Transactions on Machine Learning and Artificial Intelligence - Vol. 10, No. 3
Publication Date: June, 25, 2022
DOI:10.14738/tmlai.103.12245.
Hammound, A. M., & Ghandour, A. R. (2022). Masking the Backgrounds to Produce Object-Focused Images and Comparing that Augmentation Method to Other Methods. Transactions on Machine Learning and Artificial Intelligence, 10(3). 01-24.

Masking the Backgrounds to Produce Object-Focused Images and Comparing that Augmentation Method to Other Methods

Ahmad M. Hammoud
Dept. of Computer Science, Global University, Beirut, Lebanon

Ahmad R. Ghandour
Dept. of Computer Science, American University of Beirut, Beirut, Lebanon

ABSTRACT

Image augmentation is a very powerful method to expand existing image datasets. This paper presents a novel method for creating a variation of existing images, called Object-Focused Image (OFI). This is when an image includes only the labeled object and everything else is made white. This paper elaborates on the OFI approach, explores its efficiency, and compares the validation accuracy of 780 notebooks. The presented testbed makes use of a subset of ImageNet Dataset (8,000 images of 14 classes) and incorporates all available models in Keras. These 26 models are tested before augmentation and after applying 9 different categories of augmentation methods. Each of these 260 notebooks is tested in 3 different scenarios: scenario A (ImageNet weights are not used and network layers are trainable), scenario B (ImageNet weights are used and network layers are trainable) and scenario C (ImageNet weights are used and network layers are not trainable). The experiments presented in this paper show that using OFI images along with the original images can be better than other augmentation methods in 16.4% of the cases. It was also shown that OFI method could help some models learn although they could not learn when other augmentation methods were applied. The conducted experiments also proved that the Kernel filters and the color space transformations are among the best data augmentation methods.

Keywords: Machine Learning; Computer Vision; Object Classification; Image Augmentation.

INTRODUCTION

Nowadays, CNNs (Convolutional Neural Networks) are among the most common tools used for image classification. For ML (Machine Learning) problems such as image classification, the size of the training image set is very important to build high-accuracy classifiers. Many of the machine learning models are extremely data hungry. However, the amount of the training data available is, unfortunately, rarely abundant. Fortunately, image augmentation is a powerful method that can be used by computer-vision engineers to expand their image datasets. As the popularity of CNNs grows, so does the interest in data augmentation. Although augmented data sets are sometimes artificial, they are very similar to the original data sets. Thus, augmentation can make a network learn more useful representations because of the increase of training data. In this paper, the researchers propose a method (called OFI) to produce new images from existing ones and therefore augment data. Several experiments were conducted to validate that

the model could benefit further from using the mixed data set of original and augmented images. This paper compares 780 notebooks with the same network design but different settings. The researchers not only aim at showing the efficiency of their proposed approach, but they also provide a comparison for 26 models available in Keras using 10 augmentation categories in 3 different scenarios.

This paper tries to answer the following questions:

- Is the proposed OFI method efficient enough?
- Will using OFI method lead to a better object classification accuracy, compared to other augmentation methods?
- In which scenarios does the OFI method work better than other augmentation methods?

In this paper, the researchers introduce the idea of the OFI method and compare it to almost all kinds of other augmentation methods.

The experiments shown in this paper prove that the proposed method can lead to results that are more accurate than those of other methods in different scenarios.

The OFI method works as follows: it receives an image that includes many objects, including the animal that needs to be classified. The algorithm will then use a special tool to remove all objects in the image and keep just the animal, which is the prominent object in the foreground. Around 8,000 images are processed this way to generate the OFI version of them. Then, the generated images are used to augment the images at hand.

Nine well-known augmentation methods have been used to generate augmented images. These generated images have been used along with the original ones to check the accuracy of the classifiers. Then, the accuracy of the classifier that used a mix of OFI and original images has been compared to the accuracy of other classifiers that used other image augmentation methods.

Because there are many settings that might play an important role in the results, all the used augmentation methods were compared to each other in the following three scenarios:

- Scenario A: ImageNet weights are not used and network layers are trainable.
- Scenario B: ImageNet weights are used and network layers are trainable.
- Scenario C: ImageNet weights are used and network layers are not trainable.

The total count of tested combinations is 780. This number results from applying 10 different augmentation cases (before augmentation and after using 9 augmentation methods) on 26 models (VGG, ResNet, …) in three different scenarios (A, B, and C).

The conducted experiments show that in 16.4% of the cases (128 combinations out of 780), the OFI method leads to a higher validation accuracy, depending on the scenario and the used model.

Other than being a successful augmentation method, the OFI method comes with another advantage. When ImageNet weights were not used, using the OFI method enabled some classifiers to learn. The same model was not able to learn when other augmentation methods

were used. Concretely, the two models InceptionV3 and EfficientNetB3 could not learn after 150 epochs, even when the set of supplied images was augmented using the eight augmentation methods (all except OFI). When supported with the OFI method, the InceptionV3 model could achieve a validation accuracy above 60%, while the EfficientNetB3 model reached more than 78%. Both models could benefit from using the OFI method, while they could not make any progress with other methods after 150 epochs.

Related Work

Deep learning has been widely used in many domains [1-3]. Using that technology to implement image recognition is an emerging topic in AI (artificial intelligence) [4]. Data augmentation is a method that can be used to produce new images based on images in a dataset and improve generalization of ML models [5]. That vital technology is nowadays applied in many different fields [6-8], including object detection [9] and semi-supervised learning [10].

Augmentation is a vital method, not only for cases where datasets are small but also for any size of dataset. Indeed, even those models trained on enormous datasets such as ImageNet [11] could benefit from data augmentation.

Although data augmentation is one of the fundamental building blocks for almost all state-of-the-art image recognition implementations [12-14], augmentation strategies require extensive selection and tuning [15] because they may have a large impact on the model performance.

Recently, the topic of image data augmentation has drawn large attention from the ML research community. Fig. 1 shows the number of researches that have been conducted since 2015 in the field of image data augmentation within the computer science field using the Scopus database [16]. It is so obvious that the research works in image data augmentation are exponentially increasing. In 2020, the number was 1269 research works, which is 24 times larger than what has been carried out in 2015 (52 research papers). This jump in the number of researches is due to the effectiveness of data augmentation in producing accurate results.

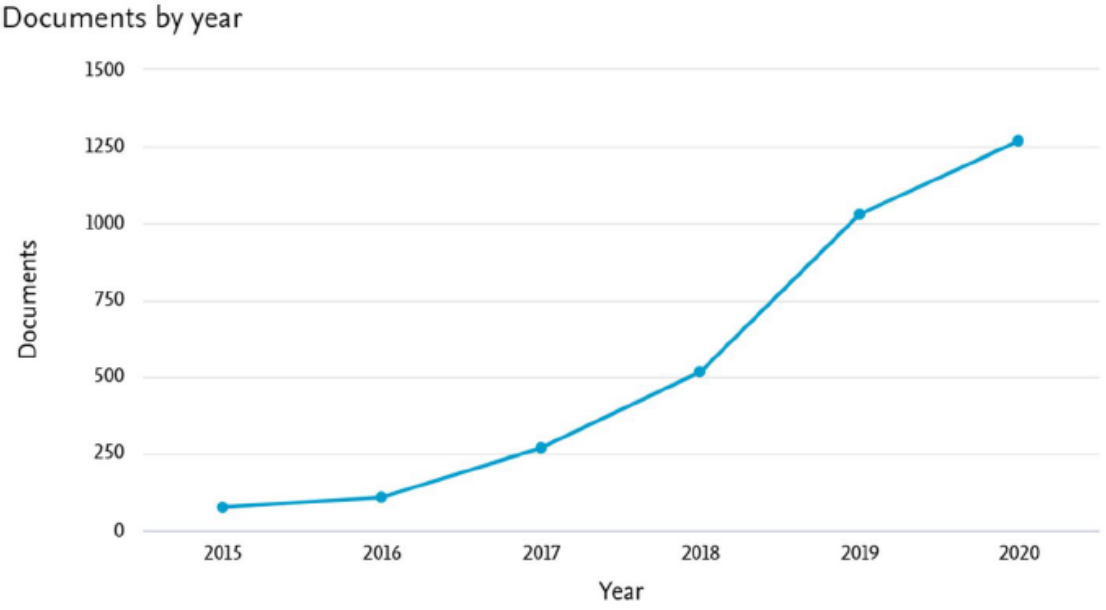


Fig. 1: Image data augmentation research works between 2015 and 2020

URL: http://dx.doi.org/10.14738/tmlai.103.12245

Without data augmentation, deep neural networks may suffer from the lack of generalization [17] and adversarial vulnerability [18]. Data augmentation has been considered a solution to the overfitting problem [19]. Thus, the reasons why image data augmentation is extremely needed can be summarized as follows [16]:

