializácie váh	Trénovaný model
Prístup 1- A	Áno/ náhodná inicializácia	Áno / náhodná inicializácia	-	Autoenkóder
Prístup 2 – KET	Áno/ náhodná inicializácia	-	Áno / náhodná inicializácia	Klasifikátor
Prístup 3- KEN	Nie / náhodná inicializácia	-	Áno / náhodná inicializácia	Klasifikátor
Prístup 4 – AKET	Áno/ náhodná inicializácia	Áno / náhodná inicializácia	-	Autoenkóder
	Áno/ váhy ovplyvnené tréningom autoenkódera	-	Áno / náhodná inicializácia	Klasifikátor
Prístup 5- AKEN	Áno/ náhodná inicializácia	Áno / náhodná inicializácia	-	Autoenkóder
	Nie/ váhy ovplyvnené tréningom autoenkódera	-	Áno / náhodná inicializácia	Klasifikátor
Prístup 6- DKET	Áno / váhy inicializované autoenkóderom z prístupu 1	-	Áno / váhy inicializované autoenkóderom z prístupu 1	Klasifikátor
Prístup 7 - DKEN	Nie/ váhy inicializované autoenkóderom z prístupu 1	-	Áno / váhy inicializované autoenkóderom z prístupu 1	Klasifikátor
Prístup 8 -AK	Áno/ náhodná inicializácia	Áno/ náhodná inicializácia	Áno/ náhodná inicializácia	Autoenkóder, Klasifikátor

3 Vplyv množstva anotovaných dát na presnosť klasifikácie

V predchádzajúcej kapitole bolo navrhnutých viacero prístupov k tréningu. V nasledujúcej kapitole sa pozrieme na podmienky, za ktorých bol tréning vykonávaný – patria sem napríklad parametre tréningu, priebeh tréningu alebo množstvo anotovaných dát. Pre tréning bude použitý dataset Fashion Mnist a architektúra Basic Model. Ukážeme výsledky jednotlivých prístupov k tréningu, zhodnotíme výsledok každého tréningu a porovnáme výsledky medzi prístupmi k tréningu navzájom s dôrazom na presnosť klasifikácie. Potom ukážeme výsledky tréningu pre iné datasety ako Fashion Mnist. V závere kapitoly navrhneme a realizujeme postup, ktorým rozšírime pôvodný dataset pomocou autoenkódera, na ktorom potom natrénujeme klasifikátor za účelom zvýšenia presnosti klasifikácie.

3.1 Tréning

3.1.1 Parametre, sledované veličiny, priebeh tréningu

Pred začiatkom tréningu jednotlivých modelov je potrebné definovať okrem trénovacích dát aj parametre tréningu. Medzi parametre, ktoré budú použité pri tréningu každého modelu a pri každom prístupe bude veľkosť batchu a počet epoch.

Veľkosť batchu definuje počet vzoriek, ktoré budú prechádzať cez sieť. Veľkosť batchu bude vždy 128.

Pod jednou epochou rozumieme prejdenie všetkými trénovacími dátami sieťou. Počet epoch budeme uvádzať pri výsledkoch jednotlivých prístupov.

V rámci tréningu, po každej epoche, môžeme sledovať rôzne ukazovatele. Môže to byť hodnota funkcie/ funkcií, ktoré sa minimalizujú, rekonštrukčná chyba (pri autoenkóderi je to zároveň hodnota funkcie, ktorá sa minimalizuje) a presnosť, v závislosti od modelu, ktorý sa trénuje. Pre každý model potrebujeme určiť, ktorý z týchto ukazovateľov označíme za sledovaný parameter, ktorý svojim spôsobom môžeme považovať za najpodstatnejší. Pre autoenkóder bude sledovaný parameter hodnota rekonštrukčnej chyby na testovacom datasete, pre klasifikátor to bude presnosť na testovacom datasete. Pre autoklasifikátor to bude, rovnako ako pre klasifikátor, presnosť na testovacom datasete. Niekto by mohol namietať, prečo nesledujeme aj rekonštrukčnú chybu, a odpoveďou je, že ju nesledujeme preto, lebo skúmame využitie autoenkódera pri klasifikácii a preto nás teraz zaujíma presnosť.

Na konci každej epochy vieme vyhodnotiť hodnotu sledovaného parametra na testovacej množine. Po prejdení všetkých epoch obnovíme váhy toho natrénovaného modelu, ktorý má pre sledovaný parameter najlepšiu hodnotu. Váhy budú obnovené pre všetky tri komponenty – enkóder, dekóder aj plne prepojenú vrstvu. Uvedieme, ktorá epocha dosahuje najlepšiu hodnotu sledovaného parametra a vzhľadom k modelu z tejto epochy budú uvádzané aj ostatné ukazovatele (napr. presnosť na trénovacom datasete, rekonštrukčná chyba na trénovacom datasete, atď.).

Počas tréningu budeme používať tzv. skoré zastavenie tréningu. Ak sa sledovaný parameter po dopredu určenom počte epoch nezmení (tzv. patience), tréning predčasne ukončíme. Inak tréning skončí po dopredu určenom počte epoch (vo väčšine prípadov 100).

V niektorých situáciách to môže skrátiť tréning. Skoré zastavenie tréningu nebude použité v prístupe 4 a 5, kde sme stanovili pevný počet epoch.

3.1.2 Množstvo trénovacích dát

V rámci jednotlivých prístupov k tréningu budeme meniť množstvo trénovacích dát tak, ako je to uvedené v Tabuľka 9. Pri prístupe 1, zistíme, aký vplyv má množstva trénovacích dát, ktoré považujeme za neanotované, na rekonštrukčnú chybu autoenkódera. V prístupe 2, 3, 8 zistíme, aký vplyv má množstvo anotovaných dát na presnosť klasifikátora. V prístupe 4, 5, 6 a 7 budeme sledovať, do akej miery dokáže autoenkóder vylepšiť presnosť klasifikácie pri danom množstve anotovaných dát.

Ak netrénujeme na 100 % trénovacích dát, potom vyberáme vzorky z datasetu náhodne takýmto spôsobom. Uvediem príklad pre 10 % dát. Trénovací dataset rozdelíme na 10 rovnako veľkých častí a náhodne vyberieme jednu z nich. Každá kategória v takejto časti má približne rovnaké zastúpenie. Ak je pri modely uvedené „-“, potom sa tento model v danom prístupe netrénuje.

Tabuľka 9: Percento trénovacích dát modelov v rámci prístupov k tréningu

Prístup	% trénovacích dát		
	Autoenkóder	Klasifikátor	Autoklasifikátor
1. A	1,5,10,25,50,100	-	-
2. KET	-	1,5,10,25,50,100	-
3. KEN	-	1,5,10,25,50,100	-
4. AKET	100	1,5,10,25,50,100	-
5. AKEN	100	1,5,10,25,50,100	-
6. DKET	-	1,5,10,25,50,100	-
7. DKEN	-	1,5,10,25,50,100	-
8. AK	-	-	1,5,10,25,50,100

Pre každý prístup a množstvo trénovacích dát bol vykonaný tréning 2- krát. Ako dataset bol použitý Fashion Mnist (FM) a použité architektúry boli Basic Model (BM) a Large Model (LM).

Dataset Fashion Mnist používame z toho dôvodu, lebo sa jedná o hlavný dataset. V ďalších častiach práce použijeme modely natrénované na FM s architektúrou BM a ak to dáva zmysel, použijeme modely natrénované aj na inom datasete. Budeme používať dataset Mnist (M), ako dataset, ktorý je jednoduchší ako FM a dataset Cifar Gray (CG), ktorý je zložitejší ako FM. Neskôr budú uvedené aj výsledky pre iné datasety, ale nebudú zopakované 2 krát a percento trénovacích dát bude zvyčajne 100 %. Použitím 2 rôznych architektúr pri tréningu dokážeme vypozorovať trend, akým sa správa daný prístup k tréningu a uistíme sa, že nejde o náhodu alebo anomáliu, ktorá súvisí s konkrétnou architektúrou.

3.2 Výsledky tréningu

V tejto podkapitole budú zobrazené a popísané výsledky jednotlivých prístupov k tréningu. Vzhľadom na to, že každú kombináciu tréningových dát trénujeme 2 krát, kvôli veľkému množstvu dát budú zobrazované priemerné výsledky. Ak nebude uvedené inak, potom hodnoty výsledkov v tabuľkách vznikli ako priemer z výsledkov 2 tréningov.

Nebudeme uvádzať pojmy priemerná presnosť, ale uvedieme iba jednoducho - presnosť. Výsledky tréningu vykazujú veľkú mieru stability, líšia sa na 3. až 4. desatinnom mieste, a okrem ojedinelých prípadoch sú výsledky tréningu veľmi podobné. V grafoch budeme pre ilustráciu znázorňovať nespriemerované hodnoty z výsledkov 2 tréningov.

Prístup 1 -A

Parametre tréningu:

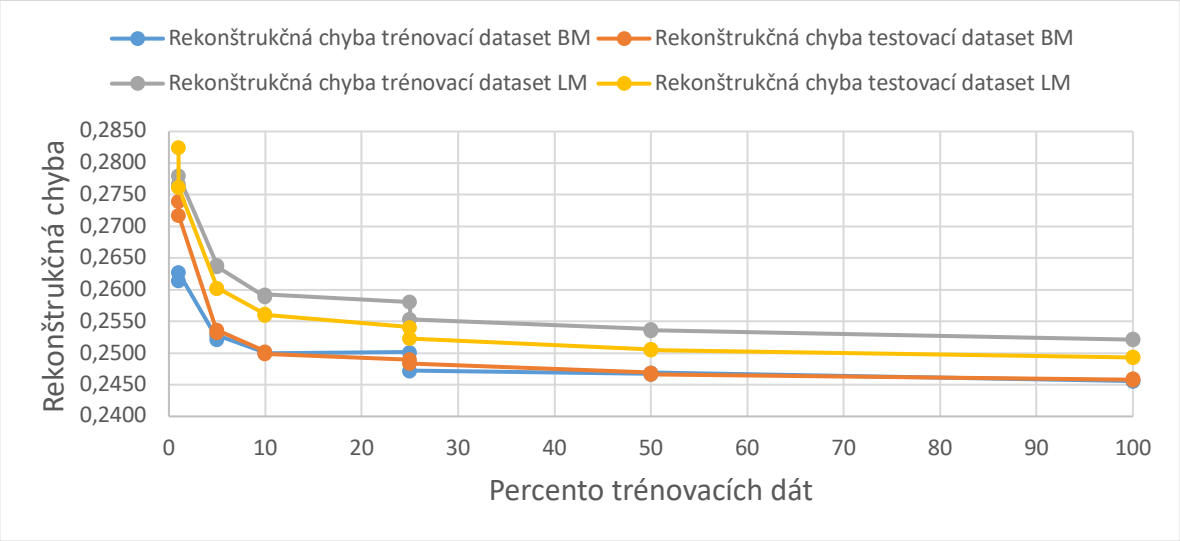
- dataset: Fashion Mnist,
- skoré ukončenie: 20,
- sledovaný parameter: rekonštrukčná chyba na testovacom datasete.

Tabuľka 10: Výsledok prístupu 1-A na FM – BM

Percento trénovacích dát	Počet epoch	Najlepšia epocha	Rekonštrukčná chyba trénovací dataset BM	Rekonštrukčná chyba testovací dataset BM
1	100	98	0,2621	0,2729
5	100	92	0,2525	0,2534
10	100	96	0,2500	0,2501
25	86,5	66,5	0,2487	0,2487
50	100	98,5	0,2468	0,2468
1000	100	99	0,2456	0,2459

Tabuľka 11: Výsledok prístupu 1-A na FM – LM

Percento trénovacích dát	Počet epoch	Najlepšia epocha	Rekonštrukčná chyba trénovací dataset LM	Rekonštrukčná chyba testovací dataset LM
1	100	97,5	0,2773	0,2793
5	100	91,5	0,2637	0,2602
10	100	93	0,2591	0,2561
25	90	78	0,2567	0,2532
50	100	94,5	0,2537	0,2506
1000	100	92,5	0,2522	0,2494



Obrázok 14: Graf vývoja rekonštrukčnej chyby prístupu 1-A na FM – BM, LM

Prístup 2 -KET

Parametre tréningu:

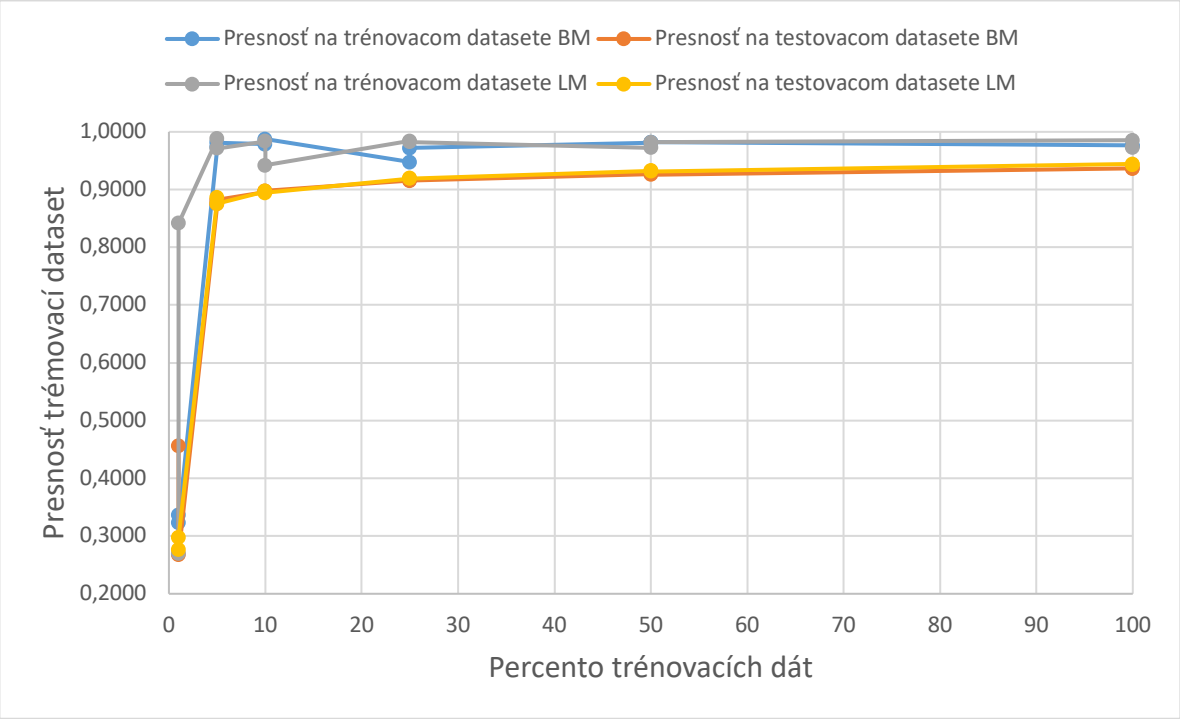
- dataset: Fashion Mnist,
- skoré ukončenie: 20,
- sledovaný parameter: presnosť na testovacom datasete.

Tabuľka 12: Výsledok prístupu 2-KET na FM - BM

Percento trénovacích dát	Počet epoch	Najlepšia epocha	Presnosť na trénovacom datasete BM	Presnosť na testovacom datasete BM
1	21	1	0,3300	0,3618
5	66	46	0,9778	0,8789
10	75,5	55,5	0,9831	0,8969
25	51	31	0,9599	0,9158
50	81	61	0,9816	0,9265
1000	67	47	0,9756	0,9369

Tabuľka 13: Výsledok prístupu 2-KET na FM - LM

Percento trénovacích dát	Počet epoch	Najlepšia epocha	Presnosť na trénovacom datasete LM	Presnosť na testovacom datasete LM
1	27	7	0,5558	0,2869
5	84,5	74,5	0,9798	0,8808
10	70,5	50,5	0,9627	0,8957
25	90	70	0,9834	0,9195
50	80	60	0,9772	0,9320
1000	83	72	0,9790	0,9438



Obrázok 15: Graf vývoja presnosti prístupu 2-KET na FM – BM, LM

Prístup 3 -KEN

Parametre tréningu:

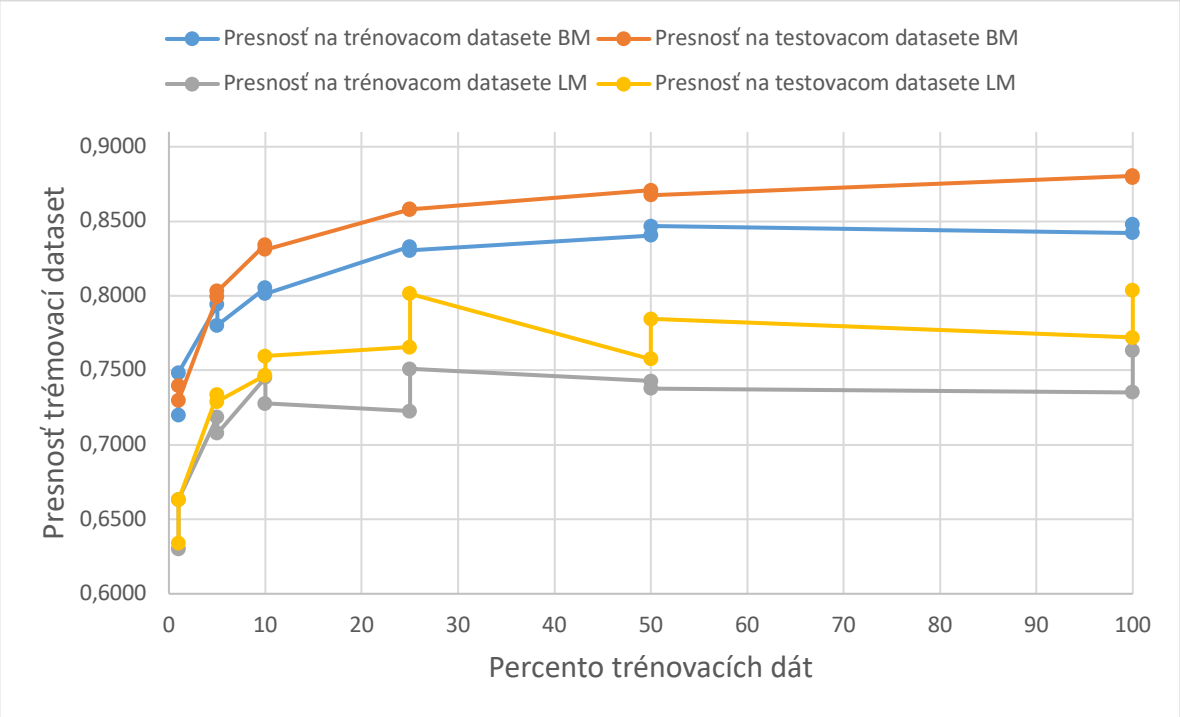
- dataset: Fashion Mnist,
- skoré ukončenie: 20,
- sledovaný parameter: presnosť na testovacom datasete.

