Klasifikace psů podle plemena

Moravec Vojtěch

LS 2020

1 Úvod

V tomto projektu do předmětu Metody Analýzy Dat 4 se budeme zabývat kategorizací, neboli klasifikací obrazových dat. Přesněji se zaměříme na kategorizaci psů podle plemen. Toto téma jsme zvolili podle úkolu ze stránky Kaggle, který můžeme nalézt na adrese `https://www.kaggle.com/c/dog-breed-identification/overview/description`. Cílem tohoto úkolu je pomocí trénovacích dat naučit klasifikátor, který bude následně schopen klasifikovat obrázky psů do 120 vybraných plemen. Jelikož se jedná o obrazová data a klasifikaci, tak jsme se rozhodli využít konvolučních sítí s mnoha vrstvami, které jsou v současnosti *state of the art* metodou pro klasifikaci a detekci objektů v obrazových datech. V této práci nejprve uvedeme informace o originálních datech a jak jsme je zpracovali. Následně se budeme zabývat samotnou klasifikací a výsledky.

2 Popis datasetu

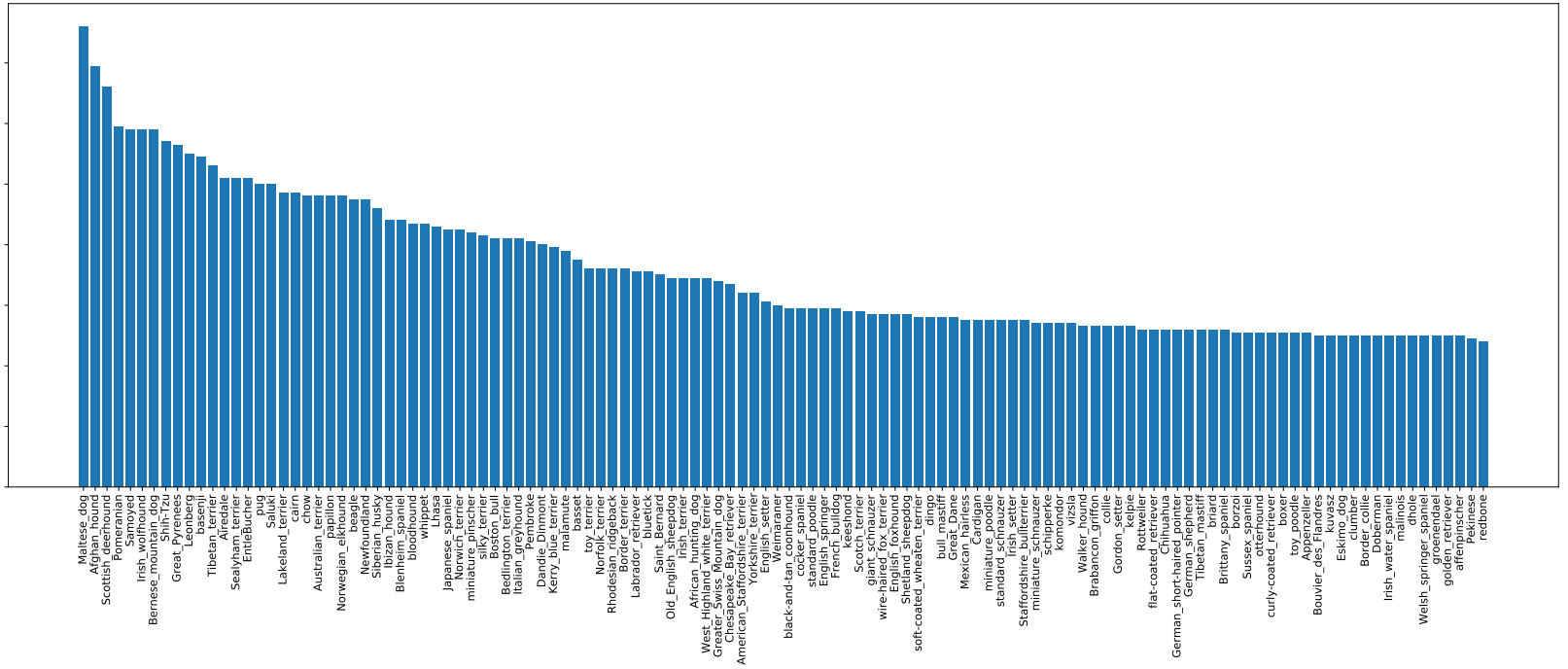
Dataset, který je k dispozici k úkolu na stránce Kaggle (viz URL v předchozí kapitole), vychází z datasetu psů vytvořeného na Standfordské univerzitě. Konkrétně se jedná o *Stanford Dogs Dataset* [1]. Kde tento dataset byl vytvořen z ještě většího datasetu ImageNet [2].

