VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FAKULTA ELEKTROTECHNIKY

A KOMUNIKAČNÍCH TECHNOLOGIÍ

MPC-STU

SROVNÁNÍ KNIHOVEN PRO ROZŠÍŘENÍ

OBRAZOVÉHO DATASETU

Pavel Dušek, Jiří Pražák, Stanislav Svědiroh,

Václav Večeřa, Václav Zmítko

Brno, 2021

Obsah

[Zadání 1](#_Toc70962337)

[1. Knihovny pro augmentaci dat 2](#_Toc70962338)

[1.1 Augmentor 2](#_Toc70962339)

[1.1.1 Zkosení (Perspective Skewing) 3](#_Toc70962340)

[1.1.2 Elastické zkreslení (Elastic Distortions) 3](#_Toc70962341)

[1.1.3 Rotace (Rotating) 4](#_Toc70962342)

[1.1.4 Roztažení (Shearing) 5](#_Toc70962343)

[1.1.5 Oříznutí (Cropping) 6](#_Toc70962344)

[1.1.6 Zrcadlení (Mirroring) 6](#_Toc70962345)

[1.2 Albumentations 6](#_Toc70962346)

[1.3 Imgaug 8](#_Toc70962347)

[1.3.1 Heatmaps 11](#_Toc70962348)

[1.3.2 Segmentační mapy 11](#_Toc70962349)

[1.4 AutoAugment 12](#_Toc70962350)

[1.4.1 ImageNetPolicy 12](#_Toc70962351)

[1.4.2 CIFAR10Policy 12](#_Toc70962352)

[1.4.3 SVHNPolicy 13](#_Toc70962353)

[1.5 PyTorch 13](#_Toc70962354)

[1.5.1 Torchvision.transforms 14](#_Toc70962355)

[2. Benchmark knihoven 16](#_Toc70962356)

[2.1 Horizontal flip 17](#_Toc70962358)

[2.2 Vertical flip 17](#_Toc70962359)

[2.3 Rotate 18](#_Toc70962360)

[2.4 Shift scale rotate 18](#_Toc70962361)

[2.5 Brightness 19](#_Toc70962362)

[2.6 Contrast 19](#_Toc70962363)

[2.7 Random crop 20](#_Toc70962364)

[2.8 Resize 20](#_Toc70962365)

[2.9 Gray scale 21](#_Toc70962366)

[3. Závěr 22](#_Toc70962367)

# Zadání

Datasety jsou alfou a omegou strojového učení. Pro metody hlubokého učení je vhodné mít datasety dostatečně obsáhlé. Bohužel stává se, že vzorových dat je omezené množství. V tomto případě přichází na řadu metody rozšíření datasetu. Cíl projektu je nastudovat, zdokumentovat a vytvořit srovnání minimálně 5 knihoven vybraných po domluvě s vedoucím projektu pro rozšíření datasetu v jazyce Python. Součástí řešení je vytvoření časového srovnání jednotlivých implementací.

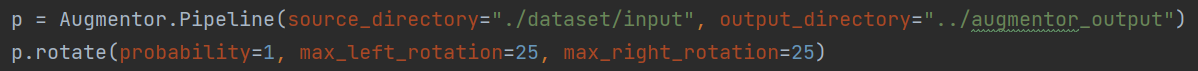
# Knihovny pro augmentaci dat

Augmentace dat je operace pro rozšíření datasetu, které jsou využity pro trénování modelů strojového učení. Důvodem jejího využití je zabránění overfitingu, což znamená případu, kdy je neuronová síť trénována na datech, která nejsou dostatečně rozmanitá. Existuje několik možností pro augmentaci dat, těmito možnostmi jsou různé translace, rotace, změny škálování, zašumění, překlápění obrazu, změna jasu, kontrastu atd… Bylo zadáno pět knihoven pro rozšíření datasetu (augmentaci dat). Těmito knihovnami jsou *Augmentor, Albumentations, Imgaug, PyTorch a AutoAugment.* Tyto knihovny jsou balíčkem v Pythonu navržených pro podporu rozšíření a umělého generování obrazových dat pro úlohy strojového učení.

## Augmentor

Tato knihovna nabízí augmentaci, a to několika způsoby, především zkosení (Perspective Skewing), elastické zkreslení (Elastic Distortions), rotaci (Rotating), roztažení (Shearing), oříznutí (Cropping), a zrcadlení (Mirroring). Do argumentu každé funkce v následujících podkapitolách je zadávána pravděpodobnost (Probability) v mezích 0 – 1 podle klasické pravděpodobnosti.