1. When compared with regular data collection and label annotation, data augmentation is an inexpensive methodology.
2. Since it is generated from ground-truth data by nature, it is extremely accurate.
3. It is controllable, which results in generation of balanced data.
4. It helps overcome the problem of overfitting.
5. It helps achieve higher testing accuracies.

Since data augmentation is of paramount importance, very useful augmentation methods have been applied to modern deep CNNs. They were proven to be very efficient and demonstrated the added value of data augmentation.

As proposed by Shorten and Khoshgoftaar, Fig. 2 summarizes those numerous available data augmentation methods [6]. Each line with a shape in Fig. 2 tells which data augmentation method is used by the corresponding meta-learning scheme. For instance, meta-learning using mixing images is covered in smart augmentation.

As shown in Fig. 2, image data augmentation consists of two categories: basic image manipulations and deep learning approaches.

Basic Image Manipulations

Basic image manipulations include Kernel filters, geometric transformations, random erasing, color space transformations, and mixing images [6].

Flipping is one of the most popular geometric transforms for data augmentation [6]. Flipping reflects an image around its horizontal or vertical axis (or both) [20]. Horizontal flipping and random cropping [21] are used the most among other data augmentation methods.

Cutout [19] improves the robustness of network by randomly removing contiguous sections of images.

GridMask [22] and Hide-and-Seek [23] can be considered as upgraded versions of Cutout.
Random Erasing [24] selects a random rectangle region in an image and replaces its pixels with random values.

There are other geometric transforms such as translating and rotating [25, 26]. However, translation often adds noise [27]. Noise injection inserts random values into images. This augmentation method has been deeply studied in [28].

Mixing images is a basic manipulation method that either transforms images and mixes them together [29] or averages pixel values between images [30].

MSDA (Mixed Sample Data Augmentation) has attracted increasing attention lately [31]. Random image cropping and patching is a known method that randomly crops four images and then patches them in order to create a new training image [32].

CutMix [33] is a variation of Cutout and Mixup.
FMix [34], Puzzle Mix [35], and Attentive Cut-Mix [36] aim to capture the most important region of an image and paste it on another one.

Color space transformations can also be used. Color distributions can be jittered, and saturation, contrast, and brightness can be adjusted to generate new images [21, 25].

Kernel filters sharpen and blur images. A new kernel filter called PatchShuffle has also been applied with success [37]. It randomly swaps the matrix values in the window.

Lately, a novel data augmentation strategy named Cut-Thumbnail has been proposed [31]. It tries to improve the shape bias of the network. Their experiments showed that it works better than other augmentation strategies across classification, fine-grained image classification, and object detection.

Yingwei Zhou proposed a new approach (called «slot-based image augmentation») that can be used to enrich the dataset with extra foreground and background combinations. This proposed augmentation system has extra flexibility adapting to various scenarios [38].

Ghiasi et al. proposed a Copy-Paste augmentation method, through which they randomly paste objects onto an image [39]. They could demonstrate that Copy-Paste can lead to significant improvements on the LVIS benchmark.

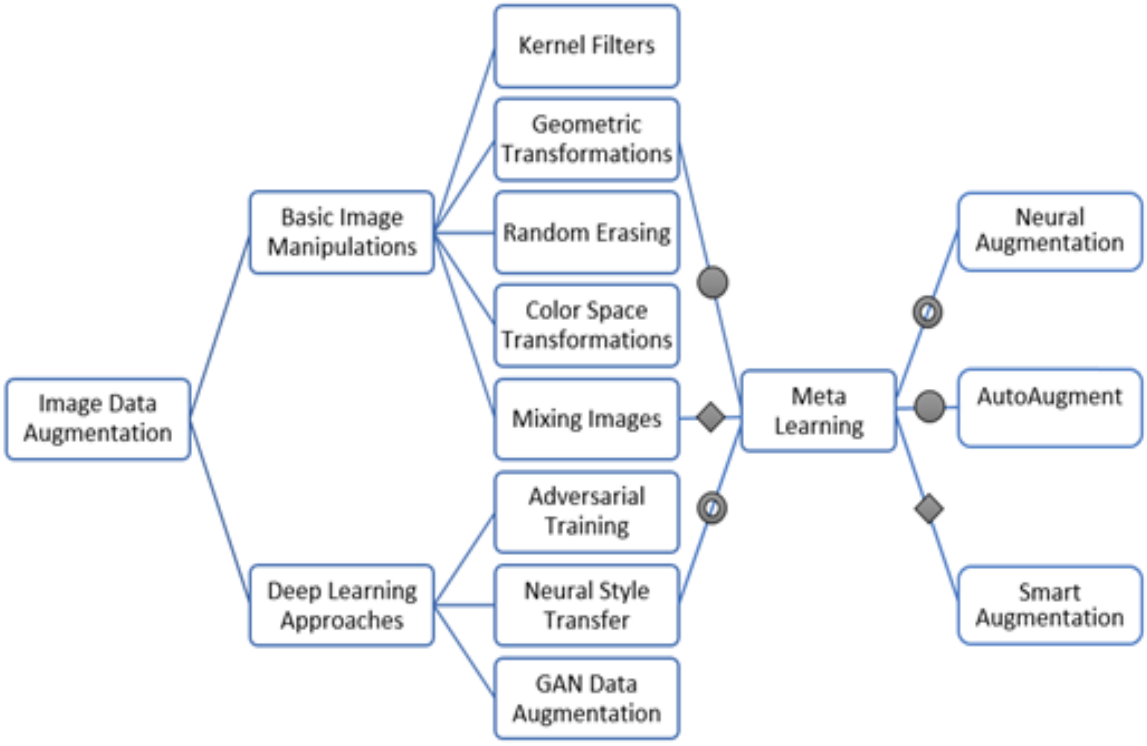


Fig. 2: A taxonomy of image data augmentations

URL: http://dx.doi.org/10.14738/tmlai.103.12245

Deep Learning Manipulations

Deep learning image data augmentation includes three main categories:

- The first category is GAN (Generative Adversarial Networks) [40].
- The second category is Neural Style Transfer [41].
- The third category is meta metric learning [42].

The third category includes three models, which are:

- Neural Augmentation
- Auto Augment
- Smart Augmentation.

Auto Augment (AA) is originally presented by Cubuk and et al. [43]. They created a software package (called AA) that automatically searches for the right augmentation policies for a specified picture. They evaluated many automatic augmentation methods and compared their proposed AA model with other related works such as GAN and their achieved result was very promising.

SA (Smart Augmentation) is another type of meta metric learning under deep learning data augmentation. It was originally presented by Lemley and et al. [44]. They used a modern regression approach to boost training precision and minimize overfitting. SA showed strong and substantial improvements in precision on all datasets.

