

Obsah

1	Úvod do problematiky	2
2	ResNet, řešení pro hluboké sítě	3
3	I3D	5
3.1	I3D ResNet50	5
3.2	Trénování sítě	7
4	Dataset	9
4.1	HAA500	9
4.1.1	Zpracování RGB datasetu	10
4.1.2	Zpracování segmentačního datasetu	11
4.1.3	Zpracování optického toku	12
4.2	Kinetics-400	13
5	Použitá metrika	15
6	Augmentace	16
7	Experimenty s I3D ResNet50	17
7.1	První trénovací experiment	18
7.2	Trénování na celém datasetu	19
7.3	Experimenty s segmentačním modelem I3D Resnet	21
7.4	Experimenty s modelem optického toku I3D Resnet	22
7.5	Klasifikátory	26
8	Závěr	27
9	Seznam literatury	28
10	Seznam obrázků	29
11	Seznam tabulek	30

1 Úvod do problematiky

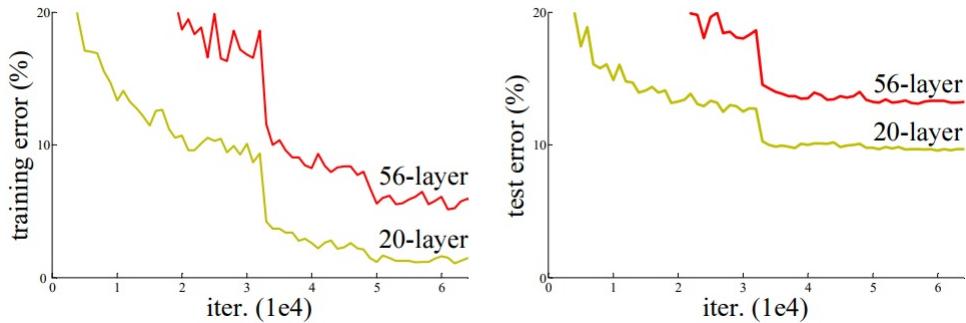
Lidé snadno rozpoznávají činnosti, které jsou nahrané na videu. Tato schopnost je pro člověka přirozená. V průběhu sledování videa pozorujeme objekty a jejich pohyb a díky tomu dokážeme díky zkušenostem vyhodnotit co se na videozáznamu děje. Počítačová automatizace tohoto procesu je náročná, ale je také zajímavá. Uplatnění může mít například pro automatické labelování krátkých videí na sociálních sítí, vyhledávání videa na základě obsahu, shrnutí děje videa nebo třeba monitorování chování starších osob. U sledování osob můžeme rozpoznávat denní činnosti jako je chůze, běh nebo pád. Při automatickém labelování videí můžeme rozpoznávat různé sporty nebo libovolné volnočasové aktivity.

Při rozpoznávání činností z videa dosahují velmi dobrých výsledků metody založené na principu konvolučních neuronových sítí. Budeme se zabývat metodou *I3D* [8], která takzvaně nafoukne standartní architektury *CNN* (*Convolutional neural network*) pro klasifikování obrázků jako jsou VGG-16 [10], Inception [11] nebo architekturu ResNet [6], která bude pro tuto práci klícová.

2 ResNet, řešení pro hluboké sítě

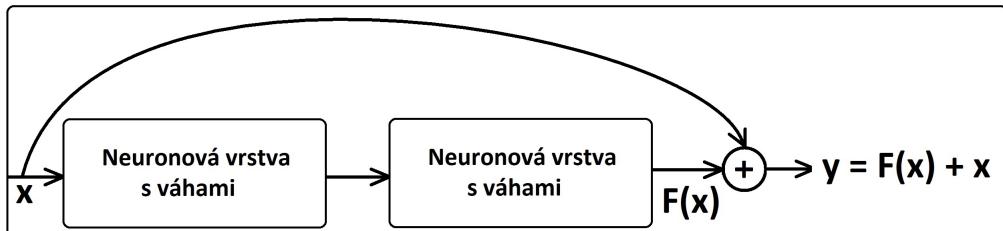
ResNet (Residual Network) [6] je konvoluční neuronová síť. Základ konvolučních sítí představují konvoluční filtry, které jsou ve skrytých vrstvách aplikovány na vstupní data. Vstupní data v případě počítačového vidění je nejčastější RGB obrázek. Podle úlohy mají síťe jiné výstupy. Častá úloha na kterou je ResNet používán je klasifikace obrázků.

ResNet architektura řeší problém velmi hlubokých neuronových sítí. Zjistilo se, že pokud navýšíme hloubku neuronové sítě, můžeme tím dosáhnout lepších výsledků a menších hodnot loss funkce při trénování. Optimalizace ale takové hluboké neuronové sítě je obtížná. Ukázalo se, že hluboké sítě například s počtem 56 vrstev, dosahovaly při trénování horších výsledků než ta samá architektura s pouze 20 vrstvami. To je ukázáno na Obrázku 1, kde je průběh trénování architektury CIFRA-10. Jak je vidět, neuronová síť s 20 vrstvami dosáhla lepších trénovacích i testovacích výsledků než síť s 56 vrstvami.



Obrázek 1: CIFRA-10 trénovací a testovací error
původní obrázek: <https://arxiv.org/pdf/1512.03385.pdf> (strana 1)

Důvodem může být špatná optimalizace sítě, ztrácející se gradient nebo špatné trénování po přidáním nových vrstev do natrénované sítě. ResNet přišel s řešením a to přidáním residiuálních bloků (viz. Obrázek 2).

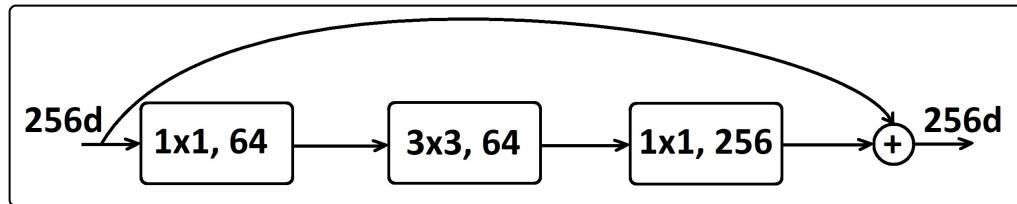


Obrázek 2: Residiuální blok

Residiuální blok obsahuje takzvaný *skip connection*, ten přičte vstupní hodnoty neuronových vrstev sítě k výstupním hodnotám. Tyto *skip connections* učí neuronové vrstvy tak, aby výstup y z nich byl roven rozdílu vstupu x . Z obrázku 2 je výstup značen $y = F(x) + x$, my ale požadujeme, aby výstup byl $y = F(x)$, proto je síť nucená učit se residuum. Sítím které využívají tuto techniku se říká ResNet.

Pro hlubší varianty ResNetu byly navrženy takzvané *bottleneck* bloky. Ty mají za účel navýšit rychlosť výpočtu s využitím snížením dimenze vstupu. Pokud máme například 256

dimenzionální vstup, můžeme ho promítnout do 64d a na konci bloku zpět rozšířit na 256 dimenzí (viz Obrázek 3).



