Klasifikace obrazu

Ondře Rajnet
Faculty of Informatics and Management
University of Hradec Kralove,
Hradec Kralove, Czech Republic
rajneon1@uhk.cz

***Abstrakt*—Tato práce se zabývá problematikou strojového učení. V dnešní době, je stále více a více roste obliba umělé inteligence a strojového učení. Aplikace, která bude sloužit jako demonstrace strojového učení, je zaměřena především na rozpoznání obrazu, konkrétně na rozpoznání nemocí (vředy, vaskulární onemocnění, . . .) na obrazu a jejich následné vyhodnocení, zda je nebo není nemoc přítomná. Budeme náš model postupně cvičit a trénovat pro lepší a přesnější výsledky. Používat nebudeme jenom jeden model, pro učení, ale vyzkoušíme jich více a porovnáme výsledky, které jednotlivý model přináší.**

***Klíčová slova*—Inception-v3 1; rozpoznání obrazu 2; strojové učení 3; neuronové sítě 4;umělá inteligence 5;**

I. ÚVOD

A. OBSAH

V dnešní moderní době roste velikým tempem popularita strojového učení. Může za to dnešní doba a rozšíření mobilních telefonů mezi masu lidí. Tím, že každý z nás využívá chytrý mobilní telefon, tak možná nevědomky přispívá k umělé inteligenci. Tito uživatelé poskytují obrovské množství dat, které se dají využít pro trénování strojového učení. Nemusíme chodit ani nikam daleko, podívejme se na Iphony a jejich funkci “Deep fusion” [1]. Tuto funkci představil Apple v roce 2019 pro na vylepšení fotografií pořízených pomocí mobilního telefonu. Funkce využívá “neural enginu” [2], který má v sobě zabudovaný přímo procesor a s jeho pomocí vylepšuje fotografie.

Hlavním cílem této práce je vytvoření funkčního modelu, pro rozpoznání obrazu. Model bude vycházet ze zadaného datasetu. Dataset se postupně zpracuje a natrénuje pro aktuální použití. V našem případě budeme rozpoznávat nemoci. Po natrénovaní budeme schopni používat obrazy a rozpoznávat na nich případné nemoci. Bude to usnadnění pro lidi, kteří to nemusí rozpoznávat pomocí odborníků. Samozřejmě model nebude přesný na 100 procent, ale posouží nám jako orientační hodnota.

Strojové učení (ML) je jeden z typů umělé inteligence (AI), který umožnuje aplikacím přinést přesnější a lepší předpovědi výsledků a nemusí k tomu být úplně přímo naprogramovaný. ML využívá historická data pro svůj vstup k předpovědi výstupních hodnot. [3]

Strojové učení mám pod sebou velice silné případy použití, od předvídaní chování zákazníků při nákupech. Mezi výhody můžeme zařadit pomoc porozumět svým zákazníkům na hlubší úrovni. Hromaděním informací o svých zákaznících a jejich korelací v průběhu času. S přibývajícími daty a časem se algoritmy strojového učení naučí asociaci a pomáhat přizpůsobit vývoj produktu nebo v oblasti marketingu mají obrovský potenciál. Už se objevili velké internetové společnosti, které využívají strojové učení jako hlavní sílu ve svých obchodních modelech. Můžeme si říci pár příkladů. Společnosti Uber využívá algoritmy k porovnání řidičů. Google využívá strojové učení k zobrazování cílené reklamy na správného zákazníka. Samozřejmě, není to jen tak, že strojové učení by mělo samé výhody. Jedna z prvních překážek může být cena za daný projekt. Projekty tohoto typu, řídí lidé, kteří jsou datový vědci a odborníci na svých místech, a to odpovídá jejich platům. Také je vyžadována vysoká softwarová infrastruktura. Dále může být v zaujatosti algoritmů. Data, která dostaneme od určité populace nebo v data obsahují nějaké chyby vedou k nepřesním modelům světa. Tyto modely v lepším případě selžou v horších případech jsou diskriminační. Podnik může založit základní obchodní procesy na neobjektivních modelech, může dojít k poškození reputace a tím zmaření podnikání. [3]