Originální úkol na stránce Kaggle obsahuje jak trénovací, tak i testovací dataset, ale labely jsou k dispozici jen pro trénovací dataset. Label je pravdivá hodnota, která nám říká, do které kategorie obrázek spadá. Konkrétně jaké je plemeno psa na daném obrázku. Absence testovacích labelů, znamená nemožnost vyhodnotit naše modely. Proto jsme se rozhodli využít originální *Stanford Dogs Dataset*, ve kterém nalezneme pro všechny obrázky labely.

Na Obrázku 1 nalezneme četnosti jednotlivých psích plemen v trénovacím datasetu. Ihned si všimneme, že četnost všech plemen je stejná a jedná se o 100 obrázků na jedno plemeno. Velikost celého trénovacího datasetu je tedy 12000 obrázků. Četnosti v testovacím datasetu se již liší a najdeme je na Obrázku 2.

Počet 100 obrázků na jedno plemeno je dosti malý, vzhledem k tomu, že chceme kategorizovat celkem 120 plemen. V této práci vyzkoušíme několik strategií, jak se s tímto vypořádat. Nejprve vybereme pouze 20 plemen, které jsou nejčastější v testovacím datasetu. Poté také vyzkoušíme umělé zvětšení datasetu pomocí augmentace dat a hlavně se zaměříme na transfer learning.

Obrázek 1: Četnost jednotlivých plemen v trénovacím datasetu



Obrázek 2: Četnost jednotlivých plemen v testovacím datasetu

2.1 Příprava datasetu

V *Stanford Dogs Dataset* jsou pro nás důležité soubory `images.tar` a `lists.tar`. První archív obsahuje po rozbalení obrázky psů, rozdělených do podsložek podle plemen. Trénovací i testovací data jednoho plemene jsou tedy uloženy v jedné složce. Druhý archív obsahuje binární soubory v MATLAB formátu. Konkrétě se jedná o `train_list.mat`, resp `test_list.mat`, tyto soubory obsahují cesty k trénovacím, resp. testovacím obrázkům a zároveň jejich labely. Již upravené soubory budou k dispozici v přilohách této práce, jedná se o:

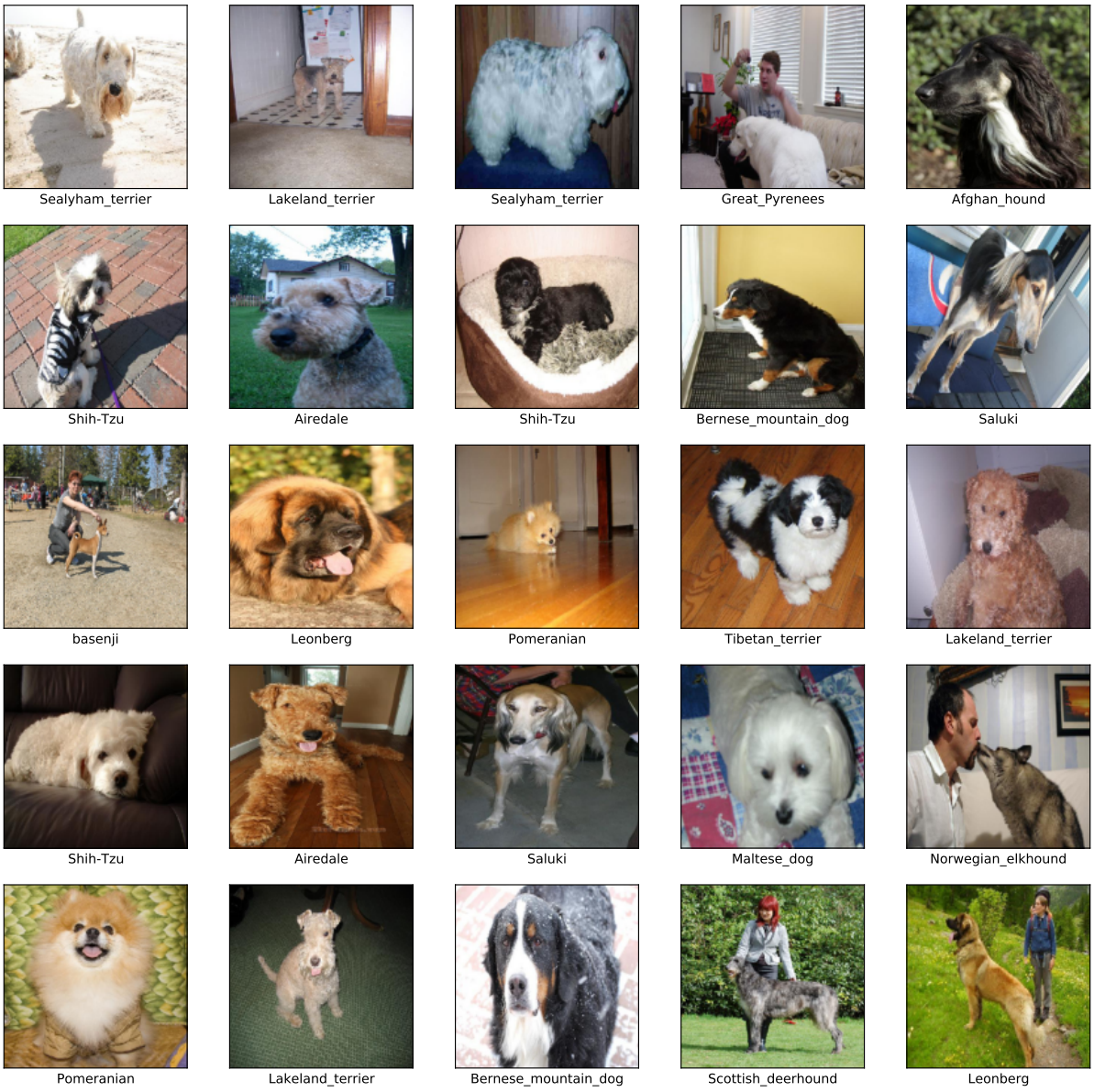
- `classes.csv` - Seznam všech tříd a příslušících číselných labelů
- `train.csv` - Seznam obrázků, které jsou využity k trénování, spolu s labely
- `test.csv` - Seznam obrázků, které jsou využity k testování, spolu s labely

Co se týče předzpracování dat, před samotným učením, tak my jsme provedli pouze sjednocení velikosti obrázků a normalizaci. Obrázky, ať už v trénovací nebo testovací množině, měli různé rozměry. Konvoluční síť ma na vstupu definován rozměr vstupních dat, proto museli byt rozměry obrázků sjednoceny. Rozhodli jsme se použít rozměr $300 \times 300 \times 3$, neboli obraz 300 pixelů vysoký a široký se třemi barevnými kanály. Obrázky jsme nepřeváděli do stupní šedi, neboť si myslíme, že barevná složka je poměrně důležitým faktorem, při klasifikaci psího plemena.

Normalizace byla provedena z rozsahu pixelů $[0; 255]$ na $[0; 1]$. Celé načtení datasetu, spolu se změnou velikosti a normalizací bylo provedeno následující funkcí, za pomocí dvou funkcí `load_img` a `img_to_array` z Python knihovny Keras [3].

```
IMG_DIM = 300
def load_image_dataset(folder, paths):
    dataset = []
    count = len(paths)
    for i in range(count):
        imagePath = os.path.join(folder, paths[i])
        image = load_img(imagePath,target_size=(IMG_DIM,IMG_DIM))
        dataset.append(img_to_array(image,dtype=np.float32) / 255)
    return np.asarray(dataset)
```

Ukázku obrázků z datasetu můžeme vidět na Obrázku 3. Pod každým obrázkem nalezenem název třídy, do které spadá, tedy psí plemeno.