Tabuľka 14: Výsledok prístupu 3-KEN na FM - BM

Percento trénovacích dát	Počet epoch	Najlepšia epocha	Presnosť na trénovacom datasete BM	Presnosť na testovacom datasete BM
1	100	98	0,7342	0,7348
5	100	98	0,7870	0,8012
10	100	97	0,8033	0,8327
25	100	99	0,8317	0,8580
50	100	99	0,8436	0,8692
1000	94,5	80	0,8450	0,8798

Tabuľka 15: Výsledok prístupu 3-KEN na FM - LM

Percento trénovacích dát	Počet epoch	Najlepšia epocha	Presnosť na trénovacom datasete LM	Presnosť na testovacom datasete LM
1	100	95	0,6467	0,6486
5	100	93,5	0,7132	0,7314
10	100	92	0,7363	0,7530
25	100	92,5	0,7367	0,7834
50	89,5	76,5	0,7402	0,7711
1000	77,5	57,5	0,7494	0,7878



Obrázok 16: Graf vývoja presnosti prístupu 3-KEN na FM – BM, LM

Prístup 4 - AKET

Parametre tréningu:

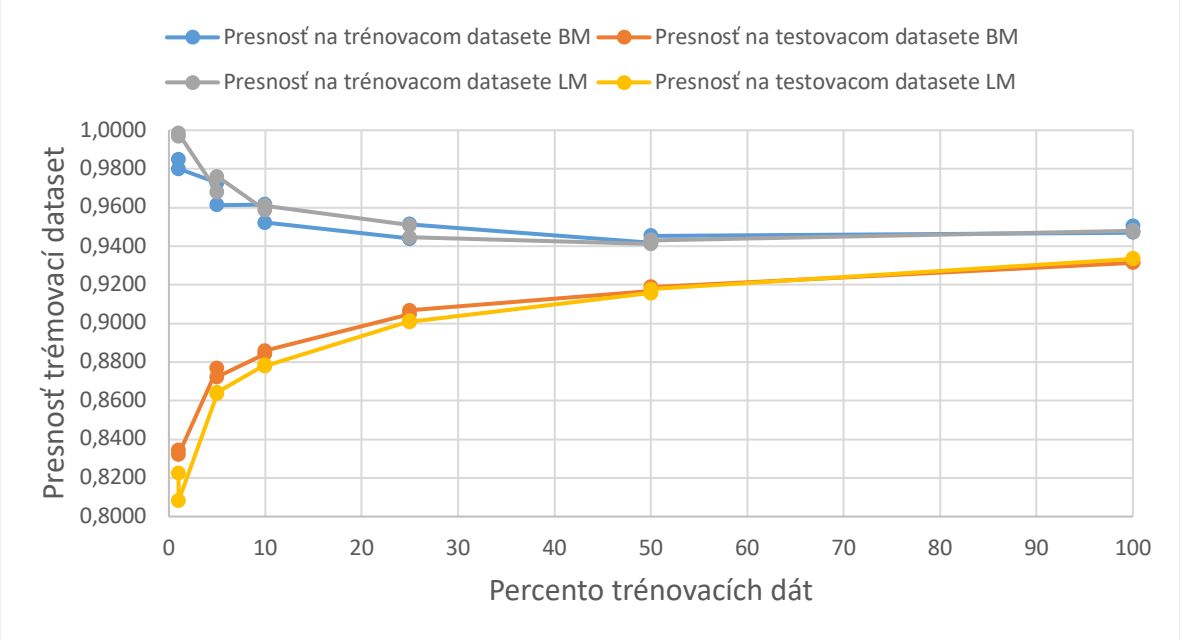
- dataset: Fashion Mnist,
- skoré ukončenie: bez skorého ukončenia, počet epoch: 150,
- sledovaný parameter: presnosť na testovacom datasete.

Tabuľka 16: Výsledok prístupu 4-AKET na FM – BM

Percento trénovacích dát klasifikátor	Rekonštrukčná chyba na testovacom datasete	Presnosť na trénovacom datasete BM	Presnosť na testovacom datasete BM
1	0,2475	0,9825	0,8333
5	0,2470	0,9670	0,8745
10	0,2468	0,9569	0,8851
25	0,2472	0,9476	0,9058
50	0,2492	0,9436	0,9178
1000	0,2520	0,9488	0,9316

Tabuľka 17: Výsledok prístupu 4-AKET na FM - LM

Percento trénovacích dát klasifikátor	Rekonštrukčná chyba na testovacom datasete	Presnosť na trénovacom datasete LM	Presnosť na testovacom datasete LM
1	0,2494	0,9975	0,8152
5	0,2513	0,9720	0,8640
10	0,2498	0,9598	0,8783
25	0,2506	0,9477	0,9011
50	0,2520	0,9421	0,9167
1000	0,2545	0,9476	0,9334



Obrázok 17: Graf vývoja presnosti prístupu 4-AKET na FM – BM, LM

Prístup 5 - AKEN

Parametre tréningu:

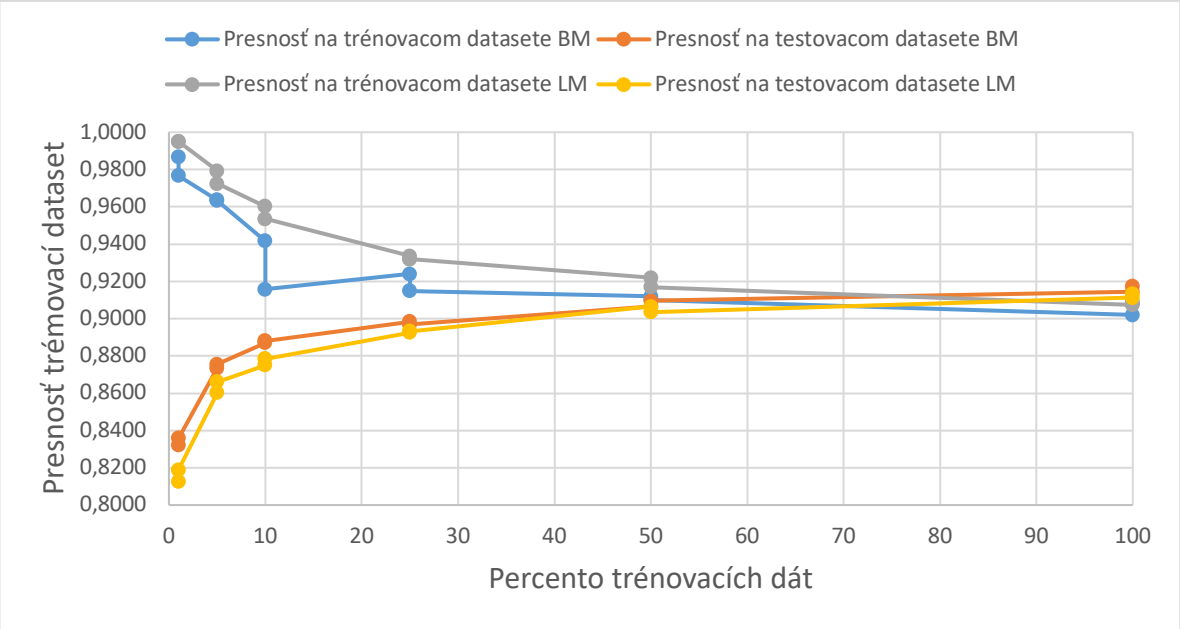
- dataset: Fashion Mnist,
- skoré ukončenie: bez skorého ukončenia, počet epoch: 150,
- sledovaný parameter: presnosť na testovacom datasete.

Tabuľka 18: Výsledok prístupu 5-AKEN na FM - BM

Percento trénovacích dát klasifikátor	Rekonštrukčná chyba na testovacom datasete	Presnosť na trénovacom datasete BM	Presnosť na testovacom datasete BM
1	0,2459	0,9817	0,8341
5	0,2461	0,9635	0,8745
10	0,2465	0,9287	0,8876
25	0,2466	0,9194	0,8976
50	0,2460	0,9111	0,9081
1000	0,2459	0,9051	0,9158

Tabuľka 19: Výsledok prístupu 5-AKEN na FM - LM

Percento trénovacích dát klasifikátor	Rekonštrukčná chyba na testovacom datasete	Presnosť na trénovacom datasete LM	Presnosť na testovacom datasete LM
1	0,2495	0,9950	0,8157
5	0,2494	0,9758	0,8632
10	0,2503	0,9568	0,8768
25	0,2508	0,9327	0,8929
50	0,2493	0,9194	0,9050
1000	0,2496	0,9082	0,9123



Obrázok 18: Graf vývoja presnosti prístupu 5-AKEN na FM – BM, LM

Prístup 6 - DKET

Parametre tréningu:

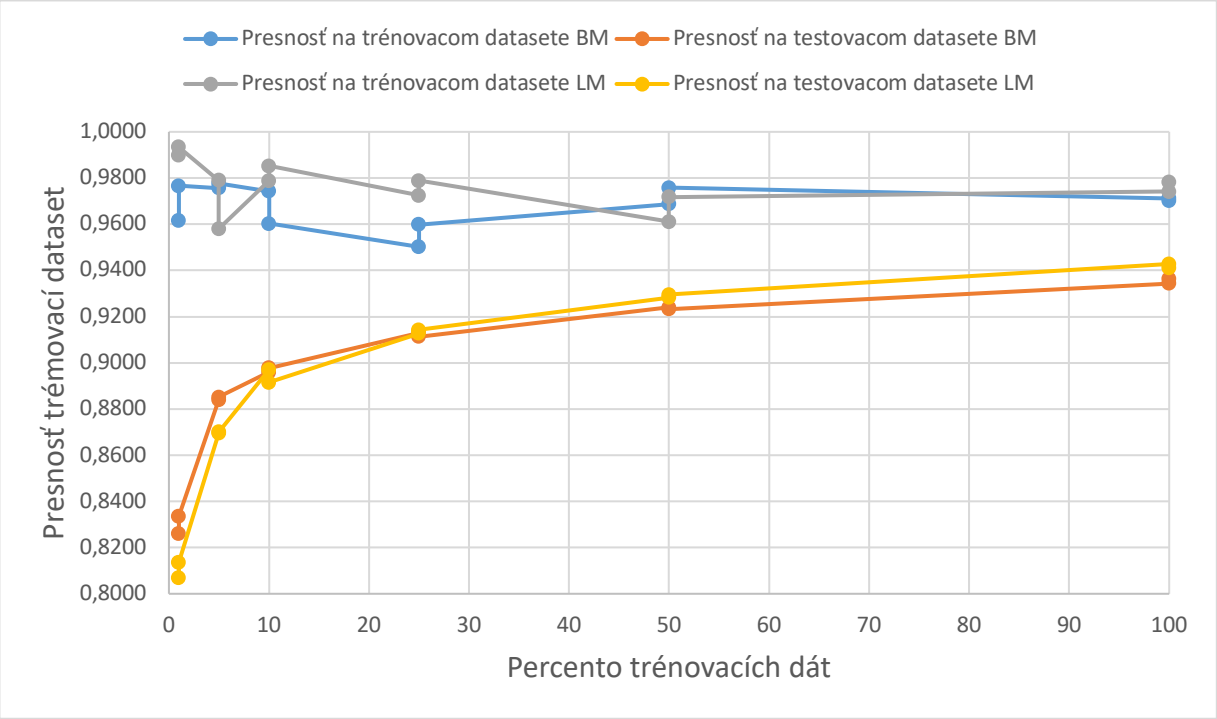
- dataset: Fashion Mnist,
- skoré ukončenie: 20,
- sledovaný parameter: presnosť na testovacom datasete.

Tabuľka 20: Výsledok prístupu 6-DKET na FM – BM

Percento trénovacích dát	Počet epoch	Najlepšia epocha	Presnosť na trénovacom datasete BM	Presnosť na testovacom datasete BM
1	54,5	37,5	0,9692	0,8298
5	60	48	0,9767	0,8845
10	56,5	38	0,9673	0,8968
25	46,5	26,5	0,9550	0,9122
50	60	53	0,9723	0,9237
1000	60	57,5	0,9707	0,9355

Tabuľka 21: Výsledok prístupu 6-DKET na FM – LM

Percento trénovacích dát	Počet epoch	Najlepšia epocha	Presnosť na trénovacom datasete LM	Presnosť na testovacom datasete LM
1	60	58,5	0,9917	0,8103
5	52	39	0,9685	0,8698
10	60	53	0,9820	0,8943
25	60	49	0,9756	0,9135
50	55	36	0,9665	0,9289
1000	60	55,5	0,9762	0,9420



Obrázok 19: Graf vývoja presnosti prístupu 6-DKET na FM – BM, LM

Prístup 7 - DKET

Parametre tréningu:

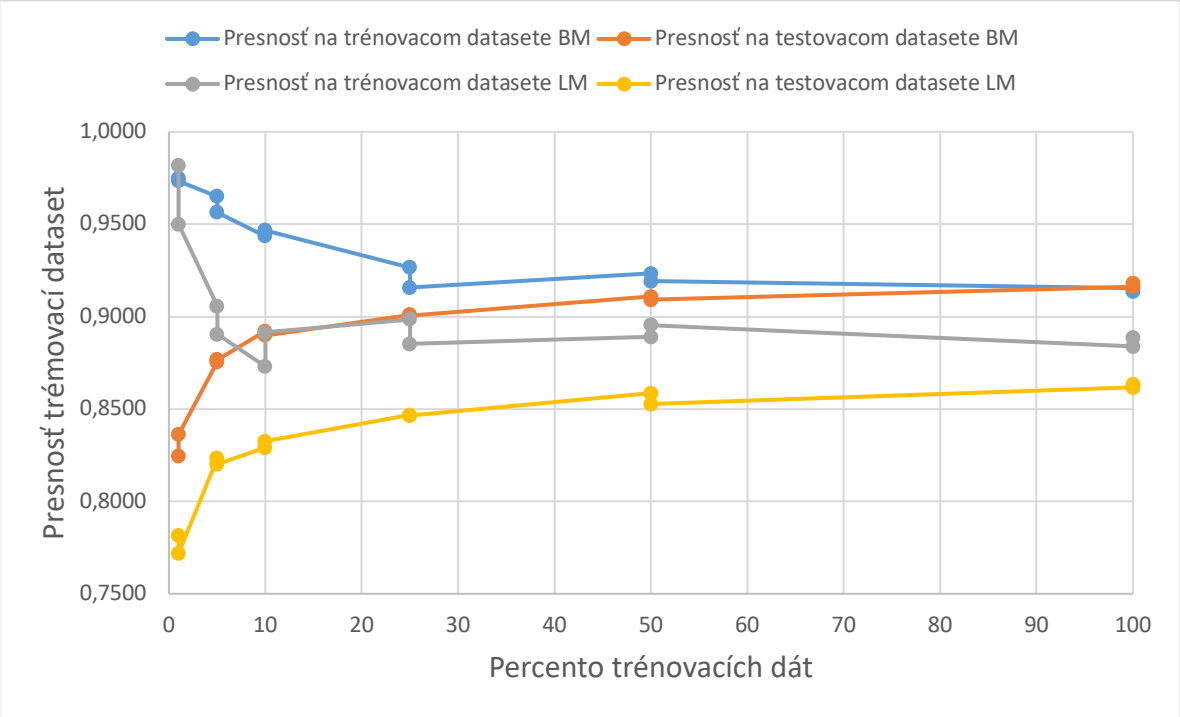
- dataset:Fashion Mnist,
- skoré ukončenie: 20,
- sledovaný parameter: presnosť na testovacom datasete.