Algoritmy strojového učení jsou s námi už více než deset let, až nyní získali novou popularitu, protože se stále více do života dostává uměla inteligence. Modely hlubokého učení (Deep learning) podporují nejmodernější aplikace současné doby. Platformy ML patří v současností mezi nejkonkurenceschopnější oblasti podnikových technologií. Většina velkých výrobců softwaru (Google, Microsoft, IBM) bojují o přihlášení uživatele právě na jejich platformu. Tyto platformy pokrývají obrovské spektrum činností strojového učení, dají se sbírat data, vyhodnocovat, vytvářet modely, klasifikace dat. Strojové učení bude mít v životě lidí větší a větší důležitost. Nejen pro obyčejného zákazníka, ale hlavně pro firmy. Bitvy o data zákazníků můžeme vidět již dneska, tato válka se bude jen dále prohlubovat. [3]

II. PROBLEM DEFINITION/ DEFINICE PROBLÉMU

A. OBSAH

V oblasti pro rozpoznání obrazu je mnoho modelů které se dají implemenovat. Modelů je více a proto je dobré si je porovnat. Porovnání výběru modelů můžeme v článcích [4]. Všechny možné řešení, které již jsou, využívají neuronové sítě, ale každá se bude lišit v konkrétní implementaci nad konkrétním případem a nebude existovat univerzální řešení pro všechny.

Typy strojového učení se rozdělují podle toho, jak se algoritmus učí dávat přesnější výsledky. Máme 4 základní algoritmy: učení pod dohledem, učení bez dozoru, učení s částečným

dohledem a posilovací učení. Níže si rozebere každý z nich. Kontrolované učení, v tomto typu datoví vědci dodávají algoritmům označená tréninková data a definují proměnné, které chtějí, aby algoritmus vyhodnotil pro korelace. Specifikuje se vstup výstup. Učení bez dozoru, patří sem algoritmy, které se cvičí na neoznačených datech. Algoritmus pracuje na principu prohledávání datové sady (datasetu) a vyhledávání smysluplné spojení. Předpovědi nebo doporučení na kterých se algoritmy trénují jsou předem učená. Učení s částečným dohledem, kombinace předchozím dvou řešení. Vědci mohou algoritmu dávat většinu dat která jsou označená jako tréninková, ale model může data sám volně prozkoumávat a rozbíjet vlastní porozumění datové sadě. Posílení učení, využívá učení výztuže, která se většinou používá k naučení algoritmu dokončit vícestupňový proces. Pro tento proces existují jasně definovaná pravidla. Algoritmus je naprogramován tak, aby splnil úkol a dostal pozitivní nebo negativní podmět, když zjistí, jak ho dokončit. Jaké kroky podnikne se většinou algoritmus rozhoduje sám. [3]

Hlavní vize celé této práce je rozpoznání obrazu nad obrazy onemocnění, které budou zanesení do datasetu a následně s tímto datasetem budeme pracovat.

V odborné práci [5] se autoři zabývají obrazy mozku. Zkoumají oblasti CT v moderním počítačovém podání. Hlavním jejich oblast zkoumání je zda může konvoluční neuronové sítě (CNN) poskytnout doplňující informace ohledně včasné diagnostiky Alzheimerovy choroby. Sestavili 3 kategorie CT snímků AD, Léze (Nádor) a normální stárnutí. Kvůli vlastnostem této korelace a její hloubky je zavedená pokročilá architektura integrující 2D a 3D sítě CNN. Spojení těchto 2 sítí je následně vyhodnocena na základě 2D obrazu podél prostorových axiálních směrů a 3 D segmentovaných bloků. Výsledky této spolupráce s touto neobvyklou propracovanou architekturou poskytují velice vysoká procenta úspěšnosti.

V odborné práci [6] se autoři zabývají předpovědí mrtvice pomocí strojového učení. Při klinickém rozhodování se v praxi uplatňuje mnoho prediktivních technik, které lékařům pomáhají s určením diagnozy a léčby pacientů. Konvekční prediktivní modely nebo techniky jsou však stále nedostateční, a proto se rozhodli udělat tento výzkum. Tato práce uvádí techniky pro cévní mozkovou příhodu aplikovanou na dataset srdečních chorob. Příznaky fibrilace síní u pacientů jsou hlavním rizikovým faktore cévní mozkové příhody. Výsledkem výzkumu je přesnější skórovací systém, které se v současné době používá pro varování pacientů, u kterých je pravděpodobné, že u nich může dojít k mrtvici.