Cubuk et al. also proposed RA (RandAugment) [45]. Compared to the previous methods, it is much simpler, but slightly cheaper.

AugMix [29] tries to improve model robustness. AWS (Augmentationwise Sharing for AutoAugment) is the second most expensive method [46].
Another approach that is called PBA (Population-based Augmentation) [47] can learn the augmentation policy online, as the training goes.

Adv. AA (Adversarial AutoAugment) [48] is another method that learns the augmentation policy online. Adv. AA is slightly cheaper, and it is based on batch augmentation [49].
Fast AA (Fast AutoAugment) [50] is based on AA and it is the cheapest method of the learned ones.

Another method that is based on multiple parallel workers is called OHL (Online Hyper-Parameter Learning for Auto-Augmentation) [51].

Recently, researchers presented TrivialAugment, that outperforms previous methods [5].
Sometimes, augmented data sets are artificial. As a matter of fact, increasing the number of training examples artificially helps reducing overfitting and improves generalization. According to [52], synthetic datasets have been widely used to learn large CNN models since they are scalable and cheap alternatives to the manual process of annotating images. In their research work, Gupta et al. introduced a fast engine to generate synthetic images of text in clutter. Their engine overlays synthetic text to existing background images in a natural way. They also use the synthetic images to train a FCRN (Fully-Convolutional Regression Network). According to [53], Dwibedi et al. introduced a new approach for generating large annotated instance datasets with minimal effort. They could also generate data that gives competitive

performance on real data. Their approach could outperform other synthesis approaches. When combined with real images, their approach improves relative performance by more than 21% on benchmark datasets.

Some researchers proceeded beyond classical approaches for data augmentation. Dvornik et al. leveraged segmentation annotations to increase the number of object instances present on training data [54].

Tripathi et al. presented a task-aware approach to synthetic data generation [55]. Their framework employs a trainable synthesizer network that is optimized to produce meaningful training samples. Their synthesizer generates realistic data because it is paired with a discriminator trained on real-world images.

Deep learning manipulations are very useful, and it is expected that deep learning for image data augmentation will keep proving its efficiency.

THE OFI METHOD

Computer-vision engineers use images to train their models. These images include the object that needs to be correctly classified along with other objects that they do not want to train the model on.

Before training a model to classify images, the following questions should be addressed:

- Why, for example, should we train the model on an American egret with bottles next to it, but when the training ends, we'll ask the model to classify an image without any bottle?
- Once the model is trained, it will classify an image of an American egret with a tree, a rock, or a river behind it, so why train the model on an animal with bottles, while real-life images might include millions of other things?
- Here, the OFI method comes to light. For every image in the training set, another image is generated where the labeled object is kept untouched and everything in the background is made white as shown in Fig. 3.
- When such an image is supplied as a part of the training set, the model is instructed to focus only on the labeled object. There will be no other objects for the network to be trained on.

URL: http://dx.doi.org/10.14738/tmlai.103.12245

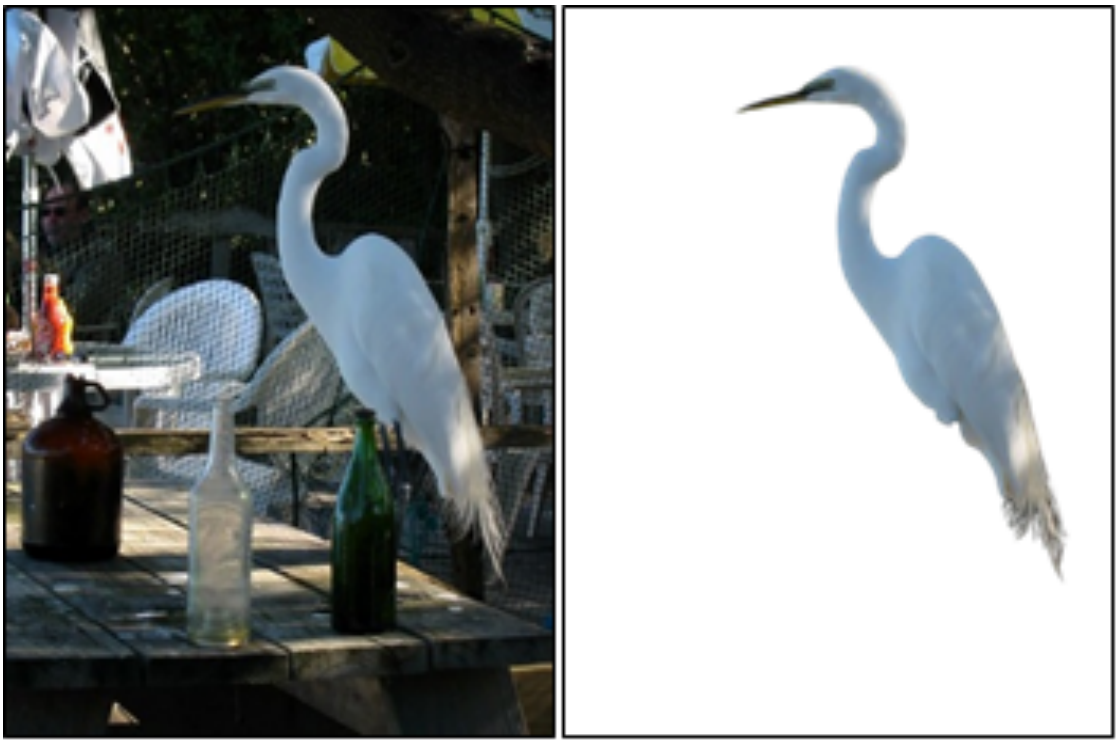


Fig. 3: The original image of an American egret and its OFI version

Experiments and Analysis

Several experiments have been conducted to validate the OFI method. These experiments are detailed in the below sections. All used notebooks can be downloaded using the following URL: http://eversmartservices.com/nb.htm

The notebooks are named as follows: NUM_MOD_AUG$.ipynb, where:

- NUM is a 3-digit serial number that is used as the ID of the notebook.
- MOD is the name of the model that is used by the notebook. This can be one of the 26 models listed in Table 1.
- AUG is a 3-character term that refers to the used augmentation method. These methods are listed in Table 3.
- The dollar sign represents 1 digit that refers to the used scenario. 1 refers to Scenario A, 2 refers to Scenario B, and C refers to Scenario C.

Used Models

To make sure most of the combinations of models and augmentation methods have been tested, this research work included 26 models and applied 10 different categories of augmentation to them in the three afore-mentioned scenarios.
The tested models are listed in Table 1.

Table 1. Used Models

1 Xception	8 ResNet101V2	15 DenseNet169	22 EfficientNetB3
2 VGG16	9 ResNet152V2	16 DenseNet201	23 EfficientNetB4
3 VGG19	10 InceptionV3	17 NASNetMobile	24 EfficientNetB5
4 ResNet50	11 InceptionResNetV2	18 NASNetLarge	25 EfficientNetB6
5 ResNet101	12 MobileNet	19 EfficientNetB0	26 EfficientNetB7
6 ResNet152	13 MobileNetV2	20 EfficientNetB1	
7 ResNet50V2	14 DenseNet121	21 EfficientNetB2	

The goal behind checking all these models is to determine which model would produce the best results in terms of validation accuracy.

As presented in Table 2, each model is tested in 10 different cases. In each case, a selected category of augmentation is applied to the model. Table 2 represents the output that result from one of the three scenarios.

Table 2. Used Augmentation Methods

Id	Model	Augmentation Category									
		NON	GEO	COL	KER	RND	MIX	WTH	GAN	STL	OFI
1	Xception	-	-	-	-	-	-	-	-	-	-
…	…	…	…	…	…	…	…	…	…	…	…
26	EfficientNetB7	-	-	-	-	-	-	-	-	-	-

All the modules presented in Table 1 will be tested using the augmentation categories presented in Table 3.