Tabuľka 22: Výsledok prístupu 7-DKEN na FM – BM

Percento trénovacích dát	Počet epoch	Najlepšia epocha	Presnosť na trénovacom datasete BM	Presnosť na testovacom datasete BM
1	60	44	0,9742	0,8304
5	59	46,5	0,9607	0,8761
10	60	48,5	0,9451	0,8910
25	57,5	45	0,9211	0,9008
50	60	53	0,9212	0,9100
1000	60	54,5	0,9144	0,9172

Tabuľka 23: Výsledok prístupu 7-DKEN na FM – LM

Percento trénovacích dát	Počet epoch	Najlepšia epocha	Presnosť na trénovacom datasete LM	Presnosť na testovacom datasete LM
1	55,5	35,5	0,9658	0,7767
5	37,5	17,5	0,8980	0,8217
10	36	16	0,8823	0,8309
25	52,5	41	0,8919	0,8466
50	55,5	41,5	0,8922	0,8556
1000	53	34,5	0,8861	0,8625



Obrázok 20: Graf vývoja presnosti prístupu 7-DKEN na FM – BM, LM

Prístup 8 - AK

Parametre tréningu:

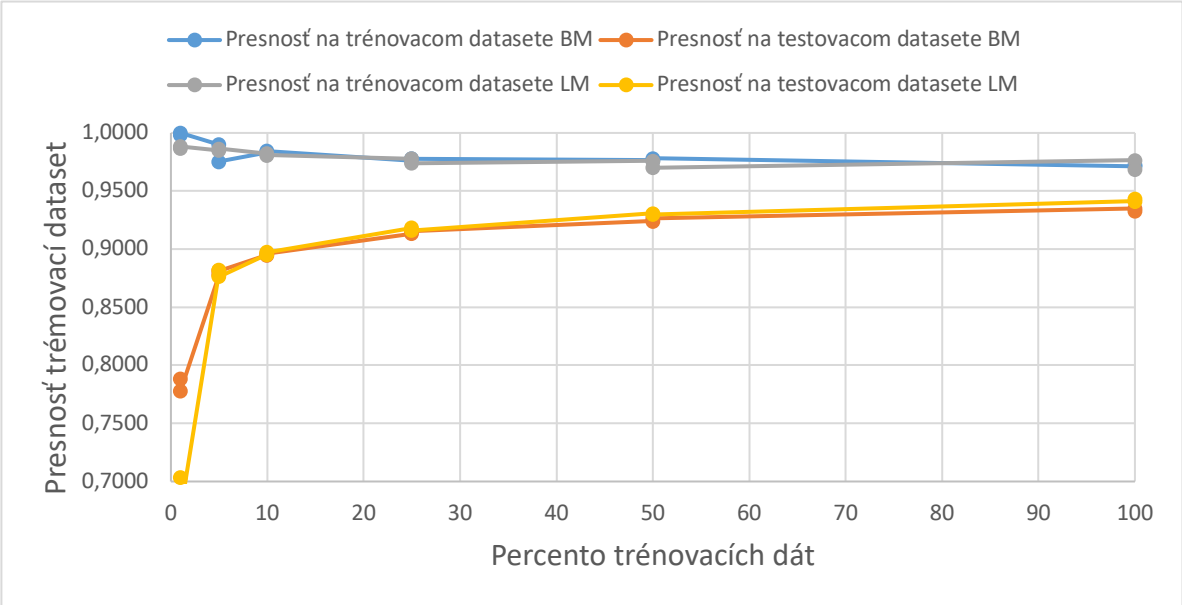
- dataset: Fashion Mnist,
- skoré ukončenie: 20, váha výstupu autoenkódera: 1, váha výstupu klasifikátora: 2,
- sledovaný parameter: presnosť na testovacom datasete.

Tabuľka 24: Výsledok prístupu 8-AK na FM – BM

Percento trénovacích dát	Rekonštrukčná chyba na testovacom datasete	Presnosť na trénovacom datasete BM	Presnosť na testovacom datasete BM
1	0,29991475	0,999166665	0,7832
5	0,2697382	0,982666665	0,87995
10	0,26781915	0,983916675	0,89565
25	0,26714401	0,976899975	0,9143
50	0,26270991	0,97731665	0,9254
1000	0,262881475	0,971741685	0,93395

Tabuľka 25: Výsledok prístupu 8-AK na FM – LM

Percento trénovacích dát	Rekonštrukčná chyba na testovacom datasete	Presnosť na trénovacom datasete LM	Presnosť na testovacom datasete LM
1	0,333116145	0,98750002	0,6889
5	0,274894415	0,98566665	0,8792
10	0,274356275	0,9815	0,8964
25	0,282562255	0,9759	0,9173
50	0,27028051	0,97303335	0,9303
1000	0,26908915	0,9725	0,94225



Obrázok 21: Graf vývoja presnosti prístupu 8-AK na FM – BM, LM

Tabuľka 26: Presnosť na testovacom datasete FM

Prístup k tréningu	2-KET		3- KEN		4-AKET		5- AKEN		6-DKET		7- DKEN		8-AK	
Architektúra	BM	LM	BM	LM	BM	LM	BM	LM	BM	LM	BM	LM	BM	LM
% trénovacích dát	Presnosť na testovacom datasete													
1	0,456	0,277	0,74	0,634	0,835	0,822	0,8323	0,813	0,826	0,807	0,8246	0,782	0,788	0,704
1	0,268	0,297	0,73	0,663	0,832	0,808	0,8358	0,819	0,834	0,814	0,8361	0,772	0,778	0,674
5	0,876	0,886	0,799	0,734	0,877	0,864	0,8734	0,86	0,884	0,87	0,8755	0,823	0,878	0,882
5	0,882	0,876	0,803	0,729	0,872	0,864	0,8755	0,866	0,885	0,87	0,8767	0,82	0,882	0,876
10	0,896	0,897	0,834	0,747	0,884	0,879	0,8871	0,875	0,896	0,897	0,8921	0,829	0,895	0,896
10	0,898	0,894	0,831	0,759	0,886	0,878	0,888	0,879	0,898	0,892	0,8899	0,833	0,896	0,897
25	0,916	0,919	0,858	0,765	0,905	0,901	0,8984	0,893	0,913	0,913	0,901	0,847	0,913	0,918
25	0,916	0,92	0,858	0,801	0,907	0,901	0,8968	0,893	0,911	0,914	0,9005	0,846	0,915	0,916
50	0,927	0,933	0,871	0,758	0,917	0,916	0,9065	0,906	0,924	0,928	0,9108	0,859	0,924	0,931
50	0,926	0,932	0,868	0,785	0,919	0,918	0,9096	0,904	0,923	0,93	0,9091	0,853	0,927	0,93
100	0,936	0,944	0,88	0,772	0,932	0,933	0,9143	0,911	0,934	0,943	0,9162	0,862	0,935	0,941
100	0,937	0,943	0,879	0,804	0,932	0,933	0,9172	0,913	0,937	0,941	0,9182	0,863	0,933	0,943

3.2.1 Zhodnotenie výsledkov

V tejto časti vyvodíme závery z výsledkov jednotlivých prístupov k tréningu. Pre prehľadnosť uvádzame v Tabuľka 26 presnosť na testovacom datasete FM pre všetky prístupy k tréningu, v rámci ktorých sa trénuje klasifikátor (2 - 8). Uvádzame hodnoty z 2 tréningov, nie ich priemer, z dôvodu, že sa na tieto konkrétne hodnoty budeme odkazovať.

Prístup 1 – A

V tomto prístupe trénujeme autoenkóder. Celkom prekvapivé je, že BM dosahuje nižšiu rekonštrukčnú chybu ako LM. Je zaujímavé si všimnúť, že už pri malom množstve trénovacích dát – 1 %, je rekonštrukčná chyba celkom nízka – ~0,28, ak pri 100% trénovacích dát dostaneme rekonštrukčnú chybu ~0,24. Ak by sme ale vizualizovali výsledok rekonštrukcie, tak by sme prišli na to, že pri 1% dát sa nedá hovoriť o kvalitnej rekonštrukcii. V podkapitole 4.1 sa pozrieme na rekonštrukčné chyby oveľa bližšie, kde budú znázornené aj rekonštruované vstupy. Zatiaľ stačí vedieť, že pri 100 % trénovacích dát je rekonštrukčná chyba na testovacom datasete 0,246, čo bude slúžiť ako referenčná hodnota pre rekonštrukčnú chybu pre ďalšie výsledky prístupy.

Prístup 2 – KET

Prístup 2 odpovedá na otázku, akú presnosť dokáže dosiahnuť klasifikátor pri štandardnom tréningu. Bude referenčným prístupom pre presnosť klasifikátora, s ktorým budú porovnávané zvyšné prístupy k tréningu.

Ak sa pozrieme na výsledky z pohľadu množstva trénovacích dát, pri extrémne nízkom množstve trénovacích dát – 1 %, čo je 600 vzoriek, môžeme povedať, že klasifikátor nedokáže vôbec klasifikovať, presnosť je u oboch modeloch – BM aj LM nízka – ~30%. S rastúcim množstvom trénovacích dát rapídne narástla presnosť. Pri 5 % je presnosť ~88% a pri 100% to je až ~94%. Na dosiahnutie presnosti ~94% nám pri 2 zo 4 tréningov stačilo menej ako 50 epoch, čo je dôležité číslo pre porovnanie s ďalšími prístupmi k tréningu. Ak porovnáme architektúru BM a LM, je prirodzené, že LM dosahuje vyššie presnosť ako BM, lebo má takmer 3-násobnú kapacitu. Ak by sme to mali ohraničiť, o koľko je LM lepšia ako BM, je to maximálne o 1 %.

Prístup 3 – KEN

Pri tomto prístupe je komponent enkóder náhodne inicializovaný a trénuje sa iba komponent plne prepojená vrstva. Môžeme si všimnúť, že medzi niektorými výsledkami tréningu, napríklad pri LM pri 25 % a 100 % trénovacích dát sa presnosť líši o viac ako 3 %. Tento jav je spôsobený tým, že komponent enkóder je inicializovaný náhodne a tým sa do plne prepojenej vrstvy dostanú diametrálne rozličné features.

Ďalšou zaujímavou skutočnosťou je, že BM dosahuje výrazne vyššiu úspešnosť ako LM. Hovoríme takmer o 10 %. Vieme že LM je väčší model ako BM, Enkóder z BM má 4 konvolučné vrstvy a enkóder z LM ich má až 6. Potom náhodne inicializovaný enkóder z LM má väčšiu kapacitu ako enkóder z BM, čo je ale v tomto prípade kontraproduktívne, lebo dochádza k pretrénovaniu sa, čo vysvetľuje nižšiu presnosť LM.

Ak porovnáme prístup 3-KEN s prístupom 2-KET, lepšiu presnosť dosahuje prístup 3-KEN iba pri 1 % trénovacích dát. V ostatných prípadoch je príspevok trénovateľného

enkódera na presnosť veľký, pri architektúre BM to je od 5,6 % do 7,9 % a na LM od 11 % do 17,5 %. Pri 100 % trénovacích dát dosahuje prístup 2-KET na BM ~94 % a prístup 3-KET iba ~87 % na BM. Môžeme potom povedať, že samotná plne prepojená vrstva s náhodnými features z enkódera nedosahuje dostatočnú presnosť, a tento prístup nie je vhodný pre klasifikáciu. Môžeme z neho ale ľahko zistiť, ako pomáha trénovateľný enkóder pri klasifikácii a bude porovnávaný s ďalšími prístupmi.

Prístup 4 – AKET

V rámci tohoto a ďalších prístupov – 5,6 a 7 dokážeme priamo vypozorovať na základe výsledkov, ako pôsobí autoenkóder na presnosť klasifikátora. Predkladáme hypotézu, že ak sa autoenkóder učí počas tréningu features, ktoré sú potrebné pre klasifikáciu, potom by mal klasifikátor dosahovať pri malom množstve trénovacích dát o niečo vyššiu presnosť ako klasifikátor z prístupu 2-KET, vzhľadom na to, že autoenkóder sa trénuje na 100 % trénovacích dát na tom istom datasete, na ktorom je vykonaná klasifikácia.

V tomto prístupe táto skutočnosť platí iba pri 1 % trénovacích dát, kde autoenkóder výrazne pomáha, na testovacom datasete dosahuje presnosť nad 80 %. Pri vyššom percente trénovacích dát tomu tak nie je a presnosť je nižšia ako pri prístupe 2-KET. Pri BM je to priemerne o 0,8 % a pri LM to je o 1,56 %. Tento prístup dosahuje horšie výsledky ako štandardný tréning. Po prejdení všetkých prístupov k tréningu sa k tejto hypotéze vrátime.

Počas tohto tréningu je trénovaný aj autoenkóder na 100 % trénovacích dát. Ak porovnáme rekonštrukčnú chybu s rekonštrukčnou chybou z prístupu 1-A, rekonštrukčná chyba je vyššia pri oboch architektúrach, lebo po natrénovaní autoenkódera sa ešte trénuje klasifikátor, ktorý mení váhy komponentu enkódera, ktorý zdieľa s autoenkóderom.

Prístup 5 – AKEN

V tomto prístupe je počas tréningu klasifikátora enkóder netrénovateľný. Klasifikátor sa teda bude musieť naučiť robiť klasifikáciu iba na základe features, ktoré mu prídu z enkódera, ktoré sú po každej epoche iné. Kvôli tomu je počet epoch stanovený na 150, viac, ako u ostatných prístupov. Ak sa vrátime k hypotéze z prístupu 4-AKET, potom hypotéza platí iba pri 1 % trénovacích dát, kde je presnosť väčšia ako pri prístupe 2-KET, v ostatných prípadoch je presnosť nižšia, konkrétne pri BM priemerne o 1,42 % a pri LM o 2,43 %, čo je ešte menej ako pri prístupe 4-AKET. Z pohľadu rekonštrukčnej chyby autoenkódera je rovnaká ako pri 100 % trénovacích dát z prístupu 1-A, lebo klasifikátor počas tréningu neovplyvňuje váhy autoenkódera.

Prístup 6 – DKET

Počet epoch sme znížili na 60 s pôvodných 100. Klasifikátor by mal skôr konvergovať, lebo komponent enkóder je predtrénovaný na autoenkóderi. Ak platí tvrdenie, že autoenkóder sa učí features potrebné pre klasifikáciu, potom by mal byť menší počet epoch dostatočný.

Ak sa vrátime k hypotéze z prístupu 4-AKET, hypotéza platí iba pri 1 % trénovacích dát, kde predtrénovanie autoenkóderom zvýši presnosť nad 80 %. Pri ostatných percentách trénovacích dát dosahuje klasifikátor podobnú presnosť ako klasifikátor z prístupu 2-KET. Zároveň nemožno povedať, že sieť konverguje výrazne skôr, ako pri prístupe 2-KET. Z toho

nám vyplýva, že predtrénovanie autoenkóderom a následné dotrénovanie klasifikátora neprináša žiadnu výhodu v porovnaní so štandardným prístupom, iba pri extrémne nízkom množstve trénovacích dát.

Prístup 7 – DKEN

Výsledky tohoto prístupu nám zo všetkých ostatných prístupov najlepšie odpovedajú na otázku, či sa autoenkóder učí features, ktoré sú potrebné pre klasifikáciu a v akej miere. V prístupe 3-KEN sme videli, akú presnosť dosahuje klasifikátor, ak sú váhy enkódera náhodne inicializované. V tomto prístupe je enkóder predtrénovaný na autoenkóderi na 100 % trénovacích dát, features, ktoré sa dostanú do plne prepojenej vrstvy, nie sú náhodné, ale určené pre dekóder na rekonštrukciu obrazu.

Po predchádzajúcich výsledkoch sa dalo očakávať, že tento prístup dosahuje lepšiu presnosť iba na 1 % trénovacích dát v porovnaní s prístupom 2-KET – na BM ~82 % a LM ~78. Dôležitejšie je pre nás ale porovnanie s prístupom 3-KEN. Na BM je presnosť priemerne vyššia o 5,83 % a na LM o 8,64 %, čo je relatívne nízke zvýšenie presnosti na to, že sme natrénovali enkóder na autoenkóderi. Môžeme povedať, že features z autoenkódera sú lepšie, ako náhodné inicializované, ale sú horšie, ako features, ktoré sú potrebné pre klasifikáciu. Potom features z autoenkódera nie sú relevantné pre klasifikáciu.

Pri porovnaní BM a LM, BM dosahuje výrazne vyššiu presnosť ako LM. Presnosť LM je nižšia v porovnaní s BM, na 100 % trénovacích dát, BM dosiahne ~91,7 % a LM ~86,2 %. Dôvod je podobný ako pri prístupe 3-KEN. LM má veľkú kapacitu a sa pretrénuje sa.

Výsledky tohoto prístupu a následné porovnanie s inými prístupmi naznačujú, že autoenkóder sa neučí všetky features, ktoré sú potrebné pre klasifikáciu.

Prístup 8 – AK

Jedná sa o veľmi zaujímavý spôsob tréningu, kde súčasne trénujeme autoenkóder a klasifikátor na tej istej trénovanej množine. Pri prístupe 4-AKET a 5-AKEN sme trénovali autoenkóder a klasifikátor striedavo, teraz to spravíme v jednom kroku. Pri porovnaní rekonštrukčnej chyby s prístupom 1-A, je rekonštrukčná chyba vysoká, lebo pri propagácii chyby sieťou má väčšiu váhu klasifikátor ako autoenkóder. Pri vizualizácii uvidíme, ako takto natrénovaný model dokáže rekonštruovať obraz.