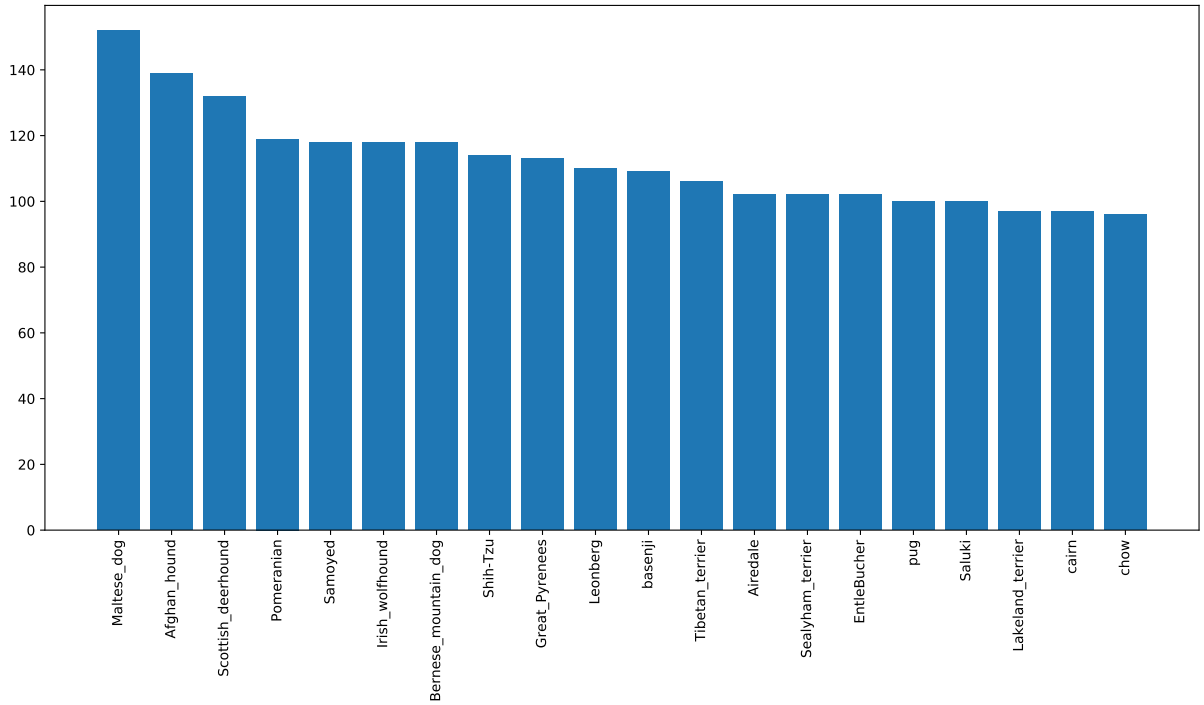


Obrázek 3: Ukázka z datasetu

3 Klasifikace 20 plemen

V této kapitole se zaměříme na klasifikaci 20 nejčastějších plemen v testovacím datasetu, jejich četnosti můžeme vidět na Obrázku 4. V trénacím datasetu se stále nachází 100 obrázků na jedno plemeno, a proto je velikost trénovací sady 2000 obrázků. Z důvodu malé velikosti, jsme vzali pouze 10% této sady na validační dataset.