The dashes in Table 2 will be replaced by the validation accuracy that resulted from applying the augmentation method in the column to the model in the row. For example, the last cell in Table 2 (the rightmost cell in the bottom) will show the percentage of the validation accuracy of the notebook that applies the OFI augmentation method to the EfficientNetB7 model.

The tool that has been used to produce the OFI images is remove.bg. It uses AI technology to detect foreground layers and separate them from the background.

When there are multiple objects in the scene, the tool will remove all of them and keep only the prominent object that is in the foreground. The tool has an API that automates background separation.

URL: http://dx.doi.org/10.14738/tmlai.103.12245

Table 3. Applied Augmentation and Used Tools

#	Code	Applied Augmentation	Tool used to generate augmented images
1	NON	None of the below methods is used	None
2	GEO	**Geometric transformations** Flipping / Reflection Cropping Rotation Translation Noise injection Scaling / Zooming Shearing	imgaug.readthedocs.io
3	COL	**Color space transformations** Contrast shift Brightness shift Changing colors Apply CLAHE to all channels Channel Shuffle	imgaug.readthedocs.io
4	KER	**Kernel filters** Sharpening Blurring	imgaug.readthedocs.io
5	RND	**Random erasing** cutout dropout coarse dropout drop random channels	imgaug.readthedocs.io
6	MIX	**Mixing images**	opencv.org
7	WTH	**Weather transformations** Rain Fast Snowy Landscape Clouds	imgaug.readthedocs.io
8	GAN	**GAN-based augmentation**	github.com/giannisdaras/ylg
9	STL	**Neural style transfer**	github.com/cysmith/neural-style-tf
10	OFI	**OFI augmentation method**	remove.bg

The different tools presented in Table 3 have been used to prepare the 9 folders that include the augmented images. Then, every notebook is designed to read from a specific folder where the augmented images have been stored. When the applied augmentation includes many transformations, these different transformations are evenly applied to the images. For example, there are 2 kernel filters transformations in Table 3, namely, sharpening and blurring. This means that sharpening will be applied to half of the available images and blurring will be applied to the other half. Thus, sharpening will be applied to every other image and the same applies to blurring.

All the combinations of notebooks that will result from applying all the augmentation categories in Table 2 to the 26 models in Table 1 will be tested in three afore-mentioned scenarios.

The above three scenarios have been chosen in order to detect how much the model and the applied augmentation method would be affected by the ImageNet weights and the trainability

of the network. In scenario A, the number of epochs is 150 because the ImageNet weights are not used and therefore more epochs are needed, while the other two scenarios include only 50 epochs because the ImageNet weights are taken into consideration.

The total number of tested notebooks in one scenario is 260 (26 models x 10 augmentation cases). Since each of these notebooks will be tested in the above three scenarios, the total number of notebooks is 260 x 3 = 780.

Network Design

All notebooks are almost the same except that they differ in some settings and the folder to read the images from. Below is a sample of those settings that can be found as the first section of every notebook.

```
============================================================
#settings start
============================================================
NetWeights = None   # or "ImageNet"       … this depends on the scenario
isTrainable = True    # or False          … this depends on the scenario
numberOfClass = 14  # 14 classes          … we have 14 different animals
#Xception in the next line can be any of the 26 models
#it can be VGG16, VGG19, ResNet50, …
NET = Xception(include_top = True, weights=NetWeights)
============================================================
#settings end
============================================================
```

To test all the above models, Tensorflow (version 2.3.0) has been used through Google's cloud (colab.research.google.com). Except for the settings that are between #SETTINGS START and #SETTINGS START, all the code is the same for all the 780 notebooks. The complete network design and settings are presented in appendix A.

RESULTS

Table 4 shows the validation accuracy that resulted from applying the different categories of augmentation to the 26 models in scenario A, where ImageNet weights are not used and network layers are trainable. The shaded cells indicate that the same notebook has been run for three times without being able to learn. After 150 epochs, the validation accuracy could not exceed 15 percent. In these cases, 150 epochs were not enough to let the model start learning, because ImageNet weights were not used. If any knowledge transfer has been applied, the notebook would no doubt have benefited from that.

URL: http://dx.doi.org/10.14738/tmlai.103.12245

Table 4. Results of Scenario A

Id	Model	NON	GEO	COL	KER	RND	MIX	WTH	GAN	STL	OFI	Score
1	Xception	60.8	85.1	95.2	94.7	96.9	81	94.8	65.6	99.4	82.4	2
2	VGG16											0
3	VGG19											0
4	ResNet50											0
5	ResNet101											0
6	ResNet152											0
7	ResNet50V2	58.5	73.4	92.6	97.9	90.9	79	76.4	56.8	91.7	71.1	1
8	ResNet101V2	37.5	81.9	92.1	95.1	90.6	74.2	85.6	57.5	60.3	74.1	2
9	ResNet152V2	56.6	71.5	94.2	94.2		72.5	85	54.6	91.5	73.2	4
10	InceptionV3										60.8	8
11		61.2	79	81.9	94.3	92.3	72.7	80.9	54	92.2	71.3	1
12	MobileNet	66.6	85	95.4	97.8	97	84.2	91.9	65.7	95.7	78.2	1
13	MobileNetV2	54.4	71.1	82.3	90.8	86.8	69.6	79.4	50.5	92.3	72.2	3
14	DenseNet121	68	89	93.4	97.2	95.5	84.3	92.6	67.3	99.2	78.1	1
15	DenseNet169	66.4	86.4	95	97.7	95.3	85.6	95.8	66.1	89.7	81	1
16	DenseNet201	72	87.6	94.7	94.4	96.5	79.9	83.8	65.6	99.1	80.7	2
17	NASNetMobile	41.6	80.7	89.5	82.9	86.2	79.3	87.4	58.8	87.9	73.1	1
18	NASNetLarge											0
19	EfficientNetB0	55.6	73	90.4	96.6	94.6	73.8	90.2	57	98.2	77.5	3
20	EfficientNetB1	55.8	75.8	87.6	96.4	81.4	62.6	75.1	57.1	95.1	73	2
21	EfficientNetB2	42.8	30	90.8	95.9	17	72.5	88.2	54.3	95.5	73.4	4
22	EfficientNetB3	55.4									78.6	8
23	EfficientNetB4											0
24	EfficientNetB5											0
25	EfficientNetB6											0
26	EfficientNetB7											0
	TOTAL	853.2	1,069.5	1,275.1	1,325.9	1,121	1,071.2	1,207.1	8,30.9	1,287.8	1,198.7	44

When the number inside a cell in Table 4 is highlighted, it means that the validation accuracy is below 50. The first row in Table 4 shows that the validation accuracy of the Xception model before applying any augmentation method is 60.8%. The accuracy improves using different augmentation methods. The best among these for the Xception model in Scenario A is STL (neural style transfer) where the accuracy is almost 100%. Although the OFI method could not lead to a higher validation accuracy, it could increase the accuracy of the model and achieved 82.4% compared to the 60.8%, which was the accuracy of the model before augmenting the data.

The last column in Table 4 (Score) indicates how many times the OFI method could achieve a higher validation accuracy. Thus, the score is 2 in the first row because OFI could achieve a higher validation accuracy twice. The OFI validation accuracy is 82.4, which is higher than that of MIX (81) and GAN (65.6). The score of 2 means that OFI led to a better validation accuracy, compared to the other eight augmentation methods (GEO, COL, KER, RND, MIX, WTH, GAN, and STL)

As per the rows from 2 to 6 in Table 4, the models VGG and ResNet could not learn although their notebooks have been run 3 times. This is because ImageNet weights are not used. In these rows, the score is zero. In rows 7, 8, and 9 of Table 4, the OFI method scored 1, 2, and 4 points respectively.