Obrázek 3: Bottleneck blok

Díky ResNet architektuře jsme schopni vytvářet opravdu hluboké neuronové sítě. Velmi populární jsou ResNety s hloubkou 50,101 a 152. Tato práce se zaměří hlavně ResNet s hloubkou 50, zkráceně už jen ResNet50.

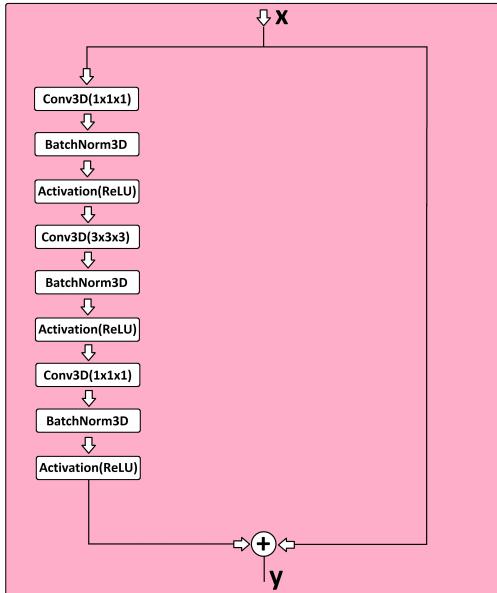
3 I3D

I3D (*Inflated 3D ConvNets*) [8] je způsob, jak z videí sbírat časo-prostorové informace. V úvodu už byla zmínka o takzvaném nafukování standartních konvolučních neuronových sítí pro klasifikaci 2D obrazu. Princip nafouknutí je takový, že vezmeme například funkční a léty prověřenou architekturu ResNet50 [6] a převedeme sdružovací filtry a konvoluční jádra z doby 2D ($N \times N$) na 3D ($N \times N \times N$). To znamená, že pokud máme ve vrstvě konvoluční jádro o rozměrech (3x3), převedeme ho na tvar (3x3x3). Vstupem do nafouknuté sítě je pak posloupnost 2D obrázků videa. Můžeme tedy říci, že přidaný třetí rozměr reprezentuje tok času videa. 3D konvoluční jádra se pohybují napříč časem a prostorem získávají z videa důležité příznaky.

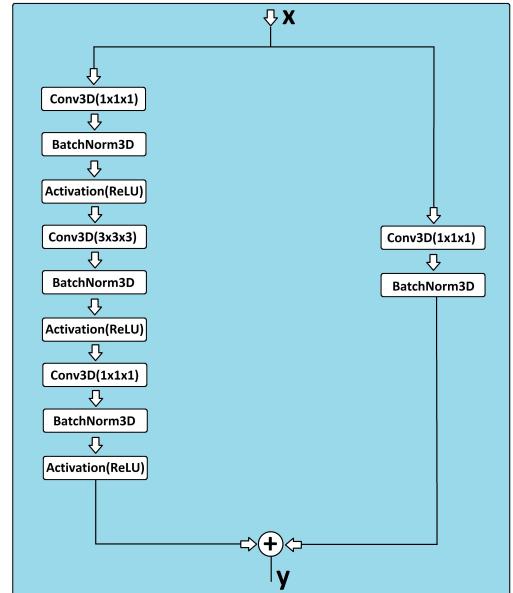
Tyto I3D modely dosahují kvalitních výsledků, pokud k nim implementujeme předtrénované váhy z ImageNet[4]. To lze docílit pomocí takzvaného *nudného* videa. Je to video složené pouze z jednoho snímku, které se opakuje a je časově neměnné. Trénování 3D sítě podle *nudného* videa se sítí učí klasifikovat obrázky. Díky linearitě se ukázalo, že váhy 2D filtrů jsou jen N násobky vah 3D filtrů. Na základě této znalosti můžeme vzít stejné váhy 2D filtrů a N krát je nakopírovat do 3D filtru a vydělit počtem N .

3.1 I3D ResNet50

Architektura [7] ResNet50 byla nafouknutá a nyní se z ní stala síť, která operuje s časově prostorovými závislosti ze snímku. I3D ResNet50 obsahuje celkem 48 konvolučních vrstev, jeden *max-pool* a jeden *average-pool*. Síť I3D ResNet je složená ze x bloků. První blok nazveme například blok A (Obrázek 4).



Obrázek 4: Blok A



Obrázek 5: Blok B

V levé větví bloku A (Obrázek 4) jsou na vstup x použité konvoluční jádro o rozměru (1x1x1) s hodnotou *stride* = (1x1x1), hodnota *stride* představuje krok po kterém se konvoluční jádro posouvá po vstupních datech. Tato konvoluční vrstva má za úkol snížit dimenzi vstupu. Jedná se tedy o *bottleneck* blok. Další vrstva v bloku se nazývá *BatchNorm3D* a ta provádí normalizaci dat, které napomáhá ke stabilizaci sítě během trénovacího procesu. Další

vrstva v pořadí je aktivační funkce, ta je zde použitá *ReLU* (*Rectified Linear Unit*) funkce. Má za úkol rozhodnout o výstupu neuronů způsobem (vzoreček(1)):

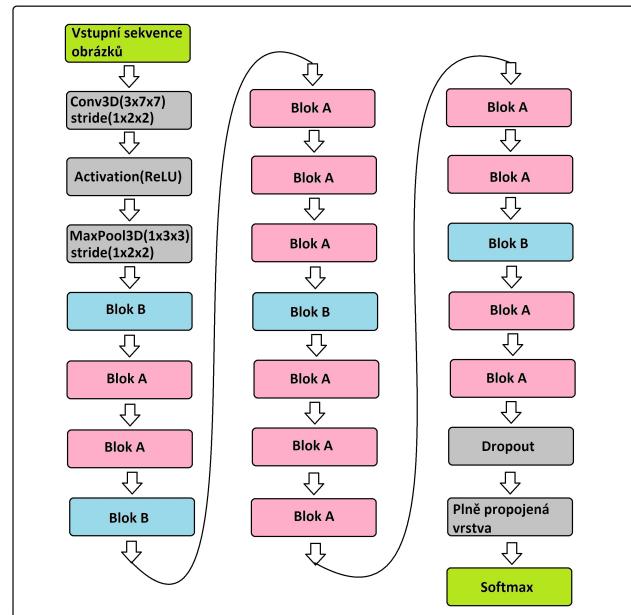
$$f(x) = \begin{cases} 0 & \text{jestli } x < 0 \\ x & \text{jestli } x \geq 0 \end{cases} \quad (1)$$

Aktivační funkce ReLU vrací nulovou hodnotu, při jakémkoliv záporném vstupu.

Tyto tři vrstvy se v levé větvi Bloku A opakují ještě 2x, s tím že podruhé máme konvoluční jádro o rozdílných (3x3x3) a na závěr konvoluční jádro (1x1x1), které má v *bottleneck* bloku za úkol zvýšit zpět svoji dimenzi. V pravé větvi Bloku A není žádná konvoluční vrstva, její účel je pouze při součtu poskytnout vstupní data a na konci bloku je sečít s výstupním signálem zpracované z levé větve.