V článku [7], je provedená studie, která se zabývá předpovědí oční axiální délky na základě vizuální interpretace obrazů sítnicového pozadí. Oční axiální délka (Al) je důležitou vlastní očí používaných ke stanovení zdraví před operací pacienta. Zásadní při je Al při výrobě umělých čoček, které nahrazují poškozené oční čočky, která má pacient. Přesně měření je však velice náročné a nákladné. Dnešní moderní fotoaparáty jsou na takové úrovní, a tak dokáži zachytit složitou strukturu očí fundus. Kvůli se tomu se rozhodli na nákladě těchto snímků vyzkoušet hlubokého učení. S přispěním vizualizační techniky lze lokalizovat diskriminační oblasti zájmu pro předpovědi. V experimentu bylo zjištěn významný vtah mezi fundusem a Al s dosaženým koeficientem s přesností 90 procent. Výsledná neurovnová síť předpovídá Al konzistentní oblasti. Výsledek této studie dokazuje souvislost mezi Al a biologickou strukturou očí.

Dnes se můžeme setkat se strojovým učením na e-shopu. V podobě zákaznické podpory, která vám v případě nouze poradí. Na druhé straně, ale nemusí být nějaký manager, který vy se vám věnoval, ale chatbot. Funkce těchto robotů je získání informací z webu a následná prezentace zákazníkovi. Jak čas utíká, tak se robot na druhé straně učí pomocí algoritmů strojového učení a má tendence poskytovat lepší odpovědi a informace. [8]

Užasnou ukázkou strojového učení neboli umělé inteligence jsou samořiditelná auta. Waymo, projekt autonomních vozidel od společnosti Google. Cílem společnosti je vytvořit vozidlo, které by bylo schopno řídit autonomně, bez zásahu člověka. Toto samo o sobě vyžaduje opravdu hodně umělé inteligence. Vozidla využívají strojové učení, aby viděli své okolí, předpovídali chování ostatních účastníků silničního provozu. S takovou obrovskou spoustou proměnných na silnici, je pro úspěch rozhodující velice vyspělí systém strojového učení. [9]

Zdravotní péče, jak jsem jich demonstroval v několika předchozích odstavcích je velmi zastoupená ve strojovém učení. Další ukázkou, jak je strojové učení důležité pro zdravotnictví je z Číny. V Číně je každoroční požadavek přezkoumání 1,4 miliardy CT snímků, kvůli hledání včasné pomoci proti rakovině plic. Kvůli nedostatku radiologů, kteří vy prováděli diagnostiku a lidské chybě, která může přijít z dlouhé práce nebo únavě. Společnost Infersion naučil algoritmy práci těchto radiologů a umožnují přesnější a účinnější diagnózu rakoviny. Neurověda se stala inspirací pro DeepMind společnosti Google, ta vytváří nástroj, který dokáže napodobit myšlenkové pochody v našem mozku. DeepMind má za sebou úspěchy v poražení lidí ve hrách. Opravdu zajímavé jsou možnosti aplikace ve zdravotnictví. Mezi přední patří zkrácení času potřebného k plánování léčby a použití správných nástrojů k diagnostice nemoci. [10]

III. NEW SOLUTION / NOVÉ ŘEŠENÍ

Pří vývoji našeho modelu strojového učení vystává několik základních otázek, které je potřeba si dopředu dobře rozmyslet a zvážit. První věc, která je potřeba učit a rozmyslet v jakém jazyce bude kód napsaný. Máme zde hned několik možných voleb [11]. Jako nejpoužívanější a nejoblíbenější je jazyk Python, který používá více než 57 procent těch, kteří se zabývají strojovým učení.

Python je oblíbený hlavně díky podpoře knihovem jako jsou TensorFlow, Pythorch a další. Knihovem existuje obrovské množství a můžeme si vybrat dle toho, na co konkrétně budeme náš model vyvíjet. Další, důvodem jsou jednoduché syntaxe a jejich snadné naučení i pro začínající programátory. Naší volbou je Python.

Pytorch je balíček věděckých výpočtů založený na Pythonu, který využívá výkonu grafické karty pro své výpočty. Je to jedna z hlavních prferovaných výzkumných plaforem pro strojové učení, aby byl schopný poskytovat maximální flexibilitu a rychlost. Poskytuje dvě hlavní funkce na té nevyšší úrovni.

