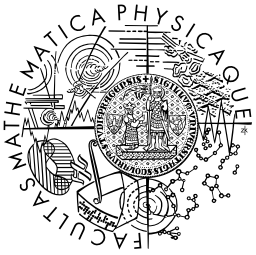


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

DIPLOMOVÁ PRÁCE

Bc. Jan Kubový

Rozpoznávání retinopatie ve snímcích sítnice pomocí strojového učení

Katedra teoretické informatiky a matematické logiky

Vedoucí diplomové práce: RNDr. Ing. Otakar Trunda, Ph.D.

Studijní program: Informatika

Studijní obor: Umělá inteligence

Praha 2024

Prohlašuji, že jsem tuto diplomovou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ....

Podpis autora

Tímto bych rád vyjádřil vděčnost mé přítelkyni a celé mé rodině, která při mě vždy stála a bezmezně mě podporovala při studiu. Dále děkuji doktoru Trundovi za vedení práce a za pomoc při jejím zpracování. Rovněž děkuji pracovníkům Institutu klinické a experimentální medicíny, IKEM, za poskytnutí dat, která mi umožnila pracovat na diplomové práci s potenciálem pomoci lidem.

Název práce: Rozpoznávání retinopatie ve snímcích sítnice pomocí strojového učení

Autor: Bc. Jan Kubový

Katedra: Katedra teoretické informatiky a matematické logiky

Vedoucí diplomové práce: RNDr. Ing. Otakar Trunda, Ph.D., Katedra teoretické informatiky a matematické logiky

Abstrakt: Ve spolupráci s Institutem klinické a experimentální medicíny (IKEM) a jejich historických dat z vyšetření byla natrénována konvoluční neuronová síť rozpoznávající diabetickou retinopatii ze snímků sítnice. Cílem práce bylo vytvořit model strojového učení, který bude možné využít ve zdravotnickém zařízení IKEM za cílem zjednodušit a případně zrychlit vyšetření. Součástí projektu byla také tvorba webové stránky, která umožňuje snadné spouštění natrénovaného modelu lékařem s pouze základní znalostí ovládání počítače. I přesto, že neuronová síť dosahuje dobrých výsledků, je třeba zdůraznit její omezenou univerzálnost z důvodu malé velikosti modelu a jednotvárnosti poskytnutých oftalmologických dat z jednoho typu fundus kamery. Navržené řešení bude podrobeno testování v provozním prostředí nemocnice. Neuronová síť není zamýšlena jako náhrada lékaře, ale jako nástroj, který mu může asistovat v diagnostickém procesu.

Klíčová slova: strojové učení zpracování obrazu retinopatie neuronové sítě diagnóza

Title: Retinopathy Detection in Retina Images using Machine Learning

Author: Bc. Jan Kubový

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: RNDr. Ing. Otakar Trunda, Ph.D., Department of Theoretical Computer Science and Mathematical Logic

Abstract: In collaboration with the Institute for Clinical and Experimental Medicine (IKEM) and leveraging their historical examination data, we developed a convolutional neural network trained to identify diabetic retinopathy from retinal images. The primary objective of this project was to establish a machine learning model applicable within the medical setting of IKEM, streamlining and potentially expediting the examination process. Additionally, we designed a user-friendly website to facilitate the straightforward utilization of the trained model by physicians possessing only basic computer skills. While the neural network demonstrates good results, it is crucial to underscore its restricted adaptability, attributed to the compact model size and the monotonic nature of ophthalmic data sourced from a specific type of fundus camera. The proposed solution is slated for testing in a real hospital operational environment. The neural network is not intended as a replacement for the physician, but as a tool that can assist the physician in diagnostic process.

Keywords: machine learning image processing retinopathy neural networks medical diagnosis

Obsah

Úvod **3**

1 Popis nemoci **6**
1.1 Symptomy diabetické retinopatie a její dopad na zdraví 6
1.2 Popis významu včasného odhalení a léčby diabetické retinopatie . 7

2 Strojové učení **9**
2.1 Strojové učení 9
2.2 Strojové učení v medicíně 10
2.3 Rozpoznání obrazu 11
2.4 Neuronové sítě 12
2.4.1 Hluboké neuronové sítě 13
2.4.2 Kapacita sítě 14
2.4.3 Regularizace 15
2.4.4 Parametry a hyperparametry 16
2.4.5 Aktivační funkce 17
2.4.6 Chybová funkce 18
2.4.7 Optimalizační algoritmus 19
2.4.8 Konvoluční vrstva 20
2.4.9 Výstupní vrstva 22
2.5 Vývoj zpracování obrazových dat 22
2.6 Postup při řešení zpracování obrazu 23
2.7 Validace modelu 25
2.8 Motivace pro použití strojového učení pro automatickou detekci retinopatie 28
2.9 Použitá vývojová platforma pro práci s neuronovými sítěmi 28

3 Přehled literatury **30**
3.1 Přehled předchozích studií o detekci retinopatie pomocí strojového učení 30
3.2 Diskuse o omezeních a výzvách stávajících přístupů 30

4 Metodika **32**
4.1 Přehled dostupných dat 32
4.2 Rozbor trénovacích dat 32
4.3 Popis algoritmů strojového učení použitých pro klasifikaci 33
4.4 Diskuse výkonnostních ukazatelů použitých pro hodnocení modelů 34
4.5 Automatické hledání hyperparametrů modelu 34
4.6 Možnosti vizualizace konvoluční vrstvy 34

5 Experimenty **36**
5.1 Obecné informace 36
5.1.1 Volba vstupního rozlišení 36
5.1.2 Použití naučeného modelu 38
5.1.3 Hyperband tuner 39
5.1.4 Augmentace dat - CLAHE 46

5.1.5 Využití veřejných dat 51
5.1.6 Složení více naučených modelů 53

6 Výsledky experimentů a analýza výkonnosti 58

7 Webový klient 59

8 Získané znalosti 61
8.1 Shrnutí klíčových zjištění studie 61
8.2 Využití výsledků studie v praxi 61
8.3 Návrhy pro budoucí práci na zlepšení výkonnosti navrhovaného přístupu 61

Závěr 63

Seznam použité literatury 65

Seznam obrázků 68

Seznam tabulek 70

A Přílohy 71
A.1 Zdrojové kódy 71
A.2 Snímky webové stránky 74
A.3 Obsah digitální přílohy 77

Úvod

Jednou z častých souběžných onemocnění u diabetu (cukrovky) je onemocnění nazvané diabetická retinopatie. Tato nemoc negativně ovlivňuje oční sítnici a bez léčby může postupně vést od zhoršení zraku až ke slepotě. Diagnóza probíhá manuálním zkoumáním snímků oční sítnice odborným personálem, což je časově náročné a závislé na dostupnosti lékařů.

Pokusíme se tento problém vyřešit nebo alespoň zjednodušit diagnostiku, v součinnosti s Institutem klinické a experimentální medicíny (dále IKEM), sestrojením nástroje, který bude schopen detekovat nemoc ze stejných snímků, které zkoumá lékař. Nebudeme se snažit o nahrazení lékaře, ale o vytvoření doplňkového nástroje, pomocí kterého by bylo možné pacienty roztřídit na ty, kteří potřebují pozornost lékaře a kteří ne. Tak by bylo možné obsloužit větší množství pacientů nebo věnovat více času jen těm, kteří to potřebují.

Využijeme metody strojového učení, konkrétně neuronové sítě, ke zpracování snímků a následnou kategorizaci nemoci. Bude třeba dbát zvýšené opatrnosti při hodnocení natrénovaného modelu, aby špatnou detekcí nedošlo k zanedbání péče. Z důvodu relativně malého počtu snímků bude náročné natrénovat neuronovou síť tak, aby řešení problému správně zobecnila a rozhodovala se primárně na základě symptomů.

Součástí řešení bude vytvoření webového portálu, přes který bude možné nástroj obsluhovat. Ideou je, aby tuto stránku používal personál starající se o pořízení snímků pacienta. Pokud by řešení vedlo k dostatečně univerzálnímu nástroji, mohl by sloužit nejen na pracovišti zařízení IKEM, ale i lékařským zařízením, které momentálně nedisponují odborným personálem schopným retinopatii rozpoznat a případné pacienty přeposlat do jiných vhodných zařízení.

Motivace

Umělá inteligence potažmo neuronové sítě se postupně osvědčují v různých oborech[1][2], kde mnohdy překonávají kvalitou lidské výkony (viz soutěž ILSVRC v sekci 2.5). Tomuto pokroku nebylo „ušetřeno“ ani zdravotnictví a začalo se objevovat množství prací, které se specializují na diagnostiku nemocí. Příkladem může být práce zaměřená na klasifikaci rakoviny kůže s přesností srovnatelnou s lidmi s až desetiletou praxí rozpoznávat nemoc[31]. Vzhledem k aktuálnímu vývoji lze předpokládat, že význam strojového učení bude při vyšetřování pacientů hrát čím dál větší roli a bude napomáhat ke kvalitnější péči. Využití strojového učení ve zdravotnictví se neomezuje pouze na ulehčení diagnóz nemocí, ale používá se i u hledání nových léků[3] nebo při překladech odborné literatury, čímž umožňuje využít znalostí i z textů v cizích jazycích.

Je třeba zdůraznit, že ne všechny pokroky v oboru umělé inteligence jsou pozitivní a lze je i zneužít. Jedním z problémů, se kterým jako společnost budeme

[1] https://news.vumc.org/2023/05/09/neural-networks-probe-proteins/

[2] https://medicalxpress.com/news/2023-10-ai-tool-similar-radiologists-chest.html

[3] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7640807/

muset bojovat, je generovaný obsah (zvuk, fotografie, video) špatně rozeznatelný od skutečnosti[4] a který již byl zneužit[5].

V rámci této práce se pokusíme vylepšit situaci při označování potenciálně nemocných pacientů diabetickou retinopatií. Rizikových pacientů je velké množství, vyšetření je zdlouhavé a vyžaduje specialistu v oboru. Proto by i třeba jen částečné zautomatizování vyšetření mohlo vést k výraznému zlepšení kvality života mnoha lidí, kteří mohou být aktuálně přehlédnuti.

Cíle

Za hlavní cíl jsme zvolili vytvoření systému, který by mohl být obsluhován i pracovníky, kteří nejsou primárně zaměřeni na problematiku rozpoznávání nemoci a kteří by dokázali systém použít pro předvýběr pacientů. Systém by pomáhal u rozdělování pacientů do dvou kategorií, a to na zdravé a na potenciálně nemocné, rizikové, kterým je třeba věnovat dodatečnou péči. Použití tohoto systému by mohlo zvýšit počet pacientů, které lze vyšetřit a následně jim poskytnout potřebnou péči. Momentálně je počet vyšetření limitován množstvím odborného personálu vyškoleného pro rozpoznávání retinopatie. Součástí hlavního cíle je potlačení falešně negativních chyb (systém rozhodne, že je pacient zdravý, i když tomu tak není). Tyto chyby jsou v medicínském prostředí velice nebezpečné, protože mohou vést k zanedbání zdravotní péče. Je třeba počítat s tím, že snaha o minimalizaci falešně negativních výsledků může vést ke zvýšení míry falešně pozitivních. To by pro naši práci neměl být velký problém, protože falešně pozitivní výsledek povede k dodatečnému vyšetření pacienta. Součástí výstupu by měl být i indikátor, v jaké míře si je systém jistý výsledkem.

Sekundárním cílem je zařazení pacienta do předem stanovených kategorií, což umožní případné upřednostnění pacientů s vážnějším průběhem. Další výhodou splnění tohoto cíle může být zvýšení důvěry v systém, jestliže bude schopen zařadit nemoc do stejných kategorií jako odborný personál.

Posledním cílem jsme zvolili grafické znázornění částí vstupního obrázku, podle kterého se systém rozhodoval. Splnění tohoto cíle by usnadnilo ověření správného chování systému a dále by to lékaři zjednodušilo další zpracování snímku.

Struktura

Tato práce je rozdělena do osmi kapitol. V první kapitole si popíšeme nemoc – diabetickou retinopatii – včetně jejich symptomů. Druhá kapitola se bude zabývat popisem strojového učení zaměřené na zpracování obrazových dat. Ve třetí kapitole si popíšeme již existující podobně zaměřené práce a na základě předchozí kapitoly se rozhodneme, jaký způsob řešení zvolíme my. Čtvrtá kapitola bude o rozboru dat a metodiky použité v práci. Následující pátá kapitola bude popisovat provedené experimenty. V šesté kapitole představíme výsledky experimentů

[4] `https://www.latimes.com/business/technology/story/2023-05-11/realtime-ai-deepfakes-how-to-protect-yourself`

[5] `https://www.respekt.cz/newsletter/umela-inteligence-je-uz-dostatecne-pokrocila-na-to-aby-narusila-volby`

včetně našeho popisu. Následná kapitola nás seznámí s webovým klientem, pomocí kterého se bude natrénovaný model používat a v osmé, závěrečné kapitole, shrneme získané znalosti touto prací a navrhneme směry, kterými by se mohly ubírat navazující práce.

1. Popis nemoci

1.1 Symptomy diabetické retinopatie a její dopad na zdraví

Diabetes mellitus, hovorově též cukrovka, je onemocnění, které sebou přináší mnoho zdravotních komplikací, z nichž jedna je diabetická retinopatie (DR). Rozpoznání tohoto onemocnění je hlavním tématem diplomové práce. Zrak, který je DR negativně ovlivněn, je pro člověka jedním z nejdůležitějších smyslů a jeho případné ohrožení není bráno na lehkou váhu. Není přesně jasné, které procesy vedou k diabetické retinopatii, ale odhaduje se, že hlavními faktory jsou délka nemoci diabetu a kolísání hladiny cukru v krvi[6].

Retinopatie negativně ovlivňuje oční sítnici, což je zadní část oční bulvy, na které jsou světlocitlivé buňky předávající informace zrakovým nervem do mozku. Veškeré tyto buňky musí být vyživovány, a proto je sítnice protkaná cévami. Nemoc způsobuje oslabování stěn a změn tvaru cév, čímž se vytváří riziko vzniku prasklinek a následného krvácení. V první fázi nemoci není zrak nijak ovlivněn a pacient si nemusí být onemocnění vědom. To je velice nebezpečné, protože v rané fázi může jednoduše dojít k podcenění a zanedbání léčby. V dalších fázích onemocnění dochází vlivem nedokrvení k odumírání malých oblastí sítnice, a to je spojeno s možným zhoršením zraku a vytvářením drobných slepých míst. Dalším rizikem je odchlípení sítnice, které znemožňuje správné prokrvení buněk. Existuje i pokročilá varianta nemoci – proliferativní diabetická retinopatie – charakteristická růstem nových cév v sítnici, které jsou křehčí, snadno praskají a způsobují tak větší krvácení. Tyto komplikace mohou vést až ke slepotě. Jen v České republice tato nemoc způsobila slepotu v 2.267 případech v roce 2016[34].

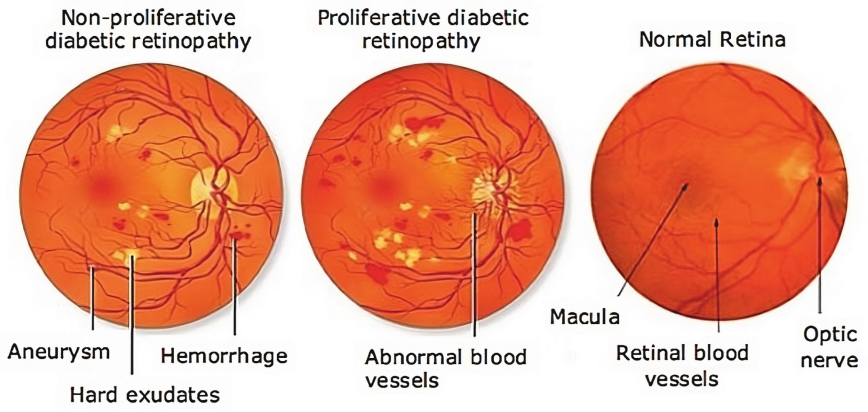
Odhaduje se, že v roce 2019 bylo na světě postiženo diabetem prvního nebo druhého stupně 9,3 % populace (463 milionů lidí) a očekává se, že v roce 2045 to bude až 10,9 % (700 milionů)[26]. Během prvních dvaceti let od diagnózy téměř každý pacient s diabetem prvního stupně a více jak 60 % pacientů s diabetem druhého stupně jsou také diagnostikování s diabetickou retinopatií[6]. Jestliže 3,6 % s prvním typem a 1,6 % s druhým typem diabetu a retinopatií oslepne, tak z toho můžeme odhadnout, že v roce 2045 bude celosvětově 6,7 milionů případů ztráty zraku kvůli zmíněné nemoci[6]. Nejnovější studie[16] dokonce odhaduje, že v roce 2050 bude 1,31 miliary lidí trpět diabetem. I za předpokladu, že by šlo „jen“ o druhý typ diabetu, oslepne téměř 12,6 milionů lidí.

Označení	**Forma**	**Dělení**
0	zdravý	
1	neproliferativní	mírná
2		střední
3		těžká
4	proliferativní	

Tabulka 1.1: Dělení diabetické retinopatie

Onemocnění DR rozlišujeme na tři formy, a to zdravého člověka značeného číslem 0, neproliferativní formu (NPDR) označenou čísly 1 až 3 podle dalšího dělení a proliferativní formu (PDR) značenou číslem 4. Závažnost neproliferativní formy se rozhoduje podle zastoupení následujících symptomů. Prvotním příznakem je výskyt mikroaneurysma (prasknutí kapilár), dále mikrohemoragiemi a hemoragiemi (výskyt krve mimo krevní řečiště), což vede k viditelným červeným tečkám nebo skvrnám na snímku sítnice. Dalším příznakem je změna cév, což se souhrnně označuje jako flebopatie a může jít o dilataci cév a nebo o změnu vzhledu (korálový vzhled, kličky). Poslední důležitá charakteristika je intraretinální mikrovaskulární abnormalita (IRMA), a ta se řadí mezi závažnější symptomy vyskytující se u těžších forem retionapatie. IRMA se vyznačuje fyzickými změnami na cévkách a lze ji lehce zaměnit za proliferativní cévy. Při určení stupně onemocnění se rozdělí oko na kvadranty a následně se určí, v kolika kvadrantech se vyskytují mikroaneurysma, flebopatie a IRMA. Z toho vznikne trojice čísel, například (4-1-0), které se převedou na zmíněný stupeň. Proliferativní znamená, že se na sítnici nebo u zrakového nervu vyskytují nové cévy. V této fázi dochází často ke krvácení z nově narostlých cév. Jedná se o nejtěžší formu onemocnění. Zároveň se může vyskytovat i fibrotická – odumřelá – tkáň vlivem špatného prokrvení[15].

Přesné rozdělení do jednotlivých kategorií se může lišit, ale pro pochopení problému a porozumění této práce není přesné rozdělené důležité. Pro úplnost uvádíme hranice mezi stupni u neproliferativní formy nemoci. Zdravý člověk bude vykazovat hodnoty vyšetření (0-0-0), počínající NPDR bude do (4-0-0), středně pokročilá varianta (4-1-0) a pokročilá varianta od (4-2-1).



Obrázek 1.1: Ukázka symptomů diabetické retinopatie (převzato [22])

1.2 Popis významu včasného odhalení a léčby diabetické retinopatie

Stejně jako u většiny nemocí, včasné odhalení vede k zastavení nebo zpomalení postupu nemoci. U diabetické retinopatie je nepříjemnou vlastností fakt, že počátek nemoci se snadno přehlédne. Dochází pouze k malým změnám na oční sítnici, které se mnohdy neprojeví zhoršeným zrakem nebo je zhoršení tak malé a postupné, že si toho pacient nemusí všimnout. Proto je třeba u lidí se zvýšeným

rizikem vzniku retinopatie provádět preventivní prohlídky. Existence nástroje, který by usnadnil diagnostiku, by pozitivně ovlivnilo dostupnost preventivních vyšetření. Nejnovější vědecké poznatky přisuzují diabetickou retinopatii ke kolísání cukru v krvi[20]. Pokud by se tato nemoc odhalila u pacienta v rané fázi, bylo by možné nasadit léčbu omezující kolísání cukru v krvi nebo upravit jídelníček pacienta se stejným záměrem.

U pokročilejší fáze nemoci je třeba sáhnout k invazivnímu způsobu léčby, jako je laserová terapie nebo chirurgický zákrok. Výhody včasné detekce jsou z tohoto pohledu zřejmé, zvláště s přihlédnutím ke skutečnosti, že nemoc může skončit trvalými následky. Obojímu se dá zabránit, nebo alespoň oddálit, při včasné léčebné terapii.

2. Strojové učení

2.1 Strojové učení

Pro správné porozumění fungování neuronových sítí je třeba popsat, jakým způsobem obecně funguje strojové učení. Hlavním cílem strojového učení je, na základě vytvořeného modelu – pomocí trénovacích dat – určit hodnotu výstupu u modelem dříve nespatřených dat. Každý takto vytvořený model má sadu parametrů, pomocí kterých se počítá výstup modelu. Může jít o hodnoty vytvořené trénováním modelu nebo také může jít o samotná trénovací data, jako tomu je u algoritmu K-nejbližších-sousedů[14]. Některé modely mohou mít i hyperparametry, kterými se ovlivňuje způsob vytváření samotného modelu. Parametry si můžeme představit jako hodnoty, které chceme nalézt trénováním a hyperparametry jako nastavení, jakým způsobem se k parametrům dostaneme. Parametry a hyperparametry popisujeme obecně, protože se pod těmito pojmy může skrývat nepřeberné množství typů proměnných od číselných po komplexní, jako jsou pravidla v rozhodovacím stromu.

Strojové učení lze dělit na dva druhy podle typu vstupních dat, a to na *učení s učitelem* (supervised) a *učení bez učitele* (unsupervised). U prvního zmíněného typu víme výslednou hodnotu ke každému vstupu. Můžeme proto při vytváření modelu využít této znalosti a zjednodušit hledání parametrů tím, že známe očekávaný výstup. Příkladem může být lineárně regresní problém, při kterém odhadujeme hodnotu funkce na základě vstupních dat. U tohoto problému budeme mít pozorování, jaké byly výsledky funkce u konkrétních hodnot. V tomto kontextu je slovem pozorování myšlena dvojice hodnot – vstup a k tomu odpovídající výstup. Úkolem modelu je predikovat hodnotu funkce i na vstupních datech, které se nevyskytovaly v trénovací množině. Dalším příkladem je problém kategorizace, u kterého vstup řadíme do předem známých kategorií, jako třeba rozpoznání číslice z obrázku. Druhým jmenovaným typem je soubor problémů, u kterého nedokážeme sdělit správný výsledek, protože sami nevíme, co by správný výsledek měl být. Příkladem problému *učení bez učitele* je shlukování, ve kterém máme v prostoru n bodů a chceme je rozdělit do m skupin podle podobnosti. Pokud bychom již věděli, do jaké skupiny patří konkrétní bod, nemusíme model vytvářet. V této práci se zabýváme problémem z kategorie *učení s učitelem*, protože na každý vstupní obrázek máme výsledek vyšetření, a proto se budeme dále v textu zabývat právě touto kategorií.

Při vysvětlování vytváření modelu používáme termín trénovací data, a proto je třeba tento pojem lépe definovat. Abychom mohli u modelu nějakým způsobem odhadnout jeho kvalitu a přesnost, musíme po jeho naučení vyzkoušet, jaké nám dá výsledky na nových datech. U druhu *učení s učitelem*, jak jsme si napsali výše, víme správné výstupy, a proto můžeme data, která máme k dispozici, rozdělit na trénovací dataset (množina vstupních dat s výsledky), který použijeme na vytvoření modelu a testovací dataset, pomocí kterého můžeme vyhodnotit kvalitu modelu. Ve skutečnosti se data většinou rozdělují na tři části, a to trénovací, validační a testovací dataset, kde validační dataset slouží k průběžnému zjišťování kvality modelu při jeho trénování. Na základě těchto průběžných výsledků se upravují hyperparametry modelu. Běžně se data rozdělují v poměru 70:15:15,

kde největší část připadá na trénovací data, ale můžeme se setkat na různé přístupy dělení dat. Existuje i takzvaná křížová validace (Cross validation), kde se data rozdělí na mnoho částí, ze kterých se postupně vybírá testovací část s tím, že zbytek dat je považován za trénovací. Pokaždé se nový model natrénuje na trénovacích datech a otestuje na testovacích. Poté se zjistí průměrná přesnost modelu ze všech měření. Tento způsob se většinou u trénování neuronových sítí nepoužívá, protože trénování bývá časově náročné a křížová validace požaduje trénování mnohokrát opakovat.

Zobecnění problému druhu *učení s učitelem* je modelování podmíněného pravděpodobnostního rozložení $y = P(Y|X)$, kde X odpovídá vstupním datům problému, Y výstupu sítě a výsledek y pravděpodobnosti, s jakou na vstup X náleží výstup Y. Následující rovnice značí konkrétní výstup sítě $\hat{y}$.

$$\hat{y} = argmax_y P(Y = y|X)$$

Problém je, že pravděpodobnostní rozdělení P neznáme a typicky disponujeme pouze omezeným množstvím dat vygenerovaných z P, které má kvůli menší mohutnosti odlišené rozdělení P'. Abychom snížili možnost, že P' bude příliš vzdálené od P, je třeba zajistit velký počet dat. Strojové učení se snaží minimalizovat rozdíl modelovaného pravděpodobnostního rozdělení P_{model} od P'. Jelikož neznáme P, výsledná kvalita modelu závisí kromě samotného učení i na trénovacích datech a jejich rozdílnému rozdělení P' od P. Z toho vyplývá, že model bude mít problém se správným určením výstupu u dat, které pochází z jiného rozdělení než je P'.

Označíme-li náhodnou proměnnou x z pravděpodobnostního rozložení P jako $x \sim P$, můžeme definovat křížovou entropii (Cross entropy)

$$H(P_{model},P') = -\mathbb{E}_{x\sim P'}[\log P_{model}(x)]$$

kterou bychom chtěli trénováním minimalizovat – co nejvíce se přiblížit pravděpodobnostnímu rozdělení modelu P_{model} k P'.

2.2 Strojové učení v medicíně

Využití strojového učení v medicínských oborech sahá do sedmdesátých let minulého století, kdy se začal tento obor rozvíjet a vznikaly první vědecké práce změřené na jejich reálné využití. Medicína je věda, ve kterém je třeba využívat informace z různých oborů a jednotlivec může jen velmi těžce dosáhnout hlubokých znalostí ve více různých oborech současně. Přesně na tyto případy se hodí využití nějaké formy strojového učení, která je schopná zpracovávat velké množství dat a případě kombinovat více druhů vstupů. Pojmem strojového učení nemyslíme jenom neuronové sítě, které se v současné době těší velké popularitě, ale i jiné modely jako jsou rozhodovací stromy či genetické algoritmy. Pokud bychom chtěli mluvit o zpracování obrazových dat (statické nebo video), nesmíme zapomenout na konvoluční sítě jako druh architektury neuronových sítí. Podrobnější rozbor neuronových sítí a zvláště konvolučních sítí si popíšeme v následující sekci.

V roce 2019 vyšel článek od doktora Nama a kol.[21] popisující použití neuronové sítě pro detekování maligních rakovinných uzlů v plicích z rentgenových snímků. K dispozici měli přes 43 tisíc snímků, ze kterých bylo 9225 s pozitivním

nálezem. Zajímavostí v této práci je použití trénovacích dat, u kterých byl znám výsledek šetření, ale pouze 37,2 % obrázků s rakovinným uzlem bylo anotováno (ručně označené části rentgenu, kde se nachází uzel). Jde o kombinaci *učení s učitelem* a *učení bez učitele* (semisupervised). Samotné získání anotovaných dat bývá časově velice náročné, a protože určování může provádět pouze odborník, získání dat může být i finančně náročné. Autoři však zmiňují výhodu použití částečně anotovaných dat – trénovaný model má větší volnost při hledání vzorů nemoci a může nalézt i takové části, které byly odborníky přehlédnuty. V závěru práce píší, že model dosahoval lepších výsledků než nespecializovaný personál a při použití v kooperaci s doktorem bylo dosahováno měřitelně lepších výsledků v přesnosti určení maligních rakovinných uzlů.