I3D ResNet50 je tvořen ještě dalším blokem, který si pojmenujme například Blok B, ten je znázorněný na Obrázku 5. Je skoro totožný jako Blok A s rozdílem, že v pravé části větve má konvoluční vrstvu. Díky tomu tento blok na výstupu navýšuje svoji dimenzi v porovnání s výstupními dimenzenami vstupních dat, zatímco Blok A má jak na výstupu tak i na vstupu stejnou dimenzi.

Celá architektura sítě I3D ResNet50 je znázorněná na Obrázku 6. Na začátku je na vstupní posloupnost obrázků aplikována vrstva s konvolučním jádrem o rozdílu (3x7x7) s krokovou hodnotou *stride* = (1x2x2). Následuje *ReLU* aktivační funkce, maximální sdružení a sekvence několika A a B bloků. Na závěr je na data aplikován dropout a plně propojená lineární vrstva, která převede data na vektor o velikosti rozpoznávaných činností. Tento vektor určuje pravděpodobnosti všech tříd. Softmax z vektoru vybere třídu s největší pravděpodobností a tu klasifikuje jako rozpoznanou činnost.



Obrázek 6: Blokově znázorněná architektura I3D ResNet 50

3.2 Trénování sítě

Trénování neuronových [1] sítí je proces, při kterém jsou sítí předkládána anotovaná data, síť je zpracuje a na základě výsledku upravuje váhy neuronů. Můžeme začínat s náhodně inicializovanými hodnotami, nebo vycházet z nějakého předtrénovaného modelu na jiném datasetu. Pokud využijeme předtrénované váhy jiného modelu, očekává se rychlejší doba trénování. Velmi důležitá při trénování je takzvaná *Loss* funkce. Ta nám během trénování udává hodnotu, jak kvalitně je naše neuronová síť nastavená na rozpoznávání činností z videí. Velmi známá *Loss* funkce je *Cross-Entropy*, se kterou budeme provádět trénování I3D ResNet50. *Cross-Entropy* můžeme zapsat matematicky jako:

$$L(p, q) = - \sum_x^N p(x) \cdot \log(q(x)), \quad (2)$$

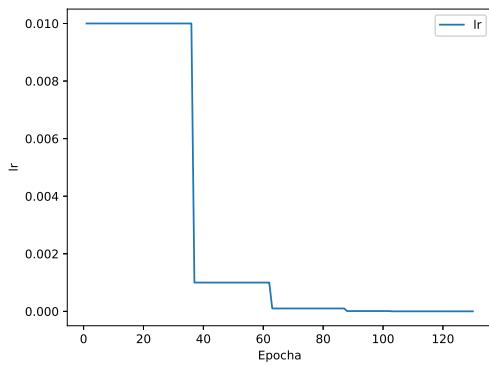
kde x je index třídy, N celkový počet tříd, $p()$ je binární indikátor správné klasifikace a funkce $q()$ reprezentuje výstup softmaxu. Čím přesněji bude síť správně klasifikovat trénovací videa, tím se bude hodnota *Loss* funkce snižovat. Cílem trénování je tedy co nejvíce snížit hodnotu této funkce.

Celý trénovací proces se dá pojmenovat jako úloha pro minimalizování *Loss* funkce. Je zde několik metod, které se snaží tento úlohu řešit, říká se jim optimizéry. Jedním z velmi oblíbených algoritmů je takzvaný SGD (*Stochastic Gradient Descent*), matematicky jej lze zapsat jako (Vzoreček 3) :

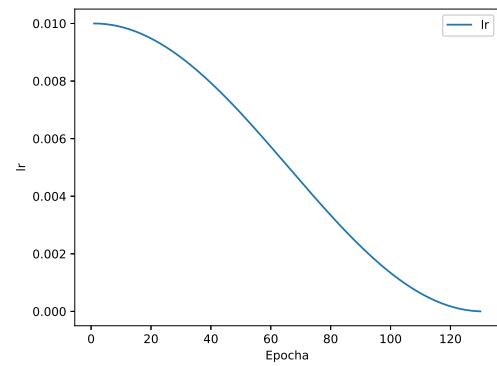
$$w_j = w_j - lr \frac{\partial L}{\partial w_j}, \quad (3)$$

kde w_j jsou váhy neuronové sítě, lr je hodnota učení a zlomek $\partial L / \partial w_j$ je takzvaný gradientu. Gradient v matematice reprezentuje směr největšího růstu, v tomto případě má stejný význam, udává směr největšího růstu *Loss* funkce. My ale požadujeme směr nejmenšího růstu, abychom minimalizovali *Loss* funkci, stačí tedy vzít zápornou hodnotu gradientu. Váhy sítě se pak budou v tomto směru upravovat. Záporný gradient je ještě vážený hyper parametrem hodnoty učení lr , ten nabývá obvykle malých hodnot. Jediné co ještě potřebujeme k trénování je numericky vyčíslit gradient $\partial L / \partial w_j$. Pro to lze použít metodu zvanou *Backpropagation*, kdy zpětně procházíme síť a získáváme gradienty vah mezi jednotlivými vrstvami.

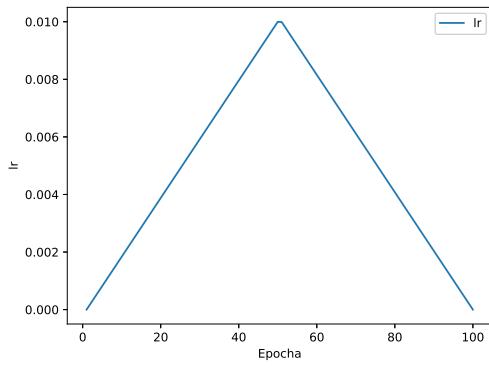
Abychom při trénování dokončovali ideálně ke globálnímu minimu *Loss* funkce, je důležité také zvolit vhodnou strategii úpravy konstanty učení lr . Těch je hned několik. Můžeme zvolit například běžné krokové snížení N epochách (Obrázek 7), kosínovou strategie (Obrázek 8), cyklickou (Obrázek 9) nebo třeba restartovací strategii (Obrázek 10).



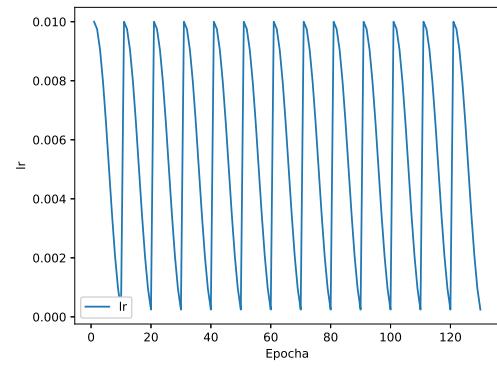
Obrázek 7: Kroková strategie lr



Obrázek 8: Kosínová strategie lr



Obrázek 9: Cyklická strategie lr (jeden cyklus)



Obrázek 10: Strategie restartováním lr

4 Dataset

Dataset je důležitá část trénování modelů neuronový sítí. Proto zvolení kvalitního datasetu, kde si budeme jistý správně anotovanými daty, je naprostý základ. Set data pro rozpoznávání činností musí obsahovat videa a informaci o jejich správné klasifikaci přiložené v externím souboru. Těmto informacím o správné klasifikace se říká *anotace*. Celý dataset by pak měl být rozdělen třech částí a to konkrétně na:

1. trénovací data,
2. validační data,
3. testovací data.