In row 10, the OFI method is the only augmentation method that worked. This is an important characteristic of the OFI method. It seems that OFI is the reason why the notebook could learn and achieve a validation accuracy of 60.8. None of the other eight augmentation methods that have been used led to the same result or even allowed the classifier to start learning. When applied to some models (like InceptionV3), the OFI method might be able to help the classifier learn, which is not the case with other augmentation methods. In the case of InceptionV3, the model could learn with the help of OFI images although ImageNet weights are not loaded, which might be a remarkable feature of the OFI approach. All the other methods did not help in making the model learn. This capability of the OFI approach is found again in row 22.

The last row in Table 4 calculates the total validation accuracy for each augmentation category. In Scenario A, Kernel filters scored the highest total, which means that this category of augmentation methods can be considered among the best in Scenario A. In general, the OFI method is in the 5th rank in this scenario.

As per the total score, it is 44 in Table 4, which means that the OFI method led to a higher validation accuracy in 44 out of the 260 cases presented in Table 4.

Table 5 shows the validation accuracy of the 26 models in Scenario B, where ImageNet weights are used and network layers are trainable.

Again, the highlighted numbers are below 50 and the shaded cells represent a notebook that was run for 3 times without being able to achieve a validation accuracy above 15%. While the number of epochs in Scenario A is 150, it is 50 in scenarios B and C. Because ImageNet weights are used, the achieved validation accuracy of all models is higher although the number of epochs is lower. Obviously, the number of shaded cells in Table 5 (21) is much lower than their number in Table 4 (118). This is because knowledge transfer has been applied in Scenario B through the use of ImageNet weights and the classifiers could benefit a lot from that.

The total validation accuracy for each augmentation category is calculated and presented in the last row of Table 5. Instead of using the total, the average could have been used and the results would have been the same.

Just like in Scenario A, Kernel filters scored the highest total in Scenario B. This is to say that this category of augmentation methods is among the top methods in both scenarios A and B.
As per the total score in Table 5, the OFI method helped the classifiers achieve higher validation accuracy in 47 cases.

In rows 8, 11, 15, and 25 of Table 5, the OFI method scored 4, which means that it could perform better than half of all other methods in these rows.

URL: http://dx.doi.org/10.14738/tmlai.103.12245

Table 5. Results of Scenario B

Id	Model	NON	GEO	COL	KER	RND	MIX	WTH	GAN	STL	OFI	Score
1	Xception	79.8	94	94.9	89.1	97.5	83.8	92.5	66	98.2	88.3	2
2	VGG16											0
3	VGG19											0
4	ResNet50	57.1	72.7	93.7	94.6	91	76.8	76.1	59.8	92.1	68.4	1
5	ResNet101	33.1	62.2	87.7	85	90.2	70.1	78.6	52.5	90.1	70.6	3
6	ResNet152	61.8	77.7	90.3	92.3	92.5	66.3	84.2	55.8	94	69.9	2
7	ResNet50V2	55	79.8	92.5	97.5	94.6	58.8	81.2	50.2	94.3	66.3	2
8	ResNet101V2	55.4	64.7	79.5	83.4	93.4	64.8	82.6	56.9	69	75.6	4
9	ResNet152V2	50.9	78	85.9	96.9	83.9	70.5	85.2	51.9	82.7	70	1
10	InceptionV3	56.7	80.5	69.7	85.9	71.8	64.6	85.2	54.3	81.4	52.5	0
11		33	84.8	76	93.5	75.9	77.8	89.1	73.1	86.3	78.2	4
12	MobileNet	81.2	89.7	96.3	94.7	91.7	87.7	96.6	72.9	94.5	87	1
13	MobileNetV2	69.1	71.4	87.6	69.7	89	81.1	86.7	64.8	78.2	68.3	1
14	DenseNet121	46.5	88.6	82.2	85.8	82	76.8	82.5	68.6	77.6	75.7	1
15	DenseNet169	62.4	79.7	81.1	71.7	56.9	71	83.5	60.9	81.9	74.1	4
16	DenseNet201	31.8	74.6	72.3	84.4	80.9	73.4	73.6	62.3	91	70.4	1
17	NASNetMobile	64.8	52	85.8	89.9	67.1	78.7	83.5	41.6	89.6	72.4	3
18	NASNetLarge	16.5	45.3	23.2	66.1		45.9	47.3	24.4	31.6	27.5	3
19	EfficientNetB0	84.9	95.6	97.8	98.9	87.5	93.7	96.5	85.2	96.4	93.1	2
20	EfficientNetB1	85.6	95.3	98.9	98.3	98	93.1	98.5	83	98.5	89.8	1
21	EfficientNetB2	86.9	92.8	98.3	94.1	97.7	94.8	98.3	86.5	97.2	94	2
22	EfficientNetB3	85.3	93	97.6	98.2	96.7	93.3	97.5	80.9	97.9	86.9	1
23	EfficientNetB4	84.8	94.7	98.6	99.1	97.5	91.7	97.1	81.6	98.1	92.7	2
24	EfficientNetB5	82.5	89	97.1	96.6	97.6	87.9	97.6	83.8	97.1	88.7	2
25	EfficientNetB6	84.7	92.8	95.3	90.8	85.8	88.6	95.9	81.3	97.1	91.3	4
26	EfficientNetB7	74.9	36.1	94.2	42.6	37.5	71.5	37.8	70.4	31.1	22.6	0
	TOTAL	1,444.9	1,791	1,981.6	2,010	1,859.2	1,778.9	1,935.1	1,502.7	1,947.7	1,686	47

Table 6 shows the results of Scenario C, where ImageNet weights are used and network layers are not trainable. As it is clear in Table 6, there are no shaded cells which means that all notebooks were capable of learning, thanks to two facts:

- First, ImageNet weights are used so the model could benefit from a set of very accurate weights.
- Second, network layers are not trainable, so the used weights are not being changed in a way that misleads the model.

The numbers in the last row of Table 6 indicate that COL has the highest total. Thus, the color space transformations is the category of augmentation methods that has the highest total in

Scenario C, which means that this category is in general one of the finest methods to use in Scenario C.

As for the highlighted numbers, the corresponding notebooks seem to need more epochs in all the three scenarios.