Presnosť klasifikátora je lepšia iba pri 1 % trénovacích dát, na BM ~78 % a na LM ~70 %, čo je potešiteľné, lebo pri ostatných prístupoch sme mali autoenkóder natrénovaný na 100 % dátach a úspešnosť nebola výrazne vyššie, ak sa pozrieme na výsledky ostatných prístupov. Pri vyšších percentách trénovacích dát je presnosť približne rovnaká, mierne nižšia ako pri prístupe 2-KET.

Celkové zhodnotenie výsledkov tréningu

Po prejdení výsledkov všetkých prístupov k tréningu sa ukázalo, že najlepší prístup k tréningu z pohľadu presnosti klasifikátora je prístup 2-KET, teda štandardný prístup k tréningu klasifikátora. Použitie autoenkódera má zmysel iba pri extrémne nízkom množstve trénovacích dát – 1%, kde dosahuje vysokú presnosť prístup 5-AKEN alebo 6-DKET, ~83 %. Pri týchto prístupoch bol ale autoenkóder natrénovaný na 100 % trénovacích

dát, aj keď neanotovaných, na ktorej sa vykonáva klasifikácia. Reálne nemusíme mať k dispozícii takúto veľkú množinu neanotovaných dát a potom by som zvolil prístup 8, kde sme síce dosiahli menšiu presnosť, 78 % na BM a 70 % na LM, ale na dosiahnutie takejto presnosti nám stačilo iba 1% trénovacích dát- 600 vzoriek.

Dôsledok, ktorý odvádzame iba na základe výsledkov viacerých prístupov a ktorý by pre svoje úplne potvrdenie potreboval viac overení na rôznych architektúrach je, že autoenkóder rozpoznáva features, ktoré nie sú až také zaujímavé pre klasifikáciu, ale pôsobí skôr ako prirodzený regularizátor. Je to možné pozorovať pri 1 % trénovacích dát, kde prístup 2-KET úplne zlyháva a model klasifikátora sa pretrénuje a má problém naučiť sa relevantné features. Ďalej je to krásne pozorovateľné medzi architektúrami BM a LM. Napriek tomu, že LM je väčšia architektúra, v prístupe 7- DKEN, BM architektúra dosahuje v priemere o 5,52 % vyššiu presnosť ako LM. Autoenkóder tak do určitej miery pôsobí ako regularizátor. Pri prístupe 4-AKET a 5-AKEN je tento jav možné pozorovať pri nižších percentách trénovacích dát.

3.2.2 Porovnanie presnosti navrhnutých architektúr s existujúcimi architektúrami na datasete Fashion Mnist

Na git repozitári datasetu Fashion Mnist majú ľudia možnosť zverejniť presnosť svojej siete na datasete Fashion Mnist- benchmark. Vybrali sme rôzne architektúry, menšie aj oveľa väčšie ako nami navrhnuté architektúry – BM a LM.[2]

Tabuľka 27: Porovnanie presnosti rôznych architektúr na FM

Počet vrstiev	Počet parametrov	Preprocessing	Presnosť na testovacom datasete
2 konvolučné	~113 000	Normalizácia	0,922
2 konvolučné a 3 plne prepojené	~1 800 000	Normalizácia	0,932
VGG 16	~26 000 000	-	0,935
GoogleNet with cross-entropy loss	~5 000 000	-	0,937
Resnet18	~11 000 000	Normalizácia, augmentácia dát	0,949
Architektúra BM	~ 270 000	-	0,937
Architektúra LM	~700 000	-	0,944

Na základe porovnania môžeme konštatovať, že architektúra BM aj LM dosahuje mierne nadpriemerné výsledky pri danom počte parametrov. Je možné , že preprocessingom a použitím augmentácie dát by sme dosiahli vyššiu presnosť, ale to nie je cieľom práce.

3.2.3 Výsledky vybraných prístupov k tréningu na iných datasetoch a architektútrach

Od tohto miesta v práci platí, ak budú v tabuľke uvedené výsledky tréningu, potom to znamená, že tréning bol vykonaný pre dané percento trénovacích dát iba jedenkrát. Ako sme povedali vyššie, výsledky sú aj pri viacnásobnom opakovaní podobné a vzhľadom na veľký počet prístupov budeme používať len 1 opakovanie.

Pre ďalší tréning sme vybrali architektúru BM, variant 2 pre plne prepojenú vrstvu, dataset FM. Výsledky nám dajú odpoveď na otázku, či nemôže mať plne prepojená vrstva

nižšiu kapacitu. Uvádzame aj výsledky pre architektúru LM na datasete CG. Modely z tohto tréningu budú použité v ďalších častiach práce.

Architektúra BM s variantom 2 na datasete Fashion Mnist

Tabuľka 28: Výsledky pre BM - variant 2 - prístup 2-KET

Percento trénovacích dát	Počet epoch	Najlepšia epocha	Presnosť na trénovacom datasete	Presnosť na testovacom datasete
1	21	1	0,2767	0,1039
5	68	48	0,9777	0,8620
10	59	39	0,9633	0,8793
25	100	91	0,9767	0,9117
50	58	38	0,9495	0,9204
100	95	75	0,9607	0,9346

Tabuľka 29: Výsledky pre BM - variant 2 - prístup 3- KEN

Percento trénovacích dát	Počet epoch	Najlepšia epocha	Presnosť na trénovacom datasete	Presnosť na testovacom datasete
1	47	27	0,1433	0,1069
5	65	45	0,1887	0,2369
10	94	74	0,2727	0,3527
25	82	62	0,2608	0,4495
50	29	9	0,2254	0,3581
100	100	84	0,3060	0,4627

Tabuľka 30: Výsledky pre BM - variant 2 - prístup 6- DKET

Percento trénovacích dát	Počet epoch	Najlepšia epocha	Presnosť na trénovacom datasete	Presnosť na testovacom datasete
1	60	59	0,7933	0,6408
5	60	60	0,9300	0,8560
10	60	53	0,9355	0,8667
25	60	53	0,9391	0,8905
50	60	60	0,9428	0,9093
100	60	56	0,9428	0,9254

Tabuľka 31: Výsledky pre BM - variant 2 - prístup 7- DKEN

Percento trénovacích dát	Počet epoch	Najlepšia epocha	Presnosť na trénovacom datasete	Presnosť na testovacom datasete
1	60	60	0,2983	0,4289
5	60	56	0,4383	0,6057
10	60	58	0,4637	0,6308
25	60	45	0,4627	0,6338
50	42	22	0,4639	0,6382
100	40	20	0,4629	0,6349

Zhodnotenie výsledkov tréningu

Pri prístupe 2 je presnosť klasifikátora 93,46 %, čo je takmer rovnaká presnosť ako pri variante 1, ktorý dosahuje 93,87 %. Potom sa naskytuje otázka, či plne prepojená vrstva musí mať veľkú kapacitu a či nemá enkóder dostatočnú kapacitu pre extraciu potrebných informácii pre následnú klasifikáciu. Vidíme, že v prípade prístupu 3-KEN, kde má enkóder náhodne inicializované váhy, presnosť klasifikátora pri 100 % trénovacích dát je 46,27 %, čo je veľmi nízka presnosť. V prístupe 7- bol autoenkóder plne natrénovaný a klasifikátor dosiahol presnosť 63,49 % na 100 % trénovacích dát, čo hovorí o tom, že pre prístupy s dotrénovaním plne prepojenej vrstvy (vtedy, keď je enkóder netrénovateľný) musí mať plne prepojená vrstva väčšiu kapacitu.

Architektúra LM na datasete Cifar Gray

Uvedené sú len tie výsledky tréningov pre architektúru LM na datasete CG, ktoré budeme neskôr v práci používať.

Tabuľka 32: Výsledky prístupu 1-A pre LM na CG

Percento trénovacích dát	Počet epoch	Najlepšia epocha	Rekonštrukčná chyba trénovací dataset	Rekonštrukčná chyba testovací dataset
100	57	37	0,5747	0,5730

Tabuľka 33: Výsledky prístupu 2-KET pre LM na CG

Percento trénovacích dát	Počet epoch	Najlepšia epocha	Presnosť na trénovacom datasete	Presnosť na testovacom datasete
100	150	111	0,9656	0,8403

Tabuľka 34: Výsledky prístupu 4-AKET pre LM na CG

Percento trénovacích dát autoenkóder	Percento trénovacích dát klasifikátor	Rekonštrukčná chyba na testovacom datasete	Presnosť na trénovacom datasete	Presnosť na testovacom datasete
100	100	0,6028	0,7792	0,7787

Tabuľka 35: Výsledky prístupu 7-AKEN pre LM na CG

Percento trénovacích dát	Počet epoch	Najlepšia epocha	Presnosť na trénovacom datasete	Presnosť na testovacom datasete
100	52	32	0,4880	0,5199

Tabuľka 36: Výsledky prístupu 8-AK pre LM na CG

Percento trénovacích dát	Váha autoenkódera	Váha klasifikátora	Rekonštrukčná chyba na testovacom datasete	Presnosť na trénovacom datasete	Presnosť na testovacom datasete
100	1	1	0,5876	0,91008	0,8228
100	1	5	0,5940	0,9234	0,8273
100	5	1	0,5963	0,90756	0,8190

3.3 Klasifikácia na datasetoch získaných pomocou autoenkódera

V nasledujúcej časti popíšeme postup, ako získame z existujúcich datasetov odvodené datasety pomocou autoenkódera. Definujeme 2 odvodené druhy datasetov- derivated dataset a merged dataset.

Derivated dataset je dataset, pre ktorého vytvorenie potrebujeme originálny dataset (Fashion Mnist, Mnist, Cifar Gray) a autoenkóder, ktorý bol natrénovaný na tomto originálnom datasete. Potom necháme obrazy z trénovacej a testovacej množiny z originálneho datasetu prejsť autoenkóderom a takto získame derivated dataset. Autoenkóder zachováva rozmer vstupu a výstupu, preto môžeme na derivated datasete trénovať klasifikátor.

Merged dataset je dataset, ktorý vznikne spojením originálneho datasetu a derivated datasetu a preto obsahuje dvojnásobný počet dát. S takto definovanými datasetmi vykonáme nasledujúci experiment. Na vstupe budeme mať originálnu dataset a autoenkóder z niektorého z prístupov k tréningu natrénovanom na originálnom datasete. Vytvoríme derivated dataset, pre ktorého vstup je originálny dataset a autoenkóder. Z originálneho datasetu a derivated datasetu vytvoríme merged dataset. Potom natrénujeme prístupom 2-KET jeden klasifikátor na derivated datasete a druhý klasifikátor na merged datasete na 100 % trénovacích dát. Následne spravíme porovnanie presnosti. Pre porovnanie presnosti zoberieme novo natrénované klasifikátory a pridáme klasifikátor z prístupu 2 natrénovaný na 100 % dát na originálnom datasete. Presnosť budeme počítať na originálnom datasete, derivated datasete a merged datasete. Autoenkóderi budú pochádzať z prístupov 1-A, 4-AKET a 8-AK.

Týmto postupom sledujeme, či sa rozšírením pôvodného datasetu zvýši presnosť klasifikácie. Ďalej, medzi klasifikátormi natrénovaných na merged datasete nám to nepriamo hovorí o kvalite rekonštrukcie, čím menšia presnosť na testovacej množine na originálnom datasete, tým horšie autoenkóder rekonštruuje obraz.

V nasledujúcich tabuľkách budú znázornené výsledky klasifikácie pre odvodené datasety. Riadok tabuľky predstavuje, na akom datasete bol klasifikátor natrénovaný (originál, derivated, merged) a stĺpec predstavuje, na akom datasete je vyhodnocovaná presnosť klasifikátora (originál, derivated, merged).

- Originálny dataset: FM,
- autoenkóder: prístup 1 – A, BM, 100 % trénovacích dát,
- originálny klasifikátor: prístup 2-KET, BM, 100 % trénovacích dát.

Tabuľka 37: Výsledky klasifikácie na odvodenom datasete na FM, autoenkóder z 1-A

	Original dataset		Derivated dataset		Merged dataset	
Klasifikátor	Presnosť na trénovacom datasete	Presnosť na testovacom datasete	Presnosť na trénovacom datasete	Presnosť na testovacom datasete	Presnosť na trénovacom datasete	Presnosť na testovacom datasete
Original	0,9952	0,9373	0,9772	0,9234	0,9862	0,9304
Derivated	0,9801	0,9279	0,9908	0,9303	0,9854	0,9291
Merged	0,9992	0,9384	0,9988	0,9341	0,999	0,9363

- Originálny dataset: FM,
- autoenkóder: prístup 4-AKET – BM, 100 % trénovacích dát,
- originálny klasifikátor: prístup 2-KET, BM, 100 % trénovacích dát.

Tabuľka 38: Výsledky klasifikácie na odvodenom datasete na FM, autoenkóder z 4-AKET

	Original dataset		Derivated dataset		Merged dataset	
Klasifikátor	Presnosť na trénovacom datasete	Presnosť na testovacom datasete	Presnosť na trénovacom datasete	Presnosť na testovacom datasete	Presnosť na trénovacom datasete	Presnosť na testovacom datasete
Original	0,9952	0,9373	0,9756	0,9239	0,9854	0,9306
Derivated	0,9755	0,9266	0,9898	0,9303	0,9827	0,9285
Merged	0,9913	0,9376	0,9896	0,932	0,9904	0,9348

- Originálny dataset: FM,
- autoenkóder: prístup 8–AK, BM, 100 % trénovacích dát,
- originálny klasifikátor: prístup 2-KET na 100 % trénovacích dát.

Tabuľka 39: Výsledky klasifikácie na odvodenom datasete na FM, autoenkóder z AK

	Original dataset		Derivated dataset		Merged dataset	
Klasifikátor	Presnosť na trénovacom datasete	Presnosť na testovacom datasete	Presnosť na trénovacom datasete	Presnosť na testovacom datasete	Presnosť na trénovacom datasete	Presnosť na testovacom datasete
Original	0,9952	0,9373	0,9395	0,9041	0,9673	0,9207
Derivated	0,9387	0,9069	0,984	0,9204	0,9613	0,9137
Merged	0,9982	0,9383	0,9968	0,926	0,9975	0,9322

- Originálny dataset: CG,
- autoenkóder: prístup 1- A, LM, 100 % trénovacích dát,
- originálny klasifikátor: prístup 2 – KET, 100 % trénovacích dát.

Tabuľka 40: Výsledky klasifikácie na odvodenom datasete na CG, autoenkóder z A

	Original dataset		Derivated dataset		Merged dataset	
Klasifikátor	Presnosť na trénovacom datasete	Presnosť na testovacom datasete	Presnosť na trénovacom datasete	Presnosť na testovacom datasete	Presnosť na trénovacom datasete	Presnosť na testovacom datasete
Original	0,9998	0,8403	0,6526	0,5643	0,8262	0,7023
Derivated	0,8656	0,7298	0,9919	0,7403	0,9287	0,7351
Merged	0,9989	0,7996	0,9979	0,7457	0,9984	0,7727

- Originálny dataset: CG,
- autoenkóder: prístup 4-AKET, LM, 100 % trénovacích dát,
- originálny klasifikátor: prístup 2-KET, 100 % trénovacích dát.

Tabuľka 41: Výsledky klasifikácie na odvodenom datasete na CG, autoenkóder z AKET

	Original dataset		Derivated dataset		Merged dataset	
Klasifikátor	Presnosť na trénovacom datasete	Presnosť na testovacom datasete	Presnosť na trénovacom datasete	Presnosť na testovacom datasete	Presnosť na trénovacom datasete	Presnosť na testovacom datasete
Original	0,9998	0,8403	0,6709	0,5884	0,8354	0,7144
Derivated	0,6855	0,6094	0,9942	0,7507	0,8398	0,6801
Merged	0,9959	0,8219	0,9939	0,7642	0,9949	0,7931

- Originálny dataset: CG,
- autoenkóder: prístup 8 - AK, LM, 100 % trénovacích dát,
- originálny klasifikátor: prístup 2-KET, 100 % trénovacích dát.