Trénovací data slouží k trénování modelu a obvykle tato množina bývá nejobjemnější. Pomocí validačních dat se během trénovacího procesu validuje natrénovaný model a my tak máme hrubou představu, jak si model povede na neviděných datech. Poslední jsou testovací data. Ty slouží k otestování a následné vyhodnocení natrénovaného modelu.

4.1 HAA500

Human-Centric Atomic Action (HAA500) [2] je ručně anotovaná sada dat videí s lidskou činností. Videá v *HAA500* byla pečlivě vybírána tak, aby zachycoval nepřerušovaný pohyb lidských postav při různých činnostech. Jednotlivé třídy datasetu jsou hodně dopodrobna popsané, tak aby byly jedinečné a rozlišitelné od ostatních. Například činnost hraní fotbalu je rozčleněna na fotbalová střela, fotbalová přihrávka, fotbalová, fotbalové vhazování, dribbling s míčem při fotbale, fotbalová hlavička a fotbalový zákrok brankáře (viz. Obrázek 11, 12, 13). Videá byla vybírána také tak, aby osoba provozující klasifikovanou činnost byla dominantní osobou videa. Hlavní část videa je zaměřena na člověka a u videí je průměrně 69.7% detektovatelných kloubů hlavní osoby. *HAA500* poskytne pro tuto práci data, které využijeme k trénování a testování modelů I3D ResNet50.



Obrázek 11: Třída fotbalový dribbling



Obrázek 12: Třída fotbalová hlavička



Obrázek 13: Třída fotbalový zákrok brankáře

Celkový sada dat *HAA500* obsahuje 500 tříd. V každé třídě je celkem 20 videí HD videí (720x1280). Dohromady dataset poskytuje 10000 videí. Videá jsou tedy perfektně distribuována mezi jednotlivými třídami. Data do trénovací, validační a testovací množiny byla rozdělena v poměru 16:1:3 (viz Tabulka 1) tak, že prvních 16 videí se přiřadilo k trénovací množině, 17. video k validační a zbytek (18,19,20) k testovací množině dat.

	Počet videí
Trénovací data	8000
Validační data	500
Testovací data	1500
Celkový počet dat	10000

Tabulka 1: Počet dat

4.1.1 Zpracování RGB datasetu

4.1.2 Zpracování segmentačního datasetu



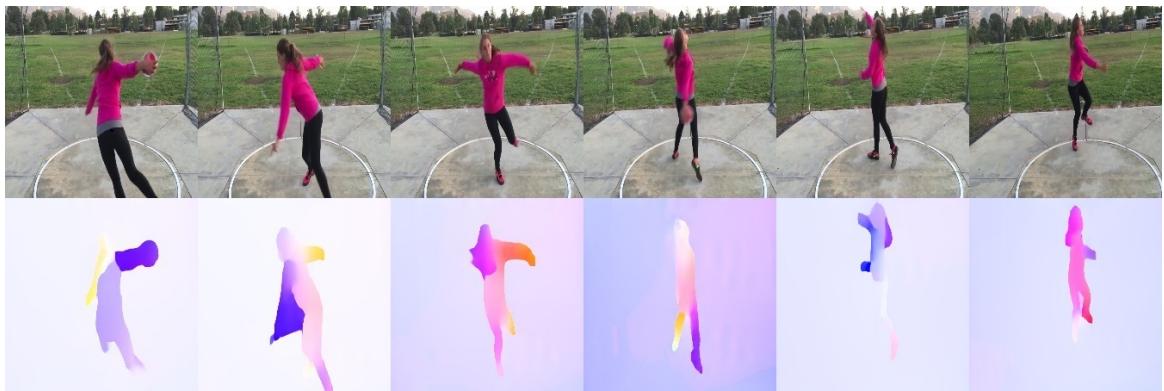
Obrázek 14: Ukázka segmentace - hod diskem



Obrázek 15: Ukázka segmentace - pád na kole

4.1.3 Zpracování optického toku

Ukázka Optického toku:



Obrázek 16: Ukázka segmentace - hod diskem



Obrázek 17: Ukázka segmentace - baseball odpal míčku

4.2 Kinetics-400

Pro základní otestování metod *action recognition* byl zvolen *Kinetics 400* [8] dataset. *Kinetics 400* byl vybrán, protože je často využíván pro testování metod rozpoznávání činností, kterými se zabývá i tato diplomová práce a bude pak snadné porovnat naše rozpoznávací systémy s jinými.

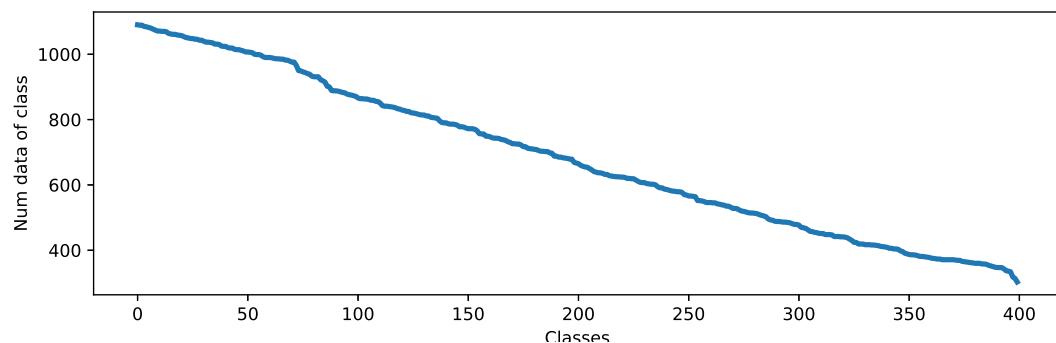
Kinetics 400 je kolekce videí s lidskou činností. Tyto videa jsou reprezentovány pomocí URL odkazy na videa internetového serveru *YouTube.com*. U každého URL odkazu je zaznamenaný počáteční a koncový čas označené činnosti, přičemž každé video trvá zhruba 10 vteřin. Činnosti jsou zaměřeny na člověka a pokrývají širokou škálu tříd včetně interakcí mezi člověkem a objektem, jako je hraní na hudební nástroje a také interakcí mezi lidmi, jako je podání ruky. V tomto datasetu je celkem 400 tříd. Každá třída obsahuje minimálně 400 videoklipů. Všechny tyto informace o URL odkazu, času ve kterém se vyskytuje činnost a korektně pojmenovaná činnost, jsou uloženy ve formátu *.json* nebo *.csv*.

Na základě dostupného datasetu *Kinetics 400* proběhla základní analýza dat. Celkový počet videí je zaznamenán v Tabulce 2. Stažený dataset byl už rozdělen do třech množin dat, tronovací, validační a testovací.