Další práce[18] se také zabývala detekováním rakovinných buněk z vizuálních dat. Konkrétně byl model učen na snímcích řezu sentinelových uzlin (uzlina nejpravděpodobněji napadená metastází rakoviny) po provedené biopsii k rozpoznání rakoviny prsu. Dataset obsahoval 399 snímků s anotovanými oblastmi s přesností na pixel, kde se nachází karcinom. Výsledkem modelu bylo zobrazení oblastí, které mohou obsahovat nebezpečné buňky. I přes malé množství trénovacích dat se autorům podařilo dosáhnout slušné přesnosti a v závěru píší, že při kooperaci algoritmu s lékařem se zkrátila doba zpracování výsledků na polovinu.

Posledním příkladem využití strojového učení v medicíně je práce autorů Chen, Chen, Yu, Chen, Chang, Hsu, Hsiao, Yeh a Chen [4] učící rozpoznání více typů rakoviny plic bez anotovaných vstupních obrázků. Jak jsme již psali, získání anotovaných obrázků je časově i finančně náročné. Pokud by se ukázalo, že tento typ vstupu není třeba a postačí k natrénování modelu pouze vstupní obrázky s výsledkem vyšetření, mohlo by to usnadnit tvorbu podobných diagnostických neuronových sítí. Dalším problémem anotovaných částí obrázku je nevýrazný přechod mezi zdravými a rakovinnými buňkami. To vede k výrazným rozdílům mezi odborníky, kteří anotaci prováděli. V závěru autoři zmiňují, že při použití metody Grad-CAM pro zobrazení oblastí, podle kterých neuronová síť dělala závěry, neoznačila pouze rakovinné buňky, ale i nekrotické části. Síť se „naučila“ rozpoznávat nekrotické buňky, protože se téměř vždy vyskytovali u rakovinných buněk. Problém je, že odumřelé části budou přítomné i z jiných důvodů (například infekce).

Ve všech popsaných studiích byl vytvořený model používán v kooperaci s odborným personálem. Nebyla snaha o nahrazení lidské práce, ale o zjednodušení nebo zpřesnění výsledku vyšetření. Stejné ambice máme i v naší práci.

2.3 Rozpoznání obrazu

Klasifikace může pracovat na různorodých vstupech a rozpoznání obrazu je pouze speciální případ, ve kterém je vstupem matice pixelů. Obrazová data mívají vysokou dimenzi, protože je třeba počítat každý barevný kanál pixelu za jednu vstupní hodnotu. Oproti tabulkovým datům (například anamnéza pacienta – věk, pohlaví atd.) je dimenze obrázků o mnoho řádů vyšší, a proto se tradiční metody (rozhodovací stromy[2], SVM[5], logistická regrese[1]) strojového učení nepoužívají přímo na vstupních datech. Napřed se používají metody pro snížení dimenze a extrahování důležitých znaků z obrázku, na kterých se již metoda strojového učení dá použít. Příkladem metod snižující dimenzi je extrahování hran metodou

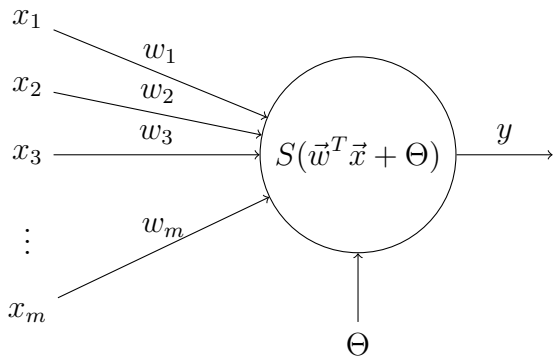
Sobel[13] nebo SIFT[19]. Problémem tohoto přístupu může být neznalost, jaké znaky je třeba z obrázků vyextrahovat. V dnešní době převládá použití konvolučních neuronových sítí na tento typ problému, protože jsou schopné potřebné znaky extrahovat pomocí učení automaticky[32].

2.4 Neuronové sítě

Základním stavebním prvkem neuronové sítě je neuron (obr. 2.1), který má m vstupů $\vec{x}$ a jeden výstup y. Dále obsahuje m vah $\vec{w}$, jednu prahovou hodnotu Θ (parametry, které se mohou měnit v průběhu učení) a aktivační funkci $S(x)$. Výstupní hodnota neuronu se počítá následovně:

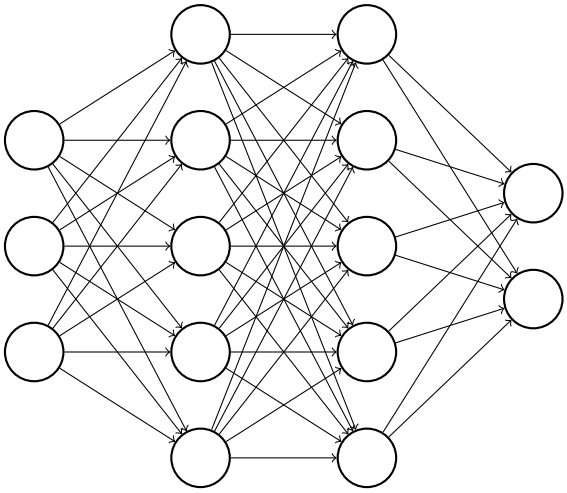
$$y = S(\sum_{i=1}^{m}(w_i x_i) + \Theta)$$

Nejjednodušším možným modelem neuronové sítě je použití právě jednoho neuronu. Pokud bychom zvolili jeho aktivační funkci jako prahovou (výstup je 0 a od nějaké prahové hodnoty je výstup 1) dostaneme model nazvaný *perceptron*. Propojením více neuronů vniká neuronová síť a její architektura může být rozmanitá. Různé způsoby propojení neuronů dává síti různé vlastnosti. Nás bude nejvíce zajímat dopředné, vrstvené neuronové sítě, které se snaží co nejlépe aproximovat neznámou funkci f se vstupem x. Slovo dopředná je zde z toho důvodu, že se výstup funkce počítá pouze ze vstupu x a není zde žádná zpětná cesta z výstupu sítě zpět na její vstup. Síť se zpětnou hranou se nazývá rekurentní a používají se zejména u zpracování textu nebo zvuku (mohou mít variabilní délku vstupu), což není náš případ.



Obrázek 2.1: Ilustrace neuronu

Standardní architektura neuronových sítí je sdružení neuronů do jednotlivých vrstev, kde vstupem neuronů jsou výstupy neuronů z předchozí vrstvy jak je vidět na obrázku 2.2. Neurony nejsou mezi sebou v jedné vrstvě nijak propojeny. Typicky je první vrstva označená jako vstupní a počet neuronů odpovídá vstupní dimenzi aproximované funkce. Naopak poslední vrstva je výstupní (z ní už nevedou žádné další spojení) a počet neuronů je nastaven podle toho, jaký má být výstup sítě. U aproximování funkce je výstupní vrstva složena pouze z jednoho neuronu, jehož výstupem je hodnota funkce. Vrstvy mezi vstupní a výstupní vrstvou se nazývají skryté.



Obrázek 2.2: Ilustrace vrstvené dopředné sítě

Z principu, jak se počítá výstup neuronové sítě, není těžké nahlédnout, že je složité zjistit, proč síť na vstup dala konkrétní výstup. Hluboká síť může dosahovat milionů parametrů a všechny se používají při výpočtu. Na rozdíl od rozhodovacích stromů, u kterých lze postupně procházet cestu uzly, u neuronové sítě toto není jednoduše uskutečnitelné. Jedním z momentálně zkoumaných řešení, které se samo nabízí, protože na vstupu máme velké množství dat, je natrénovat jinou neuronovou síť, která by interpretovala výstup sítě [1].

2.4.1 Hluboké neuronové sítě

Použije-li se v architektuře více skrytých vrstev, mluvíme o hlubokých neuronových sítích [7, str.168]. Ty se začaly používat především tlakem na zvýšení kvality modelu společně s rychlejším učením. Při zvětšování velikost jedné vrstvy se sice zvedala kapacita sítě, ale kromě pomalejšího učení nedocházelo k dramatickým zlepšením výsledků. Výhody více vrstev si můžeme představit tak, že každá vrstva se soustředí pouze na získání charakteristických znaků z vrstvy předchozí. První skrytá vrstva může reagovat na drobné detaily, ze kterých následující vrstva je schopna rozpoznat větší celky. Při použití konvolučních vrstev, které se hodí na zpracování obrazových dat, může být první vrstva schopna rozpoznat horizontální a vertikální čáry o rozměrech například 3x3 pixely. Další vrstva z těchto čar může například složit čtverec a tak dále. Neuronová síť s alespoň jednou skrytou vrstvou, drobným omezením na použité aktivační funkci a konečným počtem neuronů, je schopna aproximovat libovolnou spojitou funkci[8] a je tedy univerzálním aproximátorem. Na druhou stranu lze libovolnou funkci aproximovat i neuronovou sítí s více skrytými vrstvami při použití celkově menšího počtu neuronů[30].

Při zvyšování počtu skrytých vrstev začalo docházet k problému s učením. Čím byla vrstva blíže začátku, tím pomaleji se měnily její parametry. Problém byl způsoben použitím (v té době nejčastěji používané) aktivační funkce sigmoid. U každé vrstvy (bráno od poslední) se při zpětném průchodu počítá derivace funkce. Pokud je tato funkce často pod hodnotou 1, dochází k problému nazvaném

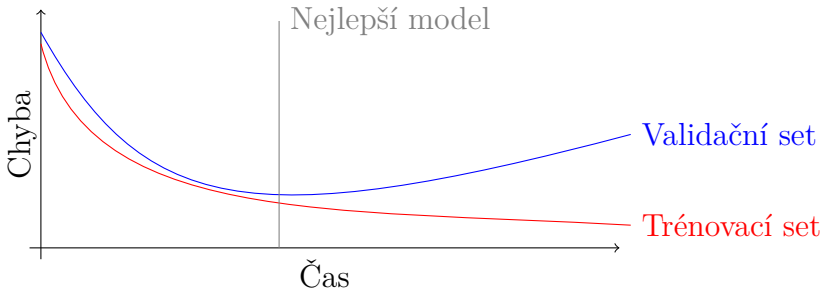
[1] `https://arstechnica.com/information-technology/2023/05/openai-peeks-into-the-black-box-of-neural-networks-with-new-research/`

mizející gradient. Naštěstí tento problém bylo poměrně snadné vyřešit změnou aktivační funkce na ReLU, která má u všech kladných hodnot derivaci právě 1.

Ukázalo se, že hluboké neuronové sítě dosahují podstatně lepších výsledků než jednovrstvé a začaly se hojně používat. To vedlo k experimentování s architekturou, které do té doby nebylo možné jednoduše provádět. Začala se zkoušet různá propojení vrstev. Objevily se sítě s propojením ob vrstvu, se všemi následnými vrstvami a tak dále. To vedlo většinou k lepším výsledkům a umožnilo to sítím použít nové cesty k lepšímu učení. Jedním z představitelů těchto vylepšení je DenseNet[9].

2.4.2 Kapacita sítě

Počtem parametrů se často nazývá velikost nebo také kapacita modelu a tu je třeba uzpůsobit řešenému problému. U neuronových sítí jsou parametry váhy propojení a prahy neuronů. Neexistuje ideální velikost modelu, protože pro každý problém a typ vstupních dat je třeba nalézt jinou vhodnou velikost. Jestliže se zvolí velikost příliš malá, model nebude mít možnost nalézt správné zobecnění hledaných vzorů (řešení) a nebude dosahovat dostatečné přesnosti. Na druhé straně leží situace, kdy je velikost modelu příliš velká a model není tlačen do zobecnění řešení, což vede k přeučení. Charakteristikou přeučení je malá chyba na trénovacím datasetu, ale výrazně větší chyba na validačním a testovacím datasetu (obr. 2.3). Při rozhodování o velikosti modelu se používá princip Occamovy břitvy, která nám říká, že bychom při stejných výsledcích měli použít menší model, protože pravděpodobně dokáže lépe zobecnit řešení. Kapacita sítě přímo úměrně souvisí s možností řešit komplexní problémy, ale není to jediná podmínka pro řešení. V souvislosti s pravděpodobnostním prostorem vyšší kapacita dovoluje síti modelovat detailnější „pravidla“ pravděpodobnosti.



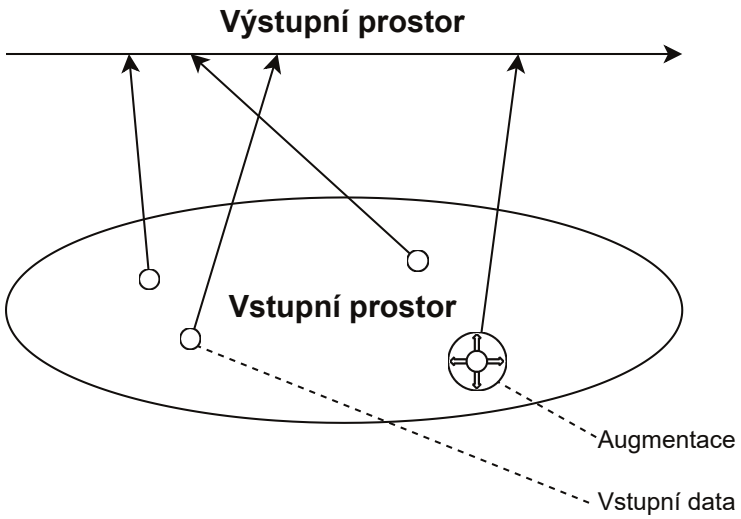
Obrázek 2.3: Průběh učení - ukázka přeučení modelu. Nejlepší model je v bodě, kde chyba dosahuje minima na validačním setu.

Je třeba si uvědomit, že počet parametrů není jediná vlastnost sítě, kterou můžeme upravovat a lepších výsledků můžeme dosáhnout s menším modelem s jinou architekturou. Například pro zpracování obrazových dat se dnes převážně používají již několikrát zmíněné konvoluční sítě, které si podrobněji popíšeme v jedné z následujících sekcí textu. Způsobem, jakým konvoluční vrstvy pracují, umožní síti lépe rozpoznávat vzory s menší kapacitou, než při použití klasické architektury.

2.4.3 Regularizace

Úzce spjatý problém s kapacitou sítě je přeučení (overfitting) modelu. K tomu dojde, když si model „zapamatuje“ trénovací vstupy – nedojde ke zobecnění řešení problému a správnému naučení potřebných vzorů (*features*) ze vstupu. Tomu je třeba předejít a jedním z možných způsobů je regularizace, která může být formou úpravy architektury sítě nebo úpravou vstupních dat, případě kombinace obojího.

Dalším důvodem může být příliš malý trénovací dataset, který umožní modelu jednodušeji si „zapamatovat“ všechny vstupy místo hledání vzorů. V takovém případě je nejlepší možnost získat více dat. Pokud to není možné, přichází na řadu augmentace dat – úprava vstupních dat tak, aby se nezměnil výstup.



Obrázek 2.4: Ilustrace vlivu augmentace dat. Představíme-li si výstup modelu jako projekci ze vstupního prostoru do výstupního, tak augmentace rozšiřuje okolí jednoho vstupu a model je nucen zobecnit řešení. Augmentace nenahradí nová trénovací data, ale umožní lépe využít existující.

Opačný problém než přeučení je nedoučení (underfitting) modelu. To přímo nesouvisí s regularizací, ale uvádíme ho zde pro úplnost. Jak název napovídá, problém spočívá ve špatně naučeném modelu a lze ho rozpoznat horší přesností na trénovacím datasetu.

Regularizace v síti

První možností regularizace je úprava architektury, při které měníme různé aspekty související s modelem. Jedním z možných příčin přeučení může být příliš veliká kapacita sítě, která není tlačena k nalezení vhodných vzorů, ale jednoduše se pokusí „zapamatovat“ si všechny vstupy z trénování. Zmenšením kapacity omezíme případně přeučení, ale je třeba si dát pozor na to, abychom zmenšili kapacitu úměrně ke složitosti problému a neznemožnili modelu problém řešit.

Dalším způsobem může být využití Dropout vrstev v modelu, které při trénování náhodně část svého vstupu zahodí a tím nutí model nespoléhat se vždy na stejné části sítě. Při použití těchto vrstev musíme zajistit dostatečnou kapacitu / propustnost sítě, protože model si musí nějaké informace duplikovat, aby

při výpadku výpočetní cesty nepřišel o možnost správně se rozhodnout. Při přidání této vrstvy je nutné se vypořádat s problémem, který použitím způsobíme. Zahazujeme-li části dat, tak do další vrstvy dorazí slabší signál (suma vstupu násobná vahami), než by přišel normálně. Logicky se Dropout vrstva nepoužívá mimo učení, a proto by signál na vstupu vrstev byl vyšší, než s čím síť normálně počítá. Způsoby řešení jsou dva, a to umělé zesílení nezahozených dat z Dropout vrstvy při učení, nebo zeslabení signálu mimo učení.

Vrstva Cutout funguje na podobném principu jako předchozí vrstva. Také zahazuje část dat, ale používá se přímo na vstupu, kde vyřízne část vstupu. Typické použití je vyříznutí části obrázku.

Regularizace dat

Pokud není možné získat větší množství trénovacích dat, někdy je možné si nová data rozšířit. Snažíme se upravovat existující data tak, aby se zachovala podstata, která odpovídá stejnému výstupu, ale zároveň, abychom vstupní data změnili. Tato technika se nazývá augmentací dat. Nejsnadněji se tento způsob představuje na obrázcích, které lze rozmanitě upravovat. Můžeme přidat šum, rozostřit obrázek, otočit, část vykrojit a podobně. Všemi těmito technikami dáváme nerunové síti trochu jiný vstup, a proto bude více nucena hledat společné znaky. Takto rozšiřovat data lze jen do jisté míry, protože pořád vycházíme ze stejných vstupních dat. Jinými slovy, stále by se nám hodilo mít co nejvíce různorodých dat pro učení, ale pomocí augmentace si můžeme trochu pomoct.

2.4.4 Parametry a hyperparametry

Neuronová síť má u každého neuronu několik parametrů (váhy u vstupu a práh), které se trénováním upravují. Úpravu parametrů se nejčastěji provádí trénovacím procesem zvaným zpětné šíření chyby (backpropagation), u kterého je snaha při každém kroku o malinko zlepšit kvalitu modelu. Pokud by se tento krok prováděl na základě všech trénovacích dat najednou, tak by v ideálním případě každý krok vedl k lepšímu modelu. Reálně se krok upravuje vždy na základě pouze části trénovacích dat (které se postupně prochází) a při jedné iteraci učení parametrů může dojít i k celkovému zhoršení. U modelu může dojít k příliš pomalému učení, přeučení, příliš velkému kroku změny parametrů, které vede k výraznému zhoršení kvality modelu a jiným nechtěným situacím.

Hyperparametry se označují parametry neuronové sítě, které se nemění při trénování, ale jsou nastavovány (většinou ručně) před každým učením. Hyperparametry upravují způsob trénování a jejich nastavení bývá složité, protože neexistuje pevný postup, jak upravovat tyto hyperparametry, aby se došlo k lepšímu modelu. Existuje mnoho doporučení, kterými je vhodné se řídit, ale bohužel to úspěch nezaručí. Hyperparametrem může být například nastavení velikosti zlepšujícího kroku při učení.

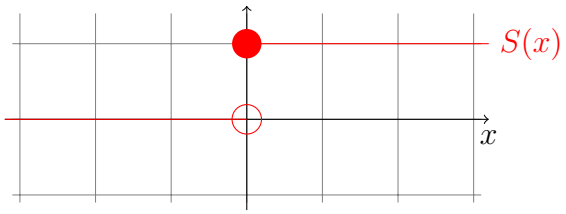
Na základě přesnosti na validačním setu se upravují hyperparametry tak, aby se došlo k co nejlepšímu modelu. Skutečná kvalita modelu se odhaduje až teprve podle přesnosti, se kterou model správně určí výstup u dat z testovací množiny. Hledání vhodných hyperparametrů je možné dělat ručně nebo využít nějaký automatický nástroj.

2.4.5 Aktivační funkce

Nedílnou součástí každého neuronu je aktivační funkce, která převádí zpracovaný vstup na výstup neuronu. Výpočet výstupu neuronu y je následovný $y = S(\sum_{i=1}^{m}(w_i x_i) + \Theta)$, kde $\vec{x}$ představují vstupní hodnoty, $\vec{w}$ vektor vah ke každé vstupní hodnotě, Θ práh neuronu a S jeho aktivační funkci.

U perceptronu jsme popisovali prahovou aktivační funkci, která měla výstup 0 do nějaké hodnoty α a poté 1. Vycházelo se z biologické předlohy neuronu, který se vybudí až od určité hodnoty signálu.

$$S(x) = \begin{cases} 0 & \text{if } x < \alpha \\ 1 & \text{if } x \geq \alpha \end{cases}$$

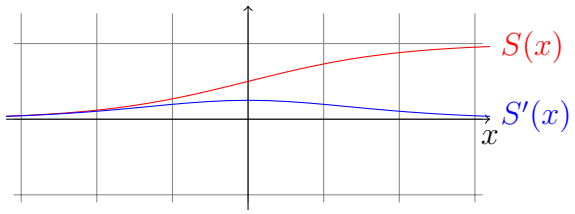


Obrázek 2.5: Prahová aktivační funkce ($\alpha = 0$)

Jelikož tato aktivační funkce má ostrý přechod, učení je složitější, protože dochází k velkým změnám výstupu. Vhodnější funkce je taková, která má pozvolný přechod.

Jednou z prvních používaných funkcí byla funkce sigmoid.

$$S(x) = \frac{1}{1 + e^{-x}}$$

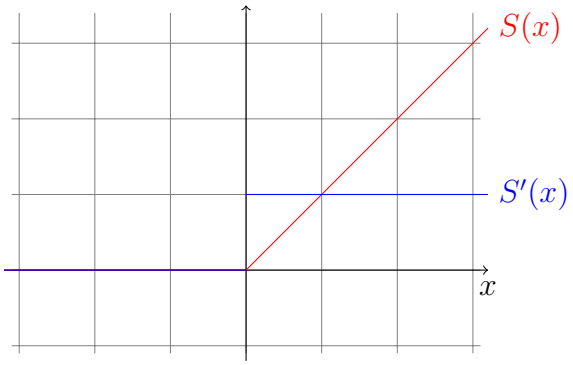


Obrázek 2.6: Sigmoid aktivační funkce a její derivace

Hlavním problémem u funkce sigmoid je mizející gradient, který se projeví při použití hlubokých neuronových sítí. Při trénovaní se počítají derivace této funkce, a které se při zpětném průchodu sítí mezi sebou násobí. Jestliže bude derivace vždy ostře menší než 1, bude se hodnota exponenciálně zmenšovat.

Proto se přišlo s funkcí, která nebude mít derivaci často menší než 1 a nebude trpět na popsaný problém, ReLU.

$$S(x) = x^{+} = max(0,x)$$

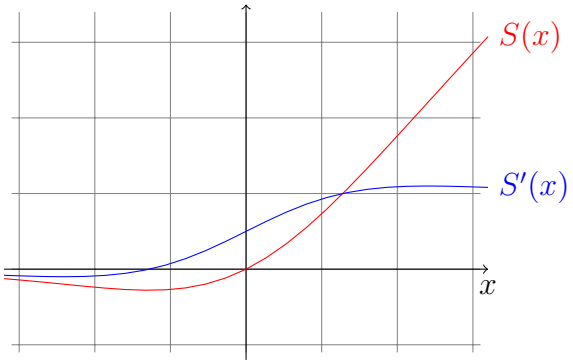


Obrázek 2.7: ReLU aktivační funkce a její derivace

Jelikož není ReLU shora omezená a její derivace je u kladných hodnoty vždy 1, odstranil se problém s mizejícím gradientem a ukázalo se, že je tato funkce vhodná i pro hluboké neuronové sítě čítající desítky vrstev. Další iterace aktivačních funkcí byly takové, které nemají ostrý přechod v bodě 0 nebo v záporných hodnotách jdou mírně do mínusu.

Příkladem takové funkce je SiLU (swish), která má následující předpis, kde σ označuje funkce sigmoid.

$$S(x) = x\sigma(x)$$



Obrázek 2.8: Swish aktivační funkce a její derivace

2.4.6 Chybová funkce

Pomocí chybové (loss) funkce se měří přesnost aproximované funkce oproti správnému výstupu funkce. Volba vhodné chybové funkce vzhledem k řešenému problému je klíčová, protože od ní se odvíjí jak průběh učení, tak následná úprava hyperparametrů. V našem případě se dá uvažovat o dvou různých typech loss funkce a to pro regresní formu neuronové sítě nebo kategorizační formu (výstupem je jedna kategorie z více možných).

V případě regresní optimalizace porovnáváme dvě čísla – výstup sítě a správný výstup. Nejjednodušší možný výpočet chyby je absolutní hodnota rozdílu (Mean Absolute Error MAE). Nejčastěji se však používá druhá mocnina rozdílu (Mean Square Error MSE), která větším způsobem penalizuje vzdálené hodnoty a naopak zmenšuje chybu, pokud jsou si hodnoty blízko. Existují i složitější funkce,

jako je třeba druhá mocnina z rozdílu logaritmů (Mean Square Logarithmic Error MSLE).

$$L_{MAE}(y,\widehat{y}) = |y - \widehat{y}|$$
$$L_{MSE}(y,\widehat{y}) = |y - \widehat{y}|^2$$
$$L_{MSLE}(y,\widehat{y}) = |(\log y + 1) - (\log \widehat{y} + 1)|^2$$

U kategorizačních úloh záleží, jestli je problém binární – rozhoduje se mezi dvěma třídami – nebo zda jde o kategorizaci více tříd. U binární kategorizace se používá dominantně binární křížová entropie (Binary Cross Entropy BCE), u které se očekává, že výstup bude jeden a bude v rozmezí [0,1], čehož lze docílit použitím sigmoid aktivační funkce na výstupním neuronu. U SVM se často používá Hinge funkce (HL), která funguje na celém oboru hodnot a očekává se, že správným výstupem jsou hodnoty 1 a -1. Tato funkce penalizuje hodnoty, které mají opačné znaménko a zároveň penalizují i ty, které mají sice správné znaménko, ale hodnota je v rozmezí $(-1,1)$. Důvodem je snaha separovat třídy prázdným pásmem.

$$L_{BCE}(y,\widehat{y}) = -(y \log \widehat{y} + (1 - y) \log (1 - \widehat{y}))$$
$$L_{HL}(y,\widehat{y}) = \max (0, 1 - y * \widehat{y})$$

Poslední námi jmenovanou úlohou je kategorizace s více třídami, kde výstupní vrstva obsahuje více neuronů, typicky jeden neuron na třídu. Obvykle se používá kódování 1 z n (one hot encoding), který převádí třídu na vektor s nulami a právě jednou jedničkou na místě, která patří dané třídě. Při výpočtu loss funkce se proto použijí dva vektory, a to výstup neuronů a zakódovaná správná třída. Většinou se na výstupní vrstvě neuronové sítě používá aktivační funkce softmax, která převede libovolný vektor (může obsahovat i záporné hodnoty a součet nemusí být roven 1) na rozdělení pravděpodobnosti příslušnosti do tříd. Po této operaci jsou všechny hodnoty ve vektoru v rozmezí [0,1] a součet je roven 1. Loss funkce porovnávají dvě pravděpodobnostní rozdělení a vrací „velikost odlišnosti“. Funkce, které si zde představíme jsou křížová entropie (Cross entropy CE) a Kullbackova-Leiblerova divergence (KL).