Tabuľka 42: Výsledky klasifikácie na odvodenom datasete na CG, autoenkóder z AK

	Original dataset		Derivated dataset		Merged dataset	
Klasifikátor	Presnosť na trénovacom datasete	Presnosť na testovacom datasete	Presnosť na trénovacom datasete	Presnosť na testovacom datasete	Presnosť na trénovacom datasete	Presnosť na testovacom datasete
Original	0,9998	0,8403	0,3896	0,3562	0,6947	0,5983
Derivated	0,2555	0,2519	0,9812	0,7367	0,6184	0,4943
Merged	0,9929	0,8074	0,9804	0,7447	0,9866	0,7761

Zhodnotenie

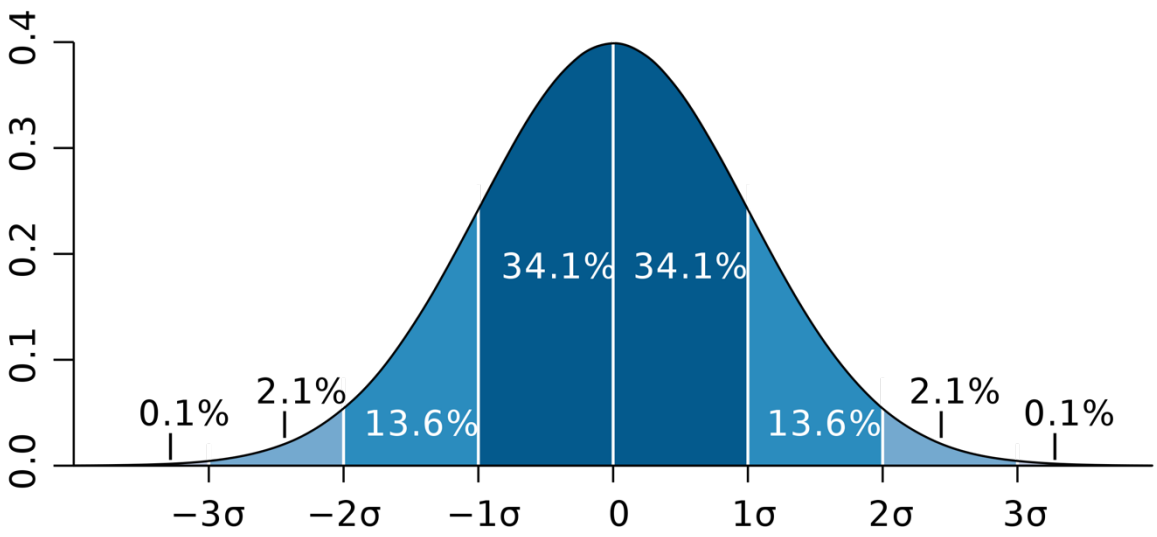
K zvýšeniu presnosti klasifikácie nastalo iba v dvoch prípadoch. Prvý bol klasifikátor natrénovaný na merged datasete FM, autoenkóder z prístupu 1-A, kde bola presnosť na originálnom datasete FM 93,83 %. Druhý bol klasifikátor natrénovaný na merged datasete FM, autoenkóder z prístupu 4-AKET, kde bola presnosť na originálnom datasete FM 93,76 %. Originálny klasifikátor z prístupu 2-KET, dosiahol presnosť 93,73 %, potom sa teda nedá v týchto dvoch prípadoch hovoriť o významnom zvýšení presnosti a určite by som uprednostnil prístup 2-KET pred takýmto postupom, kde potrebujeme rozšíriť dataset. V prípade datasetu CG bol tento postup úplne nevhodný, kde sa merged klasifikátor nepriblížil ani presnosti originálneho klasifikátora. Najlepšiu presnosť dosiahol klasifikátor natrénovaný na merged datasete CG, autoenkóder z prístupu 4-AKET, 82,19 %. Originálny klasifikátor na CG dosahuje presnosť 84,03 %.

4 Oddelenie neznámych dát

Ako ľudia sme zvyknutí zaraďovať do kategórii všetko okolo seba a od algoritmov strojového učenia prirodzene očakávame rovnaké schopnosti. Väčšina modelov klasifikátorov je ale natrénovaná tak, že rozozná len veľmi obmedzený počet kategórií. Datasety použité v práci majú iba 10 kategórii. Trénovací proces je nastavený tak, že každý vstup patrí do jednej z kategórii a pri predpovediach musí vstup do jednej z kategórii patriť. Klasifikátor nepozná pojem neznámy ak ho to explicitne nenaučíme. V tejto kapitole predstavíme, navrhneme a overíme rôzne techniky, za použitia ktorých by sme chceli oddeliť dáta pochádzajúce z jedného datasetu od dát pochádzajúcich z iného, pre klasifikátor neznámeho datasetu.

4.1 Porovnanie rekonštrukčnej chyby autoenkódera na rôznych datasetoch pre označenie obrazu za neznámy

V tejto časti pokúsime na základe rekonštrukčnej chyby autoenkódera určiť, či daný obraz na vstupe patrí do datasetu, na ktorom bol autoenkóder natrénovaný alebo pochádza z iného datasetu. Vychádzame pri tom z predpokladu, že na trénovacom datasete vypočítame priemer a smerodajnú odchýlku a na základe pravidla 3- σ, ktoré hovorí o tom, že u približne normálne rozdeleného štatistického súboru by sa mali takmer všetky relevantné hodnoty zmestiť do 3 smerodajných odchýlok od priemeru. Pre naše účely by sme použili 2-σ, v okolí 2–σ od priemeru sa u normálneho rozdelenia nachádza približne 95,45 %, ako možno vidieť na Obrázok 22. [3]



Obrázok 22: Graf normálneho rozdelenia[4]

O všetkých vstupoch, ktorých hodnoty rekonštrukčných chýb sa nenachádzajú v intervale priemer ± 2σ by sme prehlásili, že nepatria do pôvodnej domény – sú neznáme. Poďme sa pozrieť na realizáciu tejto myšlienky.

Priemer a smerodajnú odchýlku pre vybraný dataset – označme ho pôvodný, získame na základe autoenkódera natrénovanom prístupom 1-A na 100 % trénovacích dát. Tento

prístup sme vybrali preto, lebo rekonštrukčná chyba na testovacej množine je najmenšia zo všetkých zvyšných prístupov. Na pôvodnom datasete a ostatných (neznámych) datasetoch vypočítame priemer a smerodajnú odchýlku rekonštrukčnej chyby na trénovacej a testovacej množine. Pre porovnanie rekonštrukčnej chyby použijeme autoenkódery natrénované na rôznych datasetoch prístupom 1.

Potom, na základe priemeru a smerodajnej odchýlky rekonštrukčnej chyby vypočítanej na trénovacej množine pôvodného datasetu určíme interval <Priemer - 2σ; Priemer + 2σ >. Ak rekonštrukčná chyba vstupu patrí do tohoto intervalu, potom označíme, že vstup patrí do pôvodného datasetu, inak označíme vstup za neznámy. To, či obraz označíme za neznámy budeme určovať na trénovacej a testovacej množine pôvodného datasetu a na testovacej množine neznámych datasetov.

Budeme pracovať s autoenkóderom natrénovanom na datasete Mnist, architektúra BM, neznáme datasety budú potom Fashion Mnist a Cifar Gray.

Ďalší autoenkóder bude natrénovaný na datasete Fashion Mnist, architektúra BM, neznáme datasety sú Mnist a Cifar Gray.

Ďalší autoenkóder bude natrénovaný na datasete Fashion Mnist five classes, architektúra BM, neznáme datasety sú Mnist, Fashion Mnist five other classes a Cifar Gray.

Posledný autoenkóder bude natrénovaný na datasete Cifar Gray, architektúra LM, neznáme datasety sú Mnist a Fashion Mnist.

Výsledky

Originálny dataset Mnist

Tabuľka 43: Rekonštrukčné chyby autoenkódera natrénovaného na M

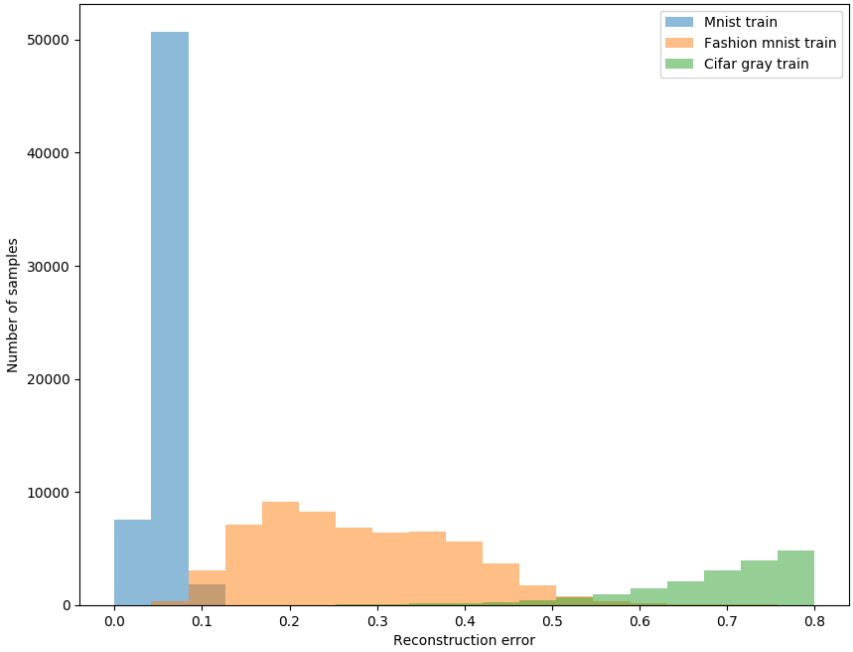
Dataset	Priemer trénovací dataset	σ trénovací dataset	Priemer testovací dataset	σ testovací dataset
Mnist	0,0600	0,0144	0,0598	0,0142
Fashion mnist	0,2789	0,1109	0,2809	0,1119
Cifar	0,8744	0,1977	0,8776	0,1970

Tabuľka 44: Pomer neznámych vstupov k celkovému počtu vstupov -autoenkóder - M

Neznáme trénovací dataset Mnist	Neznámy Mnist	Neznámy Fashion Mnist	Neznámy Cifar Gray
0,0481	0,0502	0,9918	1

Dataset Mnist má nízku smerodajnú odchýlku v porovnaním s inými datasetmi a nižšiu rekonštrukčnú chybu ako ostatné datasety, tak potom nie je problém na základe navrhnutého postupu označiť vstupy za neznáme. 4,81 % vstupov z trénovacej množiny pôvodného datasetu Mnist je označených za neznámych a 5,02 % vstupov z testovacej množiny je označených za neznáme. Neznáme datasety- Fashion Mnist a Cifar Gray vieme ale veľmi spoľahlivo oddeliť na základe rekonštrukčnej chyby, pri FM 99,18 % a pri CG

100 %. Vizualizáciu rekonštrukčných chýb pre trénovacie datasety je možné vidieť na Obrázok 23.



Obrázok 23: Rekonštrukčné chyby na datasetoch - autoenkóder natrénovaný na M

Originálny dataset Fashion Mnist

Tabuľka 45: Rekonštrukčné chyby autoenkódera natrénovaného na FM

Dataset	Priemer trénovací dataset	σ trénovací dataset	Priemer testovací dataset	σ testovací dataset
Mnist	0,0636	0,0153	0,0633	0,0151
Fashion mnist	0,2438	0,0913	0,2460	0,0925
Cifar	0,5753	0,0778	0,5761	0,0776

Tabuľka 46: Pomer neznámych vstupov k celkovému počtu vstupov -autoenkóder - FM

Neznáme trénovací dataset Fashion Mnist	Neznámy Mnist	Neznámy Fashion Mnist	Neznámy Cifar Gray
0,0262	0,4061	0,0317	0,9502

Dataset Fashion Mnist má v porovnaní s datasetom Mnist väčšiu odchýlku na trénovacej množine, 0,091 na FM a 0,015 na M. 40,61 % vstupov z neznámeho datasetu Mnist je označených za neznámych a 95,02 z datasetu Cifar Gray je označených za neznámych. 40, 61 % je podľa nášho názoru dosť nízke číslo, v tomto prípade oddelenie dát nie je dobré.

Čo ale stojí za zmienku je skutočnosť, že autoenkóder natrénovaný na Mniste aj autoenkóder natrénovaný na Fashion Mniste majú veľmi podobnú priemernú rekonštrukčnú chybu - ~0,06 na trénovacej aj testovacej množine na datasete Mnist, aj napriek tomu, že autoenkóder natrénovaný na Fashion Mniste nikdy nevidel vzorky datasetu Mnist a vie ich rekonštruovať s nízkou rekonštrukčnou chybou.

Originálny dataset Fashion Mnist five classes

Tabuľka 47: Rekonštrukčné chyby autoenkódera natrénovaného na FMFC

Dataset	Priemer trénovací dataset	σ trénovací dataset	Priemer testovací dataset	σ testovací dataset
Mnist	0,0662	0,0163	0,0659	0,0161
Fashion Mnist five classes	0,2571	0,0907	0,2605	0,0919
Fashion Mnist five other classes	0,2362	0,0895	0,2371	0,0904
Cifar	0,6029	0,0752	0,6038	0,0746

Tabuľka 48: Pomer neznámych vstupov k celkovému počtu vstupov -autoenkóder - FMFC

Neznáme trénovací dataset Fashion Mnist five classes	Neznámy Mnist	Neznámy Fashion Mnist five classes	Neznámy Fashion Mnist five other classes	Neznámy Cifar Gray
0,0222	0,7127	0,0284	0,0258	0,9649

V tomto prípade sme autoenkóder natrénovali na datasete Fashion Mnist five classes, je to dataset, ktorý obsahuje vzorky prvých 5 tried datasetu FM. Dôležité je v tejto situácii porovnanie s rekonštrukčnej chyby s datasetom Fashion Mnist five other classes, čo je dataset, ktorý obsahuje zvyšných 5 tried datasetu FM. Ak porovnáme rekonštrukčné chyby a odchýlky FMFC a FMFOC, zistíme, že sú dosť podobné, ako vidieť v Tabuľka 47, aj napriek tomu, že autoenkóder natrénovaný na FMFC nikdy nevidel vstupy z FMFOC. Iba 2,58 % vstupov z datasetu FMFOC bolo označených za neznámych. Naznačuje nám to, že rekonštrukčná chyby nebude dobrý prostriedok pre oddelenie neznámych dát, ktoré sú veľmi podobné.

Originálny dataset Cifar Gray

Tabuľka 49: Rekonštrukčné chyby autoenkódera natrénovaného na CG

Dataset	Priemer trénovací dataset	σ trénovací dataset	Priemer testovací dataset	σ testovací dataset
Mnist	0,0872	0,0169	0,0865	0,0166
Fashion mnist	0,2676	0,0902	0,2697	0,0913
Cifar	0,5721	0,0802	0,5730	0,0802

Tabuľka 50: Pomer neznámych vstupov k celkovému počtu vstupov -autoenkóder - CG

Neznáme trénovací dataset Cifar gray	Neznámy Mnist	Neznámy Fashion Mnist	Neznámy Cifar Gray
0,0476	1	0,9267	0,0471

Zhrnutie

Ak sa globálne pozrieme na rekonštrukčné chyby datasetov v rámci rôznych autoenkóderov, môžeme si všimnúť istý trend. Priemerná rekonštrukčná chyba na trénovacom datasete M je ~0,06, na FM ~0,25 a na CG ~0,57 (autoenkóder z M 0,87, ale

k tomu sa dostaneme neskôr), bez ohľadu na to, na akom datasete bol autoenkóder trénovaný. Taktiež vieme, že priemerná rekonštrukčná chyba medzi datasetmi FMFC a FMFOC je podobná. Na základe vyššie uvedených skutočností a výsledkov môžeme povedať, že rekonštrukčná chyba hovorí o tom, aké náročné je dataset rekonštruovať. Vieme, že M je najjednoduchší dataset pre klasifikáciu, FM je o niečo ťažší, FMFC a FMFOC sú rovnako náročné a CG je z nich najťažší, čomu zodpovedá aj rekonštrukčná chyba. Na autoenkóderi natrénovanom na M je pri CG preto taká vysoká rekonštrukčná chyba, lebo Mnist je jednoduchý dataset a nedokáže rekonštruovať CG. Opačne to ale neplatí a autoenkóder natrénovaný na FM aj CG vie rekonštruovať M.

Aj keď sme použili iba 3 datasety, môžeme povedať, že použitie rekonštrukčnej chyby pre označenie vstupov za neznáme nebude vhodný prostriedok pre túto úlohu, rekonštrukčná chyba vypovedá iba o tom, aké náročné je obraz rekonštruovať.

4.2 Rekonštrukčná chyba na správne a nesprávne klasifikovaných dátach

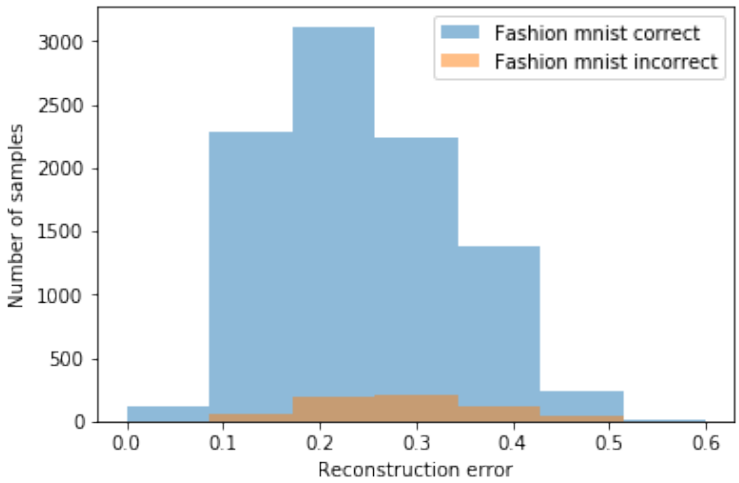
V nasledujúcej časti skúmame, aká je rekonštrukčná chyba na správne a nesprávne klasifikovaných dátach. Na rozdiel od predchádzajúcej podkapitoly tu všetky dáta pochádzajú z rovnakého datasetu a našou úlohou je identifikovať vzorky, ktoré sieť nesprávne klasifikuje. Ak zistíme, že existuje štatisticky významný rozdiel medzi rekonštrukčnou chybou správne klasifikovaných a nesprávne klasifikovaných dát, mohli by sme to použiť pre oddelenie neznámych dát. Postup je nasledujúci. Na základe toho, či sú dáta správne klasifikované ich rozdelíme do dvoch skupín (správne a nesprávne klasifikované). Následne pre všetky dáta zistíme rekonštrukčnú chybu. Na záver vyhodnotíme, či sa rekonštrukčná chyba týchto dvoch skupín odlišuje.