První model, který jsme vyzkoušeli a taky největší, který nám dovolili hardwarové limitace paměti, je tento:



Obrázek 4: Četnost 20 nejčetnějších plemen v testovacím datasetu

```
model = keras.Sequential(layers=[
    Input(shape=(IMG_DIM, IMG_DIM, 3), dtype=np.float32), # 300 x 300
    Conv2D(64,kernel_size=(5,5),activation='relu',),
    Conv2D(64,kernel_size=(5,5),activation='relu'),
    Dropout(rate=0.15),
    Conv2D(64,kernel_size=(5,5),activation='relu'),
    MaxPooling2D(pool_size=(2,2)),                          # 150 x 150
    Conv2D(64,kernel_size=(5,5),activation='relu'),
    Conv2D(64,kernel_size=(5,5),activation='relu'),
    MaxPooling2D(pool_size=(2,2)),                          # 75 x 75
    Conv2D(128,kernel_size=(5,5),activation='relu'),
    Dropout(rate=0.15),
    Conv2D(128,kernel_size=(5,5),activation='relu'),
    Conv2D(128,kernel_size=(5,5),activation='relu'),
    MaxPooling2D(pool_size=(2,2)),                          # 37 x 37
    Conv2D(256,kernel_size=(3,3),activation='relu'),
    Conv2D(256,kernel_size=(3,3),activation='relu'),
    MaxPooling2D(pool_size=(2,2)),                          # 18 x 18
    Conv2D(256,kernel_size=(3,3),activation='relu'),
    Conv2D(256,kernel_size=(3,3),activation='relu'),
    MaxPooling2D(pool_size=(2,2)),                          # 9 x 9
    GlobalMaxPooling2D(),
    Dense(units = CLASS_COUNT, activation='softmax')
])
```

V tomto modelu můžeme vidět bloky konvolučních vrstev, následovány MaxPooling2D vrstvou. MaxPooling2D vždy snižuje rozměry výstupu vrstvy dvojnásobně. Tyto rozměry jsou uvedeny v komentaařích na pravé straně. Jako poslední je plně propojená vrstva, kde počet neuronů je roven počtu různých vrstev, v tomto případě tedy 20. Důležitá je aktivační funkce `softmax` v poslední vrstvě, která vlastně generuje pravděpodobnosti jednotlivých tříd. Co se týče ztrátové funkce, tak tento model a spolu s ním všechny modely, které budou uvedeny v této práci, používá *sparse categorical crossentropy*. Tato funkce je zvolena, neboť jednotlivé třídy jsou exkluzivní a pes by měl být pouze jednoho plementa. *Nechceme-li detekovat křížence, což není cílem této práce.* Stejně tak v celé práci je zvolen optimizátor *Adam*, úspěšnost měříme pomocí přesnosti a v základu trénujeme ve 20 epochách.

Výsledky trénování první konvoluční neuronové sítě nalezneme v Tabulce 1. Od 3. epochy se trénovaní zasekne na validační přesnosti 0,0250 a síť již není schopná se naučit nic nového. Takhle to pokračuje až do 20. epochy. Víme, že toto není způsobeno špatným formátem nebo snad zpracováním dat, neboť si následně ukážeme, jakých výsledků dokážeme dosáhnout pomocí transfer learningu. Přikládáme se tedy k názoru, že nemáme dostatečný počet trénovacích dat.

Jako záchrana se nabízí umělá augmentace dat, například pomocí `ImageDataGenerator` z knihovny keras. S tímto generátorem jsme vyzkoušeli dvě strategie. V první strategii jsme generovali obrázky, které mohli být zrcadlově přetočeny podle obou os, zároveň mohlo dojít k rotaci až o 20 stupňů a náhodnému přiblížení či oddálení o 15 %. V druhé strategii jsme navíc přidali horizontalní a vertikální posuny až o 15 %. Bohužel, ani jedna strategie augmentace dat nevedla ke zlepšení a model stále nebyl schopen klasifikovat psi podle plemen. Následně jsme vyzkoušeli vynechat `Dropout` vrstvy, či zjednodušit architekturu sítě, ale nic nevedlo ke zlešení.