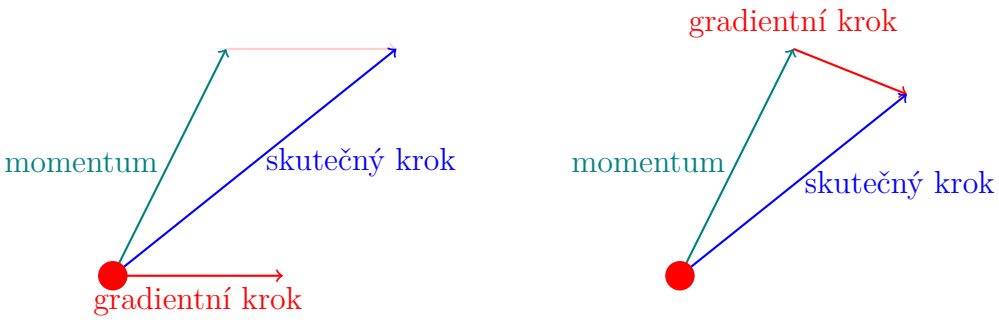
$$L_{CE}(\vec{y},\vec{\widehat{y}}) = -\sum_i y_i \log \widehat{y}_i$$
$$L_{KL}(\vec{y},\vec{\widehat{y}}) = -\sum_i y_i \log (\frac{y_i}{\widehat{y}_i})$$

2.4.7 Optimalizační algoritmus

Máme-li definovanou architekturu sítě a loss funkci, můžeme si popsat, jakým způsobem probíhá učení sítě. Je třeba si uvědomit, že výpočet chybové funkce ve skutečnosti závisí na všech parametrech sítě, protože ty tranzitivně určují její výsledek. Trénování je hledání takových hodnot parametrů, pomocí kterých bude chybová funkce dosahovat minima. Jsou situace, kde se hledá maximum, ale my se budeme dál bavit pouze o minimalizaci loss funkce.

Algoritmus používaný v drtivé většině trénování neuronových sítí je nějaká variace na gradientní sestup (gradient descent). Chybová funkce a parametry sítě nám dávají prostor s dimenzí shodnou s počtem parametrů, kde hledáme minimum. Náhodným nastavením parametrů dostaneme bod v tomto prostoru reprezentující stav modelu a úpravou parametrů se v tomto prostoru pohybujeme. V tuto chvíli předáme na vstup trénovací vzor, zjistíme výsledek a spočítáme chybu pomocí loss funkce. Následně vypočítáme parciální derivaci chybové funkce přes všechny její parametry. Tím získáme směr ve zmiňovaném prostoru, jak máme upravit parametry, aby se chyba zmenšila. Implementaci počítání parciálních derivací probíhá zpětným průchodem sítě (backpropagation)[25], ale tento výpočet nebudeme v práci podrobněji popisovat. Existují i jiné způsoby trénování, ale používají se ve velmi specifických situacích, například pokud není možné spočítat derivace chybové funkce, a tudíž nelze gradientní sestup použít. Příkladem takového učení jsou genetické algoritmy.

Existují různá vylepšení gradientního sestupu tak, aby bylo učení co nejrychlejší, ale zároveň stabilní. Jedním ze zlepšení bylo přidání momentu do změny vah tak, aby se braly v potaz i minulé kroky. To zamezí oscilacím při úpravě parametrů a vede to ke stabilnějšímu učení. Existuje i varianta momentu nazvaná Nesterov momentum[23], která místo toho, aby přidala moment k učícímu kroku, počítá parciální derivace z místa, kam by se dostala po aplikování momentu 2.9.



Obrázek 2.9: Rozdíl mezi použitím momentu po výpočtu gradientního kroku (vlevo) a aplikování momentu před výpočtem (vpravo, Nesterov)

Při výpočtu gradientního kroku je potřeba určit, jak velký má krok být. Díky gradientu víme směr, ale to nám nic neříká o struktuře prostoru, ve kterém se pohybujeme. Při volbě malého kroku bude trénování trvat dlouho a při moc velkém kroku můžeme postoupit příliš daleko a skončit v horší pozici. Neexistuje žádná ideální velikost kroku, protože záleží na mnoha faktorech včetně struktury samotného prostoru.

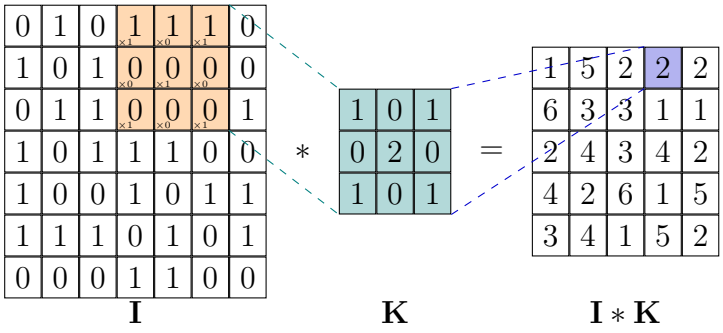
Ve většině optimalizačních algoritmech je možné nastavit hyperparametr velikosti kroku (learning rate) a zároveň některé algoritmy samy upravují tuto velikost za běhu s ohledem na průběh učení. Adaptace může probíhat například tak, že při stejném směru gradientních kroků je možné krok prodlužovat a při velkých změnách směru naopak zmenšovat.

2.4.8 Konvoluční vrstva

Hlavním konceptem konvoluce je sdílení vah u všech neuronů na jedné vrstvě. Při definování vrstvy se určí rozměry jádra (kernel, často označovaný jako K)

s počtem filtrů. Dimenze jádra je stejná, jako je dimenze předchozí vrstvy. Je to z toho důvodu, že se v podstatě jedná o váhy k jednomu neuronu z „posuvného okénka“ z vrstvy předchozí. Na obrázku 2.10 je příklad výpočtu, kde vstupní vrstva I má dimenzi (7,7,1) a jádro K (3,3,1). Na příkladu je patrné „okénko“, které se zdánlivě posouvá po vstupu a vypočítává výstup. Ve skutečnosti probíhá výpočet simultánně, protože se váhy rozkopírují pro každé výstupní políčko. To je ovšem implementační detail použité programátorské knihovny. Počet parametrů je roven (výška jádra $*$ šířka jádra $*$ hloubka předchozí vrsvy $+ 1$) $*$ počet fitrů, kde $+1$ značí práh neuronu. Typicky je možné zvolit velikost korku posunu „okénka“. V příkladu je nastaven posun o 1, ale je možné mít posun větší a tím zahodit některé sloupce a řádky. Dále lze nastavit, jestli se má výsledek počítat pouze z dat z předchozí vrstvy (často označování slovem *valid*) a nebo, jestli se má vstup rozšířit tak, aby při nastaveném posunu o 1 vznikl výsledek se stejnými rozměry jako vstup (*same*).

Hlavními výhodami tohoto způsobu výpočtu je malý počet trénovacích parametrů a fakt, že je výpočet invariantní vzhledem k pozici na vstupu. Proto se tato vrstva výborně hodí pro zpracování obrazových dat, neboť často potřebujeme vědět, bez ohledu na umístění, jestli na obrázku něco je nebo ne. V případě propojení neuronů každý s každým mezi vrstvami není tato vlastnost jednoduše možná, protože mohou všude být rozdílné váhy.

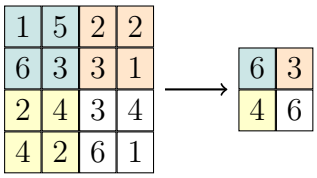


Obrázek 2.10: Ukázka výpočtu konvoluční vrstvy

Posloupností více po sobě jdoucích konvolučních vrstev dochází k extrakci charakteristických znaků (*features*), ze kterých si mohou další vrstvy odvodit, jaké objekty jsou na vstupu.

Pooling vrstva

Hlavním účelem pooling vrstev je redukce dimenze bez použití parametrů. Většinou se používá redukce buďto za použití maxima nebo průměru (obr. 2.11). Při definování vrstvy se zvolí rozměry oblasti, která se bude redukovat, typicky jde o jednotky jako 3 nebo 5. Poslední rozměr výstupu se zachovává stejný, jako je u předchozí vrstvy. Při průchodu vrstvou se rozdělí vstup na části podle zvolené velikosti oblasti a na každou se aplikuje zvolená funkce. Existují i globální varianty pooling vrstev, které redukují vstup přes celou jeho velikost (opět se zachováním poslední dimenze).



Obrázek 2.11: Ukázka pooling vrstvy o velikosti (2,2) používající maximum. Z původní dimenze (4,4,1) vznikla (2,2,1)

2.4.9 Výstupní vrstva

Poslední vrstva v neuronové síti se jmenuje výstupní a její velikost je určena typem úlohy, kterou řešíme. V případě regresních problémů, kde výstupem je jedno číslo, použijeme právě jeden neuron bez aktivační funkce. U kategorizačních problémů máme počet neuronů stejný jako je počet tříd, do kterých chceme vstup roztřídit. V takovém případě se často používá aktivační funkce softmax, která převede výstup neuronů na pravděpodobnostní rozdělení. Speciální případ je rozdělení do dvou kategorií, kde lze použít pouze jeden neuron s aktivační funkcí sigmoid, protože příslušnost do druhé kategorie můžeme dopočítat $(1-y)$. U obrazového výstupu definujeme výstupní vrstvu pomocí neuronů shodnou s dimenzí požadovaného obrázku a aktivační funkci upravíme podle potřeby. Pokud bychom chtěli získat osmi bitové RGB kanály, můžeme použít sigmoid, výsledek vynásobit 255 a oříznout na jeden bajt.

2.5 Vývoj zpracování obrazových dat

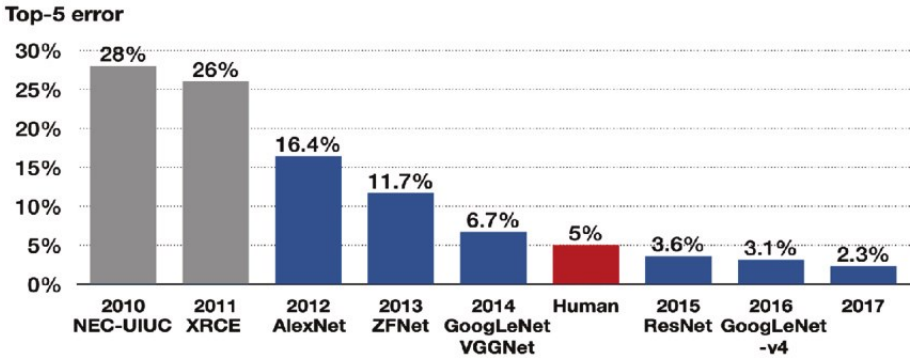
Na počátku se obrázky zpracovávaly pomocí jednoduchých metod, jako byla detekce hran či hraničních bodů (*features extraction*), ze kterých se na základě pravidel odhadoval výsledek[19]. Tyto metody a pravidla byly ručně psány expertem na danou problematiku.

Následujícím vylepšením bylo místo ručně definovaných pravidel použití klasifikačních algoritmů v čele s SVM a k-nejbližších sousedů. Stále bylo třeba ručně definovat získání rysů (*features*) obrázků, ale následná kategorizace byla určena strojově.

V roce 1998 publikoval Yann LeCun [33] dataset MNIST Dataset obsahující ručně psané číslice v rozlišení 28x28 pixelů. Z tohoto souboru obrázků se stala *hello world*[2] úloha strojového učení. Na dlouhou dobu se dataset stal metrikou při poměřování kvality architektur pro rozpoznání obrazových dat. V dnešní době je již dataset příliš jednoduchý a existují obsáhlejší soubory obrázků s mnohem větším rozlišením, které se využívají u klasifikačních problémů.

S rozšířením výkonných počítačů a opětovné popularitě neuronových sítí se začaly vytvářet hluboké sítě zpracovávající obraz a postupně se opustily ručně definované metody extrahující hlavní rysy obrazu. Následně získaly velkou popularitu konvoluční sítě (ConvNet, AlexNet, VGGNet), které dosahovaly velké přesnosti a překonaly v některých oblastech lidské schopnosti (obr. 2.12).

[2]Takto je v programování označována jednoduchá úloha, kterou se většinou začíná při učení nového programovacího jazyka



Obrázek 2.12: Graf přesnosti algoritmů, které vyhrály soutěž ILSVRC v letech 2010-2017. Modré sloupečky označují konvoluční sítě. Převzato z práce[12]

V dnešní době je stále nejrozšířenější způsob zpracování obrazu pomocí konvolučních sítí, které jsou schopny rozpoznávat složité vzory v reálném čase a v některých oblastech významně překonaly schopnosti člověka. Výpočet se přesunul z procesorů na grafické karty, které jsou vhodnější na masivní paralelizaci výpočtů potřebné při trénování a použití neuronových sítí. Již existují i čipy speciálně navržené pro akceleraci neuronových sítí.

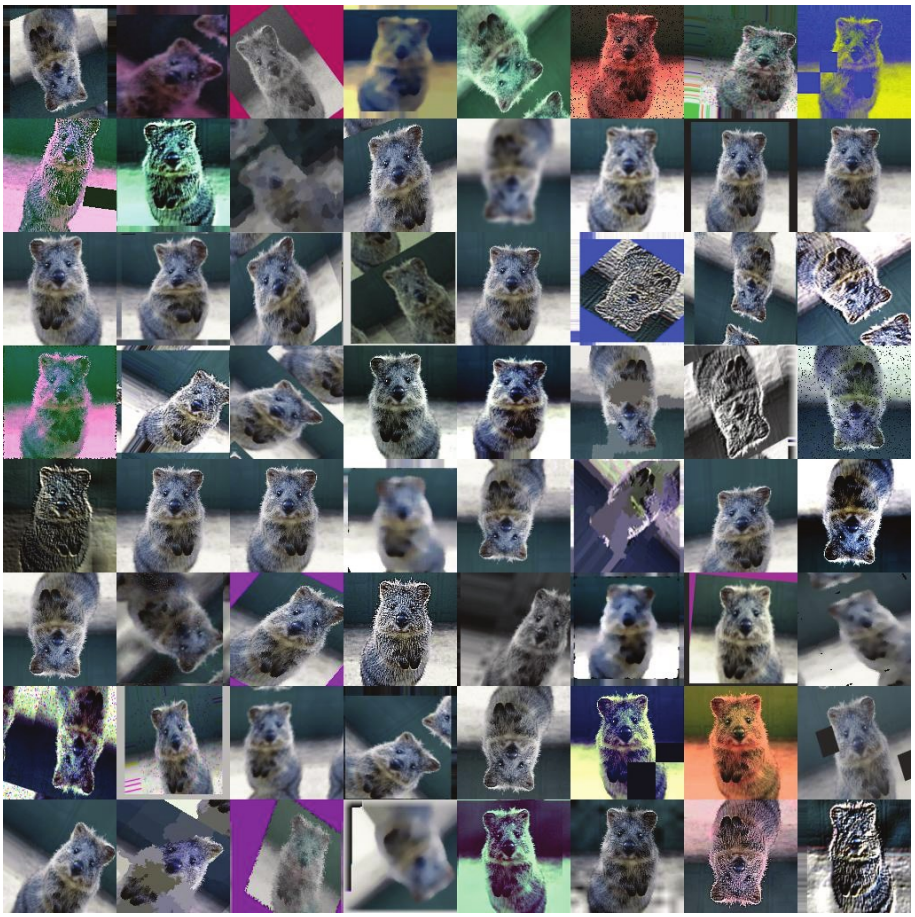
2.6 Postup při řešení zpracování obrazu

Před samotným návrhem algoritmu, který se bude používat na zpracování obrazu, je třeba si definovat, jaký chceme získat výstup. Může jít o rozpoznání textu z fotografie nebo třeba určení, jaká osoba je na obrázku. Podle zvoleného výstupu budeme vytvářet samotnou architekturu sítě, zejména pak její výstupní vrstvu. V rámci výběru výstupu je zapotřebí i určit, jakou případnou chybu či přesnost, s ohledem na vybrané metriky, akceptovat.

Neméně důležitým krokem je shromáždění dostupných dat a jejich příprava. Minimálně je třeba data rozdělit na tři části nebo si definovat způsob, jak se budou data dělit. Typicky se data dělí na trénovací (trénování modelu), validační (výběr hyperparametrů) a testovací část (odhad kvality). Je možné přistoupit na křížovou validaci, ale ta se u neuronových sítí běžně nepoužívá kvůli vysoké časové náročnosti.

Je třeba se zamyslet, jaké augmentace dat je vhodné využít, aby se částečně zvýšil počet dat a aby se model po natrénování stal více robustním. Můžeme vybírat z náhodného otočení a překlopení obrazu, přidání šumu, změny barevných tónů a mnoho dalších způsobů. Ukázku augmentací lze vidět na obrázcích 2.13 a 2.14. Musíme vycházet z toho, jaká máme data k dispozici a jakým směrem chceme zvyšovat odolnost modelu. Zároveň chceme zachovat extrahovanou informaci, aby měl model vůbec možnost ji nalézt. Přílišná snaha o silné úpravy obrázků může způsobit zhoršení celkové kvality modelu, protože se může vytratit podstatná informace z augmentovaných snímků.

Obrázek 2.13: Příklad vstupního obrázku k různým augmentacím. Převzato z projektu imgaug (`https://github.com/aleju/imgaug`)



Obrázek 2.14: Příklad různých augmentací obrázku. Převzato z projektu imgaug (odkaz stejný jako na obrázku 2.13)

U návrhu architektury sítě se doporučuje začít s již ozkoušenými kombinacemi a velikostmi vrstev a vhodně zvolit celkovou kapacitu modelu vzhledem k datům a složitosti problému (viz 2.4.2). Při zpracování obrazu můžeme vycházet z několika po sobě napojených konvolučních vrstev kombinované s pooling vrstvami, které se budou starat o automatické nalezení a extrahování vzorů. Následně přidat jednu nebo více plně propojených vrstev fungujících jako paměť modelu „skládající“ nalezené vzory. Na závěr bude definovaná výstupní vrstva, která vychází z vybra-

ného výstupu modelu. V případě kategorizačního problému má výstupní vrstva počet neuronů podle počtu kategorií.

Při výběru aktivační funkci u vrstev můžeme zvolit hojně používanou funkci ReLU, která je vhodná pro hluboké neuronové sítě. Jako optimalizační algoritmus můžeme zvolit například Adam (z Adaptive moment estimation), který je stejně jako dříve jmenovaná aktivační funkce ověřena mnoha lety používáním.

Model může mít několik hyperparametrů, které je třeba nastavit. Může se jednat o velikosti filtrů u konvolučních a pooling vrstev, velikost plně propojených vrstev, optimalizační algoritmus, velikost učícího kroku nebo třeba i aktivační funkce. Existují nástroje (nazvané tunery), které hledají kombinace hyperparametrů, při kterých je chyba na validačním setu dat nejmenší. Hledání probíhá zkoušením různých kombinací a následné trénování modelu, což bývá časově náročné.

Po definování modelu spolu s jeho hyperparametry můžeme přistoupit k samotnému trénování a případnému složení více natrénovaných modelů dohromady. Po tomto kroku změříme chybu na test setu a na jeho základě můžeme úlohu prohlásit za vyřešenou nebo se vrátit k definování architektury a proces opakovat.

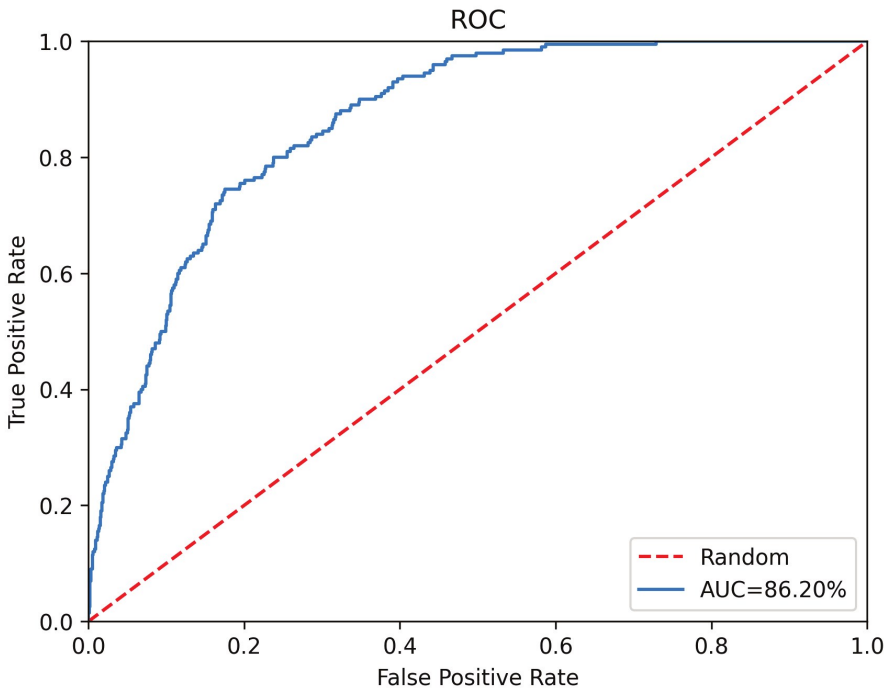
2.7 Validace modelu

Změření kvality modelu je kriticky důležité pro správný postup při řešení problému. Zjištěná kvalita modelu pomáhá zvolit další průběh jeho úprav tak, abychom se dostali k co možná nejlepšímu řešení. Odvíjí se od něj rozhodnutí, jestli je třeba měnit architekturu sítě, zaměřit se na regularizaci dat nebo se zaměřit na úpravu hyperparametrů.

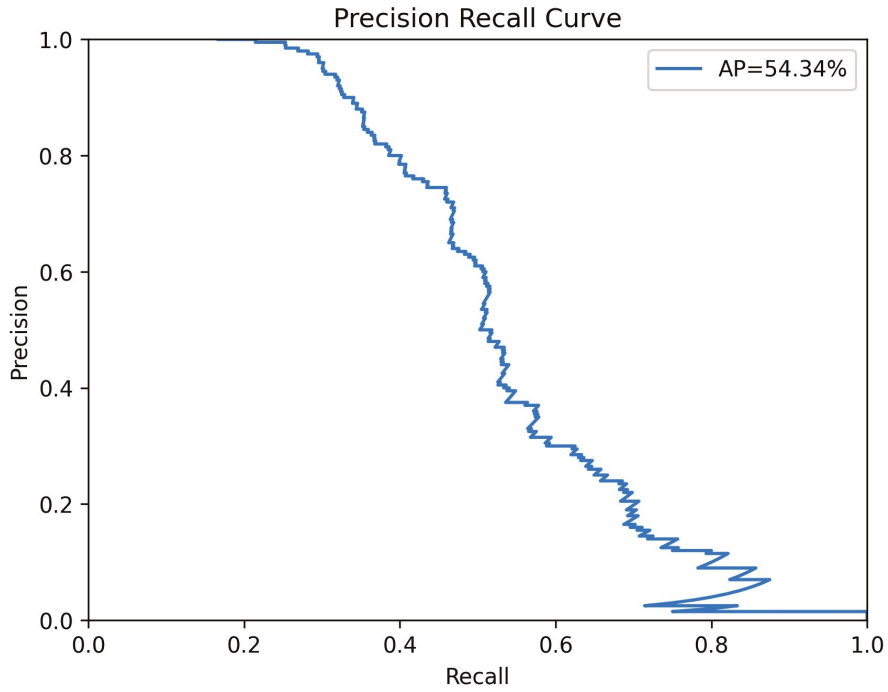
Je třeba si uvědomit, že ne každá chyba má stejnou váhu, a proto je třeba tomu uzpůsobit učení modelu. Typickým příkladem je diagnóza nemoci u pacienta. V tomto případě vyjde-li test falešně pozitivně (False Positive FP), je možné pacienta dále vyšetřovat a odhalit, že ve skutečnosti došlo k chybě u diagnózy. V opačném případě, dostane-li výsledek falešně negativní (False Negative FN), již nebude pacient dále vyšetřován a chyba se tak snadno neodhalí. S touto nerovností chyb je třeba počítat a případně uzpůsobit model tak, aby minimalizoval falešné negativní výsledky i přes možné zhoršení celkové přesnosti nebo zvýšení falešné pozitivity.

Hodnotíme-li model třídící vstup na dvě kategorie (pozitivní/negativní) používají se základní metriky (obr. 2.17), ze kterých je dále možné vypočítat pokročilejší. My se budeme hlavně zaměřovat na *Sensitivity* (rovnice 2.1), $Specificity$ (rovnice 2.3), *Accuracy* (rovnice 2.6) a F_1 *score* (rovnice 2.10).

Přehledným ukazatelem kvality binárního klasifikátoru je ROC křivka ukazující míru správně zařazených vzorů do první a druhé kategorie v závislosti na zvolené hranici reprezentující přechod mezi kategoriemi. K tomu se váže metrika AOC označující plochu pod ROC křivkou. Příklad takového grafu je na obrázku 2.15. Dokonalý klasifikátor by měl hodnotu AOC 1,0 (100 %).



Obrázek 2.15: Ukázka ROC křivky (1000 negativních výsledků a 200 pozitivních). Negativní výsledky byly náhodně vygenerovány pomocí normálního rozdělení se střední hodnotou 0,1 a směrodatnou odchylkou 0,61. U pozitivních výsledků jsme použili střední hodnotu 0,9 a směrodatnou odchylku 0,45



Obrázek 2.16: Ukázka PRC křivky (1000 negativních výsledků a 200 pozitivních). Negativní výsledky byly náhodně vygenerovány pomocí normálního rozdělení se střední hodnotou 0,1 a směrodatnou odchylkou 0,61. U pozitivních výsledků jsme použili střední hodnotu 0,9 a směrodatnou odchylku 0,45

U nevyváženého datasetu může ROC křivka a AUC zkreslovat kvalitu modelu, protože nebere v potaz False Negativ chyby a soustředí se pouze na pozitivně určené výsledky. Existuje jiná křivka – Precision Recall Curve – PRC, která lépe vizuálně reprezentuje kvalitu modelu vzhledem k oběma typům chyb. Obrázek 2.16 zobrazuje PRC křivku pro stejný ukázkový klasifikátor jako je na obrázku 2.15. Velký rozdíl zde právě hraje nevyváženost dat. Stejně tak jako u ROC existuje AOC-PR nebo též AP, což je číslo udávající plochu pod křivkou.

Při vizualizaci kvality modelu, který rozřazuje vstup na více kategorií, je třeba zmínit matici záměn (Confusion matrix), která přehledně zobrazuje, jaký měl model výstup u každé kategorie. Tuto metodu použijeme u modelů, které budou udávat přímo kategorii nemoci a ne jenom výsledek, jestli je pacient nemocný nebo ne.

ALL = P + N	Predikce pozitivní (PP)	Predikce negativní (PN)
Skutečně pozitivní (P)	True Positive (TP)	False Negative (FN)
Skutečně negativní (N)	False Positive (FP)	True Negative (TN)

Obrázek 2.17: Základní metriky

$$\textbf{Sensitivity/Recall}\ (TruePositiveRate): \quad TPR = \frac{TP}{P} = \frac{TP}{TP+FN} \tag{2.1}$$

$$(FalsePositiveRate): \quad FPR = \frac{FP}{N} = \frac{FP}{FP+TN} \tag{2.2}$$

$$\textbf{Specificity}\ (TrueNegativeRate): \quad TNR = \frac{TN}{N} = \frac{TN}{TN+FP} \tag{2.3}$$

$$(FalseNegativeRate): \quad FNR = \frac{FN}{P} = \frac{FN}{TP+FN} \tag{2.4}$$

$$\textbf{Precision}\ (PositivePredictiveValue): \quad PPV = \frac{TP}{PP} = \frac{TP}{TP+FP} \tag{2.5}$$

$$\textbf{Accuracy}: \quad ACC = \frac{TP+TN}{ALL} \tag{2.6}$$

$$(FalseDiscoveryRate): \quad FDR = \frac{FP}{PP} = \frac{FP}{TP+FP} \tag{2.7}$$

$$(FalseOmissionRate): \quad FOR = \frac{FN}{PN} = \frac{FN}{FN+TN} \tag{2.8}$$

$$(NegativePredictiveValue): \quad NPV = \frac{TN}{PN} = \frac{TN}{FN+TN} \tag{2.9}$$

$$\textbf{F}_\textbf{1}\ \textbf{score}: \quad F_1 = \frac{2TP}{2TP+FP+FN} \tag{2.10}$$

Cohenovo Kappa :

$$KAP = \frac{2 * (TP * TN - FN * FP)}{(TP + FP) * (FP + TN) + (TP + FN) * (FN + TN)} \quad (2.11)$$

2.8 Motivace pro použití strojového učení pro automatickou detekci retinopatie

Zvolení vhodné metody pro řešení problému je důležité, protože se od ní odvíjí celá práce a závisí na ní celková kvalita natrénovaného modelu. Je třeba se zamyslet nad tím, jaká data máme k dispozici a podle toho zvolit vhodnou metodu. Jelikož naším vstupem budou snímky sítnice, musíme vybírat takové metody, které s tímto typem vstupu umějí dobře pracovat. Velikost dimenze vstupu je u obrázku dána výpočtem (výška obrázku ∗ šířka obrázku ∗ počet barevných kanálů).