Práce s knihovnou probíhá následovně, je vytvořen objekt třídy Pipeline, kterému je v parametru zadána cesta k vstupním datům a cesta pro ukládání vygenerovaných dat. Dále je možné tomu objektu přidávat operace, viz podkapitoly níže.



Tato knihovna je jediná z námi testovaných knihoven, u které lze přímo využít paralelizaci (Multithreading) a to následnovně:



Výsledná pipeline jde přímo importovat do pipeline knihovny *PyTorch* a to následovně:

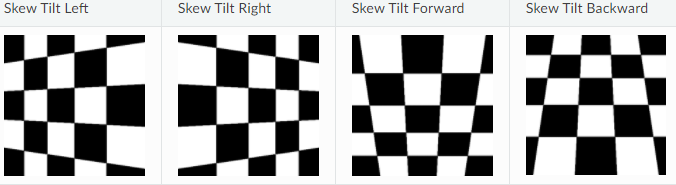
Obsah obrázku text

Popis byl vytvořen automaticky

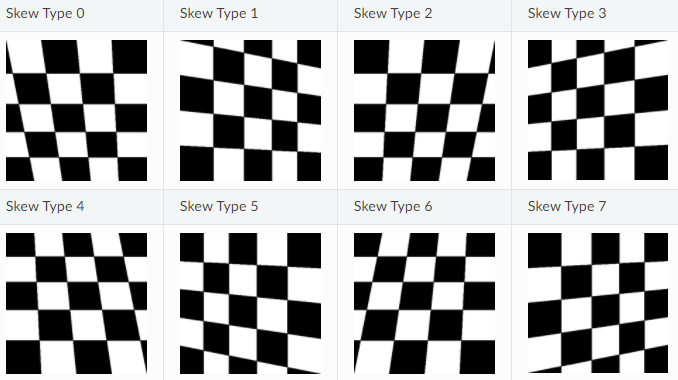
### Zkosení (Perspective Skewing)

Tento balíček funkcí zajišťuje transformaci vstupních dat tak, jako by na ně kamera nahlížela pod jiným úhlem. Všechny níže uvedené funkce mají parametr *magnitude*, kterým je *nastavováno maximální možné zkosení (v rozmezí 0.1 – 1.0).*:

* skew\_tilt() - podporované argumenty (left, right, forward, backward)



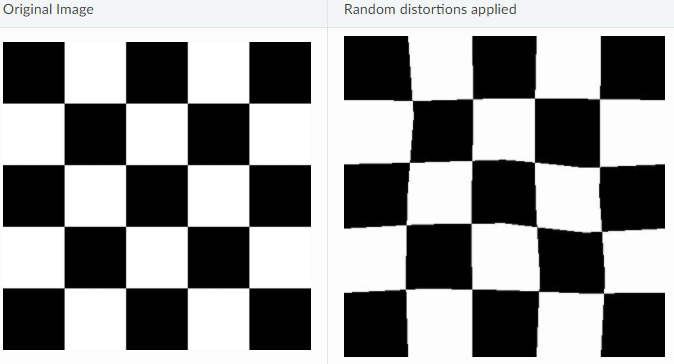
* skew\_left\_right()
* skew\_top\_bottom()
* skew\_corner() - výběr 8 možných bodů zkosení



* skew() -zkosí obrázek náhodně podle jednoho z 12 směrů zobrazených výše.

### Elastické zkreslení (Elastic Distortions)

Tento balíček funkcí slouží k provedení elastického zkreslení obrazu, avšak je zachován poměr stran, jak lze vidět na obrázku níže.



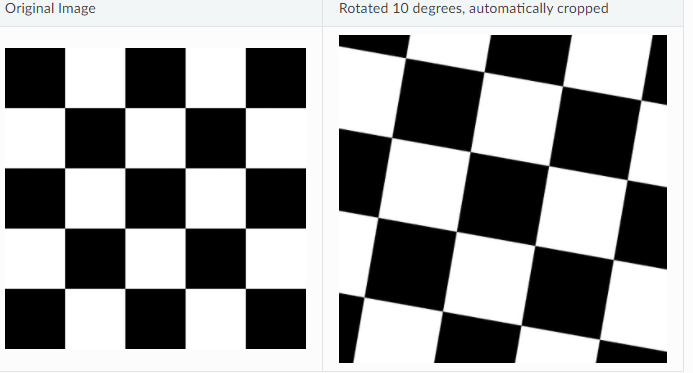
K tomuto zkreslení slouží funkce:

* *random\_distortion()* - jsou zadávány parametry *grid\_width, grid\_height, magnitude*. První dva parametry mohou nabývat hodnot 2 až 10 a určují velikost mřížky. Hodnota parametru *magnitude* by měla být v rozmezí 1 až 10.