Table 6. Results of Scenario C

Id	Model	NON	GEO	COL	KER	RND	MIX	WTH	GAN	STL	OFI	Score
1	Xception	92.7	95	96.2	95.8	95.3	95.3	95.7	93	94.7	93.5	1
2	VGG16	56	60.2	61.7	60.5	59	56.8	59.5	52.2	57.3	57.2	2
3	VGG19	51.4	58	58.7	57.6	53.3	52.6	56.3	49.3	53.9	54.3	4
4	ResNet50	19.7	21.3	22.4	20.2	19.1	18.9	19.8	17.5	17.8	18.3	2
5	ResNet101	20.4	21.8	23	19.5	19.5	19.5	19.7	19	18.1	20.7	6
6	ResNet152	20.1	20.2	21.8	22.1	19.7	18.5	19.6	19.5	18.4	18	0
7	ResNet50V2	92.4	94	95.3	94.3	93.9	93.3	93.8	90.9	92.6	92.3	1
8	ResNet101V2	92.5	94.6	95.8	95.8	95.3	94.8	94.8	92.5	93.7	93.3	1
9	ResNet152V2	93.3	95.8	96.3	96.2	95.2	94.8	95.5	92.7	94.4	94.3	1
10	InceptionV3	93.2	94.9	96.1	95.6	94.9	94.2	94.9	93.1	94.2	93.5	1
11	InceptionResNetV2	93.9	96.5	97.1	96.6	96.2	95.6	96.2	93.9	95.1	95.1	1
12	MobileNet	96.2	97.3	98.3	97.9	97.3	97.1	97.4	95.1	96.8	96.7	1
13	MobileNetV2	93.7	96.3	96.8	95.8	95.7	93.7	95.4	92.1	94.1	93.5	1
14	DenseNet121	92.7	96.2	96.6	96.3	95.5	94.7	95.9	91.9	94.3	93.6	1
15	DenseNet169	94.3	96.6	96.5	96.7	96.4	95.5	96.5	93.9	95.4	94.6	1
16	DenseNet201	94.3	96.4	97	97.2	96.8	95.8	97.1	93.4	95.3	95	1
17	NASNetMobile	95.3	97.3	97.7	97.4	96.8	96	96.7	94.3	96	95.8	1
18	NASNetLarge	94.6	96.8	97.2	96.8	96.9	96.3	96.3	94.9	96.2	95.7	1
19	EfficientNetB0	93.6	96.3	96.9	96	96.2	94.8	95.8	92	94.1	94.3	2
20	EfficientNetB1	95.1	96.3	96.5	96.2	95.9	95	95.4	93.6	94.1	94.8	2
21	EfficientNetB2	94.6	96.5	96.7	96.3	96.3	95.9	96.6	93.7	95.5	94.8	1
22	EfficientNetB3	94	96.4	96.5	96.3	96	95.5	96	94.1	95.1	94.8	1
23	EfficientNetB4	92.6	96.1	96.1	95.8	96	95	95.6	92.7	94.6	94.1	1
24	EfficientNetB5	91.2	93.7	94.1	93.5	93.2	92.7	93.5	90.4	91.8	91.7	1
25	EfficientNetB6	90.8	93.7	93.9	94.3	94.1	93	93.9	91.1	92.8	91.9	1
26	EfficientNetB7	92.4	94	94.2	94	93.3	93.3	93.7	92	93.3	93	1
	TOTAL	2,131	2,192.2	2,209.4	2,194.7	2,177.8	2,158.6	2,181.6	2,108.8	2,149.6	2,144.8	37

To check the impact of the settings of the used scenarios (A, B, and C), the three tables (4, 5 and 6) presenting the results of the three scenarios need to be compared to each other. This will show the improvement in terms of validation accuracy between every 2 scenarios. Table 7 compares the results of Scenarios A and B. The cell is shaded whenever the validation accuracy in Scenario B is higher than that of Scenario A. Table 7 proves that the settings of Scenario B led to better results as 166 out of 260 notebooks could achieve a higher validation accuracy. This is because we used ImageNet weights.

URL: http://dx.doi.org/10.14738/tmlai.103.12245

Table 7. Scenario A vs Scenario B

Id	Model	NON	GEO	COL	KER	RND	MIX	WTH	GAN	STL	OFI
1	Xception										
2	VGG16										
3	VGG19										
4	ResNet50										
5	ResNet101										
6	ResNet152										
7	ResNet50V2										
8	ResNet101V2										
9	ResNet152V2										
10	InceptionV3										
11											
12	MobileNet										
13	MobileNetV2										
14	DenseNet121										
15	DenseNet169										
16	DenseNet201										
17	NASNetMobile										
18	NASNetLarge										
19	EfficientNetB0										
20	EfficientNetB1										
21	EfficientNetB2										
22	EfficientNetB3										
23	EfficientNetB4										
24	EfficientNetB5										
25	EfficientNetB6										
26	EfficientNetB7										

Table 8 compares the results of Scenario B with those of Scenario C to detect how further the results got improved. As Table 8 shows, 192 notebooks (out of 260) achieved even higher validation accuracy in Scenario C compared to Scenario B. This improvement is because network layers are not trainable in Scenario C.

Table 8. Scenario B vs Scenario C

Id	Model	NON	GEO	COL	KER	RND	MIX	WTH	GAN	STL	OFI
1	Xception										
2	VGG16										
3	VGG19										
4	ResNet50										
5	ResNet101										
6	ResNet152										
7	ResNet50V2										
8	ResNet101V2										
9	ResNet152V2										
10	InceptionV3										
11											
12	MobileNet										
13	MobileNetV2										
14	DenseNet121										
15	DenseNet169										
16	DenseNet201										
17	NASNetMobile										
18	NASNetLarge										
19	EfficientNetB0										
20	EfficientNetB1										
21	EfficientNetB2										
22	EfficientNetB3										
23	EfficientNetB4										
24	EfficientNetB5										
25	EfficientNetB6										
26	EfficientNetB7										

Table 9 compares Scenario A with Scenario C and shows how many notebooks in total could achieve a higher validation accuracy in Scenario C compared to Scenario A. As Table 9 shows, 247 notebooks (out of 260) could achieve a higher validation accuracy in Scenario C compared to Scenario A. Obviously, most of the notebook (95% of them) could achieve a higher validation accuracy in Scenario C. The reason behind that is the fact that Scenario A does not use ImageNet weights and allows the network layers to be trained, whereas Scenario C makes use of ImageNet weights and benefit from the used model, by preventing the network layers from being trainable.

URL: http://dx.doi.org/10.14738/tmlai.103.12245

Table 9. Scenario A vs Scenario C

Id	Model	NON	GEO	COL	KER	RND	MIX	WTH	GAN	STL	OFI
1	Xception										
2	VGG16										
3	VGG19										
4	ResNet50										
5	ResNet101										
6	ResNet152										
7	ResNet50V2										
8	ResNet101V2										
9	ResNet152V2										
10	InceptionV3										
11											
12	MobileNet										
13	MobileNetV2										
14	DenseNet121										
15	DenseNet169										
16	DenseNet201										
17	NASNetMobile										
18	NASNetLarge										
19	EfficientNetB0										
20	EfficientNetB1										
21	EfficientNetB2										
22	EfficientNetB3										
23	EfficientNetB4										
24	EfficientNetB5										
25	EfficientNetB6										
26	EfficientNetB7										

To complete the analysis of the three scenarios (A, B and C) and determine where the OFI approach can be considered an efficient data augmentation method, the findings need to be numerically summarized.

Table 10 sums up the content of the above three tables (Table 4, Table 5, and Table 6). It includes a column for every scenario and a row for every model. The numeric values inside the cells indicate how many times the OFI approach resulted in a higher validation accuracy.

Each shaded cell in Table 10 indicates how many times the OFI approach led to a higher validation accuracy compared to all other augmentation methods applied to the corresponding model in the corresponding scenario. The last column in that table is the sum of the scores of the three scenarios.

This total score shows how many times the OFI approach could achieve a higher validation accuracy compared to all other augmentation methods in the three scenarios.

Table 10. The OFI Approach Scoreboard

Id	Model	Score in Scenario A	Score in Scenario B	Score in Scenario C	Total
1	Xception	2	2	1	5
2	VGG16	0	0	2	2
3	VGG19	0	0	4	4
4	ResNet50	0	1	2	3
5	ResNet101	0	3	6	9
6	ResNet152	0	2	0	2
7	ResNet50V2	1	2	1	4
8	ResNet101V2	2	4	1	7
9	ResNet152V2	4	1	1	6
10	InceptionV3	8	0	1	9
11	InceptionResNetV2	1	4	1	6
12	MobileNet	1	1	1	3
13	MobileNetV2	3	1	1	5
14	DenseNet121	1	1	1	3
15	DenseNet169	1	4	1	6
16	DenseNet201	2	1	1	4
17	NASNetMobile	1	3	1	5
18	NASNetLarge	0	3	1	4
19	EfficientNetB0	3	2	2	7
20	EfficientNetB1	2	1	2	5
21	EfficientNetB2	4	2	1	7
22	EfficientNetB3	8	1	1	10
23	EfficientNetB4	0	2	1	3
24	EfficientNetB5	0	2	1	3
25	EfficientNetB6	0	4	1	5
26	EfficientNetB7	0	0	1	1
	TOTAL	44	47	37	128
Percentage of notebooks where OFI approach outperforms other methods					16.4%

In summary, the OFI approach scored better than other augmentation methods in 128 notebooks out of 780. This is to say that the OFI approach can be considered a "better" approach in 16.4% of the notebooks analyzed in this study.