Hlavním kandidátem na metodu je použití neuronové sítě, protože v podobných úlohách dosahuje výborných výsledků. Zejména po příchodu konvolučních sítí došlo k výraznému posunu v přesnosti (viz sekce 2.5), které bychom v naší práci také využili. Další nespornou výhodou této metody je momentálně jeho velká obliba a s tím spojená existence a vývoj mnoha nástrojů[3][4][5] usnadňující experimentování s neuronovými sítěmi. Nejen v práci autorů Voulodimos, Doulamis, Doulamis a Protopapadakis [32] došli k závěru, že konvoluční sítě jsou vhodné pro tento typ úlohy.

Dalším možným kandidátem je metoda podpůrných vektorů (SVM), která hledá optimální nadroviny oddělující různé třídy dat. Idea SVM je vzít transformující funkci (kernel), aplikovat ji na data a tím je převést do více dimenzionálního prostoru, ve kterém již bude možné rozdělit data rovinou podle kategorií. Značnou výhodou této metody je skutečnost, že pokud existuje řešení, tak ho vždy najdeme a neuvízneme v lokálním extrému, jako je to u neuronových sítí[3]. SVM lze použít pouze na rozdělení na dvě kategorie a bez modifikování algoritmu ho nelze použít na kategorizaci 5 stupňů retinopatie. Hlavní nevýhodou jsou paměťové nároky, které exponenciálně rostou s dimenzí vstupu a počtem trénovacích vzorů. Vzhledem k dostupným datům jsme dali přednost použití neuronové sítě.

2.9 Použitá vývojová platforma pro práci s neuronovými sítěmi

Po rozhodnutí, že budeme vytvářet model neuronové sítě, bylo třeba vybrat, jakou vývojovou platformu (framework) použijeme. Nepoužití již dostupných knihoven by sice bylo možné, ale nerozumné, a vhledem k zadání práce zbytečné. Na implementaci takových knihoven pracují velké týmy lidí a sami bychom jim jen těžko mohli konkurovat.

Mezi hlavní knihovny patří PyTorch[6], který je zastřešen společností Meta

[3] `https://www.tensorflow.org/`
[4] `https://keras.io/`
[5] `https://huggingface.co/`
[6] `https://pytorch.org/`

(dříve Facebook) a TensorFlow[7], který je vyvíjen týmem společnosti Google. Pro práci jsme si zvolili framework TensorFlow, který je plně dostačující, zaměřuje se na celou škálu odvětví umělé inteligence a máme s ním nejvíce zkušeností. Jazyk, ve kterém budeme vyvíjet je Python.

TensorFlow nabízí široký výběr nástrojů pro manipulaci s daty, definování neuronových sítí a trénování modelů. Jeho flexibilita umožňuje efektivní využití výpočetního výkonu nejen prostřednictvím procesoru, ale také na grafických kartách nebo specializovaných akceleračních kartách optimalizovaných pro výpočty (TPU). Modely se definují pomocí vysokoúrovňového rozhraní Keras[8][9], které umožňuje přehledně definovat architekturu sítě. Pokud bychom potřebovali natrénované modely spouštět pomocí jiné vývojové platformy, než ve které byly vytvořeny, je třeba se připravit na nekompatibilitu formátů. Většina platforem si model ukládá do svého proprietárního formátu. Naštěstí existuje iniciativa ONNX[10] společností Meta a Microsoft pro vytvoření otevřeného standardu interoperability různých platforem, ke které se přidávají i další velké společnosti. TensorFlow se „zatím“ do této iniciativy nepřidal, ale existuje nástroj[11] převádějící uložený model do onnx formátu, který bychom mohli případně využít.

[7] `https://www.tensorflow.org/`
[8] `https://keras.io/`
[9] `https://www.tensorflow.org/guide/keras`
[10] `https://onnx.ai/`
[11] `https://github.com/onnx/tensorflow-onnx`

3. Přehled literatury

3.1 Přehled předchozích studií o detekci retinopatie pomocí strojového učení

Tým Pratt, Coenen, Broadbent, Harding a Zheng [24] trénoval konvoluční neuronovou síť poměrně standardní architektury s necelými 6,3 miliony parametrů. Pro trénování použili veřejný dataset ze stránky kaggle obsahující přes 80 tisíc snímků různé kvality. Vstupním obrázkům změnili rozlišení na 512x512 pixelů. Samotné trénování rozdělili na dvě části, kde v první části použili pouze 10.290 snímků a zbytek byl použit až v druhé části, když model dosahoval dobrých výsledků. Opatřením proti přeučení jednoho výstupu bylo použití vah jednotlivých stupňů nemoci. Autoři nepoužili 5000 snímků z datasetu na trénování, aby pomocí nich mohli na konci změřit kvalitu modelu. Naměřili 95 % Specificity a 75 % Accuracy, což může vypadat jako relativně dobrý model. Bohužel Sensitivity je pouhých 30 %, a to je při diagnostikování nemoci nepoužitelné.

Vědečtí pracovníci a lékaři ve studii[21] rozebírají snahu o vytvoření nástroje pro klasifikaci nemoci retinopatie pomocí neuronové sítě. Model byl trénován na neveřejném datasetu o 128 tisících snímcích sítnice, který byl rozdělen v poměru 80:20 na trénovací a validační část. Následně byl model hodnocen pomocí dvou veřejných datasetů EyePACS-1 (9963 snímků) a Messidor-2 (1748 snímků). Je třeba zdůraznit, že tým použil již naučenou konvoluční síť Inception-v3 používající se na kategorizaci snímků, u kterého nahradili rozhodovací část za čtyři binární klasifikátory. Model má velikost 22 milionu parametrů a tvůrci použili 10 různě naučených modelů (učení probíhalo na stejných datech i architektuře sítě), ze kterých počítali průměr výsledku. Samotný výsledek se skládá z více metriky, protože po znázornění ROC křivky naučeného modelu byly vybrány dva prahy tak, aby jeden odpovídal 98 % specificity a druhý 97 % sensitivity. Dále byl model hodnocen na dvou datasetech, ze kterých byly zahozeny některé nevyhovující snímky. U předzpracovaném datasetu EyePACS-1 dosahoval model AUC hodnoty 99,1 % a u Messidor-2 99,0 %. Je třeba mít na paměti, že AUC hodnoty mohou vypadat přehnaně optimisticky (viz 2.7), pokud vychází z nevyvážených dat. Oba zmíněné datasety jsou značně nevyvážené (8021 ku 767 u EyePACS-1 a 1503 ku 353 u Messidor-2), a proto může být AUC hodnota u modelu zavádějící.

3.2 Diskuse o omezeních a výzvách stávajících přístupů

Hlavním úskalím při tvorbě modelů schopných diagnostikovat nemoci (nejen diabetickou retinopatii) často bývá nedostatek trénovacích dat. Problém je zvláště patrný u nemocí postihující pouze zlomek populace a u kterých je třeba specializovaného personálu pro určení nemoci. To ani nemluvíme o dostupnosti dat s označenými symptomy s přesností na pixel, pomocí kterých by mohly vzniknout modely určující nejen výsledek vyšetření, ale i nalezené symptomy, které vedly k vyhodnocení výsledku. Vzhledem k faktu, že výsledek může určit pouze

odborník, nemůžeme využít širokou veřejnost pro rozšíření dostupných dat tak, jak to lze udělat u kategorizace objektů z fotografie.

Omezujícím faktorem u diagnostických testů je odlišný přístup k různým druhům chyb. Jak jsme si popisovali v sekci 2.7, falešně negativní výsledek je podstatně horší než falešně pozitivní. Při vytváření modelu je třeba přihlédnout na tuto nerovnováhu a upravit model tak, aby potlačoval falešně negativní výsledky. To zákonitě vede ke zhoršení přesnosti falešně pozitivních výsledků.

I přes podobnost problému s kategorizací obrazových dat z fotografií, která se v dnešní době již běžně používá, je detekce retinopatie podstatně složitější problém. Často se rozhoduje na základě malých útvarů a některé symptomy si jsou vizuálně podobné, i když patří k jiným stupňům nemoci.

4. Metodika

4.1 Přehled dostupných dat

K dispozici máme dva datasety, ze kterých je první soukromý a druhý veřejný. Prvním datasetem jsou data poskytnutá z českého medicínského zařízení Institutu klinické a experimentální medicíny. Jelikož je jedno z jejich specializací léčba diabetu, mají vlastní databázi vyšetření pacientů se snímky sítnic a výsledků těchto vyšetření. Hlavním důvodem poskytnutí dat byla snaha získat systém, který by jim pomohl při diagnóze diabetické retinopatie a ulehčil by práci odbornému personálu při roztřídění pacientů na zdravé a na ty, kteří potřebují další kontrolu. Při každém vyšetření se pořizují minimálně čtyři snímky – obě oči barevně a černobíle a k tomu je dostupný výsledek vyšetření. Dále jsou k dispozici data o pacientovi jako je jeho věk, či délka trvání diabetu.

Druhý jmenovaný je dataset dostupný na stránce kaggle.com[1] a jedná se o data ze soutěže z roku 2015, kterou vypsala nadace California Healthcare Foundation[2]. Cílem bylo vytvořit model s co možná nejlepší přesností rozhodnutí, zda jsou na snímku znaky retinopatie nebo ne. Dataset se skládá ze snímků sítnice a k tomu výsledky vyšetření rozdělené do 5 kategorií pro každý snímek. Další vlastností tohoto datasetu je různorodost kvality a velikost snímků. Je evidentní, že snímky byly pořízeny na různých typech fundus kamer s různým nastavením. Snímky se liší v barevném podání i rozlišení. I když se v datasetu vyskytují nějaké špatné snímky (nejedná se o sítnici, ale o oko zvenku), měl by být popis výsledného stupně přesný.

Na portálu kaggle.com je i dostupný dataset EyePACS[3], který vychází z výše zmíněného datasetu, které je zpracován pro jeho snadnější použití. Všem snímkům bylo změněno rozlišení tak, aby výška ani šířka nebyla větší než 1024 pixelů a zároveň byly ořízнuty černé okraje tak, aby maximum prostoru vyplňovala samotná sítnice. Dále byly odstraněny některé evidentně špatné snímky (snímek pacientova oka místo sítnice atd.). Původní dataset obsahoval dohromady přes 80 tisíc snímků, ale 53 tisíc z nich byly určeny pro test, takže k nim nejsou veřejné výsledky. Z toho důvodu je EyePACS založen pouze na 35 tisících snímcích z trénovacího setu. V práci budeme využívat data právě z tohoto datasetu a nebudeme si data měnit a filtrovat sami.

4.2 Rozbor trénovacích dat

U obou datasetů je silně nerovnoměrné rozložení jednotlivých kategorií a převažuje nultá kategorie označující absenci nemoci. Toto rozdělení odpovídá datům, se kterými se dá setkat v reálném světě, protože zastoupení nemoci v populaci je malé, a proto většina vyšetření bude končit s výsledkem, že je pacient zdravý.

[1] `https://www.kaggle.com/c/diabetic-retinopathy-detection`

[2] `http://www.chcf.org`

[3] Dataset byl ještě 4. 10. 2023 dostupný na adrese `https://www.kaggle.com/datasets/mariaherrerot/eyepacspreprocess`. Bohužel byl v době odevzdání práce smazán a z licenčních důvodů ho nemůžeme přidat do digitální přílohy. Vychází však ze stále dostupného datasetu a kromě drobných úprav nenabízí jiné snímky.

Je však třeba s tímto nepoměrem dat počítat a při trénovaní ho kompenzovat, protože při snaze snížit celkovou chybu by měl model tendenci upřednostnit ten výstup, který v datech dominuje.

Dataset ze stránky kaggle.com vykazuje velkou různorodost nejen v kvalitě snímků. To je z obecného hlediska kladná vlastnost, protože to bude nutit model více generalizovat řešení a tím by se měl stát více odolným. Nevýhodou jsou vyšší nároky na velikost modelu a hlavně na počet trénovacích dat. Tím, že zavedeme do dat proměnlivou kvalitu, barvy a podobně, musíme modelu dát o to víc dat, aby byl schopen „najít“ ty správné znaky určující stupeň nemoci.

Snímky z IKEMu vykazují konzistentnost jak v barvě, tak i v rozlišení. To má své výhody i nevýhody, které musíme vzít při řešení v potaz. Model je kladně ovlivněn v tom, že nemusí hlavní znaky tak složitě „hledat“ a jednodušeji se je naučí rozpoznávat. Na druhou stranu bude mít model problém určit správný výstup u odlišných dat, než jaké byly v trénovací množině. U homogenních dat si můžeme dovolit použít menší model než bychom potřebovali u nesourodých dat.

Při procházení dat poskytnutým zařízením IKEM, jsme narazili na zásadní omezení, které nám ztíží trénování modelu. Problémem je, že ke čtyřem snímkům sítnic (barevný a černobílý snímek levého a pravého oka) je dostupný pouze jeden výstup kategorie nemoci. Data bohužel neobsahují informaci, které oko bylo v horším stavu a je tedy zaznamenáno jako výsledek vyšetření. Špatné označení kategorie snímku může vést ke zhoršenému učení neuronové sítě nebo dokonce k nemožnosti generalizace vstupu a tím i správné rozpoznání kategorii nemoci. Teoreticky můžeme mít až 50 % dat špatně označených, ale ve skutečnosti to bude méně, protože rozdíl nebývá, až na výjimečné případy, větší než 1 stupeň nemoci. Po upozornění na tento problém IKEM změnil metodiku a začal zaznamenávat výsledek vyšetření podrobněji, ale dosavadní data se znovu zpracovávat nebudou.

4.3 Popis algoritmů strojového učení použitých pro klasifikaci

Jistou výhodou této nemoci je, že její znaky je možné pozorovat přímo ze snímků sítnice, a proto architektura neuronové sítě nebude vybočovat od jiných sítí zpracovávající obrazová data. Budeme moci vycházet z již existujících a ověřených architektur. I když by dodatečné informace mohly modelu pomoci při rozhodování, přínos by neměl být nikterak veliký, protože hlavními znaky nemoci jsou změny na sítnici oka. Předností pouze obrazového vstupu je možnost predikovat nemoc bez ohledu na to, jestli máme k dispozici pacientovu anamnézu.

Standardní způsob řešení použitý v jiných článcích je vytvoření binárního klasifikátoru určující přítomnost nemoci. Někteří použili výstup více binárních klasifikátorů vyjadřující přítomnost konkrétního symptomu. My jsme se rozhodli pro nový způsob řešení, a to nahlížet na problém jako na regresní. Výstupem bude číslo znázorňující stupeň nemoci. Tento způsob jsme zvolili ze dvou důvodů, a to, že tato metoda ještě není detailně prozkoumána a hlavně se nové symptomy přidávají k těm stávajícím. Hranice mezi stupni nemoci nejsou přesně ohraničené, jako při kategorizačních problémech.

4.4 Diskuse výkonnostních ukazatelů použitých pro hodnocení modelů

V sekci Validace modelu 2.7 jsme si představili několik možných metrik, které lze použít při hodnocení modelu. Primárně se budeme soustředit na matici záměn, protože nám je schopná přehledně zobrazit, jak model predikuje různé stupně nemoci. Druhou metriku jsme vybrali ROC křivku spolu s AOC metrikou, protože se u řešení tohoto problému často využívají, a proto je budeme moci porovnat s výsledky jiných prací. V některých případech použijeme i PRC křivku, která se sice nepoužívá často, ale nám připadá vhodné ji využít, protože dokáže zohlednit i přesnost u falešně negativních výsledků, které ROC křivka pomíjí a je méně náchylná na zkreslené zobrazení vlivem nevyrovnaných dat datasetu.

4.5 Automatické hledání hyperparametrů modelu

Naivní způsob hledání hyperparametrů modelu může být vyzkoušení všech možností a poté vybrat takové nastavení, kde má model nejmenší chybu. To je sice zcela validní způsob, ale reálně se mu chceme vyhnout z časových důvodů. Trénování neuronových sítí je náročné na výpočetní výkon, a proto bychom chtěli použít nějaké chytré prohledávání. Obecně se tyto algoritmy nazývají tunery.

Jedním z moderních algoritmů pro hledání vhodné kombinace hyperparametrů je Hyperband[17], který využívá iterativní způsob hledání, kde zkouší krátké úseky trénování různých kombinací a vyřazování vždy poloviny nejhorších kombinací. Modely se netrénují od začátku do konce, ale učení je rozdělené na menší úseky. Tvůrci odhadují, že tento způsob je 5x až 30x rychlejší než dříve známé tunery.

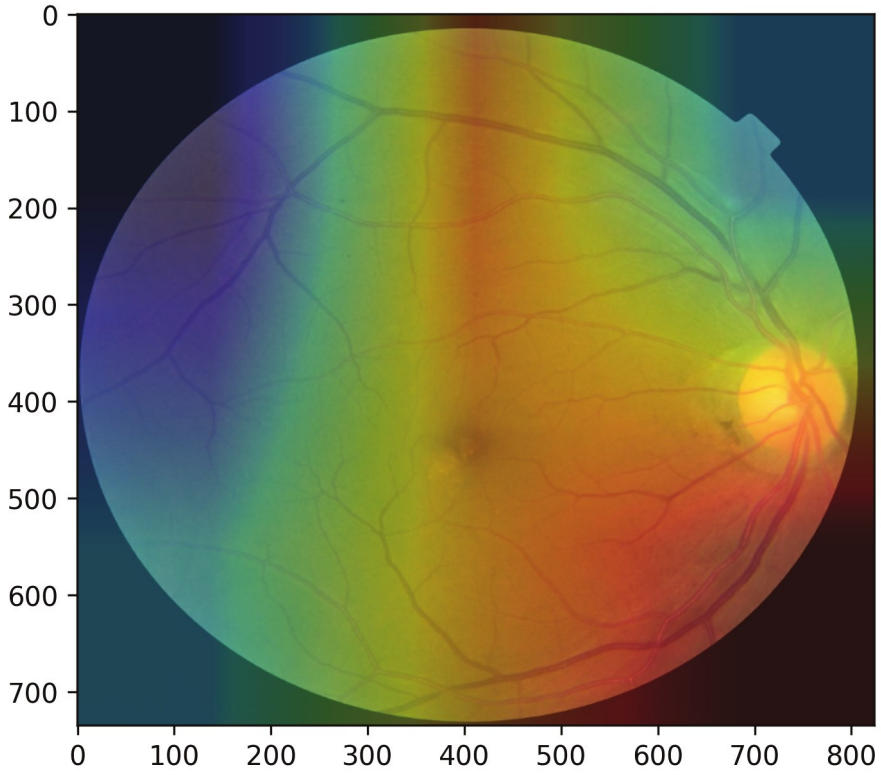
Zároveň je možné tuner kombinovat s předčasným zastavením učení, pokud se chyba na validačním setu nesníží po n krocích. Takto můžeme ještě více zrychlit hledání, ale je třeba nastavit n na vhodný počet kroků, abychom neukončovali učení dřív, než se bude moci projevit význam parametrů.

4.6 Možnosti vizualizace konvoluční vrstvy

Po natrénování neuronové sítě používající konvoluční vrstvy je chtěné si vizuálně ověřit, že konvoluční vrstvy se naučily rozpoznávat podstatné části. K tomuto účelu můžeme využít existující metodu Grad-CAM[28], ve které se používá výpočet gradientu směřující do poslední konvoluční vrstvy vzhledem ke vstupnímu obrázku a výstupu sítě. Vypočtené gradienty se převedou na mapu regionů, které se významně podílely na výsledku sítě. Oproti předchozím metodám je výhoda Grad-CAM v tom, že není třeba upravovat architekturu sítě a je možné ji použít i na již natrénované neuronové sítě. Drobnou nevýhodou je nízké rozlišení důležitých oblastí, protože se vychází z rozlišení poslední konvoluční vrstvy. Regiony je možné interpolovat na vyšší rozlišení, ale to nám nezajistí přesnost na pixel a jde spíše o zobrazení větších oblastní, které se podílely na výsledku.

Metoda byla vytvořena pro sítě, které mají kategorický výstup (více neuronů a použitý softmax). V našem případě jsme museli metodu upravit, protože naším výstupem je jeden neuron udávající stupeň nemoci. Využili jsme poznatků v práci autorů Kanda a kol. [11], jak Grad-CAM metodu upravit pro regresní problém. V naší implementaci jsme vynechali ořezání záporných hodnot, protože i ty by měly z našeho pohledu být důležitou součástí výsledku.

Na snímku 4.1 je ukázán příklad výstupu metody Grad-CAM, kde nejdůležitější regiony jsou vybarveny červeně a nejméně důležité tmavě modře.



Obrázek 4.1: Příklad zobrazení Grad-CAM techniky u snímku se stupněm 2 DR

5. Experimenty

5.1 Obecné informace

Při vytváření architektury sítě se budeme řídit již osvědčenými pravidly [7] [29] a využívat znalosti z jiných studií, které se zaměřovaly na zpracování obrazových dat. Budeme se snažit vyvarovat složitějším strukturám sítě, abychom se mohli spolehnout, že případný neúspěch při trénování nemohla zavinit neobjevená chyba v implementaci složitých částí architektury sítě nebo nevhodnost použití části na řešení problému. Proto jsme se zaměřili na postupnou aplikaci konvolučních vrstev s malým rozměrem jádra a rostoucí velikostí filtrů prokládanou MaxPool vrstvami. Po konvolučních vrstvách použijeme několik plně propojených vrstev, které by se měly starat o zpracování nalezených *features* z předchozích vrstev. Poslední vrstva bude obsahovat jeden neuron bez jakékoli aktivační funkce, která bude zároveň vrstvou výstupní.

Podstatným rozhodnutím je, na jakých datech budeme model trénovat. K dispozici máme dva datasety, které se liší jak kvalitou a konzistencí vstupních dat, tak i přesností výstupu. Jelikož je cílem vytvořit systém, který by mohl využívat IKEM ke svým potřebám, přikloníme se k upřednostnění dat poskytnutých od nich. Bude třeba pevně rozdělit vstupní data na tři části tak, aby všechny kategorie byly zastoupeny v podobném poměru a dále s tímto rozdělením nehýbat, aby případné experimenty nebyly ovlivněny jiným vstupem. Z důvodu malého počtu snímků jsme se rozhodli rozdělit data v poměru 7:2:1, kde největší část připadá na trénovací data, druhá na validační data, která budou sloužit k hledání hyperparametrů a poslední testovací data.

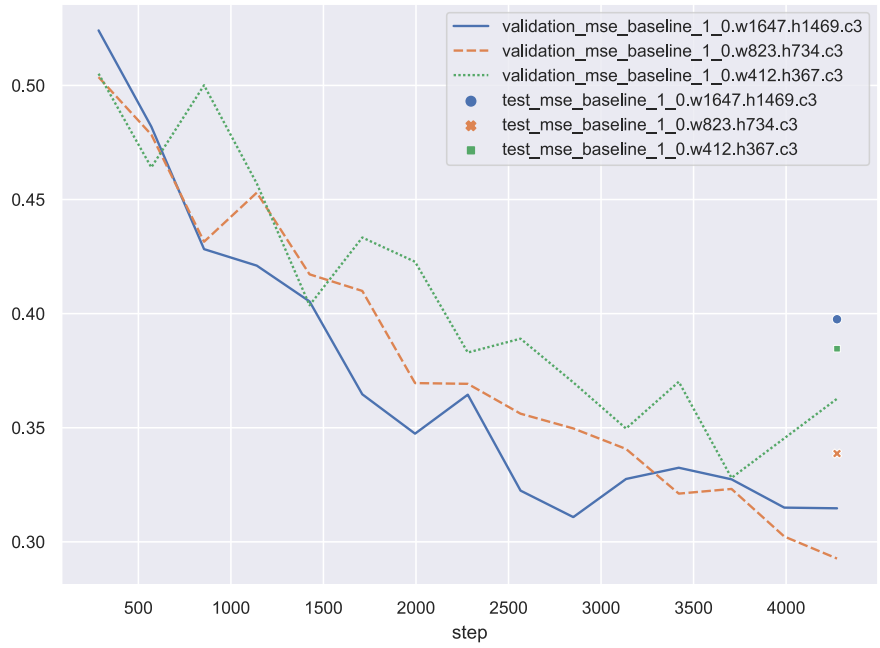
5.1.1 Volba vstupního rozlišení

Symptomy naštěstí nejsou o velikosti nízkých jednotek pixelů, a proto nejspíš nemusíme používat vstupní obrázky v plném rozlišení a snížit nároky při trénování neuronové sítě jejich zmenšením. Je třeba být se zmenšováním snímku opatrný, abychom nepřišli o nějaké podstatné detaily. Proto jsme přišli s experimentem vyzkoušet různé velikosti vstupních obrázků a počet barevných kanálu, aby se ukázalo, jestli by nešlo používat zmenšené velikosti bez újmy na kvalitě výstupu. Pokud by se změna velikosti nebo snížení počtu kanálů projevilo pozitivně, modely by byly menší a bylo by možné je učit rychleji. Architekturu jsme zvolili podle zmíněných pouček, a to šest po sobě jdoucích konvolučních vrstev s rostoucí velikostí filtrů následovanou plně propojenou vrstvou se 100 neurony. Výstupní vrstva má jeden neuron bez aktivační funkce.

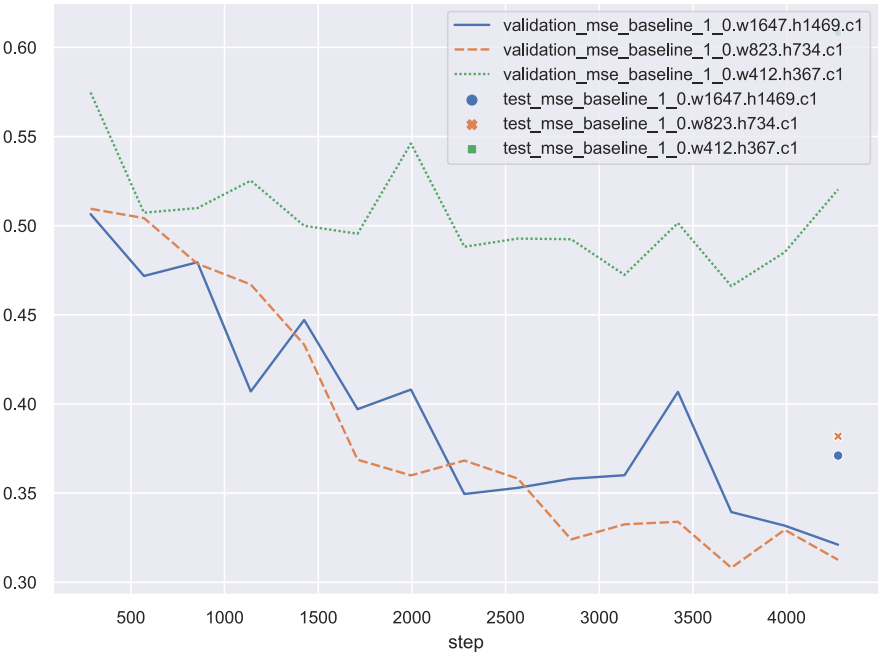
Z grafu 5.1 je patrný průběh učení vyobrazenou chybou na validačním setu po každé naučené epoše. Rozdíly zde nejsou nijak veliké a nedá se říct, že by jedna z verzí výrazně vyčnívala nad ostatními. Na druhou stranu, při pohledu na body označující chybu na testovacím setu lze označit jednoznačně „vítěze“ v tomto experimentu a je jím použití menšího rozlišení 823x734 pixelů se třemi barevnými kanály.