	Počet videí
Trénovací data	219 782
Validační data	18 035
Testovací data	35 357
Celkový počet dat	273 174

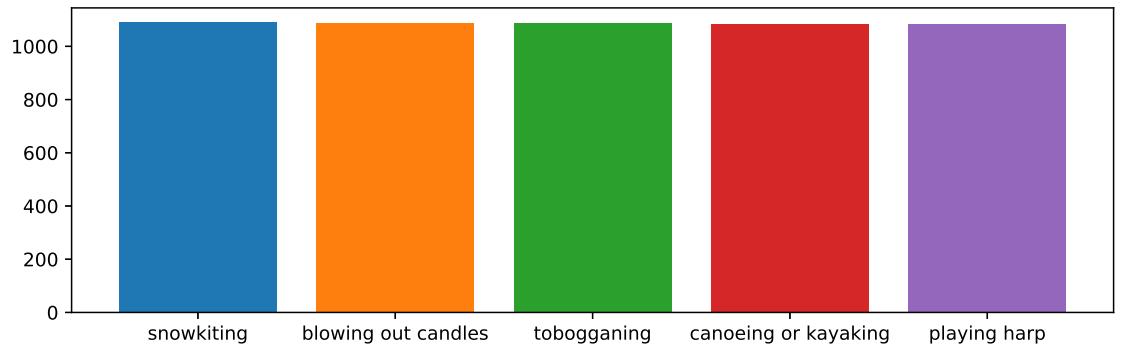
Tabulka 2: Počet dat

Při analýze datasetu je důležité zjistit jak jsou data v jednotlivých třídách rozprostřeny. Distribuce dat mezi třídami je zobrazena na Obrázku 18.



Obrázek 18: Distribuce tříd

Z Obrázku 2 je patrné, že zde není žádná třída, která by výrazně v počtu dat převyšovala ostatní třídy. Pro každou třídu je také dostatečné množství videí. Na Obrázku 19 je zobrazený graf s pěti nejčetnějšími třídami.



Obrázek 19: Ukázka nejčetnějších tříd

Pro stažení *Kinetics 400* videí byl vytvořený script s využitím knihoven *youtube-dl* a *ffmpeg*. Tento způsob extrakce videí datasetu je ale příliš zdlouhavý. Při praktickém otestování rychlosti stahování *360p* videí bylo zjištěno, že stažení jednoho průměrného videa zabere $t_{mean} = 15.57s$. Z tohoto údaje vyplývá, že stažení celého datasetu o 273174 videí potrvá při průměrném stahování 49.2 dne. Podobných výsledků dosáhl také Mark Gituma(2021) ve svém článku ***Downloading The Kinetics Dataset For Human Action Recognition in Deep Learning*** [5]. V tomto článku popisuje stahování datasetu *Kinetics 700* je zde uvedeno, že bylo dosaženo průměrného stažení jednoho videa na hodnotu $t_{mean} = 23.52s$ a stažení celého *Kinetics 700* by trvala 176.1 dní.

Alternativně byl dataset získán s využitím *Academics Torrents* [3][9]. Tento projekt poskytuje ukládání, ale i stahování velkých dat pro výzkum a vědecké účely. *Academics Torrents* je platforma využívající *peer-to-peer* nástroj *BitTorrent*, díky tomu je umožněné skupině členům takzvaně "seedovat" a díky tomu sdílet data ostatním účastníkům.



Obrázek 20: Ukázka - snowkiting



Obrázek 21: Ukázka - tobogganing



Obrázek 22: Ukázka - blowing out candles



Obrázek 23: Ukázka - canoeing or kayaking

5 Použitá metrika

Při porovnávání několika modelů je třeba zvolit vhodnou metriku, díky které můžeme určit kvalitu rozpoznávání činností. Velmi častou volenou metrikou pro rozpoznávání akcí z videí je takzvaná *Top1* přesnost, k ní jsou často přidávány *Top3* a *Top5* přesnosti. Všechny tyto metriky mají společnou jednotku [%]. *Top1* přesnost udává kolik videí neuronová síť dokázala správně rozpoznat. Matematicky lze zapsat jako:

$$Top1 = \frac{TP}{FP + TP} [\%], \quad (4)$$

kde *TP* jsou správně rozpoznaná videa a součet ve jmenovateli *FP* + *TP* je počet všech testovaných videí. Za *TP* se berou třídy, který mají největší pravděpodobnost z výstupní lineární vrstvy sítě. Z výstupní lineární vrstvy můžeme ale zjistit i činnosti, které jsou na 2., 3., 4. a 5. místě. Pak můžeme určit i *Top3* a *Top5* přesnost. Pro úplné vyjasnění si výpočet jednotlivých metrik ukážeme na příkladě.

Č.	Správná třída	1. místo	2. místo	3. místo	TP (<i>Top1</i>)	TP (<i>Top3</i>)
1)	lukostřelba	lukostřelba	střelba z pistole	hra na kytařu	+1	+1
2)	hod diskem	hod míčem	hod oštěpem	pojídání jablka	+0	+0
3)	lukostřelba	střelba ze zbraně	lukostřelba	hod oštěpem	+0	+1
4)	hod diskem	hod diskem	hod míčem	frisbee	+1	+1
5)	pojídání jablka	pojídání zmrzliny	pojídání pizzy	pojídání jablka	+0	+1
				Součet	2	4

Tabulka 3: Příklad testování úspěšnosti klasifikace videí

Na příkladu z Tabulky 3 bylo testováno celkem 5 videí. Síť dokázala rozpoznat 2 videa z pěti na prvním místě, takže má hodnotu *Top1* = 40% a 4 videa se nacházely na prvních pěti pozicích tedy *Top3* = 80%.

Podle článku **HAA500: Human-Centric Atomic Action Dataset with Curated Videos** [2] byl RGB set dat *HAA500* natrénován na model *I3D* a bylo získáno:

- *Top1* = 33.53%,
- *Top3* = 53.40%.

K těmto hodnotám bychom se s architekturou *I3D ResNet* chtěli co nejvíce přiblížit.

6 Augmentace

V trénovací množině datasetu *HAA500* je pouze šestnáct videí a proto hrozí situace, že dojde k přetrénování neuronové sítě na trénovací data. Je třeba k trénovacímu algoritmu aplikovat augmentační přístupy. Tyto přístupy se snaží o změny vstupních dat v průběhu trénování sítě.

První použitá augmentace je založená na zvětšení videa a následném výběru oblasti a jejímu vyříznutí do původního rozměru obrázku (224×224). S touto metodou je ale třeba dát pozor, protože dataset *HAA500* je významným tím, že jsou činnosti centrovány do středu videa. Proto je vhodné volit malý poměr zvětšení, abychom ve výsledku nevyřízly důležité pixely z videa. Bylo otestováno, že velké zvětšení oblasti například z videa (224×224) na (320×320) pixelů a následné náhodné vyříznutí oblasti zpět na původní rozměry docházelo k horským výsledkům, než kdyby nebyla použitá žádná augmentační metoda. Proto se v této práci se video zvětšuje z rozměrů (224×224 , 260×250), vždy na čtverec a následuje náhodné oříznutí zpět na (224×224). Dále byla přidána další augmentační metoda, která každou epochu upravuje data způsobem náhodného horizontálního otočení videa s pravděpodobností $p_{flip} = 0.5$. Statisticky bude tedy každé druhé video bude otočeno.

Pomocí těchto metod jsou neuronové sítě předkládány každou epochu lehce odlišná data. Ukázka augmentace je zobrazeny na Obrázku 24, kde v horní části je originální video a ve spodní části je ukázka augmentovaného videa.