Výsledky

Autoenkóder je z prístupu 1-A a klasifikátor z prístupu 2-KET, oba natrénované na 100 % trénovacích dát, architektúra BM, dataset Fashion Mnist.

Tabuľka 51: Rekonštrukčné chyby na testovacej množine FM

Štatistika	Správne klasifikované	Nesprávne klasifikované
Počet	9387	613
Priemer	0,2434	0,2853
Smerodajná odchýlka	0,0923	0,0859

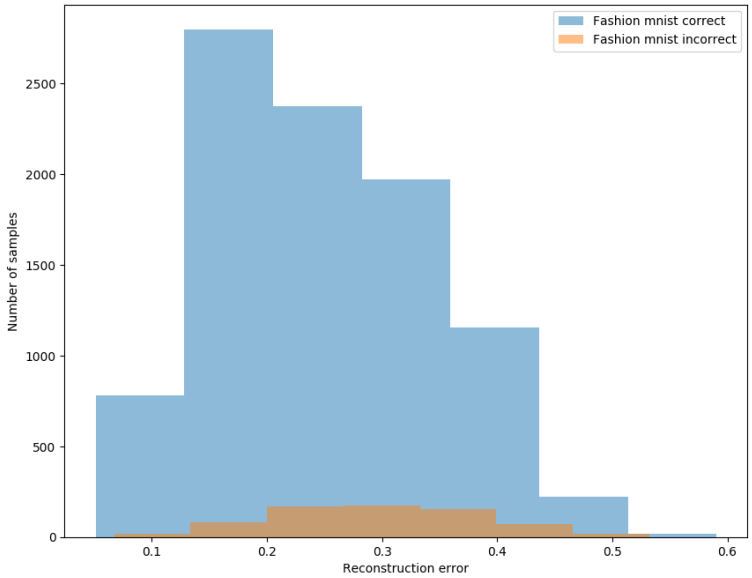


Obrázok 24: Rekonštrukčné chyby na správne a neprávne klasifikovaných dátach

Autoenkóder je z prístupu 1-A a klasifikátor z prístupu 4-AKET, oba natrénované na 100 % trénovacích dát, architektúra BM, dataset Fashion Mnist.

Tabuľka 52: Rekonštrukčné chyby na testovacej množine FM

Štatistika	Správne klasifikované	Nesprávne klasifikované
Počet	9316	684
Priemer	0,2491	0,296
Smerodajná odchýlka	0,094	0,0879

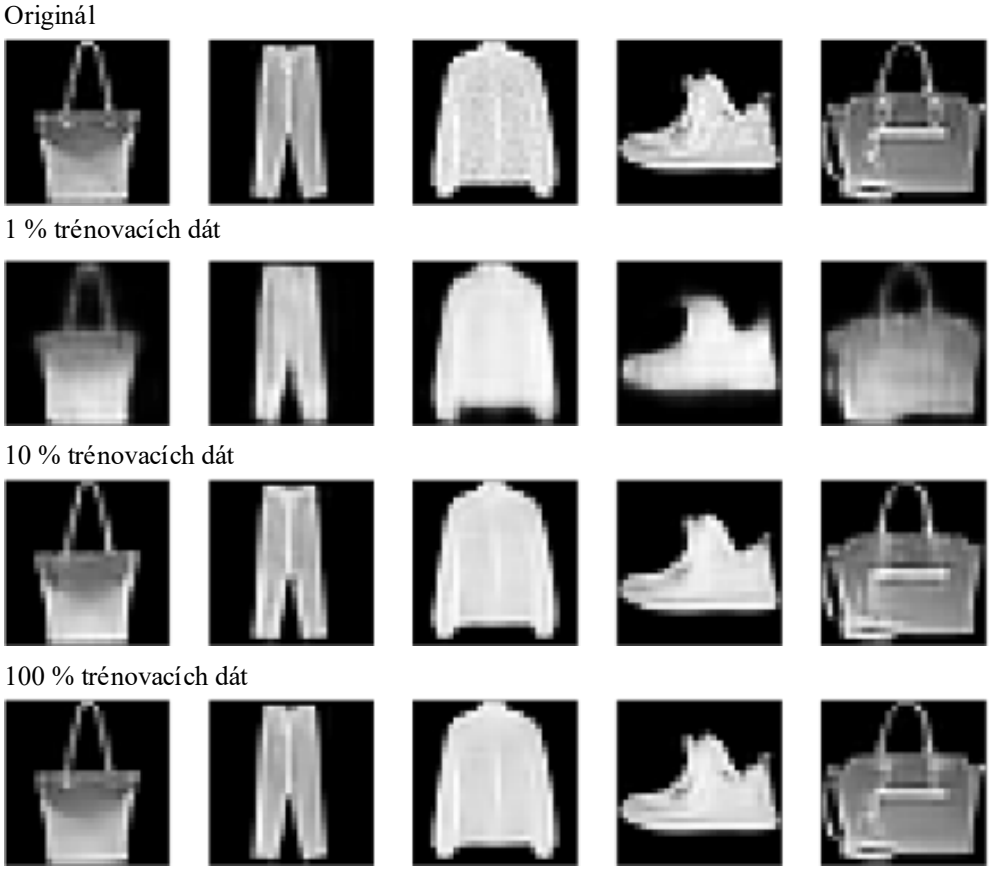


Obrázok 25: Rekonštrukčné chyby na správne a neprávne klasifikovaných dátach

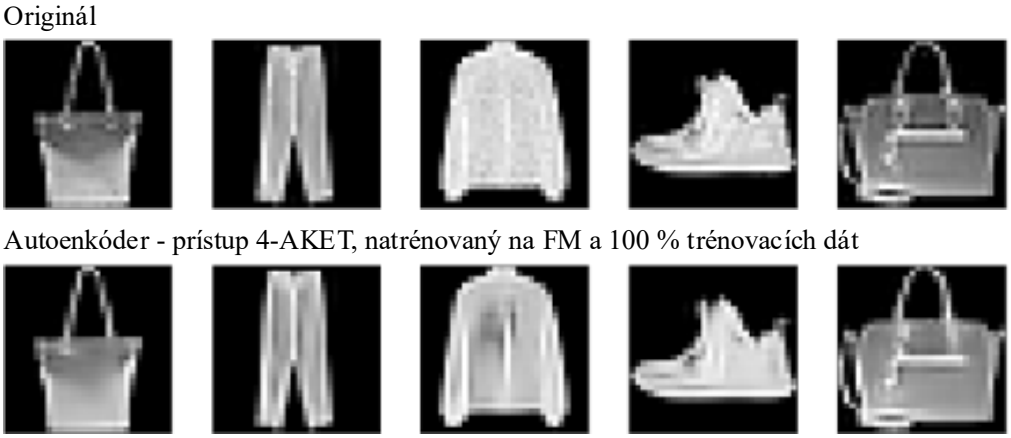
V Tabuľka 51 a Tabuľka 52 môžeme vidieť, že neexistuje významný štatistický rozdiel medzi rekonštrukčnými chybami správne a nesprávne klasifikovaných dát. Z toho vyplýva, že nemôžeme vyvodiť žiadne dôsledky, ktoré by nám pomohli rozlíšiť či vstup patrí alebo nepatrí do domény, na ktorom bol klasifikátor a autoenkóder natrénovaný.

4.3 Vizualizácia rekonštrukcie

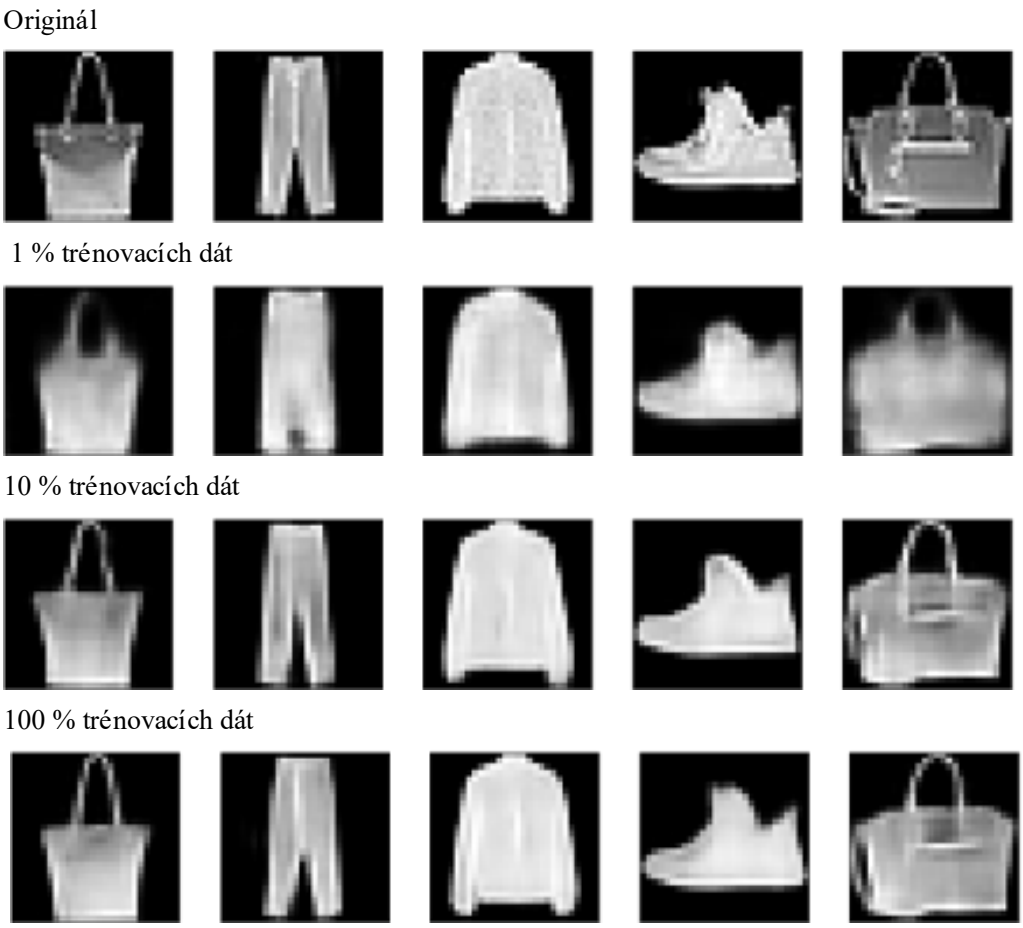
V nasledujúcej časti sa pozrieme na rekonštruovaný obraz z výstupu autoenkódera. Najskôr sa pozrieme na rekonštrukciu na datasete FM pri rôznych trénovacích percentách a na autoenkóderoch z prístupov 1-A, 4-AKET a 8-AK.



Obrázok 26: Rekonštrukcia na FM, autoenkóder z prístupu 1-A



Obrázok 27: Rekonštrukcia na FM, autoenkóder z prístupu 4-AKET



Obrázok 28: Rekonštrukcia na FM, autoenkóder z prístup 8-AK

Na tomto mieste je vhodné porovnať hodnotu rekonštrukčnej chyby s tým, ako skutočne vyzerá zrekonštruovaný obraz. Budeme sa odkazovať na výsledky z Tabuľka 10 a Obrázok 26. Pri 1 % trénovacích dát je rekonštrukčná chyba ~0,27 a pri 100 % trénovacích dát ~0,24, ale kvalita rekonštrukcie pozorovaná ľudským okom je pri 1 % trénovacích dát oveľa nižšia. Čo považujeme za samozrejmosť je, ako možno vidieť na obrázkoch vyššie, že s rastúcim percentom trénovacích dát je aj kvalita rekonštrukcia vyššia. V prístupe 8-AK, Obrázok 28, vidíme, že aj pri 100 % trénovacích dát obraz nie je dostatočne ostrý. Podľa Tabuľka 25 vieme, že rekonštrukčná chyba 8-AK je ~0,26 a tento model je určený viac pre klasifikáciu ako pre rekonštrukciu obrazu.

Teraz sa pozrieme na rekonštrukciu autoenkóderov natrénovaných na Mniste a Cifar Gray na datasete Fashion Mnist.

Originál

Autoenkóder - prístup 1-A, natrénovaný na M a 100 % trénovacích dát

Autoenkóder - prístup 1-A, natrénovaný na CG a 100 % trénovacích dát

Obrázok 29: Rekonštrukcia autoenkóderov z M a CG na na FM

Podľa Tabuľka 43 je rekonštrukčná chyba autoenkódera natrénovanom na datasete M na testovacom datasete FM 0,2809 a podľa Tabuľka 49 je rekonštrukčná chyba autoenkódera natrénovanom na datasete CG na testovacom datasete FM 0,2697. Po vizualizácii vidíme, že aj keď autoenkóder natrénovaný na M má vyššiu rekonštrukčnú chybu, rekonštrukcia vyzerá lepšie a detailnejšie ako autoenkóder natrénovaný na CG. Pre úplnosť ukážeme aj vizualizácie na datasete Mnist a Cifar Gray.

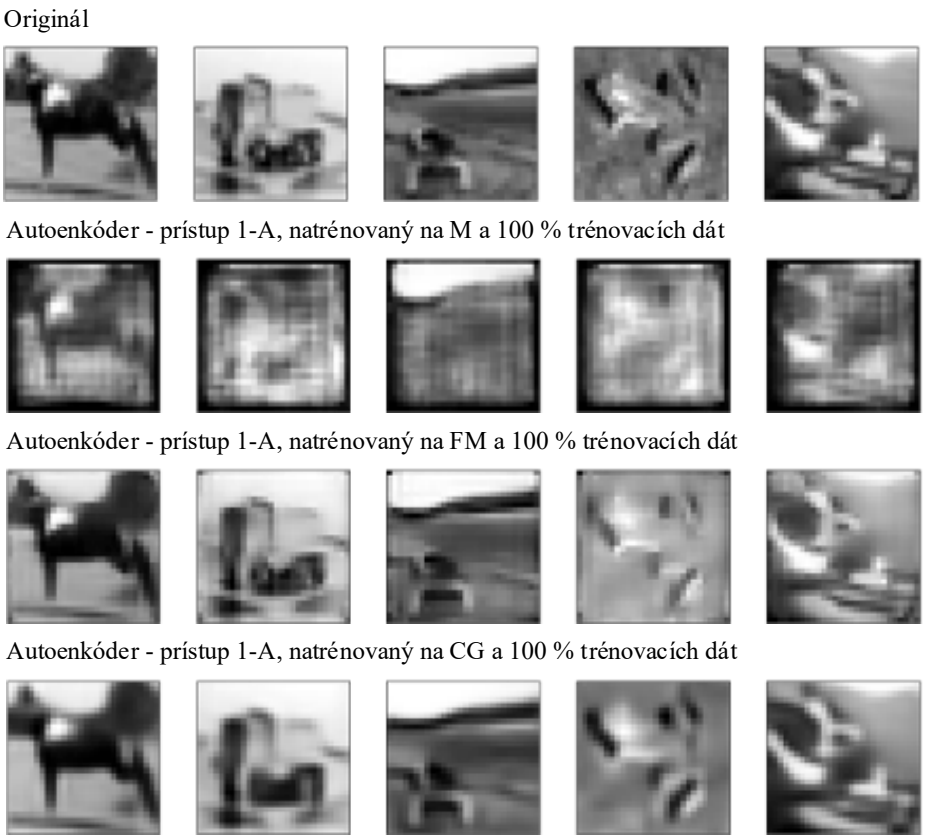
Originál

Autoenkóder - prístup 1-A, natrénovaný na M a 100 % trénovacích dát

Autoenkóder - prístup 1-A, natrénovaný na FM a 100 % trénovacích dát

Autoenkóder - prístup 1-A, natrénovaný na CG a 100 % trénovacích dát

Obrázok 30: Rekonštrukcia na Mniste



Obrázok 31: Rekonštrukcia autoenkóderov na CG

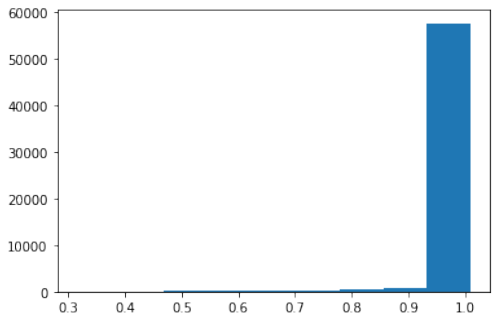
4.4 Postprocessing výstupu z klasifikátora

Majme klasifikátor, ktorý klasifikuje vstupy do *N* kategórií. Nech *X* je vektor, ktorý má *N* hodnôt. Vektor *X* nadobúda hodnoty v závislosti od výstupu z poslednej vrstvy klasifikátora. V nami navrhnutej architektúre používame na poslednej vrstve aktivačnú funkciu SoftMax. Ak je použitá aktivačná vrstva SoftMax, hodnoty vektora *X* sú v intervale <0;1>, $\sum_{i=1}^{N} X_i = 1$ a konkrétna hodnota hovorí, s akou pravdepodobnosťou zaradíme vstup do danej kategórie. Vstup zaradíme do tej kategórie, ktorá nadobúda najväčšiu hodnotu.