U grafu 5.2 použití velikosti 412x367 pixelů a jednom barevném kanálu(zelená křivka) nejspíše ukazuje paralyzaci testování z důvodu nedostatečné informace

na vstupu. Naproti tomu vyšší rozlišení lze považovat za ekvivalentní z pohledu chyby na validačním i testovacím setu.



Obrázek 5.1: Základní model s třemi barevnými kanály s velikostmi 1647, 823 a 412 px na šířku. Křivky ukazují vývoj chyby na validačním setu a body na konci chybu na test setu.



Obrázek 5.2: Základní model s jedním barevným kanálem a různými velikostmi vstupu.

Na základě tohoto experimentu jsme se rozhodli použít tři barevné kanály a rozlišení 824x735, které je čtvrtinové oproti originální velikosti snímků poskytnutých zařízením IKEM a zároveň jsou o málo větší než byly velikosti použité v jiných studiích. Neměli bychom se tedy příliš lišit od jiných funkčních řešení.

5.1.2 Použití naučeného modelu

Při učení neuronové sítě je možné použít techniku pojmenovanou Transfer learning, která využívá již nějaký naučený model řešící podobný problém jako chceme řešit my a model upravit tak, abychom ho mohli použít na problém nový. V našem případě se jedná o rozpoznání vzorů z obrázku. Naštěstí existují různé modely naučené na rozpoznávání obsahu obrázku a my se můžeme pokusit je přeučit na rozpoznávání stavu retinopatie.

V sekci 2.4.8 se zmiňujeme, že sekvence konvolučních vrstev postupně získává *features*. Natrénovaný model již toto získávání má naučené a samotné rozpoznání probíhá až ve vyšších vrstvách za konvolučními vrstvami. Je třeba, aby rozpoznávané *features* byly podobné nově potřebným, protože ty již učit nechceme. Vyšší vrstvy můžeme nahradit novými, vlastními, zmrazit váhy v konvolučních vrstvách, aby se neměnily trénováním, a trénovat takto složený model na vlastních datech. Tímto způsobem můžeme přeučit neuronovou síť na nový problém s menším úsilím, protože část modelu je již naučená a není třeba ji učit znovu. Může se stát, že tento způsob nebude fungovat podle představ, a to by mohlo být způsobené tím, že si problémy (extrahované znaky) nejsou dostatečně podobné.

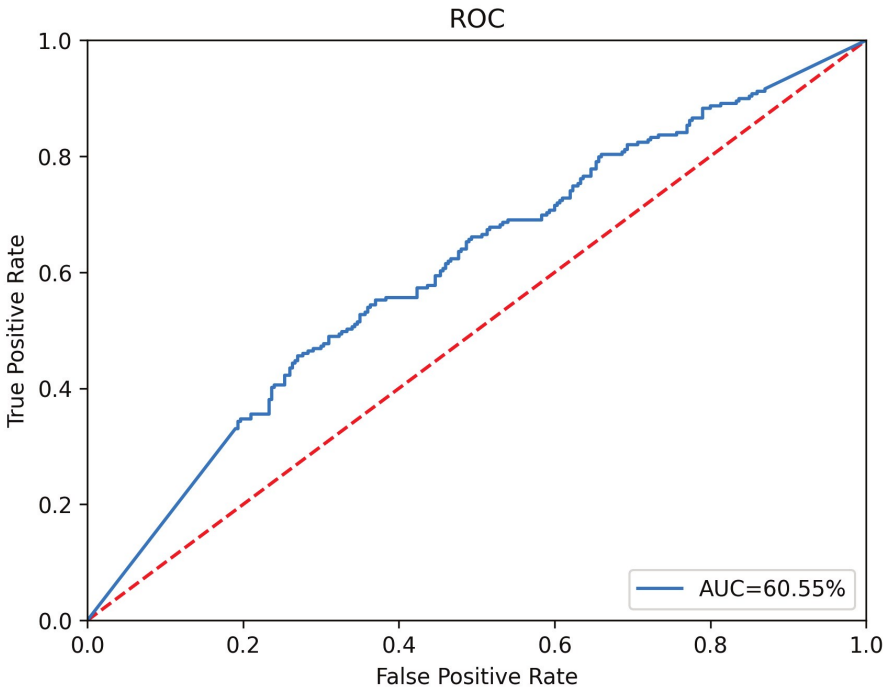
Po natrénování nově přidaných vrstev je možné povolit trénování všech vah a s nastaveným malým učícím krokem pokračovat v trénování. Trénovaní všech vah není nutné, ale může zlepšit celkovou kvalitu modelu. Problémem této části učení jsou podstatně vyšší nároky na výkon, protože předtrénované modely zpracovávající obrázky mívají typicky řádově jednotky až desítky milionu parametrů, což je mnohokrát více než budou mít naše modely. Druhým problémem je, že se může model velice rychle přeučit vzhledem k jeho obrovské kapacitě, a proto je třeba při případném použití doučení postupovat obezřetně.

Pro náš experiment, jestli je transfer learning vhodný, jsme zvolili předtrénovaný model MobileNetV2[27], který má 2,26 milionu parametrů bez horních vrstev, které jsou specializované na konkrétní úlohu. MobileNetV2 umožňuje zpracovat obrázek s jakýmkoli rozlišením větším než 32x32 pixelů a k tomu nabízí 5 sad předtrénovaných vah podle zvolené velikosti vstupu. My jsme použili rozlišení 824x735, jelikož se dle předchozího experimentu jevilo jako nejvhodnější.

Bohužel jsme neměli přesný popis architektury, které byly použité v již zmiňovaných studiích používající předtrénovaný model, a proto jsme vytvořili model podle obecně platných pouček. Nejvíc se nám osvědčilo použít jednu plně propojenou vrstvu po konvolučních vrstvách. Při použití více vrstev bylo učení příliš pomalé a po zvýšení učícího kroku bylo učení silně nestabilní. Vzhledem k celkové velikosti jsme nebyli schopni trénovat celý model, ale pouze nově přidané části, což mohlo vést k nepříliš dobrým výsledkům ukázaným v tabulce záměn 5.1 a na grafu 5.3. Rozhodli jsme se využít vlastní menší model a nepokračovat ve využívání předtrénovaných modelů.

Výstup modelu \ Skutečnost	0	1	2	3	4
0	146 49%	71 37%	8 22%	1 25%	0 0%
1	137 46%	111 57%	17 46%	0 0%	1 25%
2	16 5%	11 6%	12 32%	3 75%	1 25%
3	1 0%	1 1%	0 0%	0 0%	2 50%
4	0 0%	0 0%	0 0%	0 0%	0 0%

Tabulka 5.1: Confusion matrix pro předtrénovaný model MobileNetV2 doučený na IKEM datasetu



Obrázek 5.3: ROC metrika pro předtrénovaný model MobileNetV2 doučený na IKEM datasetu

5.1.3 Hyperband tuner

Jedním z prvních kroků, které jsme udělali při řešení této práce vlastní sítí je definice základního modelu s nastavením možných hodnot různých hyperpa-

rametrů, které chceme vyzkoušet. Dopředu nám nikdo není schopen říct, jaká kombinace hyperparametrů je ideální, protože to závisí nejen na použité architektuře sítě, ale i na řešeném problému.

Při hledání hyperparametrů jsme použili dataset IKEMu s následujícími úpravami. Každý pixel uložen jako RGB trojice celých čísel v rozsahu 0 až 255 je převeden na desetinná čísla v rozsahu -1,0 a 1,0. Tuto změnu provádíme proto, aby vstupní data byla vhodnější pro trénování neuronové sítě. Další změnou je převedení obrázku do požadovaného rozlišení podle modelu. V našem případě jde o vstup s dimenzí (824, 735, 3). Posledním krokem při přípravě datasetu bylo vypočítání `class_weight`, což jsou četnosti jednotlivých tříd v training setu, pomocí kterých se přepočítává učící krok, aby se vyvážil nepoměr počtu vstupů u různých tříd.

Zvolili jsme použití tří vrstev augmentace, které by měly snížit riziko přeučení. Použili jsme náhodné horizontální překlopení snímku, náhodnou změnu odstínu a změnu světlosti snímku. Snažili jsme se neprovádět moc agresivní úpravy a spíše se držet konzervativních hodnot.

Prvním volitelným parametrem je optimalizační algoritmus, který necháme vybrat z rmsprop, adam a nadam. Chybovou funkcí používáme buďto MSE nebo Huber. Při výběru aktivační funkce jsme nechali vyzkoušet z relu, leaky_relu, swish, selu, gelu a elu. Architekturu jsme zvolili následovně (viz tabulka 5.2 a zdrojový kód A.1):

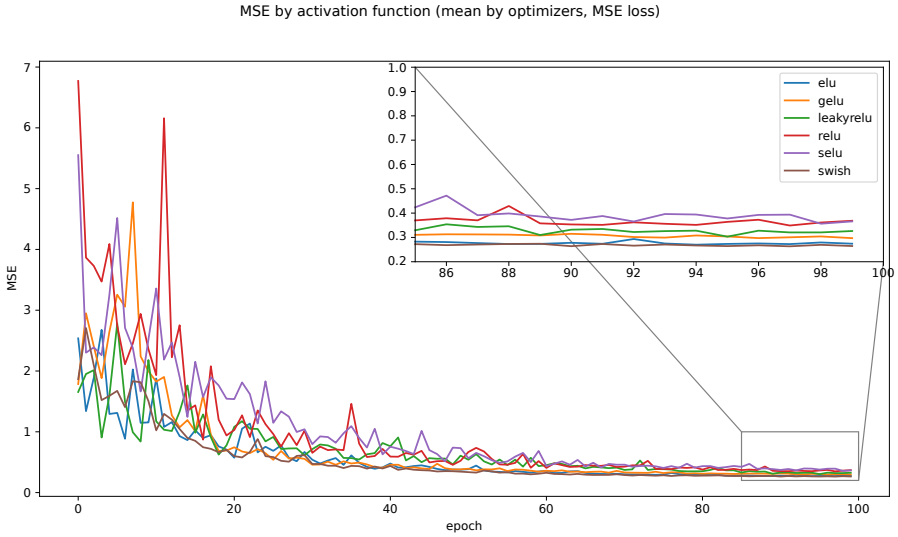
Typ vrstvy	**Tensorflow název**	**Podrobnosti**
Vstupní	Input	dimenze (824, 735, 3)
(augmentační vrstvy)		nemění dimenzi
Konvoluční	Conv2D	filter 8, rozměr (5,5), posun 1
MaxPool	MaxPool2D	rozměr (2,2)
Konvoluční	Conv2D	filter 8, rozměr (3,3), posun 1
MaxPool	MaxPool2D	rozměr (3,3)
Konvoluční	Conv2D	filter 16, rozměr (5,5), posun 1
MaxPool	MaxPool2D	rozměr (3,3)
Konvoluční	Conv2D	filter 16, rozměr (5,5), posun 1
MaxPool	MaxPool2D	rozměr (3,3)
Konvoluční	Conv2D	filter 32, rozměr (3,3), posun 1
MaxPool	MaxPool2D	rozměr (2,2)
Konvoluční	Conv2D	filter 32, rozměr (3,3), posun 1
Flatten	Flatten	
Plně propojená	Dense	velikost [50 až 120]
Plně propojená	Dense	velikost [30 až 100]
Plně propojená	Dense	velikost [20 až 50]
Plně propojená	Dense	velikost 1 bez aktivační funkce

Tabulka 5.2: Architektura neuronové sítě

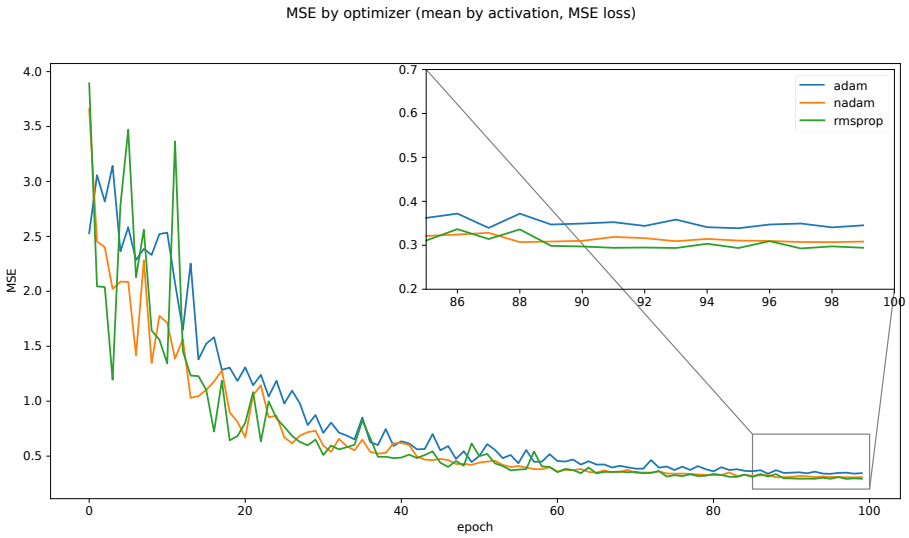
Při používání tuneru jsme narazili na problém s předčasným ukončením, který byl nastaven příliš agresivně. Docházelo totiž k situacím, že kombinace hyperparametrů, které měly teoreticky dosahovat dobrých hodnot byly brzo vyřazeny a při manuálním vyzkoušení projevovaly svoji kvalitu až po delším učení. Zvýšili jsme proto nastavení předčasného ukončení na 30 % z celkových učících epoch,

což problém vyřešilo, ale zároveň výrazně nezpomalilo prohledávání.

Výsledkem tuneru bylo použití nadam optimalizačního algoritmu, MSE chybové funkce, swish aktivační funkce, $3 * 10^{-4}$ learning rate a velikosti dense vrstev (50, 90, 50). Při tomto nastavení měl model necelých 126 tisíc parametrů a po 100 epochách učení dosahovala chyba na train setu 0,0014 a na validačním setu 0,1928.



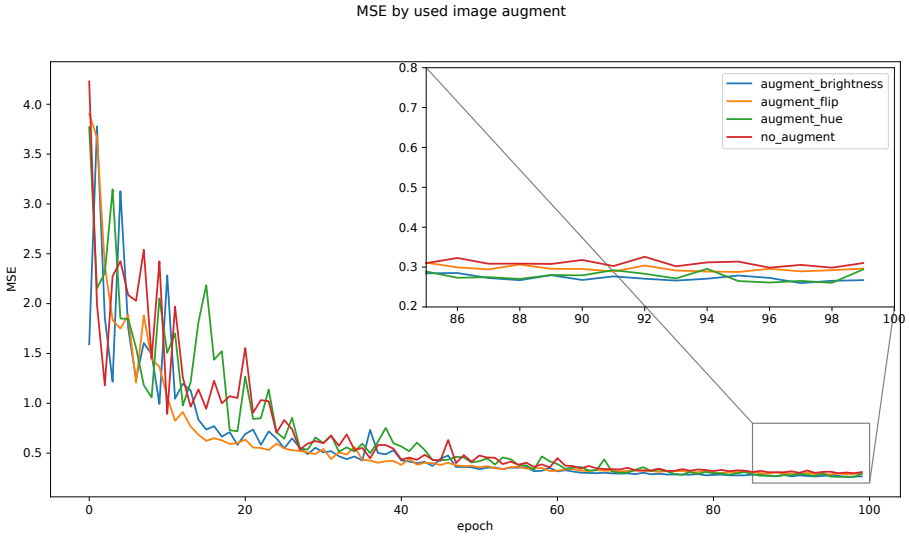
Obrázek 5.4: MSE chyba na validačních datech po epochách. Pro každou aktivační funkci byla data zprůměrována přes optimalizační funkce.



Obrázek 5.5: MSE chyba na validačních datech po epochách. Pro každou optimalizační funkci byla data zprůměrována přes aktivační funkce.

Zopakovali jsme učení se všemi kombinacemi optimalizačních algoritmů, chybových funkcí a aktivačních funkcí se zaznamenáváním dat. Z průběhu vývoje chyb na grafu 5.4 a 5.5 můžeme potvrdit očekávaný průběh, kdy všechny kombinace postupně konvergují a nedochází k přeučení. I přes skutečnost, že jsou

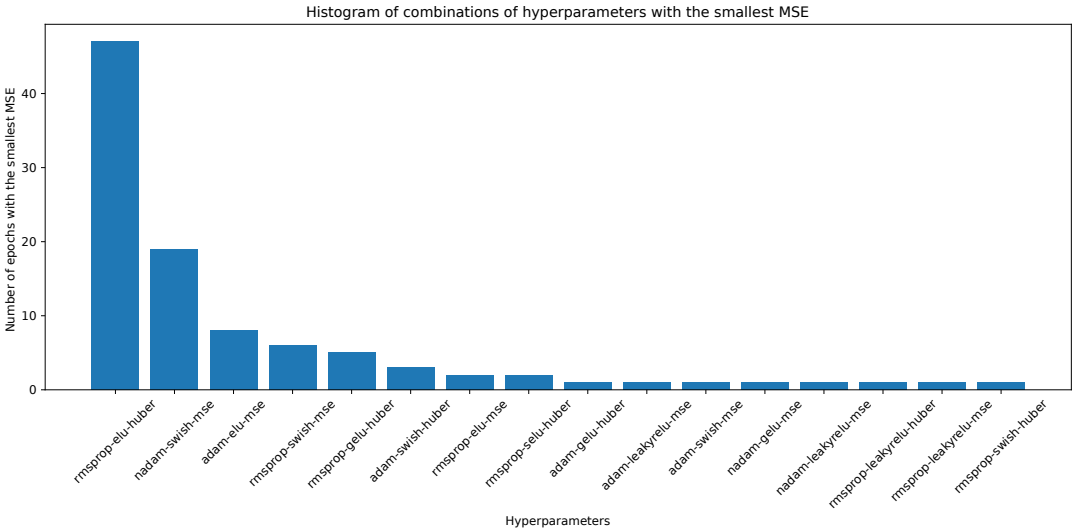
grafy založeny na malém počtu běhů a rozdíly nejsou příliš velké, lze usoudit, že nalezená kombinace hyperparametrů tunerem by měla fungovat dobře.



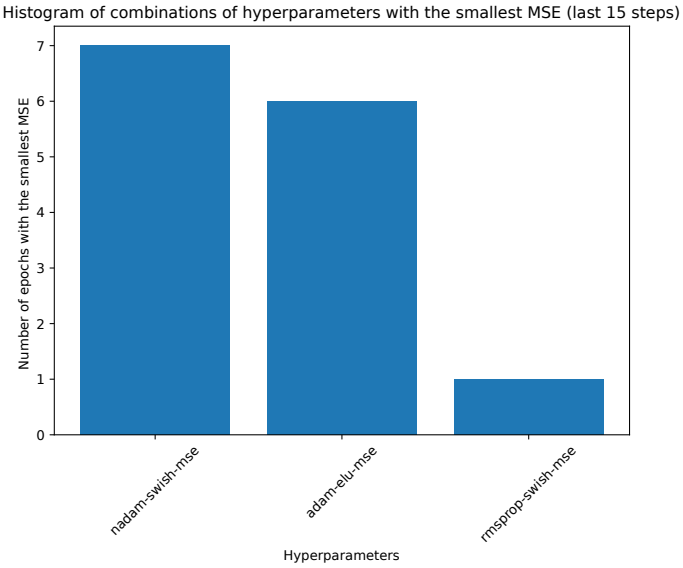
Obrázek 5.6: Vliv na MSE chybu použitím jednoduché augmentace obrázku.

Jelikož jsme u všech trénování používali jednoduché augmentace obrázků, které by teoreticky neměly mít negativní vliv na učení, chtěli jsme si tento předpoklad ověřit. Z průběhu grafu 5.6 vidíme, že jednotlivé augmentace měly sice minimální, ale převážně pozitivní vliv. Rozhodli jsme se tyto augmentace dále používat, protože nic nenasvědčuje tomu, že by měly negativní vliv na kvalitu a myslíme si, že by měly spíše pomáhat. Jmenovitě jsme použili horizontální převrácení (augment_flip), změnu ve světelnosti obrázku (augment_brightness) a lehkou změnu v odstínu snímku (augment_hue).

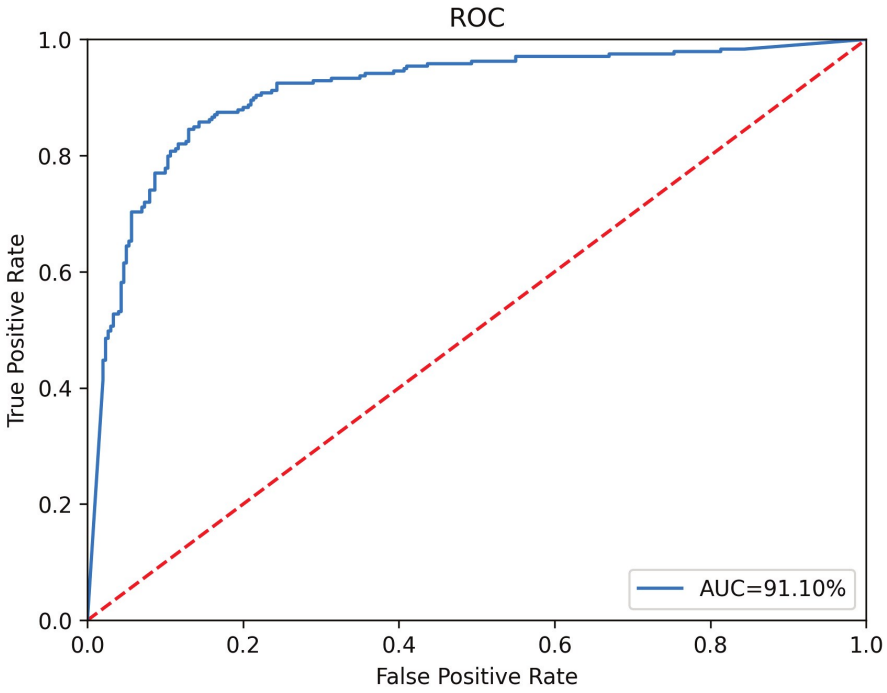
Vytvořili jsme histogram 5.7 zobrazující pro každou kombinaci hyperparametrů počet epoch, ve kterých měly nejmenší MSE chybu. Trochu nás překvapilo, že v téměř polovině epoch byla nejlepší kombinace *rmsprop-elu-huber*, protože jsme na prvním místě čekali kombinaci *nadam-swish-mse*, jelikož by měla být celkově nejlepší. Ta se umístila hned na druhém místě a je možné, že *rmsprop-elu-huber* v počátku nejrychleji klesala, ale to nemusí nutně znamenat, že by nakonec dosahovala nejlepších výsledků. Tuto myšlenku jsme si ověřili opakovaným vytvořením histogramu pomocí posledních 15 epoch (histogram 5.8).



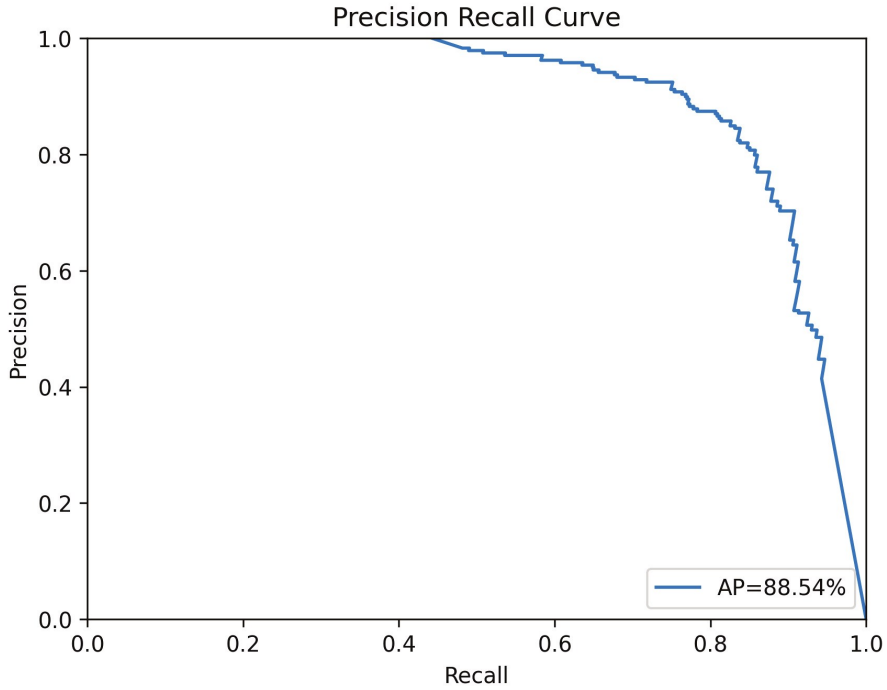
Obrázek 5.7: Počet epoch, ve kterých měla daná kombinace hyperparametrů nejmenší MSE chybu na validačním setu.



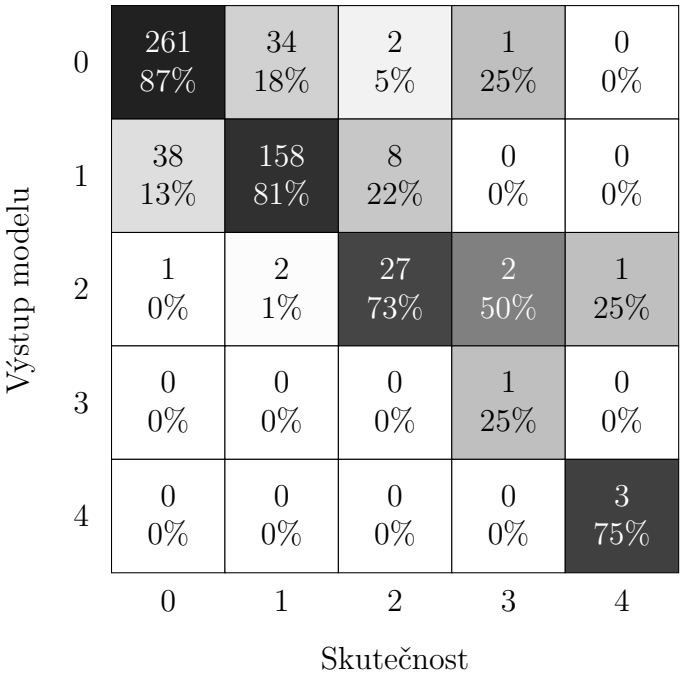
Obrázek 5.8: Počet epoch, ve kterých měla daná kombinace hyperparametrů nejmenší MSE chybu na validačním setu. Počítání z posledních 15 epoch.



Obrázek 5.9: ROC metrika pro model převedený na binární variantu ořízнutím výstupu.



Obrázek 5.10: PRC metrika pro model převedený na binární variantu ořízнutím výstupu.



Tabulka 5.3: Confusion matrix pro nejlepší model natrénovaného pomocí hyperband tuner

5.1.4 Augmentace dat - CLAHE

Zlepšení neuronové sítě jsme se nejprve pokusili úpravou vstupních obrázků tak, aby případné symptomy byly výraznější a tím pádem i jednodušší na nalezení a naučení. Existují různé metody zabývající se augmentací obrazových dat a v našem případě se potřebujeme zaměřit na ty, které nám zvýrazní krevní řečiště. Z popisu příznaků je patrné, že většina fyziologických změn je u cév nebo jsou nějak spojena s krevním řečištěm.