CONCLUSION

When very limited data sets are available, sets of images can be augmented using the mixed data set of original and augmented images. As shown in Table 10, the OFI approach is efficient in 16.4% of the cases.

As per the other approaches, the Kernel filters led to the highest sum of validation accuracy among all the 26 models included in this study in Scenarios A and B, whereas the Color space transformations produced the highest sum in Scenario C.

In brief, what has been done is testing a wide range of augmentation methods using many scenarios and models. The goal behind this comparative study is to check if the proposed OFI method would lead to better results. In more than 16% of the tested scenarios, OFI method led to a better validation accuracy. The scenarios where OFI approach outperforms other methods are listed in Table 10.

URL: http://dx.doi.org/10.14738/tmlai.103.12245

In general, the OFI method is a valid candidate when an augmentation method is needed. Although it is does not appear to be always among the top solutions, the OFI method is one of those augmentation methods that should be taken into consideration, especially when there is no knowledge transfer and there are no weights to use (as shown in Scenario A).

Future Work

During the automated process of removing a background, an error might occur. This happens when the background removal tool cannot identify the background. During this research, the tool (remove.bg) failed in removing the background for only 6% of images. Although this is not very common, it needs to be handled properly.

Among the suggested solutions is to use a secondary tool to remove the background when the primary tool produces an error.

References

[1] E. K. Wang, S. P. Xu, C. M. Chen, and N. Kumar, Neural Architecture Search Based Multiobjective Cognitive Automation System, IEEE Systems Journal, https://doi.org/10.1109/JSYST.2020.3002428, 2020.

[2] F. Zhang, T.-Y. Wu, J.-S. Pan, G. Ding, and Z. Li, Human Motion Recognition Based on SVM in VR Art Media Interaction Environment, Human-centric Computing and Information Sciences,9: 40, 2019.

[3] E. K. Wang, X. Zhang, F. Wang, T.-Y. Wu, and C.-M. Chen, Multilayer Dense Attention Model for Image Caption, IEEE Access, vol. 7, pp. 66358-66368, 2019.

[4] Y. Cui, M. Jia, T. Lin, Y. Song and S. Belongie, Class-Balanced Loss Based on Effective Number of Samples,2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, pp. 9260-9269, 2019.

[5] Müller, S. G., & Hutter, F. (2021). TrivialAugment: Tuning-free yet state-of-the-art data augmentation. Retrieved from http://arxiv.org/abs/2103.10158

[6] Shorten, C., Khoshgoftaar, T.M. A survey on Image Data Augmentation for Deep Learning. J Big Data 6, 60 (2019). https://doi.org/10.1186/s40537-019-0197-0

[7] Naveed, H. Survey: Image Mixing and Deleting for Data Augmentation. arXiv 2021, arXiv:2106.07085

[8] Khosla, C., & Saini, B. S. (2020, June). Enhancing performance of deep learning models with different data augmentation methods: A survey. *Proceedings of the International Conference on Intelligent Engineering and Management (ICIEM),* London, United Kingdom (pp. 79-85). doi:10.1109/iciem48762.2020.9160048

[9] Girshick, R., Radosavovic, I., Gkioxari, G., Dollar, P., and He, K., Detectron. https://github.com/facebookresearch/detectron, 2018.

[10] XIE, Q. Q. Le., Dai, Z. D., Eduard Hovy, Thang Luong, and Quoc Le., Hovy, E., Luong, T., & Le, Q. (2020). Unsupervised data augmentation for consistency training. In H. Larochelle, M. Ranzato, & R. Hadsell (Eds.), *Advances in Neural Information Processing Systems* (pp. 6256–6268). Curran Associates, Inc.

[11] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009, June). ImageNet: A large-scale hierarchical image database. *Proceedings of 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops)*, Miami, FL. doi:10.1109/cvpr.2009.5206848

[12] Dosovitsk, A., Fischer, P., Springenberg, J. T., Riedmiller, M., & Brox, T. (2016). Discriminative unsupervised feature learning with exemplar convolutional neural networks. *Transactions on Pattern Analysis and Machine Intelligence*, 38(6), 1734–1747.

[13] Graham, B. (2014). Fractional Max-Pooling. Retrieved from http://arxiv.org/abs/1412.6071

[14] Sajjadi, M., Tasdizen, T., & Javanmardi, M. (2016). Regularization with stochastic transformations and perturbations for deep semi-supervised learning. *Neural Information Processing Systems*.

[15] Ratner,R.J, Ehrenberg,H,Hussain,Z., Dunnmon,J. &Re,C. (2017). Learning to compose domain-specific transformations for data augmentation. *Neural Information Processing Systems*.

[16] Khalifa, N. E., Loey, M., & Mirjalili, S. (2021). A comprehensive survey of recent trends in deep learning for digital images augmentation. Artificial Intelligence Review, 55(3), 1–27. doi:10.1007/s10462-021-10066-4

[17] Perez, L., & Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning. Retrieved from http://arxiv.org/abs/1712.04621

[18] Zhang, H., Cisse, M., Dauphin, Y. N., & Lopez-Paz, D. (2017). mixup: Beyond Empirical Risk Minimization. Retrieved from http://arxiv.org/abs/1710.09412

[19] DeVries, T., & Taylor, G. W. (2017). Improved regularization of convolutional neural networks with cutout. Retrieved from http://arxiv.org/abs/1708.04552

[20] Vyas A., Yu S., Paik J. (2018) Fundamentals of digital image processing. Signals Commun Technol. https://doi.org/10.1007/978-981-10-7272-7_1

[21] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. Communications of the ACM, 60(6), 84–90. doi:10.1145/3065386

[22] Chen, P., Liu, S., Zhao, H., & Jia, J. (2020). GridMask Data Augmentation. Retrieved from http://arxiv.org/abs/2001.04086

[23] Provos, N., & Honeyman, P. (2003). Hide and seek: An introduction to steganography *[Review of Hide and seek: An introduction to steganography]. IEEE Security & Privacy*, 1(3), 32–44.

[24] Zhong, Z., Zheng, L., Kang, G., Li, S., & Yang, Y. (2020). Random Erasing Data Augmentation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07), 13001–13008. https://doi.org/10.1609/aaai.v34i07.7000

[25] Shijie, J., Ping, W., Peiyi, J., & Siping, H. (2017, October). Research on data augmentation for image classification based on convolution neural networks. *Proceedings of the Chinese Automation Congress (CAC)*. Jinan, China.

[26] Ronneberger, O., Fischer, P., & Brox, T. (2015, October). U-Net: Convolutional networks for bio-medical image segmentation. *Proceedings of the MICCAI: International Conference on Medical Image Computing and Computer-Assisted Intervention*. Munich, Germany.

[27] Mikołajczyk, A., & Grochowski, M. (2018, May). Data augmentation for improving deep learning in image classification problem. *Proceedings of the International Interdisciplinary PhDWorkshop (IIPhDW)* (pp. 117–122). Swinoujscie, Poland.