Tenzorové výpočty se silnou podporovou akcelerace GPU a budování hlubokých neurových na páskových autogradových systémech. Existuje mnoho knihem Pythonu, které mají sílu změnit chápání hlubokéhu učení a samotné umělé inteligence a pytorch je jedna z knihovem. Klíčový úspech, proč je tak oblíbená je díky tomu, že je zcela Pythonický a lde bez větších prolbémů vytvářet neuronové sítě. Je to stále mladá knihovna [12]

Jako další jaky který se používá je Java. Java je velice oblíbený jazyk ve světě programování. V případě strojového učení, není tak populární jako Python. Používá do 15 procent programátorů. Java se nepoužívá přímo na vykování strojového učení, ale je lepší pro zabezpečení sítí, kde není Python úplně vhodný. Je považovám za bezpečný jazyk díky použití bytecode a sanboxům.

Jazyk R slouží spíše pro statistické výpočty, analýzy a vizualizace ve strojovém učení. Je to jasná volba pro ty, kteří chtějí zkoumat statistické údaje pomocí grafu. Jazyk R je velice výhodný pro bioinženýrství, bioinformatiku a biomedicínské statistice. Vhodný pro jednorázové projekty jako můžou být výzkumné práce.

Jedna z hlavním je ta, v jakém prostředí budeme naší aplikaci programovat, učit ji a vyhodnocovat výsledky. Mezi ty přední patří služby Kaggle a Google Colaboratory. V obou zmíněch se dají psát programy pro strojové učení.

Kaggle je vlastně dceřiná společnost společnosti Google. Je to online komunika datových vědců a odborníků na strojové učení. Na této platformně je nám umožněno vyhledávat a publikovat datové sady, zkoumat a vytvářet modely ve webovém prostředí. Můžeme zde spolupracovat s ostatními vědci a inženýri na strojové učení. Součástí je i účast v soutěži o vyřešení datové výzvy. Platforma nám umožnuje využívat jejich výpočetní výkon pro náš model a tak nemusíme mít vlastní výkonným stroj, na kterém bude běžet výpočet. Tím, že se vše odehrává ve webovém prohlížeči, tak jsme schopni tento proces spouštět a testovat na jakémkoliv zařízení a je vždy dostupný.[13]

Google Colaboratory (Colab) umožňuje uživatelům psát a spouštět libovolný kód pythonu. S využitím webového prohlížeče a je obzvláště vhodný pro strojové učení, analýzu dat a vzdělávání. Můžeme říct je Colab hostovaná služba notebooků Jupyter, která nevyžaduje použití žádného nastavení, a zároveň poskytuje bezplatný přístup k výpočetním zdrojům včetně GPU.[14]

Pro potřeby mého modelu jsem zvolil vývoj v pythonu na platformně Kaggle. Kaggle nabízel vše co jsem požadoval a byl pro ně intuitivnější. Dále jsem použil knihovnu Pytorch pro implementaci modelů.

Model bude vyhodnocovat rozpoznání obrazu na snímcích. K tomuto účelu nám bude sloužit dataset. Dataset je kolekce, která nám bude udržovat naše snímky, jak ty které se mají určovat tak ty pro kontrolu, zda funguje náš model správně.

Nejhlavnějším bodem aplikace je správně zvolení modelu pro samotné vykonávání funkcionality. Tyto modelu zajišťuje právě knihovna Pytorch, která je mám připravené k použití. Jedná se o sadu 6 modelů.

Výběr modelu začíná už na začátku celého procesu. Modely se od sebe liší v implementaci strojového učení, každý je vhodný na trošku odlišné typy úloh. Model je takovým srdcem výpočtu a podle něj se pojde k přesnému nebo méně přesnému výsledku. Model se nastavuje na začátku a není možné ho změnit v průběhu výpočtu. Pokud se rozhodneme dataset testovat na jiném modelu, musíme natrénovat dataset na aktuální model. Můžeme obecně říci, že čím více budeme mít přítomných dat tak to bude lepší, strojové učení se lépe učí za předpokladu většího množsví dat.

Velikost dávky (batch size) definuje počet vzorků, které se poputují po neuronové síti. Počet epoch (Number of epochs) se skládá z jednoho celého cyklu tréninkových dat. To znamená, kolikrát má celý náš trénovací dataset projít daný model. Obě tyto možnosti společně s modelem nám určují jak bude výsledek uspokojivý.