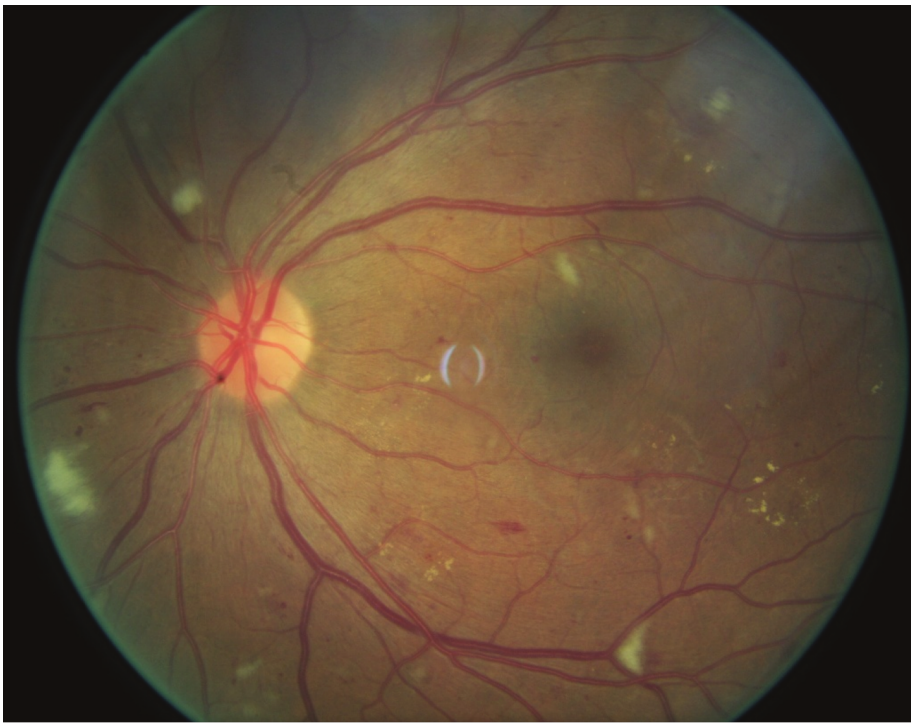
Pro zvýraznění jsme použili metodu *Contrast Limited Adaptive Histogram Equalization* (dále CLAHE), která se stará o zvýšení kontrastu obrázku oříznutím a následným adaptivním přerozdělením světlosti podle histogramu. Zvolili jsme náhodný vstupní obrázek 5.11 pro ilustraci změn. Mohli bychom CLAHE způsob použít na zeleném barevném kanálu (viz 5.12, kód 5.1), protože v něm jsou nejvýrazněji viditelné cévy [10]. Je vidět, že sice došlo ke zvýraznění cév a kontrastu obecně, ale za cenu nazelenalého nádechu celého snímku. Existuje lepší způsob, a to nejprve transformovat RGB barevné kanály do barevného prostoru LAB, kde L označuje světlost pixelu a A a B kanály jeho barvy. Použijeme-li CLAHE na L kanálu, dostaneme obrázek 5.13, který více odpovídá naší představě o zvýraznění podstatných detailů. Pokud bychom použili tuto metodu na všechny kanály L, A a B, výsledek vypadá takto 5.14. V poslední variantě jsou saturované barvy a výrazné detaily, ale za cenu nepřirozeného podání barev. Rozhodli jsme se použít variantu, u které jsme použili CLAHE na L kanálu, protože u ní došlo ke zvýraznění potřebných detailů, ale bez velkých barevných změn.

```
1  import cv2
2  import matplotlib.pyplot as plt
3
4  # Show enhanced image using RGB (G)
5  img_path = '{path to image}'
6  img = cv2.imread(img_path)
7  rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
8  r, g, b = cv2.split(rgb)
9  clahe_g = cv2.createCLAHE(clipLimit=3.0).apply(g)
10 img_enhanced = cv2.merge((r, clahe_g, b))
11 plt.imshow(img_enhanced)
12 plt.show()
```

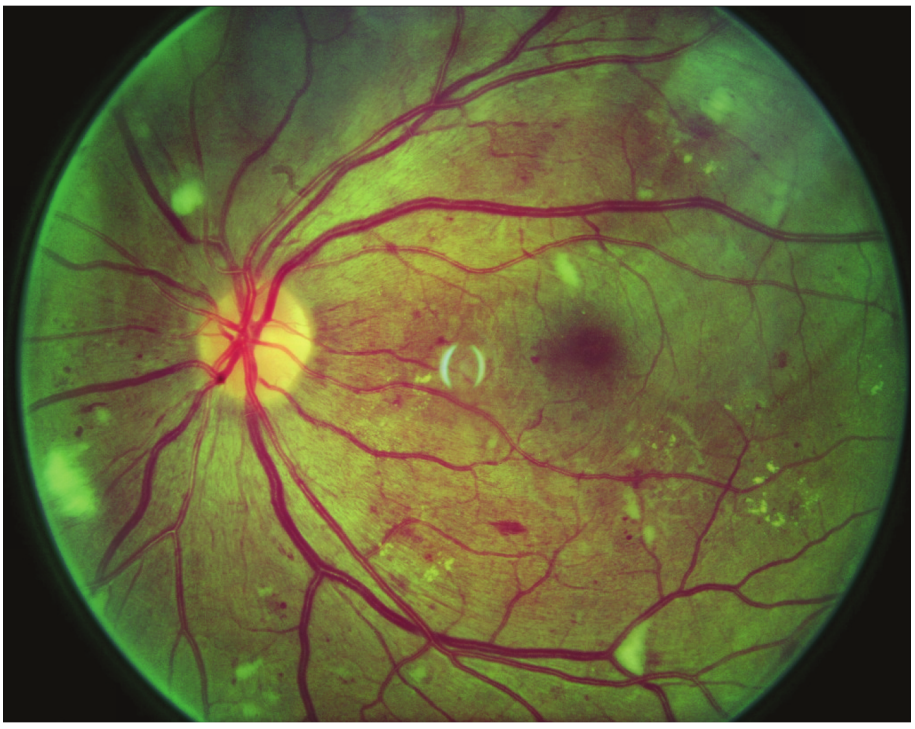
Zdrojový kód 5.1: CLAHE na zeleném kanálu

Upravili jsme všechny snímky z datasetu z IKEMu způsobem, jako je ukázán na obrázku 5.13 a spustili jsme opět hledání hyperparametrů jako v sekci 5.1.3. Domnívali jsme se, že zvýraznění detailů, ale zároveň minimální barevné odlišnosti, by mohlo vést ke snadnějšímu trénování neuronové sítě. Zároveň změnou vstupních obrázků může dojít ke změně vhodných hyperparametrů, a proto jsme opět použili nástroj pro jejich nalezení.

Z confusion matrix 5.4 a metrik ROC 5.15 a PRC 5.16 vyplývá, že úprava snímků metodou CLAHE nevedla k výraznému zlepšení kvality modelu. Je možné, že symptomy jsou pro konvoluční vrstvy stejně rozpoznatelné bez ohledu na zvýšení jejich kontrastu. Tento způsob předzpracování dat budeme považovat za nadbytečný, protože nevedl ke znatelné změně a pouze zvýšil nároky na výkon a složitost systému, který musí data připravit.

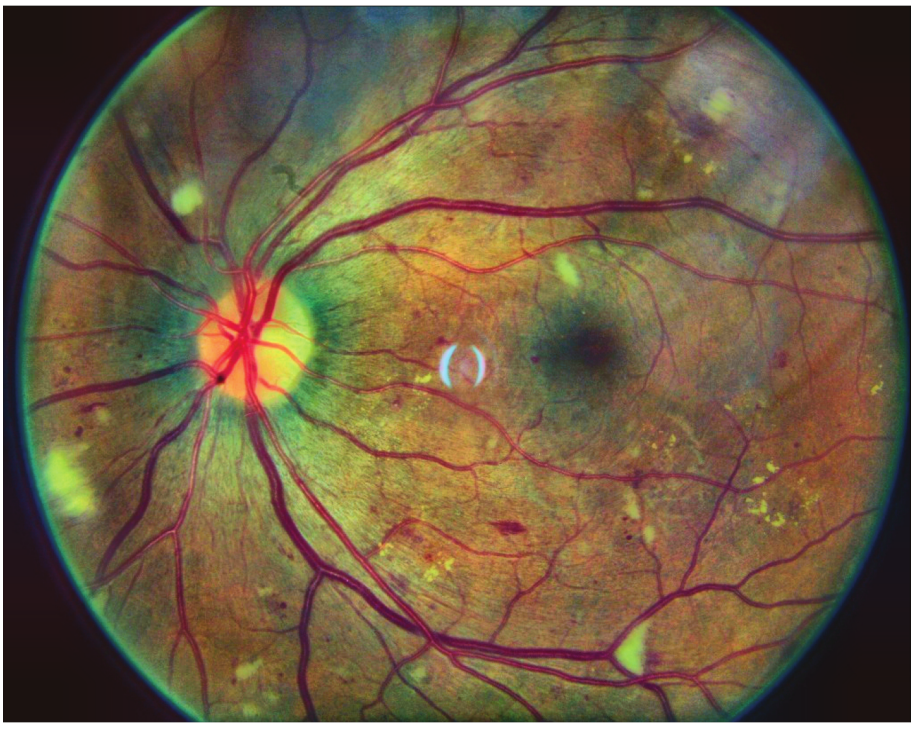


Obrázek 5.11: Vstupní obrázek pro ukázku augmentace.

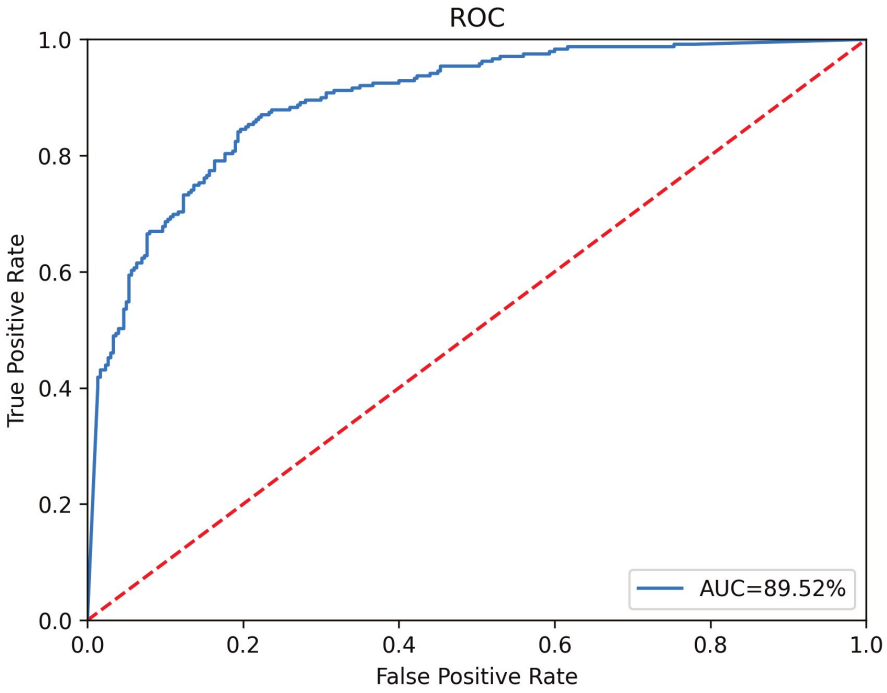


Obrázek 5.12: CLAHE metoda na zeleném barevném kanálu

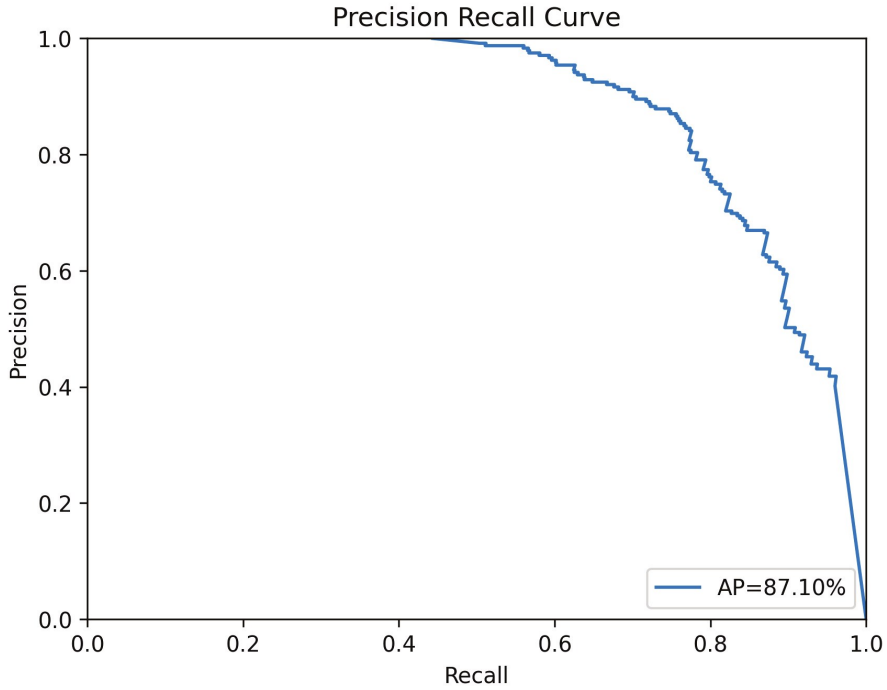
Obrázek 5.13: CLAHE metoda na L světelném kanálu



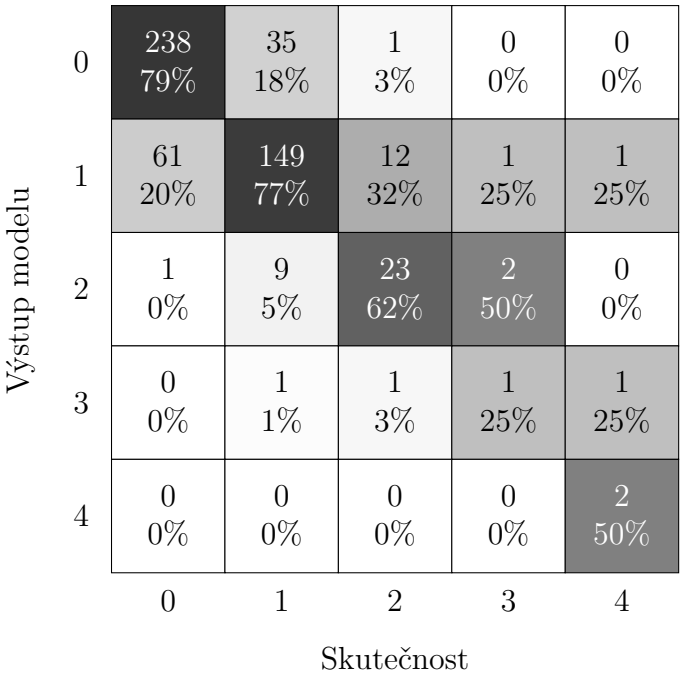
Obrázek 5.14: CLAHE metoda na všech LAB kanálech



Obrázek 5.15: ROC metrika modelu natrénovaného pomocí upravených snímků metodou CLAHE na L kanálu převedený na binární variantu oříznutím výstupu.



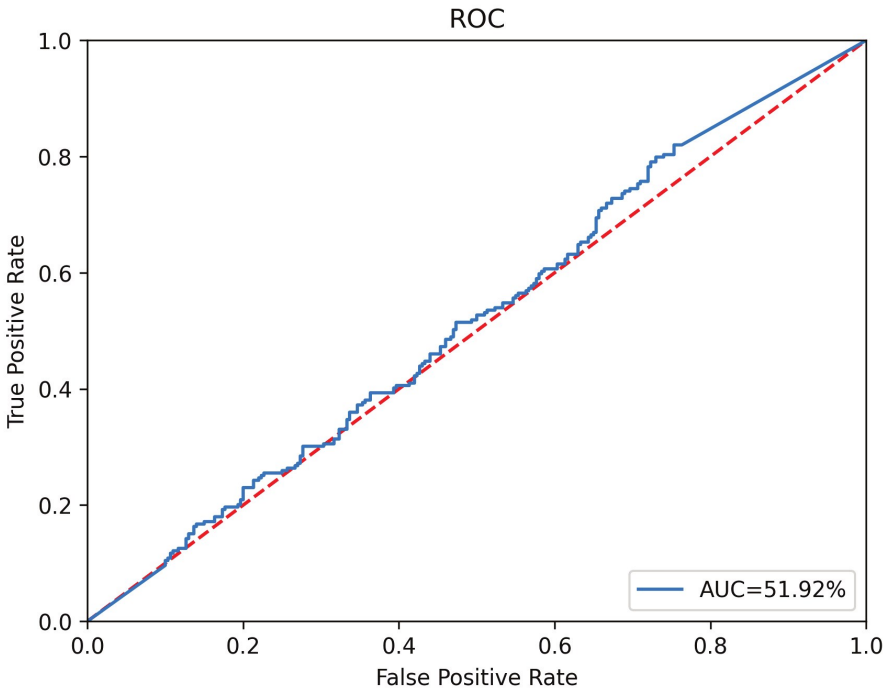
Obrázek 5.16: PRC metrika pro model natrénovaného pomocí upravených snímků metodou CLAHE na L kanálu převedený na binární variantu oříznutím výstupu.



Tabulka 5.4: Confusion matrix pro nejlepší model natrénovaného pomocí upravených snímků metodou CLAHE na L kanálu

5.1.5 Využití veřejných dat

Hlavním problémem při trénování modelu umělé inteligence pro detekování diabetické retinopatie je nedostatek dat k trénování. Proto by se hodilo využít veřejně dostupná data. Je třeba vyzkoušet, jestli model menší velikosti, než který se používal v jiných pracích, bude dostatečný pro naučení rozpoznání symptomů a následného rozdělení nemoci do správné kategorie. Je možné, že malá velikost modelu bude limitujícím faktorem pro nehomogenní data z datasetu EyePACS.



Obrázek 5.17: ROC metrika pro model natrénovaného pomocí EyePACS dataset testovaný na IKEM datasetu

Po natrénování neuronové sítě stejné architektury, která byla použita v předchozích experimentech, pomocí EyePACS datasetu dosahovala MSE chyba na train setu 0,0414. Z matice záměn 5.5 při otestování modelu na IKEM datasetu, ale hlavně z ROC metriky 5.17 je naprosto patrné, že se model vůbec nenaučil rozpoznávat stupně nemoci. Tento výsledek přisuzujeme příliš malé velikosti modelu, který nedovede nalézt potřebné charakteristiky nemoci z různorodých dat. Ukazatelem pro tuto tezi je vyšší chyba u training setu, kde se nám nepovedlo dostat na stejné hodnoty jako v případě privátního datasetu. Je pravděpodobné, že s větším a hlubším modelem by bylo možné model alespoň částečně natrénovat, ale k tomu nám chybějí výpočetní prostředky, protože velké modely mají vysoké nároky na dostupnou paměť.

Vzhledem k naprosto žalostné kvalitě modelu naučeném pouze na datech EyePACS jsme se rozhodli vyzkoušet jiný způsob využití těchto dat. Jelikož je cílem vytrénovat model, který bude co nejlépe fungovat na datech z IKEMu, nechali jsme se inspirovat postupem učení popsaným autory Pratt a kol. [24]. Při učení nejprve použili pouze část dat (cca 10 %), na kterých naučili model a zbylá data přidali až později. My jsme zvolili podobný postup, a to ten, že vezmeme nejlepší model, který jsme naučili na datech z IKEMu a budeme trénovací dataset

Výstup modelu \ Skutečnost	0	1	2	3	4
0	206 69%	139 72%	21 57%	3 75%	3 75%
1	83 28%	46 24%	13 35%	1 25%	1 25%
2	9 3%	9 5%	2 5%	0 0%	0 0%
3	2 1%	0 0%	1 3%	0 0%	0 0%
4	0 0%	0 0%	0 0%	0 0%	0 0%

Tabulka 5.5: Confusion matrix pro model natrénovaného pomocí EyePACS dataset testovaný na IKEM datasetu

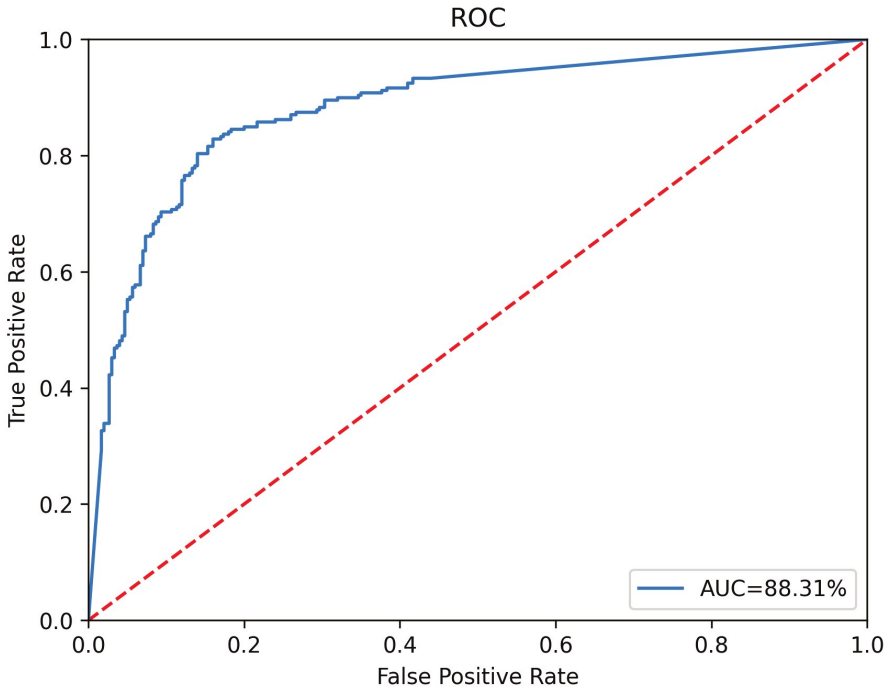
postupně rozšiřovat o část dat z EyePACS pokaždé, když chyba klesne pod určitou hodnotu. Teoreticky bychom zvýšením počtu trénovacích vzorů mohli zvýšit robustnost a přesnost sítě.

Zvolili jsme experiment tak, že jsme vzali nejlepší model natrénovaný na IKEM datasetu ze sekce 5.1.3 a pokaždé, když modelu klesl MSE loss pod 0,1, přidali jsme 200 snímků z datasetu EyePACS k trénovací množině. Tento postup jsme zopakovali desetkrát. Poté jsme nechali model učit s rozšířeným trénovacím datasetem. Výsledný model byl ten, co měl v poslední fázi učení minimální MSE metriku u validačního setu.

Ani tento způsob využití EyePACS datasetu nevedl k lepšímu modelu (viz 5.6, 5.18). Domníváme se, že převážil negativní důsledek rozdílných dat, a proto došlo k lehkému zhoršení.

Výstup modelu \ Skutečnost	0	1	2	3	4
0	275 92%	70 36%	5 14%	1 25%	0 0%
1	23 8%	118 61%	12 32%	0 0%	1 25%
2	1 0%	6 3%	20 54%	2 50%	0 0%
3	1 0%	0 0%	0 0%	1 25%	1 25%
4	0 0%	0 0%	0 0%	0 0%	2 50%

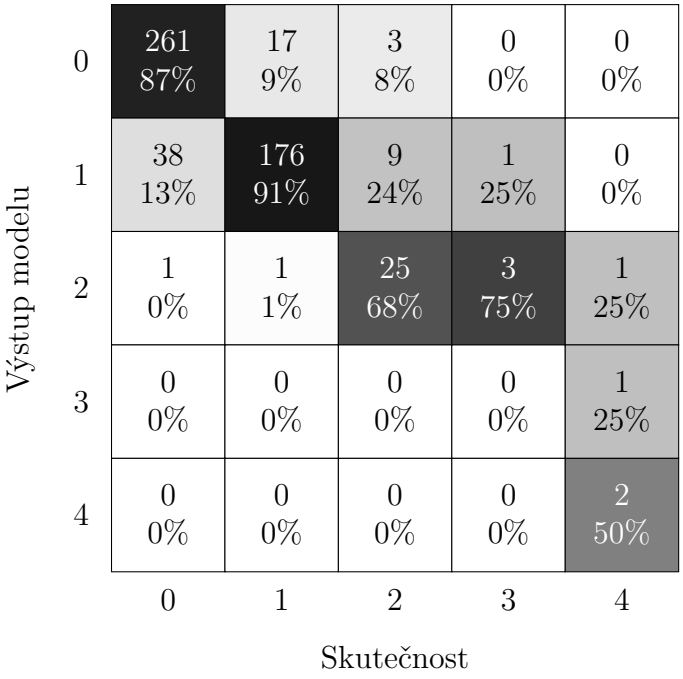
Tabulka 5.6: Confusion matrix pro model natrénovaného pomocí IKEM dataset rozšířený o část EyePACS



Obrázek 5.18: ROC metrika pro model natrénovaného pomocí IKEM dataset rozšířený o část EyePACS

5.1.6 Složení více naučených modelů

Když máme nalezený způsob, jak natrénovat co nejlepší model z dostupných dat, můžeme provést poslední krok, kterým celkovou kvalitu ještě o malinko zlepšíme. Jde o způsob spojení více modelů do jednoho – *ensemble*. Jelikož trénování



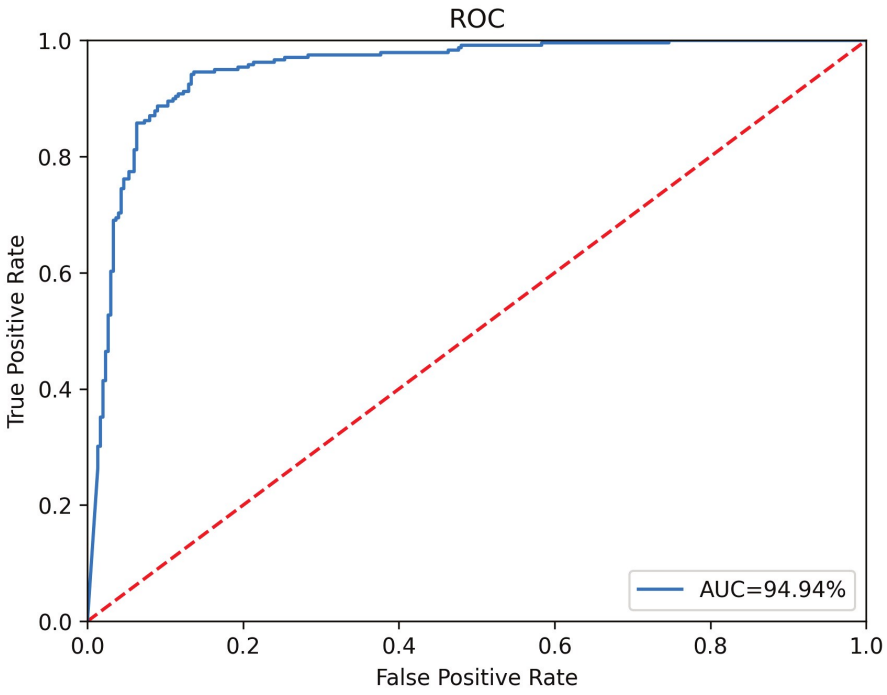
Tabulka 5.7: Confusion matrix pro ensemble model a privátní dataset

neuronové sítě bývá stochastické a závislé na počátečním nastavení vah, můžeme předpokládat, že různé modely naučené na stejných datech budou fungovat trochu odlišně. Každý může být o málo silnější v rozpoznávání některých symptomů a slabší u jiných. Této skutečnosti můžeme využít tak, že budeme průměrovat výsledek několika modelů a tím „vyvažovat“ jejich nedostatky.