[28] Moreno-Barea, F. J., Strazzera, F., Jerez, J. M., Urda, D., & Franco, L. (2018, November). Forward noise adjustment scheme for data augmentation. *Proceedings of the IEEE Symposium Series on Computational Intelligence (SSCI)*, Bangalore, India. (pp. 728–734) doi:10.1109/ssci.2018.8628917

[29] Hendrycks, D., Mu, N., Cubuk, E. D., Zoph, B., Gilmer, J., & Lakshminarayanan, B. (2019). AugMix: A simple data processing method to improve robustness and uncertainty. Retrieved from http://arxiv.org/abs/1912.02781

[30] Inoue, H. Data Augmentation by Pairing Samples for Images Classification. arXiv 2018, arXiv:1801.02929.

[31] Xie, T., Cheng, X., Wang, X., Liu, M., Deng, J., Zhou, T., & Liu, M. (2021, October 17). Cut-thumbnail: A novel data augmentation for convolutional neural network. *Proceedings of the 29th ACM International Conference on Multimedia*, Virtual Event China. doi:10.1145/3474085.3475302

[32] Takahashi, R., Matsubara, T., & Uehara, K. (2018). Ricap: Random image cropping and patching data augmentation for deep CNNs*. Asian Conference on Machine Learning*, 1, 786–798.

[33] Yun, S., Han, D., J, S., Chun, S., Choe, J., & Yoo, Y. (2019). Cutmix: Regularization strategy to train strong classifiers with localizable features [Review of Cutmix: Regularization strategy to train strong classifiers with localizable features]. *In EEE International Conference on Computer Vision* (pp. 6023–6032).

[34] Harris, E., Marcu, A., Painter, M., Niranjan, M., Prügel-Bennett, A., & Hare, J. (2020). FMix: Enhancing mixed Sample Data Augmentation. Retrieved from http://arxiv.org/abs/2002.12047

[35] Kim, J.-H., Choo, W., & Song, H. O. (2020). Puzzle Mix: Exploiting saliency and local statistics for optimal mixup. Retrieved from http://arxiv.org/abs/2009.06962

[36] Walawalkar, D., Shen, Z., Liu, Z., & Savvides, M. (2020). Attentive Cutmix: An Enhanced Data Augmentation Approach for Deep Learning Based Image Classification. *In ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 3642–3646).

[37] Kang, G., Dong, X., Zheng, L., & Yang, Y. (2017). PatchShuffle Regularization. Retrieved from http://arxiv.org/abs/1707.07103

[38] Zhou, Y. (2019). Slot Based Image Augmentation System for Object Detection. http://arxiv.org/abs/1907.12900

[39] Ghiasi, G., Cui, Y., Srinivas, A., Qian, R., Lin, T.-Y., Cubuk, E. D., … Zoph, B. (2020). Simple Copy-paste is a strong data augmentation method for instance segmentation.http://arxiv.org/abs/2012.07177

[40] Goodfellow IJ et al. (2014) Generative adversarial nets. *Proceedings of the 27th International Conference on Neural Information Processing Systems - Vol 2*, 2014, pp. 2672–2680.

[41] Jing Y., Yang Y., Feng Z., Ye J., Yu Y., Song M. (2019) Neural style transfer: a review. IEEE Trans Visual Comput Graphics. doi:10.1109/TVCG.2019.2921336

[42] Frans K., Ho J., Chen X., Abbeel P., Schulman J. (2018) Meta learning shared hierarchies.

[43] Cubuk ED., Zoph B., Mane D., Vasudevan V., Le QV. (2019) Autoaugment: learning augmentation strategies from data, doi:10.1109/CVPR.2019.00020

[44] Lemley, J., Bazrafkan, S., & Corcoran, P. (2017). Smart Augmentation Learning an Optimal Data Augmentation Strategy. IEEE access: practical innovations, open solutions, 5, 5858–5869. doi:10.1109/access.2017.2696121

[45] Cubuk, E. D., Zoph, B., Shlens, J., & Le, Q. V. (2020). Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops* (pp. 702–703).

[46] Tian, K., Lin, C., Sun, M., Zhou, L., Yan, J., & Ouyang, W. (2020). Improving auto-augment via augmentation-wise weight sharing. *In Advances in Neural Information Processing Systems* (p. 33).

[47] Ho, D., Liang, E., Chen, X., Stoica, I., & Abbee, P. (2019). Population based augmentation: Efficient learning of augmentation policy schedules. *In International Conference on Machine Learning*, (pp. 2731–2741). PMLR.

[48] Zhang, X., Wang, Q., Zhang, J., & Zhong, Z. (2020). Adversarial autoaugment. *In International Conference on Learning Representations.*

[49] Hoffer, E., Itay, T. B.-N. H., Giladi, N., Hoefler, T., & Soudry, D. (2020). Augment your batch: Improving generalization through instance repetition. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* CVPR.

[50] Lim, S., Kim, I., Kim, T., Kim, C., & Kim, S. K. (2019). Fast autoaugment. In H. Wallach, H. Larochelle, A. Beygelzimer, F. D'Alch´e-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (Vol. 32, pp. 6665–6675). Curran Associates.

[51] Lin, C., Guo, M., Li, C., Yuan, X., Wu, W., Yan, J., Lin, D., & Ouyang, W. (2019). Online hyper-parameter learning for auto-augmentation strategy. *In proceedings of the IEEE/CVF International Conference on Computer Vision.* ICCV.

[52] Gupta, A., Vedaldi, A., & Zisserman, A. (2016). Synthetic Data for Text Localisation in Natural Images. Retrieved from http://arxiv.org/abs/1604.06646

[53] Dwibedi, D., Misra, I., & Hebert, M. (2017). Cut, paste and learn: Surprisingly easy synthesis for instance detection. Retrieved from http://arxiv.org/abs/1708.01642

[54] Dvornik, N., Mairal, J., & Schmid, C. (2018). Modeling Visual Context is Key to Augmenting Object Detection Datasets. Retrieved from http://arxiv.org/abs/1807.07428

[55] Tripathi, S., Chandra, S., Agrawal, A., Tyagi, A., Rehg, J. M., & Chari, V. (2019). Learning to generate synthetic data via compositing. Retrieved from http://arxiv.org/abs/1904.05475

URL: http://dx.doi.org/10.14738/tmlai.103.12245

APPENDIX

Appendix A - The network design

As stated before, all the used notebooks have the same network design. They differ in the used model (Xception, VGG, ResNet...), but they share the same network design, which is presented below:

```
for layer in NET.layers:
layer.trainable = isTrainable          # this is True in Scenarios A and B, but it is False in Scenario C
# NET is the model that is set in the settings part.
# It can be any of the 26 models: VGG16, VGG19, ResNet50, …
x = Flatten(name='cust_NET')(NET.output)
x = layers.Dense(512, activation='relu')(x)
x = layers.Dropout(0.5)(x)
prediction = Dense(numberOfClass, activation='softmax')(x)
model = Model(inputs=NET.input, outputs=prediction)
```

The following settings are used for compilation:

```
model.compile(loss = "sparse_categorical_crossentropy",
        optimizer = "adam",
        metrics = ["accuracy"])
```

Two image generators are used as follows:

```
batch_size = 16
image_data_gen = ImageDataGenerator(rescale=1./255)
train_data_gen = image_data_gen.flow_from_directory(directory=local_dir + "/train",
                                    target_size = (224,224),
                                    color_mode='rgb',
                                    batch_size=batch_size,
                                    class_mode = 'sparse')
test_data_gen = image_data_gen.flow_from_directory(directory=local_dir + "/test",
                                   target_size = (224,224),
                                   color_mode='rgb',
                                   batch_size=batch_size,
                                   class_mode = 'sparse')
```

This is how the model is trained:

```
epochs_num = 150
if NetWeights == "ImageNet":
  epochs_num = 50
hist = model.fit_generator(train_data_gen,
                steps_per_epoch=train_data_gen.n//batch_size,
                epochs= epochs_num,
                validation_data=test_data_gen,
                validation_steps=test_data_gen.n//batch_size, verbose=2)
```