Než se data dostanou do samotného algoritmu na trénování a zpracování musí projít přípravou (Data preparation) Je vlastně fáze přípravy, kdy jsou data podle potřeby profilována, formátována a strukturována. Vybíráme příslušné vlastnosti a atribury. Tato fáze má přímí dopad na čas a výsledky trénování. Data jsou rozdělena do dvou skupin. Jedna pro trénování modelu a druhá pro jeho hodnocení. Nemůžeme tam poslat přímo sníméky z datasetu protože jsou nejčastěji ve formátu png nebo Jpeg. Algoritmy vyžadují aby data pouze byla čísla a mohli s nimi takto pracovat. Některé algoritmy mají svoje požadavky na data. Může se stát, že je potřeba opravit statistický šum a chyby v datech a složité nelineárlní vztahy mohou být vyloučeny z dat.[15]

V případě, že máme data takto připravená můžeme pokračovat dále a přejít na samotné trénování modelu na datech. Délka trénování záleží na velikosti datasetu, zvoleném modelu a vlastnostech které jsme si volili na začátku. Můžeme říci, že v této fázi probíhá samotné učení. Konzistentní tréninku zlepšuje predikce modelu. Váhy modelu jsou inicializovány náhodně. Takto se algoritmus učí upravovat váhy.

Po skončení trénování budeme model muset testovat oproti sadě na ověření modelu, kterou jsme si vytvořili předtím. Tato sada pomáhá posoudit přednost modelu.[15]

Až vše skončí a proběhne trénování modelu a dat, můžeme model vyzkoušet, jak si povede. Z našeho datasetu vybere libovolný snímek a necháme ho, ať nám určí na jeho výsledek. Výsledek dostaneme v procentech. Správně natrénovaný model by měl mít jednoznačnou odpověď a měli by jsme být schopni velice jednoduše zkontrolovat jeho odpověď. Je vhodně vyzkoušet více modelů a porovnat výsledky, kterých jsme dosáhli. I podobně napsané algoritmy mohou mít odlišné výsledky.

IV. IMPLEMENTACE ŘEŠENÍ

A. OBSAH

Jak už bylo uvedeno výše, pro implementaci aplikace, modelů a samotných algoritmů byla zvolena platforma Kaggle. Důvod ke zvolení této platformy je zajisté více, ale pro mě to bylo nejintuitivnější a nepřehlednější řešení s možností okamžitého výstupu.

Prvním krokem bylo načtení a importování požadovaných knihoven pro práci se strojovým učení, v mém případě jsem zvolil možnosti od Pytorch. Pytorch je mladá knihovna a líbí se

mi jejich prezentace a práce s ní. Dále bylo důležité, abychom měli přístup na grafickou kartu, která bude akcelerovat naše výpočty nad modely.

```
import torch
import torch.nn as nn
import torchvision
```

```
torch.cuda.is_available()
```

Obrázek 1: Ukázka kódu třídy pro zařáteční inicializaci

Imputs, neboli příprava vstupů. Důležitá součást začátku, zde se nastavuje vše, jak má výsledný model fungovat. Jaké má mít parametry a jaký konkrétní algoritmus se použije pro výpočet. V mém modely je aplikováno 6 modelů.

```
data_dir = '/kaggle/input'
num_classes = 16
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")


model_name = 'densenet' # [resnet, alexnet, vgg, squeezenet, densenet, inception]
batch_size = 8
num_epochs = 20
# Flag for feature extracting. When False, we finetune the whole model,
#   when True we only update the reshaped layer params
feature_extract = False
```

Obrázek 2: Ukázka kódu třídy pro Přípavu vstupů

Ukázka algoritmu, který se bude používat pro výpočet strojového učení. Na konkrétním příkladu je Inception v3. Ale v modelu je aplikováno více algoritmů, které mají podobnou strukturu.

```
elif model_name == "inception":
    """ Inception v3
    Be careful, expects (299,299) sized images and has auxiliary output
    """
    model_ft = models.inception_v3(pretrained=use_pretrained)
    set_parameter_requires_grad(model_ft, feature_extract)
    # Handle the auxilary net
    num_ftrs = model_ft.AuxLogits.fc.in_features
    model_ft.AuxLogits.fc = nn.Linear(num_ftrs, num_classes)
    # Handle the primary net
    num_ftrs = model_ft.fc.in_features
    model_ft.fc = nn.Linear(num_ftrs,num_classes)
    input_size = 299
```

Obrázek 3: Ukázka jedno z algoritmů

Ukázka algoritmu z přípravy dat, o které jsem mluvil výše. Dataset se snímky je nejdříve rozdělen do dvou skupin a tyto skupiny jsou následně zpracovány algoritmem. Každá skupina používá jiné metody pro úpravy a zpracování dat. Poté, co proběhne zpracovací část, jsou data načte do paměti a připravena pro následné trénování pomocí neuronových sítí a algoritmu, který jsme si zvolili za začátku.