### Rotace (Rotating)

Tento balíček funkcí slouží k augmentaci pomocí rotace, k tomu slouží funkce:

* *rotate()* - do argumentu této funkce je možné zadat parametry *max\_left\_rotation* a *max\_right\_rotation*, v těchto mezích dochází k náhodné rotaci. Pokud není obrázek rotován v násobcích 90°, pak jsou výsledná data oříznuta a je kladen důraz na co možná největší oříznutí, za účelem dodržení původního poměru stran, což vede k přiblížení objektu. Pokud by byla rotace provedena klasicky, aby byl vidět celý rotovaný objekt, pak vznikne kolem objektu pole, což může být nežádoucí. Tato problematika je vyobrazena na obrázku níže.

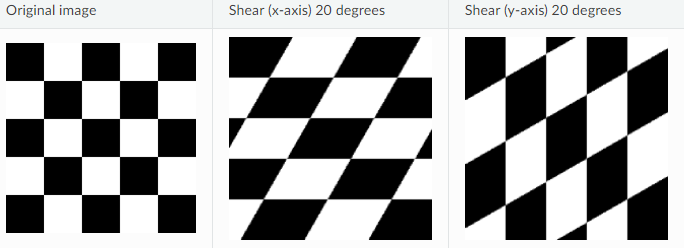


* *rotate90()* - rotace o 90°
* *rotate180()* - rotace o 180°
* *rotate270()* - rotace o 270°
* *rotate\_random\_90()* - funkce otočí objekt o náhodný násobek 90°

### Změna perspektivy (Shearing)

Modifikuje objekt změnou perspektivy ve směru x nebo y. Opět tato knihovna klade důraz na zachování poměru stran, a proto se roztažený obraz ořízne (přiblíží). Jak lze vidět na obrázku níže. K této modifikaci slouží funkce:

* shear() - do argumentu této funkce je možné zadat parametry max\_shear\_left a max\_shear\_right, v těchto mezích dochází k náhodnému roztažení, přičemž není možno zadat hodnotu meze větší než 25°.



### Oříznutí (Cropping)

Oříznutí probíhá jako přiblížení určité náhodné části objektu. Oříznutí zařizují následující funkce:

* crop\_centre() - do argumentů této funkce je nutné zadat parametry percentage\_area, randomise\_percentage\_area. Pokud je druhá zmiňovaný parametr false, pak první parametr udává absolutní hodnotu oříznutí, v opačném případě udává rozmezí mezi 0 až percentage\_area.
* crop\_by\_size() - do argumentů této funkce je nutné zadat parametry width, height, centre. První dva tyto parametry zajišťují absolutní hodnotu oříznutí v daném směru. Centre, pokud je tento parametr true, pak je ořezáváno směrem do středu, pokud je to naopak, pak je ořezaný prostor náhodný.
* crop\_random() - do argumentů této funkce je nutné zadat parametry percentage\_area, randomise\_percentage\_area. Na rozdíl od funkce crop\_centre neprobíhá oříznutí směrem ze středu, ale v náhodném vybraném místě.

### Zrcadlení (Mirroring)

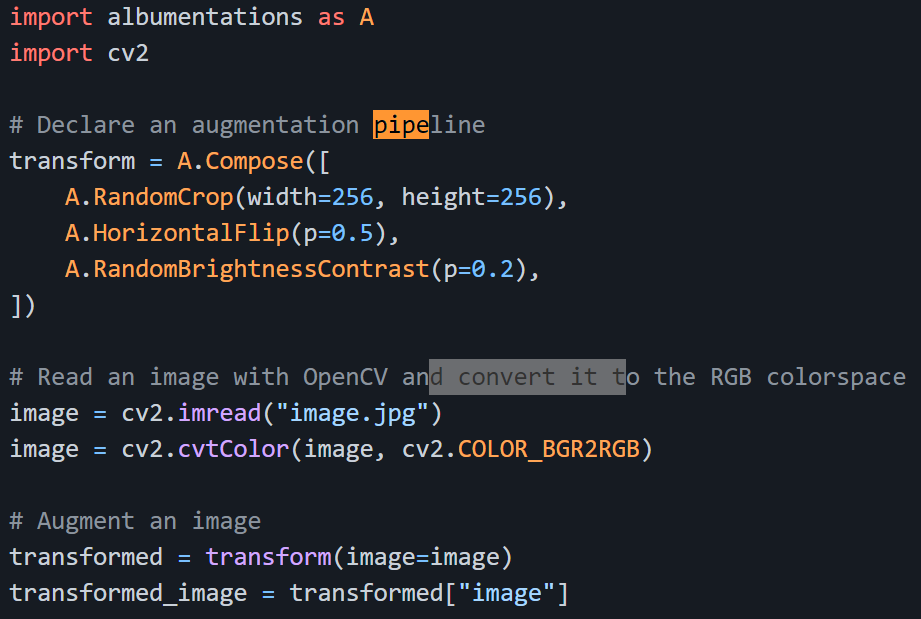
Tento balíček funkcí otáčí (zrcadlí) objekt pomocí následujících funkcí:

* flip\_left\_right() - přetočí objekt pomocí svislé osy zleva doprava
* flip\_top\_bottom() - přetočí objekt pomocí vertikální osy z vrchu dolu
* flip\_random() - přetočí objekt pomocí náhodně vybrané osy

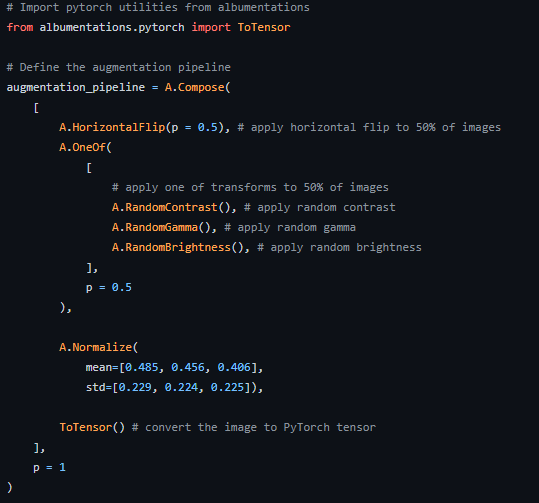
## Albumentations

Tato knihovna nabízí více než 70 různých funkcí pro augmentaci dat. Kromě základních forem augmentace dat, které již byly popsány v knihovně *Augmentor*, se specializuje především na transformaci obrazu na úrovni pixelů. Díky tomu nabízí augmentaci pomocí změny barvy a jejich odstínů (*RGBShift, ChannelShuffle, ToGray, FancyPCA, HueSaturationValue, ToSepia*), změny sytosti (HueSaturationValue, *ColorJitter*), inverze barev (*InvertImg, Solarize*), změny kontrastu (*CLAHE, RandomContrast, RandomGamma,* *ColorJitter*), změna jasu (*RandomBrightness, ColorJitter*), rozmazání/rozostření (*Blur, MedianBlur, MotionBlur, GaussianBlur, GlassBlur, RandomFog*), zaostření (*Sharpen*), vyrovnání barev (*Equalize*), přidání šumu (*GaussNoise, ISONoise, MultiplicativeNoise, RandomRain, RandomShadow, RandomSnow, RandomSunFlare*), normalizace obrazu (*Normalize*) a kompresi obrazu (*JpegCompression, Downscale, ImageCompression, Posterize*). Velkou výhodou této knihovny je možnost neuvažovat (zahodit) části vstupních dat a zpracovávat pouze relevantní části, k tomu slouží funkce (*ChannelDropout, CoarseDropout, GridDropout, MaskDroupout*).

Vytvoření pipeline probíhá velmi podobně, jako u předcházející knihovny, ukázka syntaxe je vyobrazena na následujícím obrázku:



Na konci pipeline je možné obrázky přímo překonvertovat jako *Tensor* a následně ho lze předat do knihovny PyTorch následovně:



Výsledky po provedení augmentace některými funkcemi jsou vyobrazeny na následujícím obrázku.

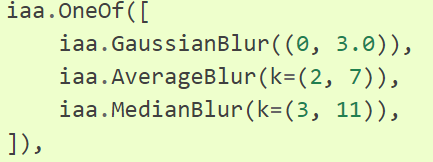


## Imgaug

Tato knihovna kromě pokročilé augmentace dat dokáže také augmentovat na základě předem definovaného rozložení pravděpodobnosti (stochastické parametry) a dokáže pracovat s hloubkovými a segmentační mapami.

Oproti předchozím knihovnám lze nastavit aplikaci některých augmentačních funkcí jenom „občas“ (funkce *sometimes()*). Například *sometimes(0.5, GaussianBlur(0.3))* – takto bude *GaussianBlur()* aplikován pouze na 50% vstupních dat. Výhodou je, že těchto funkcí může *Sometimes()* obsahovat více a s touto pravděpodobností bude aplikován celý balíček funkcí na vstupní data.