Epocha	Tr. ztráta	Tr. přesnost	Val. ztráta	Val. přesnost
1	3,0042	0,0428	2,9966	0,0400
2	2,9961	0,0511	2,9965	0,0400
3	2,9960	0,0483	2,9972	0,0250
4	2,9959	0,0528	2,9975	0,0250
5	2,9959	0,0528	2,9978	0,0250

Tabulka 1: Trénování vlastního modelu

3.1 Využití transfer learningu

Transfer learning (*volně přeloženo jako přenesené učení*) je metoda strojového učení, která se snaží využit či přenést již naučené znalosti v rámci jednoho problému, k řešení jiného problému, který mu je podobný. V našem případě, víme že obrázky psů pochází z datasetu ImageNet [2] a že Keras nabízí modely přímo naučené na tomto datasetu. Můžeme tedy

využít znalostí, které se síť naučila na celém tomto datasetu a využít je pouze ke klasifikaci psích plemen.

Tato technika je velice výhodná v případech, kdy nemáme dostatečný počet trénovacích dat k naučení celé sítě, tak jak je tomu v našem případě. Zároveň je délka trénování mnohem kratší, neboť váhy většiny vrstev jsou zmraženy a učíme pouze poslední vrtvu či vrstvy, které nahradíme v přeneseném modelu. Modely, které jsme se rozhodli vyzkoušet nalezneme v Tabulce 2.

Model	Počet parametrů	Počet vrstev
VGG19 [4]	143 667 240	26
Xception [5]	22 910 480	126
InceptionV3 [6]	23 851 784	159
InceptionResNetV2 [7]	5 873 736	572

Tabulka 2: Převzaté modely konvolučních neuronových sítí

U těchto přenesených modelů jsme tedy využili váhy, které se naučili na ImageNetu a nastavili dimenzi vstupních dat na již zmíněných $300 \times 300 \times 3$. U všech modelů jsme nevyužili originální poslední plně propojené vrstvy a nastavili jsme GlobalMaxPooling2D jako poslední vrstvu převzatého modulu. Na tuto vrstvu jsme následně napojili naší plně propojenou vrstvu s počtem neuronů rovným počtu tříd. Výsledky trénování a testování pro zkombinované modely nalezneme v Tabulce 3.

Model	Trénovací přesnost	Testovací přesnost
VGG19	0,7289	0,5040
Xception	1,0000	0,9626
InceptionV3	1,0000	0,9599
InceptionResNetV2	0,9994	0,9603

Tabulka 3: Výsledky klasifikace pro 20 plemen s použitím transfer learningu

V této tabulce si všimneme, že oba modely Xception a InceptionV3 dosáhli perfektní trénovací přesnosti, což znamená, že se naučili přesně na trénovací data dochází k *overfittingu*. Toto samo o sobě nemusí být dobré, neboť se může stát, že síť nebude schopna klasifikovat nic jiného než trénovací data. Avšak v našem případě vidíme, že přesnost je na testovací sadě velmi vysoká, kolem 96 %. Nejhůře dopadl model VGG19, který je také nejjednoduší. Jeho výsledky jsme se snažili vylepšít augmentací dat, ale ani zde augmentace nepomohla. Při použití první strategie augmentace došlo k malému zlepšení přesnosti na 0,5150, tohoto výsledku jsme dosáhli až po 40 trénovacích epochách.

Dále jsme vyzkoušeli více plně propojených vrstev za poslední MaxPooling2D vrstvou a před finální detekční vrstvou, výsledky jsou uvedeny Tabulce 4. Tyto přidané `Dense` vrstvy měli každá 256 neuronů a aktivační funkci ReLU. S těmito vrstvami můžeme pozorovat

zlepšení trénovací přesnosti z 0,7289 až na 0,8900 při použití 3 vrstev navíc. Oproti tomu testovací přesnost dosáhla nejlepší hodnoty 0,5241 při dvou přidaných vrstvách.

Počet `Dense` vrstev	Trénovací přesnost	Testovací přesnost
1	0,7933	0,5209
2	0,8700	0,5241
3	0,8900	0,5076

Tabulka 4: VGG19 - Vliv plně propojených vrstev na výsledek klasifikace

Reference

[1] A. Khosla, N. Jayadevaprakash, B. Yao, and L. Fei-Fei, "Novel dataset for fine-grained image categorization," in *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, (Colorado Springs, CO), June 2011.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[3] F. Chollet *et al.*, "Keras." `https://keras.io`, 2015.

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014.

[5] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," 2016.

[6] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," 2015.

[7] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," 2016.