Obrázek 24: Ukázka augmentace (pád na kole)

7 Experiments I3D ResNet50

7.1 První trénovací experiment

První experiment byl proveden s malým subsetem datasetu *HAA500*. Malý subset obsahoval pouze 10 tříd. Celkový počet dat byl tedy viz. Tabulka 4.

	Počet videí
Trénovací data	160
Validační data	30
Testovací data	30
Celkový počet dat	200

Tabulka 4: Počet dat

Trénování proběhlo se sítí *I3D* s *backbone* sítí *ResNet50* pro 4,8,16 a 32 klíčových snímků videa. Hlavním cílem toho experimentu bylo především otestovat funkčnost trénování neuronové sítě. Tento postup je vhodný předtím než proběhne trénování pro všechna data, protože je možné takto odhalit drobné chyby programovacího kódu, které by mohly způsobovat nepřesnosti při trénovacím procesu. Při tomto experimentu s malým subsetem *HAA500* se také zkoumala přesnost klasifikace videí v závislosti na počtu zvolených klíčových snímků systému. Záznam experimentu je znázorněný v Tabulce 5.

	Top1 [%]	Top5 [%]
4 klíčové snímkы	0.23	0.83
8 klíčových snímků	0.23	0.83
16 klíčových snímků	0.3	0.83
32 klíčových snímků	0.33	0.87

Tabulka 5: Závislost přesnosti rozpoznávání a počtu klíčových snímků

Z Tabulky 5 je patrné, že pro systém pracující s více klíčovými snímkami roste také přesnost klasifikace testovaných videí. Díky této znalosti bude v následujících experimentech pracováno se systémy s 32 klíčovými snímkami, jediná jejich nevýhoda je ale větší výpočetní složitost a tím také delší doba trénování a testování.

7.2 Trénování na celém datasetu

Následující trénovací experiment byl proveden se všemi daty z trénovací a validační množiny datasetu *HAA500*. Vycházeli jsem už z předtrénovaných vah *I3D-ResNet50* natrénovaných na datasetu *Kinetics-400* a byl proveden takzvaný *fine-tuning*. Při tomto procesu načteme stejné váhy modelu a pouze odstraníme váhy poslední plně propojené lineární vrstvy, která přiřadí všem třídám hodnoty věrohodnosti. Díky tomuto bude možné natrénovat model na 500 odlišných tříd z našeho datasetu.

Trénovací *optimizer* byl použitý *SGD* s různými *schedulery*. Ve většině případů se síť trénovala 130 epoch. Jako nejlepší model byl vyhodnocen ten, který při trénovacím procesu dosáhl největší *Top1[%]* přesnost na validačních datech. Následně byl tento nejlepší model vyhodnocen na testovací množině dat. Celý trénovací experiment za cílem dosažení co nejlepší hodnoty *Top1[%]* na testovacím setu je zaznamenaný v Tabulce 6.

Pokus	Optimizer Scheduler	lr	nastavení	Top1[%] (val)	Top1[%] (test)	Top3[%] (test)	Top5[%] (test)
1.	SGD multisteps	0.01	step= [10, 25, 40, 50, 60, 70, 80, 90, 100]		33.667	54.133	62.867
2.	SGD multisteps	0.01	step= [15, 30, 60, 90, 110]		50.333	72.034	77.800
3.	SGD multisteps	0.01	step= [15, 20, 40, 80, 110]		54.600	76.067	71.733
4.	SGD step	0.01	step= 15		59.467	78.867	84.600
5.	SGD cosine	0.01	eta_min=0		56.000	74.333	80.266
6.1	SGD cyclic	10^{-6}	max_lr= 0.015	71.4	67.800	86.467	90.000
6.2	SGD cyclic	10^{-8}	max_lr= 0.01	75.2	69.00	86.133	90.666
6.3	SGD cyclic	10^{-8}	max_lr= 0.001	71.2	67.066	87.6	91.600
6.4	SGD cyclic	10^{-8}	max_lr= 0.003	72.0	68.466	85.466	89.933
6.5	SGD cyclic	10^{-8}	max_lr= 0.005	72.2	69.266	84.800	88.800
6.1.1	SGD cyclic	10^{-8}	max_lr= 0.02	73.2	69.333	87.266	91.4
6.1.7	SGD cyclic	10^{-8}	max_lr= 0.005	73.2	70.266	88.00	91.933
7.	SGD plateau	0.01	mode=min		54.0	72.733	79.866
8.	SGD restart	0.01	T_0 = 10, T_mult= 1, eta_min= 10^{-6}		51.4	70.866	77.666

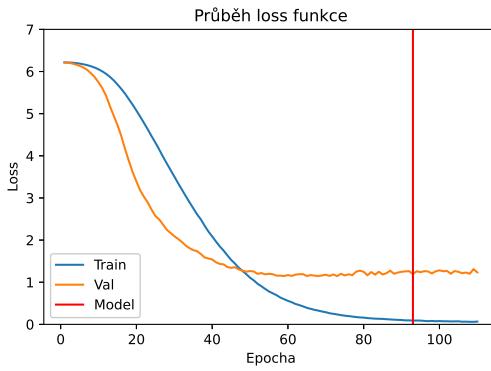
Tabulka 6: Trénování I3D ResNet50 na rgb datech

7.3 Experimenty s segmentačním modelem I3D Resnet

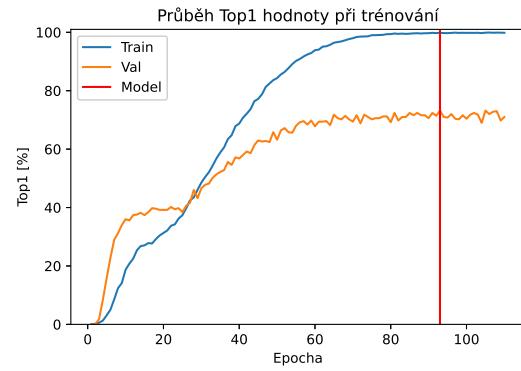
Pokus	Optimizer Scheduler	lr	nastavení	Top1 [%] (val)	Top1 [%] (test)	Top3 [%] (test)	Top5 [%] (test)
1.	SGD multisteps	0.01	step= [15, 30, 60, 90, 110]		28.800	46.867	53.667
2.	SGD multisteps	0.01	step= [10, 25, 50, 90, 110]		28.667	46.600	53.333
3.	SGD step	0.01	step= 20		28.000	44.933	52.733
4.	SGD multisteps	0.01	step= [17, 33, 66, 90, 110]		30.933	47.800	56.533
5.	SGD cosine	0.01	eta_min=0		38.867	54.667	62.467
5.1.	SGD cosine	0.005	eta_min=0	41.4	39.733	55.267	62.467
5.2.	SGD cosine	0.01	eta_min=0	37	34.733	51.067	58.000
5.3.	SGD cosine	0.02	eta_min=0	31.8	30.800	47.200	55.933
5.4.	SGD cosine	0.03	eta_min=0	25.8	27.733	45.133	52.600
6.5	SGD cyclic	10^{-8}	max_lr= 0.02	37.0	35.267	53.200	61.933
6.6	SGD cyclic	10^{-8}	max_lr= 0.03	33.2	35.733	50.800	57.467
6.9	SGD cyclic	0	max_lr= 0.01	34.6	37.600	52.400	59.600
6.10	SGD cyclic	0	max_lr= 0.008	34.133	50.733	59.933	59.600
6.11	SGD cyclic	0	max_lr= 0.013	37.8	35.400	53.800	61.333
7.	SGD plateau	0.01	mode=min		35.933	52.267	60.533
8.	SGD restart	0.01	T_0 = 10, T_mult= 1, eta_min= 10^{-6}		36.067	53.800	62.067