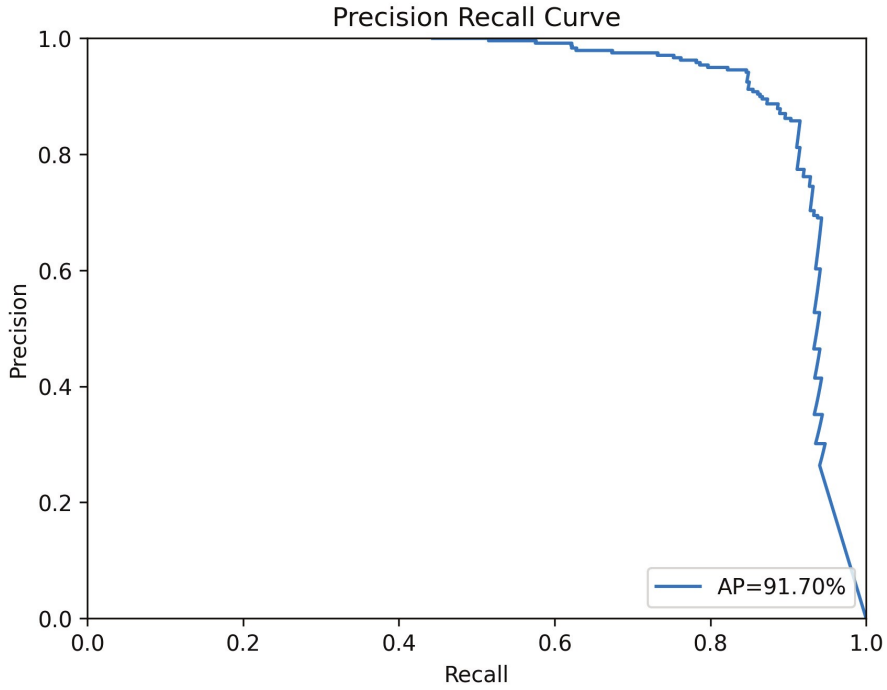
Z výsledků předchozích experimentů jsme označili za nejúspěšnější variantu vlastní model natrénovaný na privátním datasetu s nalezenými hyperparametry ze sekce 5.1.3. Natrénovali jsme 5 takových modelů a následně je složili do jednoho s tím, že výsledek průměrujeme. Otestováním *ensemble* modelu získáme matici záměn 5.7, ROC metrikou 5.19 a PRC metrikou 5.20. Oproti samotnému nejlepšímu modelu došlo k drobnému vylepšení.

Při otestování modelu na části datasetu EyePACS jsme zjistili, že model zdaleka nedosahuje potřebné přesnosti (viz matice záměn 5.8) pro jeho reálné využití u snímků z různých fundus kamer a zařízení. Zdá se, že model je opravdu úzce svázán s uniformními daty ze zařízení IKEM.

Snímky 5.21 5.22 5.23 zobrazují Grad-CAM metodu *ensemble* modelu, kde červená barva označuje místa, která neuronová síť považuje za podstatná pro vrácený stupeň nemoci a modrá barva naopak nepodstatná. Pro vygenerování snímků jsme použili kód A.2.



Obrázek 5.19: ROC metrika pro ensemble model a privátní dataset

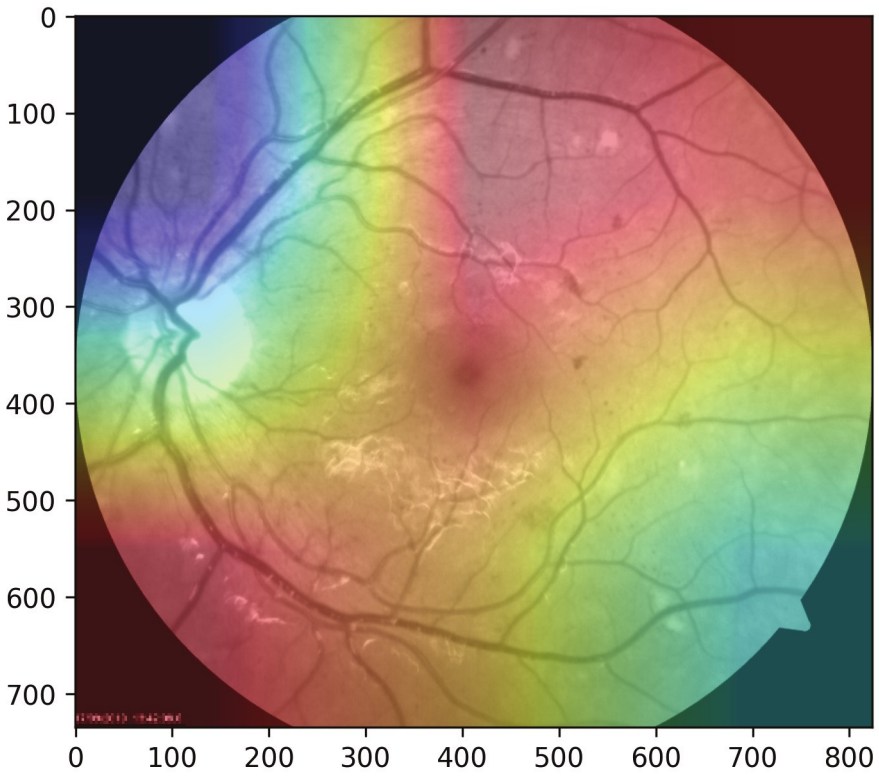


Obrázek 5.20: PRC metrika pro ensemble model a privátní dataset

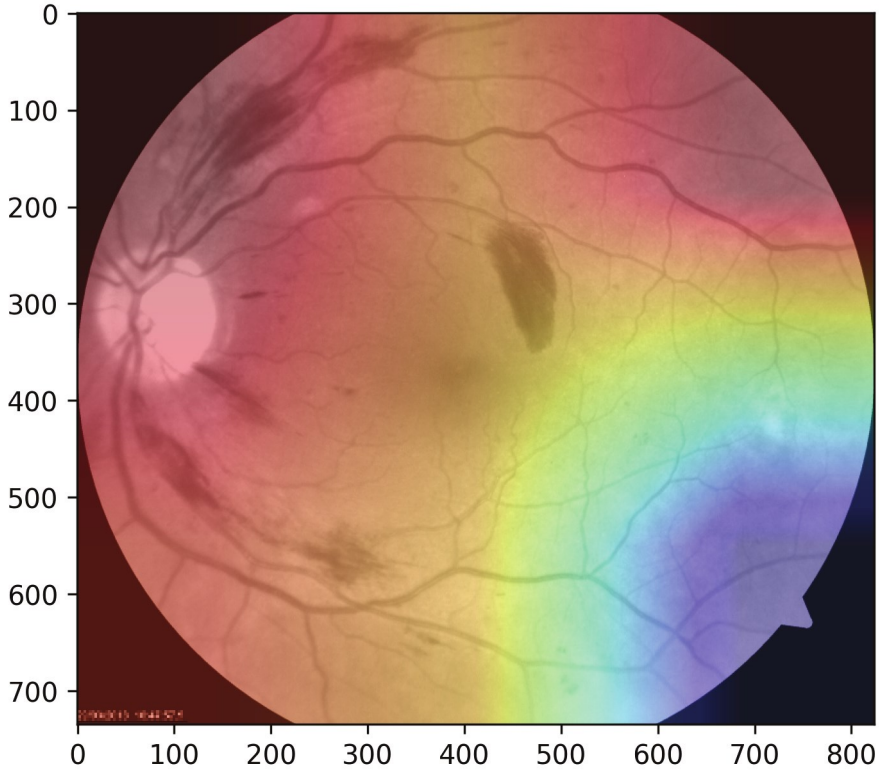
Výstup modelu	0	1	2	3	4
0	12 0%	0 0%	1 0%	0 0%	0 0%
1	639 25%	73 28%	81 16%	12 16%	3 5%
2	1252 48%	131 50%	248 49%	27 36%	23 37%
3	662 26%	56 21%	168 33%	33 45%	31 50%
4	19 1%	1 0%	7 1%	2 3%	5 8%
	0	1	2	3	4

Skutečnost

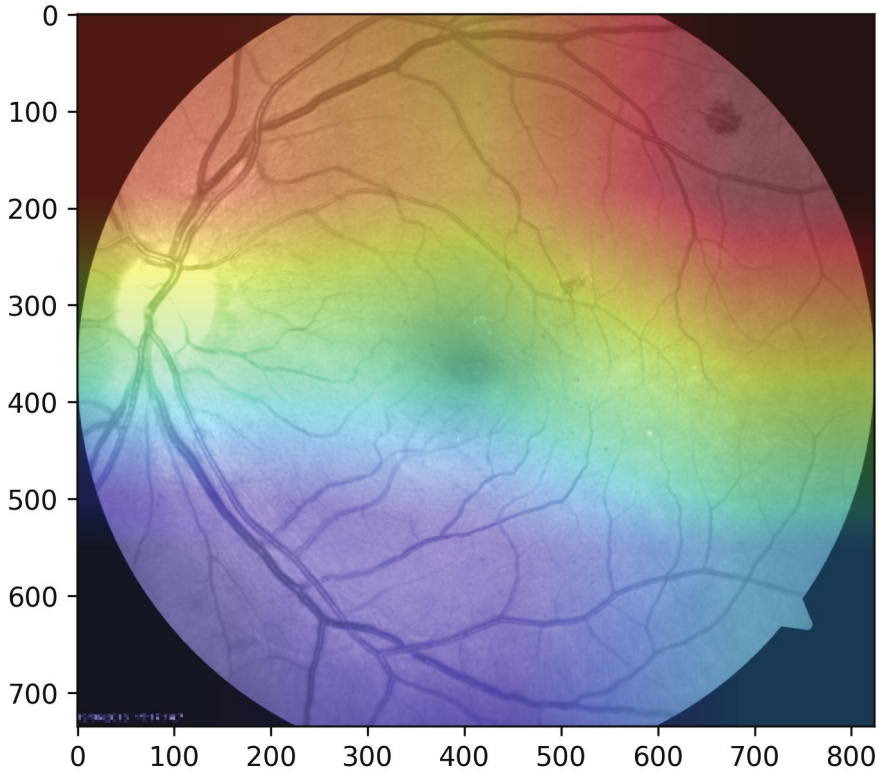
Tabulka 5.8: Confusion matrix pro ensemble model a dataset EyePACS



Obrázek 5.21: Příklad Grad-CAM pro ensemble model



Obrázek 5.22: Příklad Grad-CAM pro ensemble model



Obrázek 5.23: Příklad Grad-CAM pro ensemble model

6. Výsledky experimentů a analýza výkonnosti

Z výsledků posledního experimentu skládající nejlepší modely dohromady lze odvodit, že se podařilo natrénovat model, který dokáže rozpoznávat znaky retinopatie, podle kterých odvodí stupeň nemoci. Bohužel takto dobře funguje pouze na privátních datech IKEMu, které jsou oproti veřejným datům jednotvárné, což evidentně usnadňuje učení a nenutí model k zobecnění řešení. To vede k úzké vazbě mezi modelem a daty a vytrácí se univerzálnost, což je patrné ze špatných predikcí na datasetu EyePACS.

Při trénování modelu na IKEM datasetu se hodnota loss funkce dostala téměř k nule z čehož usuzujeme, že pro daná data měl model dostatečnou kapacitu pro nalezení vhodných parametrů. Oproti tomu při trénování stejného modelu na EyePACS datech se loss pohyboval řádově výše, a to může indikovat, že byla kapacita modelu příliš malá a/nebo architektura mělká. I přes snahy o augmentaci dat nedošlo k vybudování robustnosti modelu, který by byl schopen fungovat na různorodých datech.

Porovnáme-li hodnotu AOC metriky s výsledky z jiných studií, nedosahujeme na stejně vysoké hodnoty i přes skutečnost, že je model specializovaný na konkrétní data. Pokud bychom se snažili srovnat výsledky u veřejného dataset EyePACS, dojdeme k závěru, že je náš model výrazně horší a nevhodný pro univerzální použití.

Připravení snímků metodou CLAHE, k našemu překvapení, nevedlo k lepším výsledkům predikce. Pravděpodobně jsou symptomy stejně patrné před i po zvýraznění detailů, a proto nedošlo ke zlepšení.

Nový způsob označení stupně nemoci pomocí regrese se ukázal být funkční alternativou k používané binární kategorizaci přítomnosti nemoci nebo konkrétního symptomu. Myslíme si, že tato cesta je vhodnější než kategorizace podle symptomů, protože to snižuje nároky na přípravu trénovacích dat, kterých je v medicínských oborech nedostatek.

7. Webový klient

Hlavním cílem práce bylo natrénování modelu, který bude rozpoznávat diabetickou retinopatii a bude možné ji využít v nemocnici IKEM. Proto bylo třeba vytvořit nástroj s rozhraním, které bude moci využívat i člověk s běžnou znalostí práce s počítačem. Možnosti se nabízely dvě, a to aplikace, která by běžela lokálně na počítači a nebo webová stránka, která poběží na jednom místě a uživatelé k ní budou přistupovat přes webový prohlížeč odkudkoliv. Druhá možnost se nám jevila jako výhodnější, protože na cílovém počítači není třeba nic připravovat a postačuje jakýkoliv prohlížeč, který je dostupný na téměř všech zařízeních připojených na internet. Druhou výhodou jsou případné aktualizace, které se provedou pouze na jednom místě a uživatelé se k nové verzi dostanou okamžitě pouze přístupem na webovou adresu. Nevýhodou mohou být vyšší výpočetní nároky na server, pokud by se neuronová síť osvědčila a začalo ji využívat více lékařů současně. Na druhou stranu by bylo možné využít specializovaný hardware počítající výstup sítě rychleji a efektivněji, který však není výbavou běžných počítačů.

Pokud bychom se rozhodli pro aplikaci, případná zátěž by se rozložila mezi uživatele a výpočet by probíhal na jejich počítači, ale budoucí aktualizace aplikace by nebyly tak přímočaré.

Existuje ještě hybridní varianta, progresivní webová aplikace[1], která těží z výhod obou přístupů. Umožňovala-by aktualizovat model i webové rozhraní pro všechny současně, ale zároveň by výpočet probíhal u klienta a tím nekladl vyšší nároky na výkon serveru. Při vytváření takové webové aplikace postavené na platformě .NET jsme narazili na nedostatečné schopnosti existujících knihoven umožňující spouštění neuronových sítí. Vypočítání výstupu sítě není problém, ale my potřebujeme navíc ještě provádět i zpětné šíření chyby a počítání parciálních derivací při vizualizaci Grad-CAM. Námi zkoušené knihovny to ale v prostředí WebAssembly[2] neumožňují. Dále je možné, že by velikost webové aplikace byla výrazně větší, než je typická velikost webové stránky, protože by musela obsahovat i framework pro práci s neuronovou sítí. Každý přístup na stránku by vedl ke stažení celé webové aplikace a tím zase zatěžoval kapacitu sítě.

Na základě chybějících možností knihoven při vytvoření progresivní webové aplikace jsme se vrátili k původnímu řešení a vytvořili webovou stránku pojmenovanou **DiabeSee**. Usoudili jsme se, že možnost snadnější aktualizace webu a modelů je klíčová vlastnost při vývoji experimentálního projektu a převažuje před negativy vyšších výpočetních nároků. Web je psán v jazyce Python a díky tomu může využívat stejnou vývojovou platformu, která byl použita pro trénování neuronové sítě. To nám značně zjednodušilo vývoj, protože nebylo třeba řešit kompatibilitu uložených modelů s jinými knihovnami a můžeme použít podobný kód pro vytvoření Grad-CAM vizualizace podstatných oblastí vstupního snímku.

Využili jsme dostupný minimalistický framework Flask[3] pro tvorbu webů, který za nás řeší směrování požadavků mířící na webový server (viz 7.1) a my se můžeme soustředit pouze na vzhled a funkcionalitu stránky. Ukázku webu si lze prohlédnout v příloze na obrázcích v příloze A.1 A.2 A.3.

[1] `https://en.wikipedia.org/wiki/Progressive_web_app`

[2] Webový standard pro strojový kód běžící v prohlížeči

[3] `https://flask.palletsprojects.com`

```
1 @app.route("/")
2 def hello_world():
3     return "<p>Hello, World!</p>"
```

Zdrojový kód 7.1: Nastavení směrování kořenové adresy pro vrácení jednoduchého HTML paragrafu pomocí knihovny Flask

Protože je zpracování jednoho snímku časově i výkonově náročné, použili jsme modul Flask-Cache[4], pomocí kterého si v paměti po určitou dobu udržujeme výsledky. Nastavili jsme časový limit na 5 minut, ve kterých předpokládáme, že by mohl uživatel nedopatřením poslat ke zpracování stejný snímek. Omezili jsme počet držených výsledků na 10, abychom předešli nadměrnému využití pamětí. Oba limity jsou sdílené mezi všemi uživateli, takže nehrozí zaplnění paměti ani při mnoha požadavcích během krátké doby.

Část kódu zodpovědná za spuštění natrénovaných modelů byla napsána tak, aby je bylo možné vyměnit. Nové modely však musí mít stejné rozměry vstupu i výstupu a kvůli metodě Grad-CAM je nutné, aby využívaly konvoluční vrstvy. K této vlastnosti jsme přistoupili proto, abychom co nejvíce zjednodušili případný vývoj aplikace po nasazení do reálného provozu. Pro výměnu modelů je stačí nahradit ve složce a restartovat webovou službu.

Abychom předešli problémům se závislostmi na různých verzích knihoven a programů při nasazení a spuštění webové služby, rozhodli jsme se využít kontejnerizaci pomocí programu Docker[5]. Kontejnerizace může být chápána jako zabalení aplikace, jejích závislostí a celého operačního systému (typicky minimalistické verze Linuxu) do jednoho balíku, který lze následně spustit. Tímto způsobem je zajištěna izolace od zbytku systému a zároveň je zaručena dostupnost všech potřebných závislostí pro bezproblémový běh aplikace, aniž by bylo nutné složité nastavování prostředí.

[4] https://flask-caching.readthedocs.io
[5] https://www.docker.com/

8. Získané znalosti

8.1 Shrnutí klíčových zjištění studie

Při provádění experimentů popsaných v kapitole 5 jsme se přesvědčili o skutečnosti, že pro úspěšné natrénování modelu detekující diabetickou retinopatii na různorodých snímcích je třeba velké množství dat a zároveň model s velkou kapacitou. Oproti jiným studiím jsme pracovali s modely s menší kapacitou, a proto se nám nepodařilo vytvořit model dostatečně robustní pro všeobecné použití. V případě specializace na jednotvárná data nebyla malá kapacita modelu limitující a postačila pro vytvoření funkčního, byť omezeného, modelu.

Použití jednoho výstupního neuronu, který udává přímo stupeň nemoci se ukázal jako funkční alternativa k binární kategorizaci. Předpoklad, že lze na stupeň nemoci koukat jako na škálu symptomů, se ukázal být pravdivý.

8.2 Využití výsledků studie v praxi

Natrénovaný model bude použit v testovacím režimu v zařízení IKEM prostřednictvím vytvořeného webového prostředí. Bude se ověřovat jeho přínos při diagnóze pacientů a vždy bude sloužit pouze v kooperaci s lékařem. Z výsledků je patrné, že je model úzce svázán s jednotvárností snímků, což se může ukázat jako problém při změně fundus kamery. Proto bude třeba v budoucnu přijít s robustnějším modelem, který nebude trpět tímto omezením.

Použitá metoda CLAHE 5.1.4 může být potenciálně prospěšná pro zvýraznění detailů snímku při jeho manuálním zkoumání lékařem. Při testovacím provozu neuronové sítě bude nabídnuto IKEMu připravit nástroj, který bude transformovat snímky touto metodou.

8.3 Návrhy pro budoucí práci na zlepšení výkonnosti navrhovaného přístupu

Z experimentů a jejich výsledků je patrné, že je třeba prozkoumat architektury neuronových sítí s více vrstvami a větší kapacitou, což se ukázalo být limitujícím faktorem v této práci. Více se zaměřit na kvalitu modelu na různorodých datech a ne pouze na malém datasetu.

V experimentu předzpracující snímek jsme použili metodu CLAHE, která se nakonec neukázala jako prospěšná. Stálo by za zvážení prozkoumat více metod předzpracujících snímky a pokusit se nalézt takovou, která by ulehčila modelu rozpoznat symptomy, potažmo stupeň nemoci nebo experimentálně ověřit, že předzpracování snímků nehraje důležitou roli.

Pokud by se publikoval dataset s označenými pixely se symptomy, bylo by zajímavé prozkoumat možnosti modelu určující oblasti s nalezenými symptomy. Popisovali jsme studie, které se zaměřovaly na označování rakovinných buněk na snímcích, které by mohly posloužit jako inspirace pro budoucí práce zaměřené na diagnostiku diabetické retinopatie.

Jedním ze zajímavých způsobů, které by stálo za to v budoucí práci prozkoumat, by bylo použití architektury sítě známý pod názvem *autoencoder*. Hlavní myšlenka této sítě je natrénování efektivního zakódování vstupu pomocí postupného zmenšování dimenze a následné zvětšení do původní velikost. Tím vznikne v síti úzké hrdlo. Síť je následně trénována k tomu, aby vracela stejný výstup jako dostane na vstupu. Takto je nucena uchovat si co nejvíce detailů při průchodu úzkým hrdlem. My bychom mohli síť v nejužším bodě rozdělit a přidat vrstvy starající se o rozpoznání diabetické retinopatie. Naše představa je taková, že by síť byla více tlačena ke správnému extrahování *features* ze snímku sítnice a tím zjednodušit trénovaní té části, které je zodpovědná za určení stupně nemoci.

Předpokládáme, že se v budoucnu bude tento systém nebo jemu podobný hojně využívat, a proto by nebylo na škodu rozdělit webovou stránku na tři části. První, která by tvořila vzhled a poskytovala by uživatelům možnost interakce s natrénovanými modely. Druhá část by bylo API, které by bylo provolávané webovou stránkou a zároveň by ho bylo možné volat bez přístupu na web. Tím by se umožnilo napojit jiné nástroje na tento systém. Poslední částí by byla službu, která by obstarávala samotné spouštění neuronové sítě. Toto rozdělení by umožnilo jak dynamické spuštění více služeb podle potřeby a tím distribuovat složitý výpočet na více strojů, tak i dostupné rozhraní pro napojení jiných nástrojů, které by mohly stavět na tomto systému.

Závěr

Na základě poskytnutých dat ze zařízení IKEM jsme vyvinuli model neuronové sítě schopný rozpoznávat diabetickou retinopatii ze snímků sítnice. Zvolili jsme vstupní rozlišení 824x735 pixelů se třemi barevnými kanály. Na rozdíl od jiných přístupů jsme se rozhodli řešit problém jako regresi, nikoliv kategorizaci. Místo jednoduchého rozdělení na zdravý a nemocný či více binárních výstupů podle přítomnosti specifického symptomu, jsme zvolili mapování nemoci na číselnou škálu od 0 do 4 podle stupně nemoci.

Vzhledem k typu vstupních dat jsme zvolili architekturu konvoluční neuronové sítě (CNN). Oproti obecným neuronovým sítím umožňuje konvoluční architektura efektivní extrakci a interpretaci vizuálních informací. Námi definovaný model dosahoval velikost lehce přes 120 tisíc parametrů. V porovnání s modely jiných studií, které dosahují velikosti v řádů jednotek milionů parametrů, jsme použili model mnohem menší, který bylo možné trénovat i na běžném zařízení. Navzdory menší velikosti našeho modelu jsme při trénovaní na privátních datech dosahovali téměř nulové chyby, což svědčí o dostatečné kapacitě modelu pro řešení problému. Testování na privátních datech poskytovalo nadějné výsledky, avšak při testování na veřejných, různorodých datech odhalilo limity malé velikosti sítě. Vzhledem k cíli práce, vytvořit nástroj pro zařízení IKEM s jedním typem fundus kamery, není omezená univerzalita nutně problémem.

Pro konečné využití jsme přistoupili k natrénování pěti neuronových sítí. Při aplikaci jsme využili výsledků všech těchto sítí, které jsme následně průměrovali. Z výsledků testování vyplývá, že toto opatření vedlo ke zvýšené přesnosti v porovnání s jediným modelem.

Součástí práce bylo vytvoření webového prostředí, pomocí kterého je možné zpracovat vložené snímky natrénovanými modely. Web jsme nazvali DiabeSee a je dostupný na adrese `http://www.ms.mff.cuni.cz/~kubovyj/diabesee`. Obsluha tohoto nástroje je dostatečně snadná, aby ji mohl využívat i netrénovaný personál. Spolu s predikcí stupně nemoci je uživateli zobrazen i vložený snímek se zvýrazněnými částmi, které modely považovaly za podstatné pro zvolený stupeň (metoda Grad-CAM).

Námi definovaný cíl, vytvoření jednoduše použitelného nástroje pro predikci stupně diabetické retinopatie, jsme splnili vytvořením popsané webové stránky. Snahu o minimalizování falešně negativních výsledků jsme vyřešili zobrazením predikovaného stupně v podobě reálného čísla včetně rozptylu. Lékařský personál má možnost si určit vlastní hranici, pod kterou budou pokládat pacienta za zdravého. Zobrazením stupně nemoci jsme splnili i sekundární cíl, a to možnost řadit pacienty podle závažnosti nemoci. Třetí a poslední cíl byl stanoven na grafické znázornění těch částí vstupního snímku, které podle neuronové sítě rozhodují o stupni nemoci. Toho jsme docílili použitím zmíněné metody Grad-CAM a jejím zobrazení na webové stránce.

Vytvořené webové prostředí spolu s natrénovanými modely bude experimentálně nasazeno v zařízení IKEM, kde se bude testovat jeho kvalita a případná míra zjednodušení diagnostiky pacientů v součinnosti s odborným personálem. Zároveň budou demonstrovány možnosti metody zvýrazňující detaily krevního řečiště na snímcích sítnice, která byla zkoumána v této práci a kterou by bylo

možné využít při manuální zkoumání snímku lékařem.

Seznam použité literatury

[1] Bewick, V., Cheek, L. a Ball, J. (2005). Statistics review 14: Logistic regression. *Critical Care*, **9**(1), 112. doi: 10.1186/cc3045.

[2] Breiman, L., Friedman, J. H., Olshen, R. A. a Stone, C. J. (2017). *Classification And Regression Trees*. Routledge. doi: 10.1201/9781315139470.

[3] Burges, C. J. a Crisp, D. (1999). Uniqueness of the svm solution. *Advances in neural information processing systems*, **12**.

[4] Chen, C.-L., Chen, C.-C., Yu, W.-H., Chen, S.-H., Chang, Y.-C., Hsu, T.-I., Hsiao, M., Yeh, C.-Y. a Chen, C.-Y. (2021). An annotation-free whole-slide training approach to pathological classification of lung cancer types using deep learning. *Nature Communications*, **12**(1). doi: 10.1038/s41467-021-21467-y.

[5] Cortes, C. a Vapnik, V. (1995). Support-vector networks. *Machine Learning*, **20**(3), 273–297. doi: 10.1007/bf00994018.

[6] Fong, D. S., Aiello, L., Gardner, T. W., King, G. L., Blankenship, G., Cavallerano, J. D., Ferris, F. L. a Klein, R. (2003). Retinopathy in diabetes. *Diabetes Care*, **27**(Supplement 1), S84–S87. doi: 10.2337/diacare.27.2007.s84.

[7] Goodfellow, I., Bengio, Y. a Courville, A. (2017). *Deep Learning*. MIT Press. ISBN 9780262035613.

[8] Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, **4**(2), 251–257. doi: 10.1016/0893-6080(91)90009-t.

[9] Huang, G., Liu, Z., van der Maaten, L. a Weinberger, K. Q. (2016). Densely connected convolutional networks.

[10] Indumathi, G. a Sathananthavathi, V. (2019). *Microaneurysms Detection for Early Diagnosis of Diabetic Retinopathy Using Shape and Steerable Gaussian Features*, pages 57–69. Elsevier. doi: 10.1016/b978-0-12-816948-3.00005-2.

[11] Kanda, D., , Kawai, S. a Nobuhara, H. (2020). Visualization method corresponding to regression problems and its application to deep learning-based gaze estimation model. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, **24**(5), 676–684. doi: 10.20965/jaciii.2020.p0676.