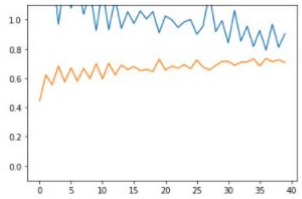
Pokud v pořádku a správně proběhnou předchozí kroky můžeme se pustit do samotného učení a trénování. Jak můžeme vidět níže, ukázku z úspěšného trénování. V tomto případě byl použit model inception a celkovým časem 14 minut 29 vteřin a měl nastavených 20 opakovacích cyklů. Výstup je i graf pro lepší orientaci.

```
import os
from torchvision import datasets, transforms

# Data augmentation and normalization for training
# Just normalization for validation
data_transforms = {
    'train': transforms.Compose([
        transforms.RandomResizedCrop(input_size),
        transforms.RandomGrayscale(0.1),
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
    'validation': transforms.Compose([
        transforms.Resize(input_size),
        transforms.CenterCrop(input_size),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
}

image_datasets = {x: datasets.ImageFolder(os.path.join(data_dir, x), data_transforms[x]) for x
in ['train', 'validation']}
dataloaders = {x: torch.utils.data.DataLoader(image_datasets[x], batch_size=batch_size, shuffle
=True, num_workers=4) for x in ['train', 'validation']}
dataset_sizes = {x: len(image_datasets[x]) for x in ['train', 'validation']}
class_names = image_datasets['train'].classes
```

Obrázek 4: Ukázka kódu třídy pro přípavu dat

```
model_ft = model_ft.to(device)
model_ft = train_model(model_ft, criterion, optimizer_ft, num_epochs=num_epochs, is_inception=
(model_name=="inception"))
```

```
validation Loss: 0.9042 Acc: 0.7071: 100%|██████████| 20/20 [14:28<00:00, 43.44s/it]

Training complete in 14m 29s
Best val Acc: 0.730245
```



Obrázek 5: Ukázka trénování neuroných sítí pomocí algorimu

Po dokončení trénování a výsledcích modelu, je potřeba tyto výsledky skontroval a poupravit. Pro kontrolu a úpravu výsledků se používá druhá část dat, které jsme si připravili v sekci příprava dat. Data se porovnání oproti nim a popřípadě se upravuje hodnota plt.pause(0.001). V tomto případě se zvětšuje o 1 tisícinu, tak aby byli výsledky co nejpřesnější mohou být.

```
def imshow(inp, title=None):
    """Imshow for Tensor."""
    inp = inp.numpy().transpose((1, 2, 0))
    mean = np.array([0.485, 0.456, 0.406])
    std = np.array([0.229, 0.224, 0.225])
    inp = std * inp + mean
    inp = np.clip(inp, 0, 1)
    plt.imshow(inp)
    if title is not None:
        plt.title(title)
    plt.pause(0.001)  # pause a bit so that plots are updated
```

Obrázek 6: Ukázka kódu třídy pro kontrolu výsledku

V. TESTOVÁNÍ VYVINUTÉ APLIKACE - ŘEŠENÍ

Obsah této kapiloly už vypovídá z názvu, bude zda potřeba stavnoti testové scénáře a následně vyhodnotit nově vy-

tvořenou aplikaci.

Testování aplikace na přesnost jednotlivých modelů a času, který na to daný algoritmus potřeboval čili rychlost. Test této funkcionality bude prováděn následně. Do zdrojového kódu budou vždy uloženy stejné výchozí hodnoty pro všechny modely, jediné, co se bude měnit je dan algoritmus. Nastavení hodnot batchsize bude rovno 10 a numepochs bude nastaveno na 20.Dále výsledek bude porovnávám se stejným snímkem a to s vascular2144.png. Pro lepší představu, vyzkoušíme 2 kolový průběh. Po dokončení testu, bude výsledek algoritmu zanesen do tabulky pro porovnání hodnot s ostatními. Tento test bude proveden 1x pro každý model, který aplikace nabízí. Ve výsledku bude test proveden 6x.