7.4 Experimenty s modelem optického toku I3D Resnet

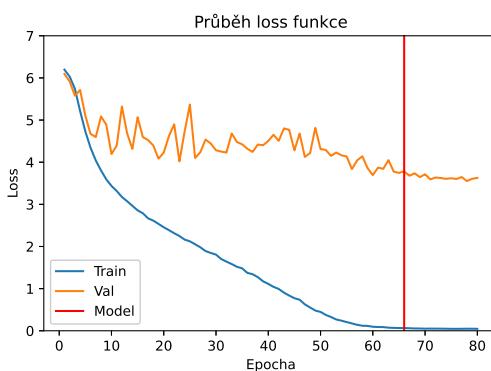
Pokus	Optimizer scheduler	lr	nastavení	Top1 [%] (val)	Top1 [%] (test)	Top3 [%] (test)	Top5 [%] (test)
1.	SGD multisteps	0.01	step= [15, 30, 60, 90, 110]		11.533	22.800	30.267
2.	SGD multisteps	0.01	step= [17, 35, 55, 70, 100, 115, 130]		20.466	33.800	42.933
3.	SGD step	0.01	step= 20		23.066	39.800	47.333
4.	SGD cosine	0.01	eta_min=0		29.2	45.800	53.800
5.	SGD cyclic	10^{-8}	max_lr= 0.1, mode='exp_range'		33.067	49.067	56.067
6.	SGD cyclic	10^{-6}	max_lr= 0.015		35.333	52.333	59.867
6.1.	SGD cyclic	10^{-8}	max_lr= 0.02	35.8	34.4	51.267	58.400
6.2.	SGD cyclic	10^{-8}	max_lr= 0.03	36.0	33.6	49.733	59.267
6.3.	SGD cyclic	10^{-8}	max_lr= 0.04	34.4	33.533	51.00	60.533
6.4.	SGD cyclic	10^{-8}	max_lr= 0.008	36.2	34.800	52.933	59.533
6.6.	SGD cyclic	0	max_lr= 0.018	37.0	34.800	51.067	57.667
6.7.	SGD cyclic	0	max_lr= 0.01	37.0	36.067	54.333	61.400
6.8.	SGD cyclic	0	max_lr= 0.005	36.8	34.00	52.800	59.667
6.9.	SGD cyclic	0	max_lr= 0.013	33.6	34.533	52.933	60.667
7.	SGD plateau	0.01	mode=min		26.200	43.067	50.200
8.	SGD restart	0.01	T_0 = 10, T_mult= 1, eta_min= 10^{-6}		30.733	46.867	54.400
9.	SGD restart	0.01	T_0 = 20, T_mult= 1, eta_min= 10^{-8}		31.733	47.867	54.667



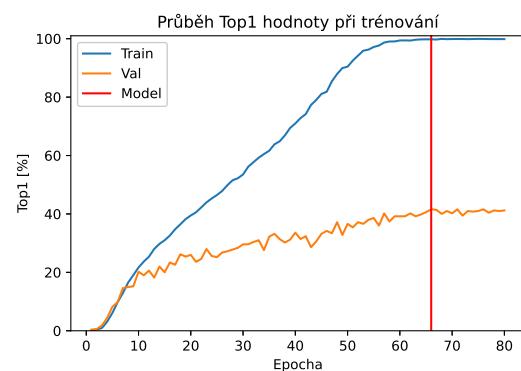
Obrázek 25: Průběh loss funkce při trénování - rgb data, cyclic



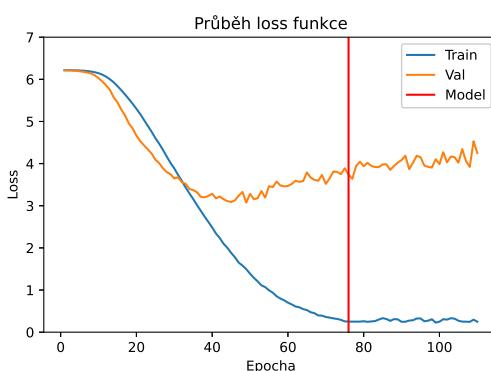
Obrázek 26: Průběh Top1 hodnoty při trénování - rgb data, cyclic



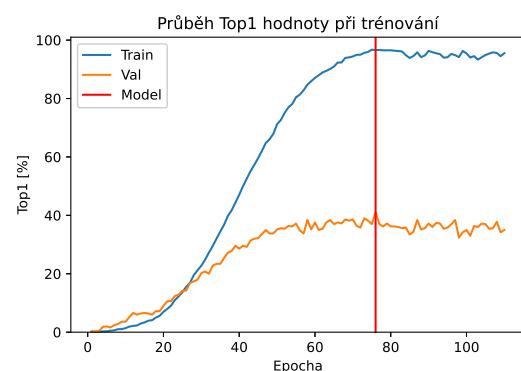
Obrázek 27: Průběh loss funkce při trénování - segmentační data, cosine



Obrázek 28: Průběh Top1 hodnoty při trénování - segmentační data, cosine



Obrázek 29: Průběh loss funkce při trénování - data optického toku, cyclic



Obrázek 30: Průběh Top1 hodnoty při trénování - data optického toku, cyclic

Použití: mean = [0.131, 0.084, 0.012], std = [0.250, 0.159, 0.028]

Pokus	Optimizer	sheduler	Top1 [%]	Top3 [%]	Top5 [%]
1.	SGD	cosine	37.267	51.933	59.533
2.	SGD	cyclic	35.533	52.467	59.867
3.	SGD	plateau	32.933	49.866	57.667

Nejlepší výsledky:

Data	Top1 [%]	Top3 [%]	Top5 [%]
RGB data	70.267	88.000	91.933
Segmentační data	39.733	55.267	62.467
Data s optickým tokem	36.066	54.333	61.400

Tabulka 7: Tabulka nejlepších výsledků

$$X = \alpha X_{rgb} + \beta X_{segmentace} + \gamma X_{opticky_tok} \quad (5)$$

α	β	γ	Top1 [%]	Top3 [%]	Top5 [%]
1.0	1.0	1.0	69.267	84.200	89.400
1.0	0	1.0	68.867	84.533	89.933
1.0	1.0	0	69.000	84.933	90.333
0	1.0	1.0	46.200	62.400	69.333

Tabulka 8: Tabulka přesnosti rozpoznávání pro různé parametry α, β, γ

	α	β	γ	Top1 [%]	Top3 [%]	Top5 [%]
Test na validační setu	0.55	0.35	0.29	76.000	86.200	91.200
Test na testovacím setu	0.55	0.35	0.29	72.200	85.200	90.400

Tabulka 9: Tabulka přesnosti rozpoznávání pro různé parametry α, β, γ

$$X = X_{rgb} \cdot X_{segmentace} \cdot X_{opticky_tok} \quad (6)$$

	Top1 [%]	Top3 [%]	Top5 [%]
Test na validační setu	73.800	87.200	92.400
Test na testovacím setu	72.867	88.667	92.133