Dalšími zajímavými funkcemi v této knihovně jsou SomeOf() a OneOf. Například SomeOf((0,5),….) vybere náhodně 0-5 pravidel, které jsou dále uvedeny v argumentu této funkce. Nedoporučuje se aplikace až příliš mnoha augmentačních funkcí najednou, protože mohou být vstupní data změněna k nepoznání. Toto se hodně projevuje u použití více než jednoho rozostření na každý objekt, této problematice zamezuje funkce OneOf(), která náhodně vybere pouze jedno pravidlo a to je aplikováno, dále je uveden příklad:



Integrace do PyTorch probíhá přes Tensor, tato knihovna exportuje augmentovaná data ve formě Numpy array a PyTorch obsahuje funkci, která dokáže Numpy array převést na Tensor.

Zajímavou formu augmentace dat přináší změna počasí, a to především zasněžení objektů, jak je vyobrazeno na následujících obrázcích.

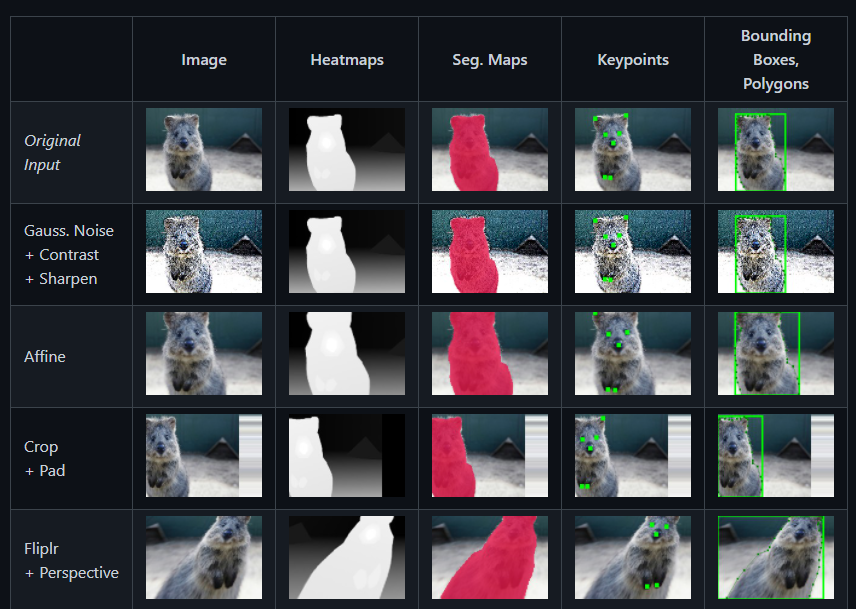




Vytvoření pipeline probíhá obdobně jako u předchozích knihoven viz následující obrázek:

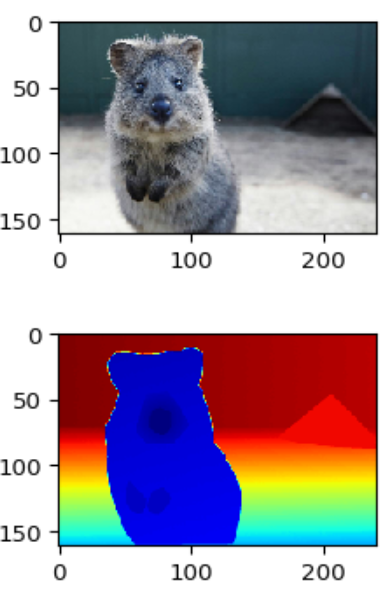


Knihovna umožňuje zobrazení Heatmap, Segmentačních map, klíčových bodů (Keypoints), ohraničujících rámečků a polygonů (Bounding Boxes):



### Heatmaps

Heatmapa je dalším rozměrem objektu, upřesňuje informace například o vzdálenosti objektu od kamery (*DepthMaps*).



### Segmentační mapy

Segmentační mapy pomáhají umělé inteligenci identifikovat objekty na vygenerovaných datech. Segmentační mapa je dalším rozměrem zpracovávaných dat. Při augmentačních operacích, měnících geometrii objektu na snímku (např. rotace, elastické deformace, …) se provede stejná operace i na segmentační mapě, aby objekty, na něž segmentační vrstva odkazuje byly na shodné pozici. Segmentační mapu je možné vytvořit například pomocí polygonů.