[12] Kang, D.-Y., Duong, H. P. a Park, J.-C. (2020). Application of deep learning in dentistry and implantology. *The Korean Academy of Oral and Maxillofacial Implantology*, **24**(3), 148–181. doi: 10.32542/implantology.2020.2015.

[13] Kanopoulos, N., Vasanthavada, N. a Baker, R. (1988). Design of an image edge detection filter using the sobel operator. *IEEE Journal of Solid-State Circuits*, **23**(2), 358–367. doi: 10.1109/4.996.

[14] Kramer, O. (2013). K-nearest neighbors. In *Dimensionality Reduction with Unsupervised Nearest Neighbors*, pages 13–23. Springer Berlin Heidelberg. doi: 10.1007/978-3-642-38652-7_2.

[15] Kubů, E. (2016). Diabetes mellitus a oční komplikace. URL `https://is.muni.cz/th/nboht/`.

[16] Lancet, T. (2023). Diabetes: a defining disease of the 21st century. *The Lancet*, **401**(10394), 2087. doi: 10.1016/s0140-6736(23)01296-5.

[17] Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A. a Talwalkar, A. (2018). Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, **18**(185), 1–52. URL `http://jmlr.org/papers/v18/16-558.html`.

[18] Liu, Y., Kohlberger, T., Norouzi, M., Dahl, G. E., Smith, J. L., Mohtashamian, A., Olson, N., Peng, L. H., Hipp, J. D. a Stumpe, M. C. (2018). Artificial intelligence–based breast cancer nodal metastasis detection: Insights into the black box for pathologists. *Archives of Pathology & Laboratory Medicine*, **143**(7), 859–868. doi: 10.5858/arpa.2018-0147-oa.

[19] Lowe, D. (1999). Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*. IEEE. doi: 10.1109/iccv.1999.790410.

[20] Lu, J., Ma, X., Zhou, J., Zhang, L., Mo, Y., Ying, L., Lu, W., Zhu, W., Bao, Y., Vigersky, R. A. a Jia, W. (2018). Association of time in range, as assessed by continuous glucose monitoring, with diabetic retinopathy in type 2 diabetes. *Diabetes Care*, **41**(11), 2370–2376. ISSN 1935-5548. doi: 10.2337/dc18-1131.

[21] Nam, J. G., Park, S., Hwang, E. J., Lee, J. H., Jin, K.-N., Lim, K. Y., Vu, T. H., Sohn, J. H., Hwang, S., Goo, J. M. a Park, C. M. (2019). Development and validation of deep learning–based automatic detection algorithm for malignant pulmonary nodules on chest radiographs. *Radiology*, **290**(1), 218–228. doi: 10.1148/radiol.2018180237.

[22] NERA (2021). Diabetic retinopathy. URL `https://web.archive.org/web/20230419125824/https://www.retinamd.com/diseases-and-treatments/retinal-conditions-and-diseases/diabetic-retinopathy/`. Accessed: 2023-05-26.

[23] Nesterov, Y. E. (1983). A method of solving a convex programming problem with convergence rate o\bigl(k^2\bigr). In *Doklady Akademii Nauk*, volume 269, pages 543–547. Russian Academy of Sciences.

[24] Pratt, H., Coenen, F., Broadbent, D. M., Harding, S. P. a Zheng, Y. (2016). Convolutional neural networks for diabetic retinopathy. *Procedia Computer Science*, **90**, 200–205. doi: 10.1016/j.procs.2016.07.014.

[25] Rumelhart, D. E., Hinton, G. E. a Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, **323**(6088), 533–536. doi: 10.1038/323533a0.

[26] Saeedi, P., Petersohn, I., Salpea, P., Malanda, B., Karuranga, S., Unwin, N., Colagiuri, S., Guariguata, L., Motala, A. A., Ogurtsova, K., Shaw, J. E., Bright, D. a Williams, R. (2019). Global and regional diabetes prevalence estimates for 2019 and projections for 2030 and 2045: Results from the international diabetes federation diabetes atlas, 9th edition. *Diabetes Research and Clinical Practice*, **157**, 107843. doi: 10.1016/j.diabres.2019.107843.

[27] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. a Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. doi: 10.485 50/ARXIV.1801.04381.

[28] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D. a Batra, D. (2016). Grad-cam: Visual explanations from deep networks via gradient-based localization. doi: 10.48550/ARXIV.1610.02391.

[29] Singh, H. (2019). *Practical Machine Learning and Image Processing: For Facial Recognition, Object Detection, and Pattern Recognition Using Python*. Apress. ISBN 9781484241493. doi: 10.1007/978-1-4842-4149-3.

[30] Suk, H.-I. (2017). An introduction to neural networks and deep learning. In *Deep Learning for Medical Image Analysis*, pages 3–24. Elsevier. doi: 10.1016/b978-0-12-810408-8.00002-x.

[31] Tschandl, P., Rosendahl, C., Akay, B. N., Argenziano, G., Blum, A., Braun, R. P., Cabo, H., Gourhant, J.-Y., Kreusch, J., Lallas, A., Lapins, J., Marghoob, A., Menzies, S., Neuber, N. M., Paoli, J., Rabinovitz, H. S., Rinner, C., Scope, A., Soyer, H. P., Sinz, C., Thomas, L., Zalaudek, I. a Kittler, H. (2019). Expert-level diagnosis of nonpigmented skin cancer by combined convolutional neural networks. *JAMA Dermatology*, **155**(1), 58. doi: 10.1001/jamadermatol.2018.4378.

[32] Voulodimos, A., Doulamis, N., Doulamis, A. a Protopapadakis, E. (2018). Deep learning for computer vision: A brief review. *Computational Intelligence and Neuroscience*, **2018**, 1–13. doi: 10.1155/2018/7068349.

[33] Yann LeCun, C. C. The mnist database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*.

[34] ÚZIS (2020). Diabetická retinopatie. URL `https://web.archive.org/web/20200815191953/https://nsc.uzis.cz/zdraveoci/index.php?pg=diabeticka-retinopatie`. Accessed: 2023-04-03.

Seznam obrázků

1.1 Ukázka symptomů diabetické retinopatie (převzato [22]) 7

2.1 Ilustrace neuronu 12
2.2 Ilustrace vrstvené dopředné sítě 13
2.3 Průběh učení - ukázka přeučení modelu. Nejlepší model je v bodě, kde chyba dosahuje minima na validačním setu. 14
2.4 Ilustrace vlivu augmentace dat. Představíme-li si výstup modelu jako projekci ze vstupního prostoru do výstupního, tak augmentace rozšiřuje okolí jednoho vstupu a model je nucen zobecnit řešení. Augmentace nenahradí nová trénovací data, ale umožní lépe využít existující. 15
2.5 Prahová aktivační funkce ($\alpha = 0$) 17
2.6 Sigmoid aktivační funkce a její derivace 17
2.7 ReLU aktivační funkce a její derivace 18
2.8 Swish aktivační funkce a její derivace 18
2.9 Rozdíl mezi použitím momentu po výpočtu gradientního kroku (vlevo) a aplikování momentu před výpočtem (vpravo, Nesterov) . 20
2.10 Ukázka výpočtu konvoluční vrstvy 21
2.11 Ukázka pooling vrstvy o velikosti (2,2) používající maximum. Z původní dimenze (4,4,1) vznikla (2,2,1) 22
2.12 Graf přesnosti algoritmů, které vyhrály soutěž ILSVRC v letech 2010-2017. Modré sloupečky označují konvoluční sítě. Převzato z práce[12] 23
2.13 Příklad vstupního obrázku k různým augmentacím. Převzato z projektu imgaug (`https://github.com/aleju/imgaug`) 24
2.14 Příklad různých augmentací obrázku. Převzato z projektu imgaug (odkaz stejný jako na obrázku 2.13) 24
2.15 Ukázka ROC křivky (1000 negativních výsledků a 200 pozitivních). Negativní výsledky byly náhodně vygenerovány pomocí normálního rozdělení se střední hodnotou 0,1 a směrodatnou odchylkou 0,61. U pozitivních výsledků jsme použili střední hodnotu 0,9 a směrodatnou odchylku 0,45 26
2.16 Ukázka PRC křivky (1000 negativních výsledků a 200 pozitivních). Negativní výsledky byly náhodně vygenerovány pomocí normálního rozdělení se střední hodnotou 0,1 a směrodatnou odchylkou 0,61. U pozitivních výsledků jsme použili střední hodnotu 0,9 a směrodatnou odchylku 0,45 26
2.17 Základní metriky 27

4.1 Příklad zobrazení Grad-CAM techniky u snímku se stupněm 2 DR 35

5.1 Základní model s třemi barevnými kanály s velikostmi 1647, 823 a 412 px na šířku. Křivky ukazují vývoj chyby na validačním setu a body na konci chybu na test setu. 37
5.2 Základní model s jedním barevným kanálem a různými velikostmi vstupu. 37

5.3 ROC metrika pro předtrénovaný model MobileNetV2 doučený na IKEM datasetu 39
5.4 MSE chyba na validačních datech po epochách. Pro každou aktivační funkci byla data zprůměrována přes optimalizační funkce. . 41
5.5 MSE chyba na validačních datech po epochách. Pro každou optimalizační funkci byla data zprůměrována přes aktivační funkce. . 41
5.6 Vliv na MSE chybu použitím jednoduché augmentace obrázku. . . 42
5.7 Počet epoch, ve kterých měla daná kombinace hyperparametrů nejmenší MSE chybu na validačním setu. 43
5.8 Počet epoch, ve kterých měla daná kombinace hyperparametrů nejmenší MSE chybu na validačním setu. Počítání z posledních 15 epoch. 43
5.9 ROC metrika pro model převedený na binární variantu ořízutím výstupu. 44
5.10 PRC metrika pro model převedený na binární variantu ořízutím výstupu. 44
5.11 Vstupní obrázek pro ukázku augmentace. 47
5.12 CLAHE metoda na zeleném barevném kanálu 47
5.13 CLAHE metoda na L světelném kanálu 48
5.14 CLAHE metoda na všech LAB kanálech 48
5.15 ROC metrika modelu natrénovaného pomocí upravených snímků metodou CLAHE na L kanálu převedený na binární variantu ořízutím výstupu. 49
5.16 PRC metrika pro model natrénovaného pomocí upravených snímků metodou CLAHE na L kanálu převedený na binární variantu ořízutím výstupu. 49
5.17 ROC metrika pro model natrénovaného pomocí EyePACS dataset testovaný na IKEM datasetu 51
5.18 ROC metrika pro model natrénovaného pomocí IKEM dataset rozšířený o část EyePACS 53
5.19 ROC metrika pro ensemble model a privátní dataset 55
5.20 PRC metrika pro ensemble model a privátní dataset 55
5.21 Příklad Grad-CAM pro ensemble model 56
5.22 Příklad Grad-CAM pro ensemble model 57
5.23 Příklad Grad-CAM pro ensemble model 57

A.1 DiabeSee web: Titulní strana 74
A.2 DiabeSee web: Strana se vstupem 75
A.3 DiabeSee web: Strana se výsledkem 76

Seznam tabulek

1.1 Dělení diabetické retinopatie 6

5.1 Confusion matrix pro předtrénovaný model MobileNetV2 doučený na IKEM datasetu 39
5.2 Architektura neuronové sítě 40
5.3 Confusion matrix pro nejlepší model natrénovaného pomocí hyperband tuner 45
5.4 Confusion matrix pro nejlepší model natrénovaného pomocí upravených snímků metodou CLAHE na L kanálu 50
5.5 Confusion matrix pro model natrénovaného pomocí EyePACS dataset testovaný na IKEM datasetu 52
5.6 Confusion matrix pro model natrénovaného pomocí IKEM dataset rozšířený o část EyePACS 53
5.7 Confusion matrix pro ensemble model a privátní dataset 54
5.8 Confusion matrix pro ensemble model a dataset EyePACS 56

A. Přílohy

A.1 Zdrojové kódy

```
def build_model(hp):
  hp_optimizer = hp.Choice('optimizer', values=['adam', 'nadam'])
  hp_learning_rate = hp.Choice('learning_rate',
              values=[1e-3, 7e-4, 5e-4, 3e-4]
              )
  if hp_optimizer == 'sgd':
    _optimizer =
    ↪  tf.keras.optimizers.SGD(learning_rate=hp_learning_rate)
  elif hp_optimizer == 'rmsprop':
    _optimizer =
    ↪  tf.keras.optimizers.RMSprop(learning_rate=hp_learning_rate)
  elif hp_optimizer == 'adam':
    _optimizer =
    ↪  tf.keras.optimizers.Adam(learning_rate=hp_learning_rate)
  elif hp_optimizer == 'nadam':
    _optimizer =
    ↪  tf.keras.optimizers.Nadam(learning_rate=hp_learning_rate)
  else:
    raise

  hp_loss = hp.Choice('loss', values=['huber', 'mse'])
  if hp_loss == 'huber':
    _loss = tf.keras.losses.Huber()
  elif hp_loss == 'mse':
    _loss = tf.keras.losses.MeanSquaredError()
  else:
    raise

  activation_function = hp.Choice('activation', values=['relu',
  ↪  'leaky_relu', 'swish', 'gelu', 'elu'])

  inputs = tf.keras.layers.Input(shape=[args.height, args.width,
  ↪  args.channels])

  inputs = tf.keras.layers.Rescaling(scale=1./127.5,
  ↪  offset=-1)(inputs)
  inputs = tf.keras.layers.RandomFlip('horizontal')(inputs)
  inputs = keras_cv.layers.RandomHue(.2, [-1.0,1.0])(inputs)
  inputs = keras_cv.layers.RandomBrightness(.2, [-1.0,1.0])(inputs)

  conv = tf.keras.layers.Conv2D(16, 5, 1,
  ↪  activation=activation_function)(inputs)
  conv = tf.keras.layers.MaxPool2D(2)(conv)
  conv = tf.keras.layers.Conv2D(16, 3, 1,
  ↪  activation=activation_function)(conv)
  conv = tf.keras.layers.MaxPool2D(3)(conv)
  conv = tf.keras.layers.Conv2D(32, 5, 1,
  ↪  activation=activation_function)(conv)
  conv = tf.keras.layers.MaxPool2D(3)(conv)
  conv = tf.keras.layers.Conv2D(32, 5, 1,
  ↪  activation=activation_function)(conv)
```

```
41    conv = tf.keras.layers.MaxPool2D(3)(conv)
42
43    conv = tf.keras.layers.Conv2D(64, 3, 1,
      ↪  activation=activation_function)(conv)
44    conv = tf.keras.layers.MaxPool2D(2)(conv)
45    conv = tf.keras.layers.Conv2D(64, 3, 1,
      ↪  activation=activation_function)(conv)
46
47    hidden = tf.keras.layers.Flatten()(conv)
48    hidden = tf.keras.layers.Dense(hp.Int('hidden_size_1', 50, 120,
      ↪  step=10, default=75), activation=activation_function)(hidden)
49    hidden = tf.keras.layers.Dense(hp.Int('hidden_size_2', 30, 100,
      ↪  step=10, default=50), activation=activation_function)(hidden)
50    hidden = tf.keras.layers.Dense(hp.Int('hidden_size_3', 20, 50,
      ↪  step=10, default=30), activation=activation_function)(hidden)
51    outputs = tf.keras.layers.Dense(1, dtype='float32',
      ↪  activation=None)(hidden)
52    model = tf.keras.Model(inputs=inputs, outputs=outputs)
53    model.compile(optimizer=_optimizer, loss=_loss, metrics=['mse'])
54    print(model.summary())
55    return model
56
57  ...
58
59  tuner_dir = 'hyperband_tuner'
60  tuner = kt.Hyperband(get_build_model_function(args),
61                       objective='val_mse',
62                       max_epochs=args.epochs,
63                       hyperband_iterations=2,
64                       factor=3,
65                       directory=tuner_dir,
66                       project_name=args.model_name,
67                       overwrite=False)
68
69  dataset = Dataset(args.batch_size, args.width, args.height,
    ↪  args.channels)
70  stop_early = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
    ↪  patience=30)
71  tuner.search(dataset.train, validation_data=dataset.validation,
    ↪  class_weight=dataset.train_weights, callbacks=[stop_early])
```

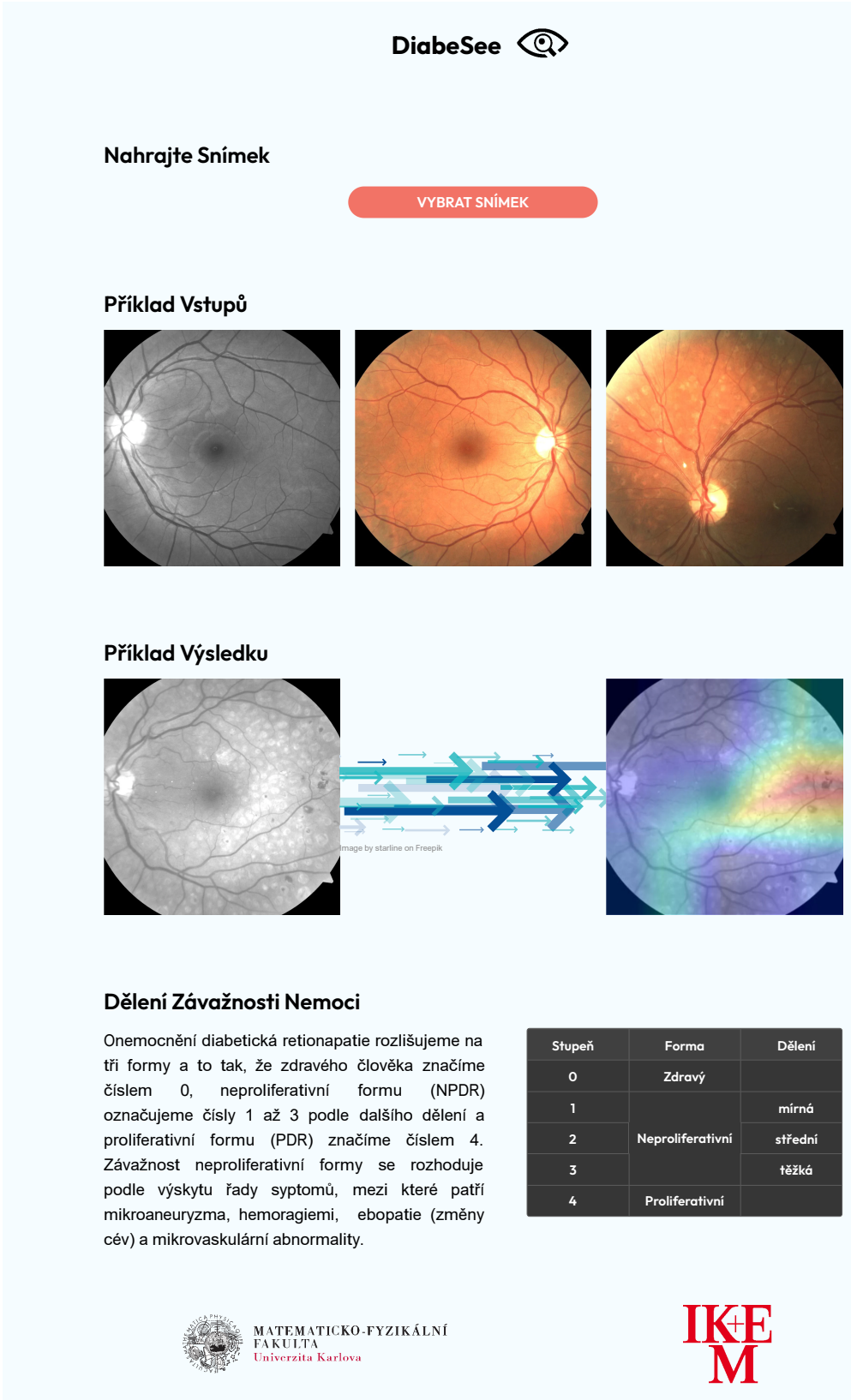
Zdrojový kód A.1: Definice základního modelu spolu s hyperband tunerem.

```
last_conv_layer_name = _get_last_conv_layer_name(model)
last_conv_layer = model.get_layer(last_conv_layer_name)
heatmap_model = tf.keras.models.Model([model.inputs],
↪  [last_conv_layer.output, model.output])
with tf.GradientTape() as tape:
    conv_outputs, predictions = heatmap_model(img)
    tape.watch(conv_outputs)
    loss = mse(tf.round(predictions[:,0]), predictions[:, 0])
    model_predictions.append(predictions[0][0])
grads = tape.gradient(loss, conv_outputs)
pooled_grads = tf.reduce_mean(grads, axis=(0,1,2))
heatmap = conv_outputs[0] @ pooled_grads[..., tf.newaxis]
heatmap = (heatmap - tf.math.reduce_min(heatmap)) /
↪  (tf.math.reduce_max(heatmap) - tf.math.reduce_min(heatmap))
heatmap = tf.image.resize(heatmap, (img.shape[1], img.shape[2]))
heatmap = tf.squeeze(heatmap)
plt.imshow(tf.keras.utils.array_to_img(img[0]))
plt.imshow(tf.keras.utils.array_to_img(tf.expand_dims(heatmap,
↪  axis=2)), cmap='jet', alpha=0.3)
plt.show()
```

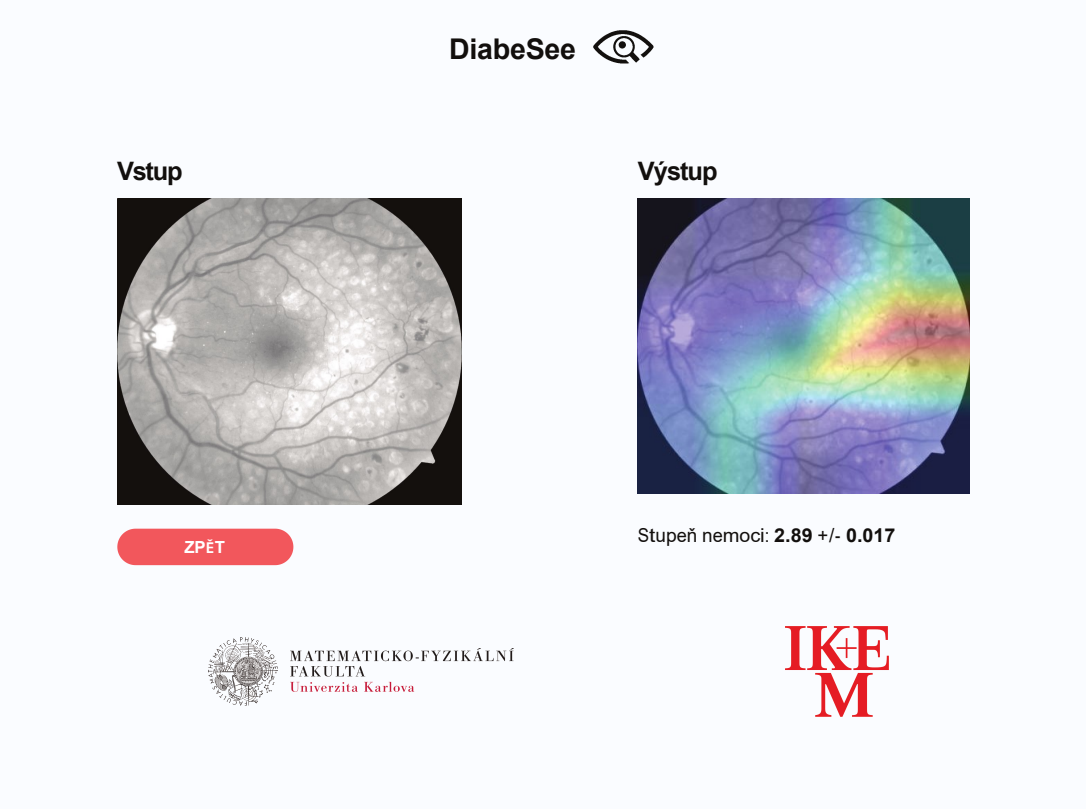
Zdrojový kód A.2: Implementace Grad-CAM metody.

A.2 Snímky webové stránky

Obrázek A.1: DiabeSee web: Titulní strana



Obrázek A.2: DiabeSee web: Strana se vstupem



Obrázek A.3: DiabeSee web: Strana se výsledkem

A.3 Obsah digitální přílohy

- `data.zip`: zašifrované snímky poskytnuté zařízením IKEM. Zašifrované jsou z toho důvodu, že snímky jsou součástí zdravotní dokumentace pacientů a před jejich obdržením je třeba podepsat dokument o mlčenlivosti. Podrobnosti k získání klíče jsou v dokumentu `README.md`
- `data_sample`: příklad vstupních dat z neveřejného datasetu IKEMu
- `experiments_output`: grafy z prvotních experimentů při rozhodování o architektuře sítě a vstupní velikosti snímků
- `logs_tensorboard`: logy z běhu různých konfigurací neuronové sítě při hledání hyperparametrů. Zobrazení logů se provádí pomocí aplikace Tensorboard popsané v dokumentu `README.md`
- `models`: složka s 5 natrénovanými modely, které jsou totožné s těmi, které jsou použité ve webové stránce
- `nn`: zdrojové kódy experimentů a skriptů v jazyce Python3
  - `experiments`: zdrojové kódy prvotních experimentů, ke kterým jsou ve složce `experiments_output` výsledné grafy
  - `experiments_runner.py`: spuštění prvotních experimentů definovaných ve složce `nn/experiments`
  - `baseline_tuner.py`: spuštění hyperband tuneru hledající vhodnou kombinaci hyperparametrů a manuálního režimu trénováním konkrétní kombinace hyperparametrů
  - `iterative_dataset_use.py`: trénování neuronové sítě iterativním přidáváním trénovacích dat z veřejnému k privátním datasetu po dosáhnutí zadané přesnosti
  - `model_evaluate.py`: zobrazení kvality modelu nebo modelů. Umí zobrazit ROC a PRC graf, matici záměn, mnoho metrik (True positive atd.) a výstup Grad-CAM metody
  - `photo_dataset_directory_iterator.py`: třídy starající se o načtení dat datasetů a poskytnutím je při trénování
  - `prepare_clahe.py`: skript slouží k transformaci všech snímků privátního datasetu metodou CLAHE a k zobrazení různých transformací na jednom snímku. Je třeba jej spustit před případným spuštěním `nn/baseline_tuner.py` s přepínačem `--clahe_dataset`
  - `transfer_learning.py`: trénování předtrénovaného modelu MobileNetV2 s nahrazenou poslední vrstvou
  - `final_tunning.py`: trénování již naučeného modelu s malým učícím krokem
- `web`: zdrojové kódy webové stránky, pomocí které je možné spouštět naučené modely, a tak získat predikci stupně nemoci podle vloženého snímku sítnice

  - `models`: stejně jako složka `models` obsahuje natrénované modely použité ve webové stránce
  - `static`: složka obsahuje použité obrázky a styly ve webové stránce
  - `templates`: šablony jednotlivých stránek, do které knihovna Flask doplní data při zobrazení
  - `app.py`: vstupním místem webové stránky používající knihovnu Flask
  - `convolution_network.py`: zdrojový kód sloužící ke spuštění netrénovaných modelů neuronové sítě uživatelem zadaným snímkem sítnice a vytvoření Grad-CAM vizualizaci
  - `docker-compose.yaml`: definuje způsob spuštění webové stránky pomocí docker kontejneru a utility *docker-compose*
  - `Dockerfile`: definuje vytvoření *docker image* webové stránky včetně přednastavení všech závislostí
  - `requirements.txt`: obsahuje seznam závislostí pro prostředí jazyka Python, které je třeba pro spouštění webu
  - `serve.py`: zdrojový kód sloužící ke spuštění webové stránky v docker kontejneru
- `README.md`: návod na zprovoznění prostředí potřebné pro spouštění přiložených skriptů, zobrazení logů z průběhu učení a spuštění webové stránky. Dále obsahuje příklady spouštění skriptů.
- `requirements.txt`: obsahuje seznam závislostí pro prostředí jazyka Python, které je třeba pro spouštění skriptů