Tabulka 10: Tabulka přesnosti rozpoznávání pro různé parametry $\alpha, \beta\gamma$

$$X = X_{rgb} \cdot (\beta X_{segmentace} + \gamma X_{opticky_tok}) \quad (7)$$

	β	γ	Top1 [%]	Top3 [%]	Top5 [%]
Test na validační setu	0.80	0.13	77.800	89.800	94.20
Test na testovacím setu	0.80	0.13	74.800	89.933	93.400

Tabulka 11: Tabulka přesnosti rozpoznávání pro různé parametry $\alpha, \beta\gamma$

7.5 Klasifikátory

Trénování na validační množině (500 vzorů), pak testováno na testovací množině :

Klasifikátor	Top1 [%]	Top3 [%]	Top5 [%]
LinearSVC(C=0.5)	50.6	70.19	76.86
LinearSVC(C=1)	50.6	70.19	76.86
SVM(kernel = linear)	40.8	51.47	53.60
SVM(kernel = rbf)	40.8	51.47	53.60
KNN(n=1)	40.8	41.0	41.33
KNN(n=3)	10.33	51.47	51.73

Tabulka 12: t1

Trénování na trénovací množině (8000 vzorů), pak testováno na testovací množině :

Klasifikátor	Top1 [%]	Top3 [%]	Top5 [%]
LinearSVC(C=0.5)	58.13	83.93	88.8
LinearSVC(C=1)	59.19	83.53	88.53
SVM(kernel = linear)	52.80	79.33	81.53
SVM(kernel = rbf)	48.40	52.93	52.20
KNN (n=1)	51.6	51.73	51.93
KNN (n=3)	48.8	56.13	56.13

Tabulka 13: t2

Trénování na trénovací množině + validační množině (8500 vzorků), pak testováno na testovací množině:

Klasifikátor	Top1 [%]	Top3 [%]	Top5 [%]
LinearSVC(C=0.5)	66.00	84.40	88.80
LinearSVC(C=1)	67.06	83.6	88.33
SVM(kernel = linear)	58.067	68.40	71.27
SVM(kernel = rbf)	49.73	56.47	60.33
KNN (n=1)	52.27	52.33	52.47
KNN (n=3)	47.27	58.93	59.13

Tabulka 14: t3

8 Závěr

Zatím nic

9 Seznam literatury

Odkazy

- [1] Vitaly Bushaev. *How do we 'train' neural networks ?* 2018. URL: <https://towardsdatascience.com/how-do-we-train-neural-networks-edd985562b73>.
- [2] Jihoon Chung et al. "HAA500: Human-Centric Atomic Action Dataset with Curated Videos". In: *ICCV 2021*.
- [3] Joseph Paul Cohen a Henry Z. Lo. "Academic Torrents: A Community-Maintained Distributed Repository". In: *Annual Conference of the Extreme Science and Engineering Discovery Environment*. 2014. DOI: 10.1145/2616498.2616528. URL: <http://doi.acm.org/10.1145/2616498.2616528>.
- [4] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, s. 248–255.
- [5] Mark Gituma. *Downloading the kinetics dataset for Human Action Recognition in deep learning*. 2021. URL: <https://towardsdatascience.com/downloading-the-kinetics-dataset-for-human-action-recognition-in-deep-learning-500c3d50f776>.
- [6] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. DOI: 10.48550/ARXIV.1512.03385. URL: <https://arxiv.org/abs/1512.03385>.
- [7] Qingge Ji et al. "Optimized Deep Convolutional Neural Networks for Identification of Macular Diseases from Optical Coherence Tomography Images". In: *Algorithms* 12 (ún. 2019), s. 51. DOI: 10.3390/a12030051.
- [8] Will Kay et al. "The Kinetics Human Action Video Dataset". In: *CoRR* abs/1705.06950 (2017). arXiv: 1705.06950. URL: <http://arxiv.org/abs/1705.06950>.
- [9] Henry Z. Lo a Joseph Paul Cohen. "Academic Torrents: Scalable Data Distribution". In: *Neural Information Processing Systems Challenges in Machine Learning (CiML) workshop*. 2016. URL: <http://arxiv.org/abs/1603.04395>.
- [10] Karen Simonyan a Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014. DOI: 10.48550/ARXIV.1409.1556. URL: <https://arxiv.org/abs/1409.1556>.
- [11] Christian Szegedy et al. *Rethinking the Inception Architecture for Computer Vision*. 2015. DOI: 10.48550/ARXIV.1512.00567. URL: <https://arxiv.org/abs/1512.00567>.

10 Seznam obrázků

Seznam obrázků

1	CIFRA-10 trénovací a testovací error původní obrázek: https://arxiv.org/pdf/1512.03385.pdf (strana 1)	3
2	Residuální blok	3
3	Bottleneck blok	4
4	Blok A	5
5	Blok B	5
6	Blokově znázorněná architektura I3D ResNet 50	6
7	Kroková strategie <i>lr</i>	8
8	Kosínová strategie <i>lr</i>	8
9	Cyklická strategie <i>lr</i> (jeden cyklus)	8
10	Strategie restartováním <i>lr</i>	8
11	Třída fotbalový dribbling	9
12	Třída fotbalová hlavička	9
13	Třída fotbalový zákrok brankáře	10
14	Ukázka segmentace - hod diskem	11
15	Ukázka segmentace - pád na kole	11
16	Ukázka segmentace - hod diskem	12
17	Ukázka segmentace - baseball odpal míčku	12
18	Distribuce tříd	13
19	Ukázka nejčetnějších tříd	14
20	Ukázka - snowkiting	14
21	Ukázka - tobogganing	14
22	Ukázka - blowing out candles	15
23	Ukázka - canoeing or kayaking	15
24	Ukázka augmentace (pád na kole)	16
25	Průběh loss funkce při trénování - rgb data, cyclic	23
26	Průběh Top1 hodnoty při trénování - rgb data, cyclic	23
27	Průběh loss funkce při trénování - segmentační data, cosine	23
28	Průběh Top1 hodnoty při trénování - segmentační data, cosine	23
29	Průběh loss funkce při trénování - data optického toku, cyclic	23
30	Průběh Top1 hodnoty při trénování - data optického toku, cyclic	23

11 Seznam tabulek

Seznam tabulek

1	Počet dat	10
2	Počet dat	13
3	Příklad testování úspěšnosti klasifikace videí	15
4	Počet dat	18
5	Závislost přesnosti rozpoznávání a počtu klíčových snímků	18
6	Trénování I3D ResNet50 na rgb datech	20
7	Tabulka nejlepších výsledků	24
8	Tabulka přesnosti rozpoznávání pro různé parametry $\alpha, \beta\gamma$	24
9	Tabulka přesnosti rozpoznávání pro různé parametry $\alpha, \beta\gamma$	24
10	Tabulka přesnosti rozpoznávání pro různé parametry $\alpha, \beta\gamma$	25
11	Tabulka přesnosti rozpoznávání pro různé parametry $\alpha, \beta\gamma$	25
12	t1	26
13	t2	26
14	t3	26