První kolo				
ID	Název	Čas	Vyhodnocení	Úspěch
1	Resnet	6:14	91,04 % na vascular	ANO
2	Alexnet	5:38	35,84% na normal-small-bowel	NE
3	Vgg	6:53	89,77 % na vascular	ANO
4	Squeezenet	6:15	89,68 % na vascular	ANO
5	Densenet	10:40	83,83 % na vascular	ANO
6	Inception	11:11	96,61 na vascular	ANO

Obrázek 7: První kolo testování algoritmů

Z výsledků v prvním kole testů můžeme vidět pár základních věcí. 5 z 6 algoritmů obstály a správně zhodnotili výsledek. U všech výše zmíněných algoritmů se jedná o rozdíly v čase i přesnosti výsledku. Většina algoritmů má rozmezí mezi 80-90 procent úspěšnosti rozpoznání snímku. Nejlepší algoritmus na procentuální skóre se jeví jako Inception, který se svými 96,61 procenty vyhrál náš srovnávací test. Na druhou stranu to zvládl za 11 minut a 11 vteřin. To je nejdelší čas z výše testovaných algoritmů.

Druhé kolo				
ID	Název	Čas	Vyhodnocení	Úspěch
1	Resnet	6:26	90,91 % na vascular	ANO
2	Alexnet	6:01	81,30 % na vascular	ANO
3	Vgg	7:03	92,78 % na vascular	ANO
4	Squeezenet	6:25	92,06 na vascular	ANO
5	Densenet	10:50	72,92 % na vascular	ANO
6	Inception	11:15	99,63% na vascular	ANO

Obrázek 8: Druhé kolo testování algoritmů

V druhém kole můžeme vidět zlepší nějaký algoritmů. V tomto kole můžeme říci, že všechny algoritmy dali správný výsledek. Minulé kolo algoritmus Alexnet nesprávně vyhodnotil snímek a určil jsou nemoc na snímku. Druhé kolo přineslo opět stejného vítěze a to Inception, který zvítězil s 99,63 procent a časem 11 minut a 15 vteřin. Algoritmus se o 2 setiny procenta zlepšil, ale časově i o 4 vteřiny pohoršil.

Porovnání obou našich kol vychází konstantní výsledky většiny algoritmů. Ve druhém kole se nejvíce zlepšil algoritmus Alexnet, který se opravil z prvního kola. Toho je velice důležitý moment, vidíme že je důležité vyzkoušet průběhy algoritmů více než jednou. Absolutním vítězen se stal algortmus Inception, který měl nevyšší procento úspěšnosti odhadu snímku, proti tomu trvalo mu to nejdelší dobu ze všech zmíněných algoritmů. Rozdíly mezi jednotlivými koly nejsou nijak výrazná, kromě jednoho případu.

VI. CONCLUSIONS / ZÁVĚRY

Cílem tohoto projektu bylo vytvořit aplikaci na strojové učení, která bude brát jako svůj vstup dataset snímků nemocí. Snímky si sama rozpočítá a pomocí nadefinovaných algoritmů na stojové učení. Poté se začne učit pomocí neuronových sítí. Až se toho vše dokončí porovná se s kontrolní složkou, kterou si aplikace vytvořila v průběhu. Poté nám může prezentovat její výsledek vůči námi vybranému snímku.

Aplikace byla naprogramována pomocí knihovny Pytorch a pomocí jejich algoritmů. Z provedených testů v tabulkách víše, je patrné že aplikace funguje. Liší se jednotlivé modely a jejich potřeba času na načtení a následné vyhodnocení proti porovnávanému snímku. Samozřejmě se liší i procentuální úspěšnost jednotlivých modelů. Časy jednotlivých modelů se liší o více jednotky minut v našem případě. Oproti tomu, procentuální výsledky nejsou tolik rozdílné a když přihlédneme k času, který potřebujeme na výsledek může se zdát, že algoritmus, který má nejvyšší přenost, nemusí být pro nás ta nejlepší volba.