Navrhujeme postup, ktorý na základe hodnôt vektora *X* bude vedieť povedať, či vstup klasifikátora je vstup z domény, na ktorej bol klasifikátor trénovaný, alebo ide o obraz z iného datasetu / domény. V tomto postupe predpokladáme, že zoberieme už natrénovaný klasifikátor a budeme brať výstupy z poslednej vrstvy po aktivačnej funkcie SoftMax, alebo ho upravíme tak, že aktivačnú funkciu na poslednej vrstve zmeníme na Sigmoid alebo úplne odstránime aktivačnú funkciu a dostaneme "surový" výstup z plne prepojenej vrstvy. Váhy klasifikátora sa meniť nebudú a nič nebudeme dodatočne trénovať. Zmenou aktivačnej funkcie, prípadne jej odstránením bude vektor *X* nadobúdať iné hodnoty ako s aktivačnou funkciou SoftMax, čo môže zlepšiť alebo zhoršiť následný postprocessing.

Ak aktivačnú funkciu SoftMax nahradíme aktivačnou funkciou Sigmoid, tak hodnoty vektora *X* budú nadobúdať hodnoty v intervale (0,1), ale nebude platiť, že $\sum_{i=1}^{N} X_i = 1$. Ak odstránime aktivačnú funkciu, potom hodnoty vektora *X* môžu teoreticky nadobúdať hodnoty v intervale (-∞, ∞).

Majme klasifikátor natrénovaný na datasete FM na 100 % trénovacích dát prístupom 2-KET, architektúra BM. Poďme sa pozrieť na vizualizáciu maximálnych hodnôt vektora *X* na trénovacom datasete FM pri aktivačnej funkcii SoftMax, Sigmoid a bez aktivačnej funkcie.

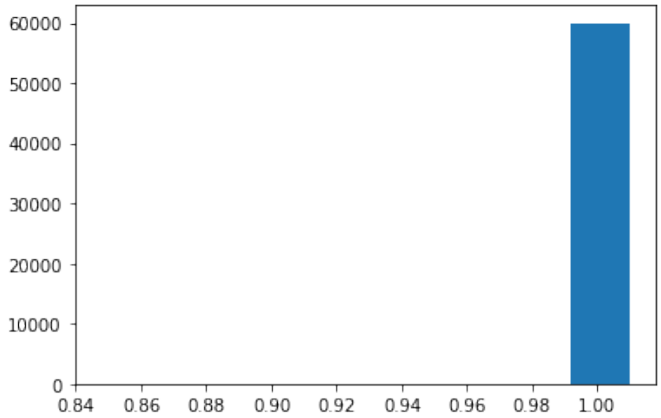


Obrázok 32: Maximálne hodnoty vektora *X* pri aktivačnej funkcii SoftMax na FM

Tabuľka 53: Štatistiky pre maximálne hodnoty vektora *X* pri aktivačnej funkcii SoftMax na FM

Štatistika	Hodnota
Priemer	0,9896
Smerodajná odchýlka	0,0519
Minimum	0,3250
Maximum	1,0000

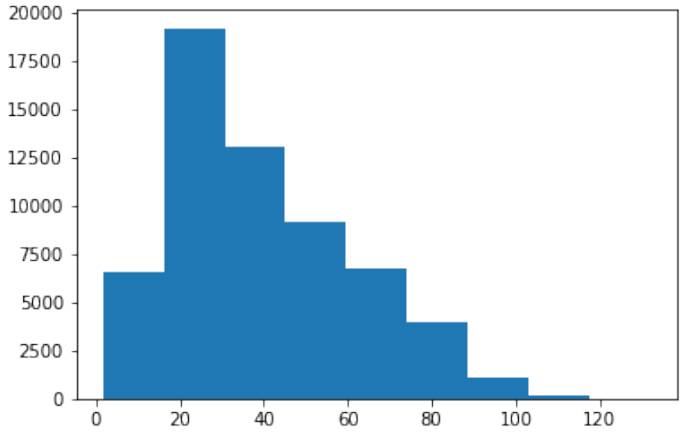
Vizualizácia maximálnej hodnoty, ktorú nadobúda vektor X pri aktivačnej funkcii sigmoid na trénovacom datasete Fashion Mnist.



Obrázok 33: Maximálne hodnoty vektora X pri aktivačnej funkcii Sigmoid na FM

Tabuľka 54: Štatistika pre maximálne hodnoty vektora X pri aktivačnej funkcii Sigmoid na FM

Štatistika	Hodnota
Priemer	1,0000
Smerodajná odchýlka	0,0011
Minimum	0,8576
Maximum	1,0000



Obrázok 34: Maximálne hodnoty vektora X bez aktivačnej funkcie na FM

Tabuľka 55: Štatistika pre maximálne hodnoty vektora X bez aktivačnej funkcie na FM

Štatistika	Hodnota
Priemer	39,8228
Smerodajná odchylka	21,5637
Minimum	1,7958
Maximum	131,8437

Pri aktivačných funkciách Softmax a Sigmoid si môžeme všimnúť, že takmer všetky maximálne hodnoty vektora X sú sústredené v blízkom okolí 1 s nízkou smerodajnou

odchýlkou a pri aktivačnej funkcii Sigmoid aj na krátkom intervale <0,84;1,0). Ak nepoužijeme aktivačnú funkciu, tak tieto hodnoty majú väčšiu smerodajnú odchýlku, čo nám môže uľahčiť oddeliť vstupy z originálnej domény od vstupov z neznámej domény.

Postprocessing vektora X bude prebiehať rovnako, bez ohľadu na to, aká bude použitá alebo nepoužitá aktivačná funkcia na poslednej vrstve klasifikátora. Použijeme funkciu, ktorej vstupom bude vektor X a jej výstup bude reálne číslo, $f: X \rightarrow R$. Funkciu nazveme predikátor. Nech p je index predpovedanej kategórie podľa použitého klasifikátora. Použité budú nasledujúce predikátory:

- **AbsDistanceFromPredicted (ADFP)**

$$f: \sum_{i=0, i\,!=p}^{N} \mathrm{abs}(X_p - X_i) \tag{10}$$

- **SquereDistanceFromPredicted (SDFP)**

$$f: \sum_{i=0, i\,!=p}^{N} (X_p - X_i)^2 \tag{11}$$

- **SumAbs (SA)**

$$f: \sum_{i=0}^{N} abs(X_i\ ) \tag{12}$$

- **SumSquare (SS)**

$$f: \sum_{i=0,}^{N} {X_i}^2 \tag{13}$$

- **MaxArgPredictor (MAP)**

$$f: X_p \tag{14}$$

Na trénovacej množine datasetu, na ktorom bol klasifikátor trénovaný, vypočítame pre každý vstup hodnotu predikátora, výsledný zoznam hodnôt nazveme vzdialenosťami. Tento dataset nazveme originálny. Potom je nami externe určený K-ty percentil, na základe ktorého je v zozname vzdialenosti vypočítaná tzv. prahová hodnota, ktorá rozdeľuje súbor hodnôt v pomere K a 100 – K. K / 100 * (veľkosť trénovacieho datasetu) je počet vstupov z trénovacieho datasetu, ktoré sme ochotní obetovať a označiť, že nepatria do daného datasetu.

Na základe prahovej hodnoty získanej na trénovacom datasete originálnej domény zoberieme iný dataset, pre každý vstup vypočítame hodnotu predikátora a ak tá je menšia, ako prahová hodnota, o vstupnom obraze prehlásime, že nepatrí do pôvodnej domény – je neznámy. Bude nás taktiež zaujímať, koľko % testovacích dát z originálnej domény bude označených za neznámych na základe prahovej hodnoty získanej na trénovacom datasete originálnej domény. Predpokladáme, že to bude viac ako na trénovacom datasete.

V našich experimetoch budeme pracovať s 10-tym percentilom, $K = 10$. To znamená, že 10 % originálnych trénovacích dát bude označených za neznáme.

Hodnoty, ktoré budeme pri experimetoch pozorovať sú nasledovné. Ak sa jedná o trénovací a testovací dataset originálnej domény, tak nás zaujíma percento správne klasifikovaných vstupov (presnosť), percento nesprávne klasifikovaných vstupov a percento vstupov, ktoré sú určené ako neznáme. Ak sa jedná o dataset z inej domény, zoberieme testovaciu množinu tohoto datasetu a zaujíma nás iba percento vstupov, ktoré sú určené ako neznáme, klasifikácia nemá zmysel.

Vzhľadom na veľký počet možností, ako možno nakombinovať klasifikátory z rôznych prístupov k tréningu (2-8), datasety (FM, M, CG, FMFC, CGFC), variantov aktivačnej vrstvy na poslednej vrstve klasifikátora (3) a predikátorov (5) sme museli zvoliť postup, ktorý zredukuje počet kombinácii, s ktorými treba spraviť postprocessing.

Rozhodli sme sa najskôr vykonať postprocessing, ktorý nám pomôže vybrať, ktorý variant poslednej vrstvy klasifikátora je najvhodnejší. Použijeme klasifikátor z prístupu 2-KET natrénovanom na FM na 100 % trénovacích dát, architektúra BM. Originálny dataset bude FM a neznáme datasety M a CG.

Potom budeme prechádzať klasifikátory z jednotlivých prístupov k tréningu na datasete FM natrénovaných na 100 % trénovacích dát a vyberieme tie prístupy k tréningu, ktorých klasifikátory vedia najlepšie označiť vstupy z iných datasetov za neznáme. Originálny dataset bude FM a neznáme datasety budú M a CG.

Potom môžeme pre tieto vybrané prístupy tréningu natrénovať klasifikátor na iných datasetoch, ako M, CG, FMFC a CGFC a zistiť, ako dokážu označiť vstupy za neznáme.

Na záver sa pozrieme súhrne na výsledky a zhodnotíme, či je nami navrhovaný postup pre označenie dát za neznámy vhodný pre túto úlohu.

Tabuľka 56: Variant klasifikátora s aktivačnou vrstvou SoftMax na poslednej vrstve

	Testovací dataset FM			Neznámy dataset Mnist	Neznámy dataset Cifar Gray
Predikátor	Presnosť	Chybovosť	Neznáme	Neznáme	Neznáme
ADFP	0,8961	0,033	0,0709	0,2241	0,1086
SDFP	0,8822	0,0269	0,0909	0,2824	0,1375
SA	0,9373	0,0627	0	0	0
SS	0,9373	0,0627	0	0	0
MAP	0,8331	0,0142	0,1527	0,4442	0,2388

Tabuľka 57: Variant klasifikátora s aktivačnou vrstvou Sigmoid na poslednej vrstve

	Testovací dataset FM			Neznámy dataset Mnist	Neznámy dataset Cifar Gray
Predikátor	Presnosť	Chybovosť	Neznáme	Neznáme	Neznáme
ADFP	0,8294	0,0867	0,0839	0,0997	0,0299
SDFP	0,8842	0,096	0,0198	0,0498	0,0167
SA	0,8994	0,1006	0	0,0025	0,0009
SS	0,8994	0,1006	0	0,014	0,0056
MAP	0,8075	0,0663	0,1262	0,6249	0,5265

Tabuľka 58: Variant klasifikátora bez aktivačnej funkcie na poslednej vrstve

	Testovací dataset FM			Neznámy dataset Mnist	Neznámy dataset Cifar Gray
Predikátor	Presnosť	Chybovosť	Neznáme	Neznáme	Neznáme
ADFP	0,8453	0,0272	0,1275	0,6346	0,5153
SDFP	0,844	0,029	0,127	0,6608	0,574
SA	0,8494	0,0342	0,1164	0,6969	0,6478
SS	0,8438	0,0327	0,1235	0,7162	0,7163
MAP	0,8446	0,028	0,1274	0,628	0,5299

Ako sme mohli tušiť už pri vizualizácia maximálnej hodnoty, ktorú nadobúda vektor X pri aktivačnej funkcii SoftMax, Sigmoid alebo bez aktivačnej funkcie, na danej úlohe si najlepšie počínal klasifikátor bez aktivačnej funkcie, ako vidieť v Tabuľka 58. Preto budeme v ďalších experimentoch používať klasifikátor bez aktivačnej funkcie. Najlepší výsledok dosiahol predikátor s názvom SumSquere, ktorý z originálneho datasetu označil 12,35 % obrazov za neznámych (na trénovacom datasete sme určili, že 10 % vstupov označíme za neznáme), na testovacej množine M označil 71,62 % za neznámych a na testovacej množine CG označil 71,63 % za neznámych. Ďalej sa pozrieme na výsledky experimentov naprieč klasifikátormi z rôznych prístupov k tréningu. Všetky klasifikátory sú natrénované na 100 % trénovacích dát, architektúra BM. Posledná vrstva bude bez aktivačnej funkcie. Neuvádzame klasifikátor z prístupu 2-KET, lebo ten je uvedený v Tabuľka 58.

Tabuľka 59: Výsledky postprocessingu pre prístup 3- KEN

	Testovací dataset FM			Neznámy dataset Mnist	Neznámy dataset Cifar Gray
Predikátor	Presnosť	Chybovosť	Neznáme	Neznáme	Neznáme
ADFP	0,8423	0,0475	0,1102	0,6759	0,92
SDFP	0,842	0,0499	0,1081	0,7472	0,9258
SA	0,8429	0,0465	0,1106	0,5329	0,9007
SS	0,842	0,049	0,109	0,6338	0,7993
MAP	0,8158	0,0819	0,1023	0,4411	0,8351

Tabuľka 60: Výsledky postprocessingu pre prístup 4 -AKET

	Testovací dataset FM			Neznámy dataset Mnist	Neznámy dataset Cifar Gray
Predikátor	Presnosť	Chybovosť	Neznáme	Neznáme	Neznáme
ADFP	0,8477	0,0352	0,1171	0,8922	0,8918
SDFP	0,8473	0,0376	0,1151	0,912	0,9205
SA	0,8494	0,037	0,1136	0,8154	0,8286
SS	0,8486	0,0386	0,1128	0,8768	0,8835
MAP	0,8326	0,0383	0,1291	0,9257	0,9217

Tabuľka 61: Výsledky postprocessingu pre prístup 5- AKEN

	Testovací dataset FM			Neznámy dataset Mnist	Neznámy dataset Cifar Gray
Predikátor	Presnosť	Chybovosť	Neznáme	Neznáme	Neznáme
ADFP	0,8423	0,0475	0,1102	0,6759	0,8547
SDFP	0,842	0,0499	0,1081	0,7472	0,8699
SA	0,8429	0,0465	0,1106	0,5329	0,7769
SS	0,842	0,049	0,109	0,6338	0,826
MAP	0,8158	0,0819	0,1023	0,4411	0,2898

Tabuľka 62: Výsledky postprocessingu pre prístup 6- DKET

	Testovací dataset FM			Neznámy dataset Mnist	Neznámy dataset Cifar Gray
Predikátor	Presnosť	Chybovosť	Neznáme	Neznáme	Neznáme
ADFP	0,8458	0,0266	0,1276	0,4621	0,344
SDFP	0,8451	0,0287	0,1262	0,5051	0,4149
SA	0,8478	0,033	0,1192	0,4518	0,3347
SS	0,8462	0,0329	0,1209	0,5372	0,4708
MAP	0,8449	0,0255	0,1296	0,503	0,4716

Tabuľka 63: Výsledky postprocessingu pre prístup 7- DKEN

	Testovací dataset FM			Neznámy dataset Mnist	Neznámy dataset Cifar Gray
Predikátor	Presnosť	Chybovosť	Neznáme	Neznáme	Neznáme
ADFP	0,8471	0,044	0,1089	0,632	0,5704
SDFP	0,8465	0,0465	0,107	0,703	0,7098
SA	0,8438	0,0501	0,1061	0,7078	0,8341
SS	0,8428	0,0516	0,1056	0,704	0,8447
MAP	0,8472	0,043	0,1098	0,6399	0,6053

Tabuľka 64: Výsledky postprocessingu pre prístup 8- AK

	Testovací dataset FM			Neznámy dataset Mnist	Neznámy dataset Cifar Gray
Predikátor	Presnosť	Chybovosť	Neznáme	Neznáme	Neznáme
ADFP	0,8449	0,032	0,1231	0,7362	0,9987
SDFP	0,8454	0,0341	0,1205	0,7454	0,9989
SA	0,8488	0,0348	0,1164	0,5813	0,9898
SS	0,8476	0,0365	0,1159	0,6169	0,9933
MAP	0,8223	0,05	0,1277	0,7987	0,3698

Po prejdení výsledkov v tabuľkách vyššie zistíme, že najlepšie označuje neznáme dáta klasifikátor z prístupu 4-AKET, Tabuľka 60, pri predikátore MAP označí viac ako 92 % vzoriek z M aj CG za neznáme, čo je vynikajúci výsledok pri zachovaní presnosti ~84 % oproti pôvodným ~93 % na klasifikátore bez postprocessingu. Okrem klasifikátora z prístupu 4-AKET si môžeme všimnúť, že si výborne počína aj klasifikátor z prístupu

8-AK, hlavne na CG. Najhoršie je na tom klasifikátor z prístupu 6-KET, menej ako 50 % dát z M aj CG označí za neznáme.