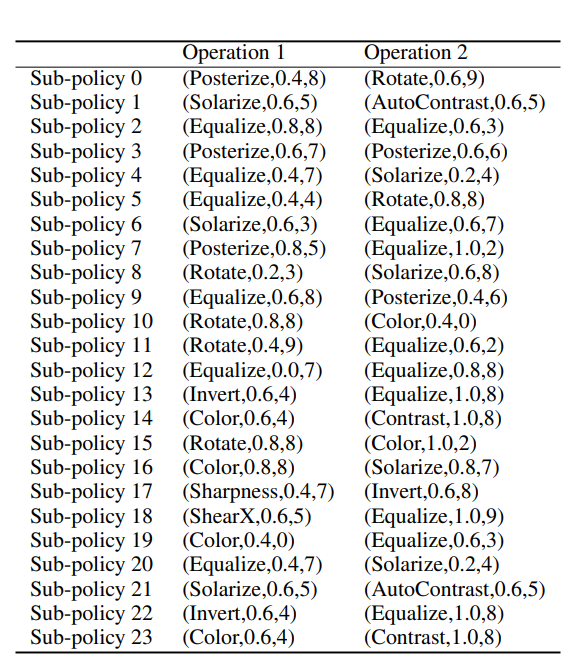


## AutoAugment

Narozdíl od předchozích knihoven, kde jsou programátorem předem definované funkce, určující, jakým způsobem budou vstupní data upravena, tak knihovna AutoAugment obsahuje sady pravidel (policy), jak obrázky upravovat, aby bylo dosaženo dobrých výsledků na testovacích datasetech. Tyto soubory pravidel byly vytvořeny umělou inteligencí na základě předem specifikovaných datasetů, proto každá z dílčí sady pravidel je naučena na práci se specifickými datasety a na nich dosahují nejlepších výsledků. Z toho důvodu nebude tato knihovna zahrnuta do porovnání v rámci této práce. Níže jsou uvedeny datasety na které byla umělá inteligence naučena. Pro každý dataset je následně uvedena tabulka pravidel, které tomu odpovídají.

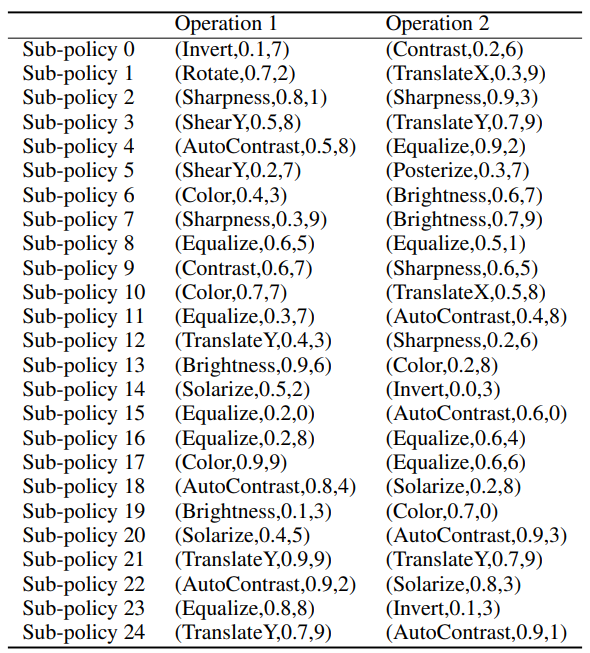
### ImageNet

Na vstupních datech jsou s určitou pravděpodobností prováděny funkce vypsané níže:



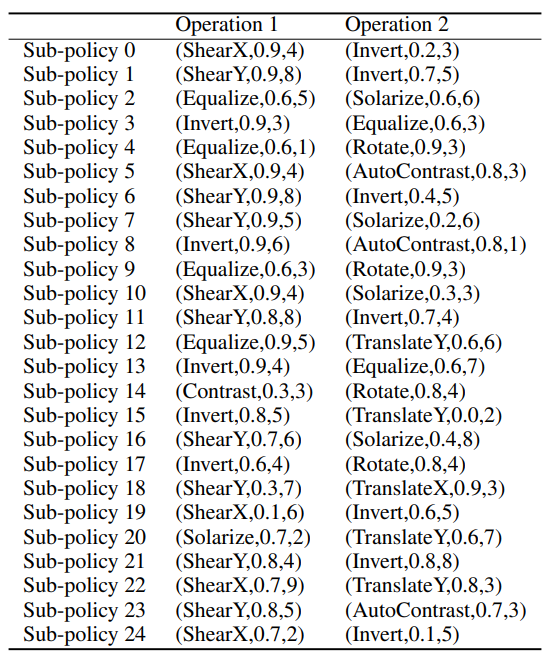
### CIFAR10

Jedná se o pravidla, která byla umělou inteligencí vytvořena na základě datasetu obsahujícího fotografie letadel, automobilů, kamionů, koček, psů, jelenů, koní, žab a ptáků. Na vstupních datech jsou s určitou pravděpodobností prováděny funkce vypsané níže:



### SVHN

Na vstupních datech jsou s určitou pravděpodobností prováděny funkce vypsané níže:



## PyTorch

Framework *PyTorch* je velmi obsáhlý, skládá se z několika knihoven, neobsahuje pouze knihovny pro augmentaci obrazových dat, ale také pokročilé metody deeplearningu nejenom obrazových dat, ale například i zvuku a textu. Předchozí zmíněné knihovny, generující rozšířené datasety mohou spolupracovat s frameworkem *PyTorch*, který je poté může použít pro strojové učení. Pro naši aplikaci je stěžejní balíček *torchvison* a jeho část *transforms*, která slouží k rozšíření obrazového datasetu.

### Torchvision.transforms

Tato část knihovny *Torchvision* je zaměřena na augmentaci obrazových dat. Obsah jejich funkcí je podobný těm, které umí předchozí knihovny. Vytvoření *pipeline* probíhá následovně:

Obsah obrázku text

Popis byl vytvořen automaticky

Oproti ostatním knihovnám je zde nezbytné vytvořit vlastní třídu, která se stará o načítání a augmentaci samotných obrázků. Je nutné, aby tato třída dědila ze třídy Dataset.

*CustomDataset* obsahuje 3 funkce, kromě konstruktoru *\_\_*init\_\_je tu funkce *\_\_*len\_\_*,* která vrací počet obrázků ve vstupních datech a funkci *\_\_getitem*\_\_která samotný obrázek načte z paměti a případně ho augmentuje.

Obsah obrázku text

Popis byl vytvořen automaticky

Po vytvoření třídy CustomDataset jí můžeme předat spolu s dalšími provozními parametry.



Finální augmentace probíhá pomocí zavolání funkce *enumerate*, které je předán *dataset\_loader*. Po zavolání této funkce je vytvořeno tolik augmentovaných dat, kolik je celkový počet vstupních dat (návratová hodnota funkce *\_\_len\_\_*).

Obsah obrázku text

Popis byl vytvořen automaticky

Funkce enumerate vrací data ve formátu *Tensor*, je nutné tento formát překonvertovat na obrázek.

# Benchmark knihoven

V následující tabulce jsou uvedena data, která vzešla z testování rychlostí augmentace jednotlivých knihoven. Podmínky pro generování augmentovaných dat byly následující. Výstupem všech knihoven bylo 100 augmentovaných obrazových dat pro každou z uvedených funkcí. Například: Všechny knihovny byly otestovány na rotaci, tudíž vzešlo z každé knihovny 100 rotovaných obrazových dat. Vždy byla změřena doba trvání augmentace všech dat u každé z knihoven. Tyto časy v milisekundách, použité augmentační funkce a knihovny jsou uvedeny v následující tabulce.

Je možné si povšimnout, že knihovna Augmentor je použita v našem testování rychlosti (benchmark) dvakrát, a to pro výpočet s použitím multithreadingu (Augmentor\_MT) a bez něj, protože se jednalo o jedinou knihovnu, která ho nabízela přímo. Ostatní knihovny byly testovány pouze bez něho.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Action** | **Augmentor** | **Augmentor\_MT** | **Albumentations** | **ImgAug** | **PyTorch** |
| HORIZONTAL\_FLIP | 7810 | 1870 | 13708 | 14443 | 42142 |
| VERTICAL\_FLIP | 8699 | 2000 | 12965 | 13719 | 41871 |
| ROTATE | 92507 | 16456 | 17850 | 17434 | 45718 |
| SHIFT\_SCALE\_ROTATE | 24794 | 4658 | 17703 | 5331 | 9378 |
| BRIGHTNESS | 13011 | 7925 | 12795 | 525084 | 45638 |
| CONTRAST | 15858 | 8603 | 13629 | 109142 | 51019 |
| RANDOM\_CROP | 3821 | 933 | 88 | 18134 | 7503 |
| RESIZE | 6443 | 1573 | 290 | 1138 | 9923 |
| GRAYSCALE | 6540 | 1329 | 12855 | 23433 | 19521 |

Pro větší přehlednost jsou časy jednotlivých augmentačních funkcí vyobrazeny v následujících grafech pro jednotlivé knihovny.



## Horizontal flip

Na následujícím grafu je vynesena časová závislost augmentační funkce *horizontal\_flip* na použité knihovně. V této funkci dost markantně vítězí *Augmentor\_MT* (s použitím multithreadingu), zatímco *PyTorch* dosahuje nejhorších výsledků.