Tato aplikace a popřípadě celý model, je demonstrací, že strojové učení může rozhodne fungovat a může být hodně užitečné i v případech lidského zdraví. V případě, že se stane chyba při vyhodnocení špatné marketingové strategie, nemusí to znamenat smrt lidí, nebo v lepším případě zdravotní komplikace jako u biomedicíny. Ke strojovému učení je potřeba přistupovat s pokorou. Je to výborný nástroj do budoucna a myslím, že obliba strojového učení, nebo spíše celého odvětví umělé inteligence jen poroste a více a více aplikací pro běžnou potřebu obyčejného člověka bude napojeno na ty to algoritmy a mnozí lidé o tom nemusí ani vědět. Největší nebezpečí se skrývá v nevědomosti uživatelů a tím, jak je nakládáno s jejich daty. Tento problém se začíná projevovat už v dnešní době.

REFERENCE

[1] Pocket-lint. „What Is Apple Deep Fusion and How Does It Work?"Pocket-lint, 29. říjen 2019. https://www.pocket-lint.com/phones/news/apple/149594-what-is-apple-deep-fusion.

[2] „Apple's 'Neural Engine' Infuses the IPhone With AI Smarts". Wired. Viděno 13. leden 2021. https://www.wired.com/story/apples-neural-engine-infuses-the-iphone-with-ai-smarts/.

[3] SearchEnterpriseAI. „What Is Machine Learning (ML)?"Viděno 14. leden 2021. https://searchenterpriseai.techtarget.com/definition/machine-learning-ML.

[4] Kothari, Mahak. „Comparison of Different Deep Learning Models for Image Classification". Medium, 7. duben 2020. https://medium.com/@mahakkothari190.mk/comparison-of-different-deep-learning-models-for-image-classification-1c49f1159d7a.

[5] Gao, Xiaohong W., Rui Hui, a Zengmin Tian. „Classification of CT Brain Images Based on Deep Learning Networks". Computer Methods and Programs in Biomedicine 138 (leden 2017): 49–56. https://doi.org/10.1016/j.cmpb.2016.10.007.

[6] Chantamit-o-pas, Pattanapong, a Madhu Goyal. „Prediction of Stroke Using Deep Learning Model". In Neural Information Processing, Iconip 2017, Pt V, editoval D. Liu, S. Xie, Y. Li, D. Zhao, a E. S. M. ElAlfy, 10638:774–81. Cham: Springer International Publishing Ag, 2017. https://doi.org/10.1007/978-3-319-70139-478.

[7] Jeong, Yeonwoo, Boram Lee, Jae-Ho Han, a Jaeryung Oh. „Ocular Axial Length Prediction Based on Visual Interpretation of Retinal Fundus Images via Deep Neural Network". Ieee Journal of Selected Topics in Quantum Electronics 27, č. 4 (srpen 2021): 7200407. https://doi.org/10.1109/JSTQE.2020.3038845.

[8] Oberoi, Archna. „9 Machine Learning Examples from Day-to-Day Life". Viděno 15. leden 2021. https://insights.daffodilsw.com/blog/9-machine-learning-examples-from-day-to-day-life.

[9] „15 Machine Learning Examples a Applications To Know — Built In". Viděno 15.leden 2021. https://builtin.com/artificial-intelligence/machine-learning-examples-applications.

[10] „27 Incredible Examples Of AI And Machine Learning In Practice". Viděno 15. leden 2021. https://www.forbes.com/sites/bernardmarr/2018/04/30/27-incredible-examples-of-ai-and-machine-learning-in-practice/.

[11] „Best Languages For Machine Learning in 2020! — by Himani Bansal — Becoming Human: Artificial Intelligence Magazine". Viděno 14. leden 2021. https://becominghuman.ai/best-languages-for-machine-learning-in-2020-6034732dd242.

[12] Shetty, Sunith. „What Is PyTorch and How Does It Work?"Packt Hub, 18. září 2018. https://hub.packtpub.com/what-is-pytorch-and-how-does-it-work/.

[13] „What Is Kaggle, Why I Participate, What Is the Impact? — Data Science and Machine Learning". Viděno 14. leden 2021. a.

[14] „Colaboratory – Google". Viděno 14. leden 2021. https://research.google.com/colaboratory/faq.html.

[15] Oden Technologies. „What Is Model Training". Viděno 14. leden 2021. https://oden.io/glossary/model-training/.