Na základe získaných výsledkov budeme ďalej pracovať iba s klasifikátormi z prístupu 2-KET, 4-AKET a 8-AK. Budeme ale uvádzať iba výsledky postprocessingu pre tie prístupy k tréningu, ktoré dávajú zaujímavé výsledky.

Doteraz sme ako originálny dataset používali FM, teraz to bude FMFC, M, CG, a CGFC. Na týchto datasetoch sa ukáže, či je skúmaný postup vhodný pre označenie dát za neznáme. Ak nemáme pre daný dataset natrénovaný klasifikátor daným prístupom, tak ho natrénujeme a vykonáme postprocessing.

- Originálny dataset: Fashion Mnist five classes
- Tréning: Prístup 4, 100 % trénovacích dát, architektúra BM

Tabuľka 65: Postprocessing na FMFC, prístup 4-AKET

	Testovací dataset FMFC			Neznámy dataset FMFOC	Neznámy dataset Mnist	Neznámy dataset Cifar Gray
Predikátor	Presnosť	Chybovosť	Neznáme	Neznáme	Neznáme	Neznáme
ADFP	0,8584	0,0232	0,1184	0,8234	0,6414	0,9986
SDFP	0,8564	0,0244	0,1192	0,809	0,6297	0,9988
SA	0,862	0,0236	0,1144	0,8098	0,4123	0,9971
SS	0,8604	0,025	0,1146	0,7916	0,4413	0,9985
MAP	0,8462	0,0224	0,1314	0,821	0,9056	0,9985

V prípade datasetu FMFC bol klasifikátor natrénovaný iba na prvých 5 triedach datasetu FM. Dataset FMFOC obsahuje zvyšných 5 tried datasetu FM. Výsledky tohto posprocessingu sú pre nás podstatné, lebo pomocou rekonštrukčnej chyby sa nám vôbec nepodarilo oddeliť dataset FMFC od FMFOC. V tabuľke vyššie môžeme vidieť, že bez ohľadu na použitý predikátor je ~80 % vstupov z testovacej množiny FMFOC označených za neznáme, čo môže naznačovať, že tento postup môže byť vhodný pre označenie vstupov za neznáme.

- Originálny dataset: Mnist,
- tréning: prístup 4, 100 % trénovacích dát, architektúra BM.

Tabuľka 66: Postprocessing na M, prístup 4-AKET

	Testovací dataset Mnist			Neznámy dataset Fashion Mnist	Neznámy dataset Cifar Gray
Predikátor	Presnosť	Chybovosť	Neznáme	Neznáme	Neznáme
ADFP	0,8954	0,0014	0,1032	0,6217	0,0356
SDFP	0,8964	0,0017	0,1019	0,5341	0,0119
SA	0,8988	0,002	0,0992	0,1953	0,0001
SS	0,8999	0,0021	0,098	0,2028	0,0001
MAP	0,8946	0,0007	0,1047	0,9986	1

V Tabuľka 66 môžeme vidieť výsledky postprocessingu pre dataset Mnist. Prekvapivé, v porovnaní s postprocessingom na FM je, že predikátory medzi sebou dosahujú veľmi rozdielne výsledky a o všetkých predikátoroch okrem MAP možno povedať, že nedokážu oddeliť neznáme dát. Predikátor MAP dokáže označiť až 99,89 % vstupov testovacej množiny FM za neznáme a všetkých 100 % z testovacej množiny CG za neznáme, pričom 10,47 % z testovacej množiny Mnistu označí za neznáme.

- Originálny dataset: Cifar Gray,
- tréning: prístup 2, 100 % trénovacích dát, architektúra LM.

Tabuľka 67: Postprocessing na CG, prístup 2-KET

	Testovací dataset Cifar Gray			Neznámy dataset Fashion Mnist	Neznámy dataset Mnist
Predikátor	Presnosť	Chybovosť	Neznáme	Neznáme	Neznáme
ADFP	0,6807	0,0443	0,275	0,7602	0,7226
SDFP	0,6866	0,0485	0,2649	0,7439	0,7132
SA	0,7452	0,102	0,1528	0,3943	0,6362
SS	0,7331	0,0876	0,1793	0,4795	0,6722
MAP	0,7029	0,0591	0,238	0,6638	0,7501

- Originálny dataset: Cifar Gray,
- tréning: prístup 4, 100 % trénovacích dát, architektúra LM.

Tabuľka 68: Postprocessing na CG, prístup 4-AKET

	Testovací dataset Cifar Gray			Neznámy dataset Fashion Mnist	Neznámy dataset Mnist
Predikátor	Presnosť	Chybovosť	Neznáme	Neznáme	Neznáme
ADFP	0,7286	0,1456	0,1258	0,3204	0,2974
SDFP	0,7274	0,1475	0,1251	0,3268	0,2958
SA	0,7351	0,1662	0,0987	0,2963	0,1968
SS	0,7255	0,1582	0,1163	0,3464	0,2442
MAP	0,711	0,1749	0,1141	0,0405	0,1824

V Tabuľka 67 a Tabuľka 68 sú výsledky postprocessingu pre dataset CG. Prvýkrát uvádzame výsledky pre architektúru LM. V prípade klasifikátora z prístupu 2-KET, Tabuľka 67, vidíme, že síce ~70 % vstupov dokáže označiť za neznáme, ale klasifikátor označí zároveň veľa vstupov z testovacej množiny originálneho datasetu za neznáme- pri predikátore ADFP 27,5 % a pri MAP 23,8 %, čo nie je želané. Pri prístupe 4-AKET, je na testovacej množine originálneho datasetu označených ~12% vstupov za neznámych, čo je lepšie ako pri postprocessingu na prístupe 2-KET. Na druhej strane tento prístup nedokáže oddeliť neznáme dáta, nízke percento vstupov z neznámych datasetov označí za neznáme a navyše, podľa Tabuľka 34, je presnosť klasifikátora bez postprocessingu ~78 % a postprocessingom sa presnosť ešte zníži.

- Originálny dataset: Cifar Gray five classes,
- tréning: prístup 4, 100 % trénovacích dát, architektúra LM.

Tabuľka 69: Postprocessing na CGFC, prístup 4-AKET

	Testovací dataset CGFC			Neznámy dataset Fashion Mnist	Neznámy dataset Mnist	Neznámy dataset CGFOC
Predikátor	Presnosť	Chybovosť	Neznáme	Neznáme	Neznáme	Neznáme
ADFP	0,7208	0,1172	0,162	0,2138	0,0146	0,3164
SDFP	0,72	0,1202	0,1598	0,2031	0,0114	0,3118
SA	0,7428	0,1492	0,108	0,1319	0,0043	0,2218
SS	0,734	0,1428	0,1232	0,1464	0,0053	0,2462
MAP	0,7218	0,1764	0,1018	0,2279	0,4901	0,0754

V tabuľke vyššie sú výsledky postprocessingu na klasifikátore natrénovanom na CGFC. Môžeme vidieť, že klasifikátor nedokáže označiť vstupy z neznámych datasetoch za neznáme alebo iba s nízkou úspešnosťou. FMFOC je analógiou k datasetu CGFOC. Ak porovnáme výsledky s klasifikátorom natrénovanom na datasete FMFC, Tabuľka 65, klasifikátor označí ~81 % vstupov z datasetu FMFOC za neznáme, pričom teraz klasifikátor označí iba ~23 % vstupov z datasetu CGFOC za neznáme, čo považujeme za nedostatočné.

4.4.1 Zhodnotenie výsledkov postprocessingu

Na začiatku, na datasete Fashion Mnist sa nami navrhovaný postup pre zistenie, či daný vstup patrí alebo nepatrí do domény, na ktorej bol dataset natrénovaný, javil ako vhodný.

Klasifikátory, z rôznych prístupov k tréningu dokázali s celkom vysokou spoľahlivosťou povedať, či daný vstup patrí alebo nepatrí do domény datasetu FM. Najlepšie výsledky dosiahol klasifikátor z prístupu 4-AKET, Tabuľka 60, kde niektoré predikátory označili viac ako 90 % vstupov z datasetu M a CG za neznáme. Veľmi dobre si postprocessing počínal aj na datasete FMFC, Tabuľka 65, kde ~80 % vstupov z testovacej množiny FMFOC bolo označených za neznámych pri zachovaní presnosti klasifikátora ~85 %. Tento výsledok považujeme za veľmi dobrý, pretože FMFOC je dataset, ktorý je vytvorený z FM a zdieľa črty s FMFC.

Vybrali sme prístupy k tréningu, ktorých postprocessing dáva dobré výsledky a potom sme chceli zistiť, ako sa bude postprocessing správať na ostatných datasetoch ako M, CG a CGFC. Už na datasete Mnist bol úspešný iba jeden predikátor a to MAP, Tabuľka 66, kde na oboch datasetoch FM a CG označil viac ako 99 % dát za neznáme. Zvyšné predikátory označili nízke % vstupov za neznáme. Ďalej sme pokračovali datasetom CG, ktorý používa architektúru LM. Napriek tomu, že sme vyskúšali klasifikátory z rôznych prístupov k tréningu, 2-KET a 4 -AKET, výsledky postprocesingu boli veľmi zlé. Prvým problémom bolo, že klasifikátor na testovacej množine originálneho datasetu označil veľké % vstupov za neznáme(viac ako 15 %), čo by nemal, aj keď presnosť neklesla o 10 %. Druhým problémom bolo, že označil nízke percento vstupov za neznáme. Pri postprocesingu na datasete CGFC sa ešte viac prehĺbil druhý menovaný problém.

Vzhľadom na získané výsledky nemôžeme všeobecne odporučiť nami navrhovaný postup, ktorý na základe výstupu z klasifikátora určí, či vstup patrí do domény, na ktorej bol klasifikátor natrénovaný. V prípade datasetu FM a FMFC a M, ktorých klasifikátor používa architektúru BM, môžeme povedať, že postprocessing dáva dobré výsledky. Pri datasetoch

CG a CGFC, kde je použitá architektúra LM, postprocessing nefunguje. Navyše, ako veľkým problémom navrhovaného postupu je, že na trénovacej množine musíme určiť, koľko % vstupov obetujeme a označíme za neznáme, klasifikátor po postprocesingu bude mať potom nižšiu presnosť na testovacej množine ako pôvodne mal.

Záver

Hlavným cieľom diplomovej práce bolo preskúmať využitie autoenkódera v klasifikačných úlohách založených na hlbokom strojovom učení. Porovnávanie prebiehalo vo viacerých rovinách – z pohľadu presnosti, posúdenia vplyvu množstva anotovaných dát a robustnosti voči neznámym dátam.

Pre porovnanie presnosti a vplyvu množstva anotovaných dát bolo navrhnutých viacero prístupov k tréningu, ktoré sme porovnali s klasickým tréningom klasifikátora. Celkovo boli navrhnuté 3 unikátne prístupy k tréningu. Základná myšlienka spočíva v tom, že autoenkóder a klasifikátor zdieľajú spoločnú časť a pri tréningu autoenkódera predtrénujeme túto časť a následne dotrénujeme klasifikátor. V prvom prístupe trénujeme striedavo jednu epochu autoenkóder a druhú epochu klasifikátor. V druhom prístupe zoberieme natrénovaný autoenkóder a dotrénujeme klasifikátor. V treťom prístupe súčasne, v rámci jedného kroku, trénujeme aj autoenkóder aj klasifikátor. V rámci jednotlivých prístupov sme trénovali s rôznym množstvom anotovaných dát. Prístupy boli vyskúšané na dvoch rôznych architektúrach, prvá navrhnutá pre dataset Fashion Mnist a druhá pre modifikovanú verziu Cifar 10. Zistili sme, že použitie autoenkódera má zmysel iba pri extrémne nízkom množstve trénovacích dát- 600 vzoriek, kde klasický tréning klasifikátora zlyháva, dochádza k pretrénovaniu siete. Už pri 3000 trénovacích vzorkách dosahuje klasický tréning podobné výsledky ako navrhnuté prístupy a pri vyššom množstve trénovaćih dát dosahuje klasický tréning o niečo vyššiu presnosť ako navrhnuté prístupy. Vysvetľujeme si to tým, že autoenkóder pôsobí ako prirodzený regularizátor a tvrdíme, že autoenkóder sa neučí všetky features potrebné pre klasifikáciu. Preto naďalej odporúčame používať klasický prístup tréningu.

Robustnosť voči neznámym dátam sme skúmali dvoma spôsobmi.

V prvom sme vyskúšali označiť vstup za neznámy na základe jeho rekonštrukčnej chyby. Zistili sme, že rekonštrukčná chyba hovorí o tom, ako náročné je daný vstup rekonštruovať a napriek tomu, že rôzne datasety majú rôzne priemerné rekonštrukčné chyby, nedokážeme na základe tejto hodnoty povedať, či vstup patrí alebo nepatrí do daného datasetu.

V druhom spôsobe sme navrhli postprocessing výstupu z klasifikátora, bez toho, aby sme klasifikátor ďalej trénovali. Nevýhodou je, že schopnosť označenia vstupov za neznáme získavame na úkor presnosti klasifikátora. Postprocessing bol skúšaný na klasifikátoroch z rôznych prístupov k tréningu a na rôznych datasetoch. Za najlepší výsledok považujeme, že postprocessing na klasifikátore natrénovanom na Fashion Mnist označil ~92 % vstupov z datasetu Mnist a Cifar Gray za neznáme, presnosť klesla z ~93% na ~83%. Neskôr sa ale ukázalo, že postprocessing nedáva dobré výsledky pre väčšiu architektúru a preto všeobecne nemožno postprocessing použiť.

Do budúcna vidím ako jednu z úloh predtrénovať autoenkóder na rozdielnom datasete, na akom bude trénovaný klasifikátor. Ďalšou úlohou by mohlo byť vyskúšať navrhované prístupy k tréningu na skutočne hlbokých sieťach, s niekoľko miliónmi parametrov. Zaujímavosťou by mohla byť vizualizácia features a na záver nájsť príčinu, prečo postprocessing v niektorých prípadoch funguje a niekedy nie.

Zoznam použitej literatúry

[1] arxiv.org: A detailed comparative study of open source deep learning frameworks [online] [cit 29.4.2020]. Dostupné na <https://arxiv.org/abs/1903.00102>

[2] github.com: Fashion Mnist [online] [cit 10.3.2020]. Dostupné na <https://github.com/zalandoresearch/fashion-mnist>

[3] cs.wikipedia.org: Pravidlo tří sigma [online] [cit 13.3.2020]. Dostupné na <https://cs.wikipedia.org/wiki/Pravidlo_t%C5%99%C3%AD_sigma >

[4] cs.wikipedia.org: Graf normálneho rozdelenia[online] [cit 13.3.2020]. Dostupné na <https://upload.wikimedia.org/wikipedia/commons/8/8c/Standard_deviation_diagram.svg>

[5] machinelearningmastery.com: How to configure image data augmentation [online] [cit 7.4.2020]. Dostupné na <https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/ >

[6] github.com: Tensorflow [online] [cit 7.4.2020]. Dostupné na <https://github.com/tensorflow/tensorflow>

[7] ketas.is: Keras: The Python Deep Learning library [online] [cit 7.4.2020]. Dostupné na <https://keras.io/>

[8] colab.research.google.com : Welcome to Colaboratory [online] [cit 7.4.2020]. Dostupné na <https://colab.research.google.com/notebooks/welcome.ipynb>

[9] cs231n.github.io: Neural networks 1 [online] [cit 29.4.2020]. Dostupné na <https://cs231n.github.io/neural-networks-1/>

[10] cs231n.github.io: Activation functions [online] [cit 29.4.2020]. Dostupné na <https://towardsdatascience.com/activation-functions-in-neural-networks-58115cda9c96>

[11] Peter Sinčák, Gabriela Andrejková : Neurónové siete, Inžiniersky prístup (1. diel). Dostupné na <http://www.student.ui.fei.tuke.sk/ZNS?action=AttachFile&do=view&target=NS1.pdf>

[12] wikipedia.org: Artificial neuron [online] [cit 29.4.2020]. Dostupné na <https://en.wikipedia.org/wiki/Artificial_neuron>

[13] ibooks.sk: Učenie neurónových sietí [online] [cit 29.4.2020]. Dostupné na <http://ibooks.sk/publ/13oravec/PDF/Kap02.pdf>

[14] Ian Goodfellow, Yoshua Bengio, Aaron Courville: Deep learning, 2016

[15] wikipedia.org: Mnist [online] [cit 29.4.2020]. Dostupné na <https://en.wikipedia.org/wiki/MNIST_database#/media/File:MnistExamples.png>

[16] machinelearningmastery.com: GAN [online] [cit 29.4.2020]. Dostupné na <https://machinelearningmastery.com/how-to-develop-a-conditional-generative-adversarial-network-from-scratch/>

[17] becominghuman.ai: Training Mxnet part 2 [online] [cit 29.4.2020]. Dostupné na <https://becominghuman.ai/training-mxnet-part-2-cifar-10-c7b0b729c33c>

Zoznam príloh

Príloha 1: CD obsahujúce zdrojové kódy pre vytvorenie aparátu pre tvorbu, tréning a analýzu použitých modelov, súbory Jupyter notebook používané pri tréningu a prácu v elektronickej podobe (PDF)