## Vertical flip

Na následujícím grafu je vynesena časová závislost augmentační funkce *vertical\_flip* na použité knihovně. Výsledky rychlostí jsou zde stejné jako u *horizontzal\_flip*. Tento výsledek byl očekáván, jelikož se jedná prakticky o stejnou operaci.

## Rotate

Na následujícím grafu je vynesena časová závislost augmentační funkce *rotate* na použité knihovně. Lze vidět, že *Augmentor\_MT, Albumentations a ImgAug* zde dosahují skoro stejných rychlostí.

## Shift scale rotate

Na následujícím grafu je vynesena časová závislost augmentační funkce shift\_scale\_*rotate* na použité knihovně. *Augmentor\_MT* je zde nejrychlejší.

## Brightness

Na následujícím grafu je vynesena časová závislost augmentační funkce *brightness* na použité knihovně. Augmentace za pomocí knihovny *ImgAug* probíhala extrémně dlouho oproti ostatním knihovnám, proto je zde vyobrazen ještě jeden graf, aby bylo zřejmé jak obstály zbylé knihovny.

## Contrast

Na následujícím grafu je vynesena časová závislost augmentační funkce *contrast* na použité knihovně. Knihovna *ImgAug* dosahuje opět nejhorších výsledků.

## Random crop

Na následujícím grafu je vynesena časová závislost augmentační funkce *random\_crop* na použité knihovně. *Albumentations* v této aplikaci předčila všechny knihovny.

## Resize

Na následujícím grafu je vynesena časová závislost augmentační funkce *resize* na použité knihovně. *Albumentations* dosáhla nejlepšího času.

## Gray scale

Na následujícím grafu je vynesena časová závislost augmentační funkce *resize* na použité knihovně. *Albumentations* dosáhla nejlepšího času.

# Závěr

První část této práce je věnována teoretickému rozboru jednotlivých knihoven a jejich augmentačním metodám. Z hlediska rozmanitosti možností pro rozšiřování datasetu subjektivně vítězí *Albumentations* a *PyTorch*. Samozřejmě je nutné brát v úvahu konkrétní úlohu, pak může být objektivně výhodnější použít knihovnu jinou.

Knihovna *Augmentor* obsahuje pouze základní augmentační metody, velkou její výhodou je velice snadné použití a konverze pipeline do knihovny *PyTorch*.

Knihovna *Albumentations* se jeví z hlediska různorodosti augmentačních metod jako dobrá volba, jako jediná obsahuje velké množství metod zaměřujících se na změnu barevné informace obrazu.

Knihovna ImgAug je velmi zajímavá z hlediska funkčnosti její augmentační pipeline. Umožňuje použití logických funkcí „občas“, „jedna z“ atd. Tato funkcionalita se jeví jako velice důležitá, pokud chceme zamezit konkrétním kombinacím augmentačních funkcí.

Knihovna PyTorch obsahuje funkčnosti pro učení modelů umělé inteligence. Kromě balíčku pro augmentaci obrazových dat dokáže augmentovat též například zvuk. Z tohoto důvodu je pro ostatní výše zmíněné knihovny žádoucí možnost přímé kompatibility s PyTorch. Pokud však potřebujeme data pouze augmentovat, PyTorch se zdá být místy příliš složitý.

Knihovna Autoaugment je jednoduše použitelná pouze na konkrétní datasety. Pokud bychom chtěli jejich možností využít i pro jiné, je třeba nejprve naučit model, což nemusí být v časových možnostech konkrétního projektu.

Další částí této práce je vyhodnocení rychlosti základních transformací. V této kapitole byly vyhodnoceny časy jednotlivých augmentačních funkcí pro různé knihovny. Z celkového námi provedeného porovnání na vybraných funkcích vychází knihovna PyTorch, jako nejpomalejší. Dále následují knihovny, ImgAug, Augmentor bez použití multithreadingu, Albumentations. Jako nejrychlejší, na vybraných funkcích, byla knihovna Augmentor (s použitím multithreadingu). Snažili jsme se knihovnám zajistit rovnaké podmínky, ale bez náhledu do vnitřních implementací je toto obtížné. Vyhodnocení je tedy spíše informativní, pokud chceme pouze základní implementaci knihovny bez manuálního řešení multithreadingu. Dokumentace knihovny PyTorch se také zmiňuje o možnosti využítí CUDA jader na grafické kartě, což by mělo výrazně změnit její rychlost. V tomto projektu ale nebyla tato možnost aktivně zkoumána.