Name: Akash Bhargava
Student ID Number: 18210613
Email: akash.bhargava2@mail.dcu.ie
Module Code: CA685 – Data Analytics Practicum
Programme: Master's in computing (Data Analytics)
Date of Submission: August 11, 2019
Github Path: https://github.com/justprophet/Synthetic-Image-Creation-and-Text-Detection
Project Title: Synthetic Data Creation for Detection and Reading Digital Weighing Scales from User-contributed Images
Supervisor: Dr. Tomas Ward

**DISCLAIMER**

A report submitted to Dublin City University, School of Computing for module CA640 Professional and Research Practice, 2017/2018. I understand that the University regards breaches of academic integrity and plagiarism as grave and serious. I have read and understood the DCU Academic Integrity and Plagiarism Policy. I accept the penalties that may be imposed should I engage in practice or practices that breach this policy. I have identified and included the source of all facts, ideas, opinions, viewpoints of others in the assignment references. Direct quotations, paraphrasing, discussion of ideas from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged, and the sources cited are identified in the assignment references. I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work. By signing this form or by submitting this material online I confirm that this assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

By signing this form or by submitting material for assessment online I confirm that I have read and understood DCU Academic Integrity and Plagiarism Policy (available at http://www.dcu.ie/registry/examinations/index.shtml)

Name(s): Akash Bhargava

Date: August 11, 2019

# Synthetic Data Creation for Detection and Reading Digital Weighing Scales from User-contributed Images

Akash Bhargava-18210613
School of Computing
*(Student)*
Dublin City University
akash.bhargava2@mail.dcu.ie

***Abstract*— In this paper, we present an approach to record weighing scale data obtained using smart phone camera. Obesity and Cardiovascular problems are burdens to the society and therefore, weight management is necessary to the purposes of tracking these conditions. However, in many parts of the world connected weighing scales like Bluetooth scales are expensive and not easily available. As a contribution to democratize the market, we present this method of using mobile images of weighing scales.**

**Due to unavailability of data, the purpose is to create useful synthetic data to replicate real-world scenario for creating and developing machine vision pipelines around it.**

**As a result of it, we were able to create 4000 Synthetic Digital Weighing Scale images along with their indexing in three categories to represent them as mobile images.**

**Next, we were able to generate a Transfer learning model with the use of ImageNet and ResNet to distinguish between Analog and Digital weighing scales with 85% and 90% accuracy respectively.**

**In addition to that we were able to implement an OpenCV deep learning model for test detection in our synthetic images. And finally, we were able to implement some text recognition model which can identify the weights from the images.**

***Keywords—deep learning, synthetic images, text-recognition, text-detection, connected health***

## I. INTRODUCTION

In recent years Obesity has arrived as one of the major challenges faced by the Heath Organisations. According to World Health Organization (WHO), worldwide obesity has tripled since 1975[1]. In Ireland itself 61% of adults and 22% of 5-12 year olds are suffering from the conditions of obesity. [2]

With the advent of Connected Heath Services, several people are trying to keep their weight in check through regular monitoring. However, the reach of these services is limited due to a lot of reasons. These services are expensive, difficult to use and require a complex set of equipment to function accurately for example Bluetooth Weighing scales. Thus, it has stayed away from the hands of low economic households, lesser educated factions and older age groups.

As a result of this we propose an alternative of using mobile images of weighing scales to record the weights of the user and move forward from that. Since there was no pre-existing dataset to proceed with this idea, we started collection images manually and indexing them, but this process was tedious, and the data collected was not enough to be augmented for Deep Learning Processes.

Thus, we came up with the project pipeline which can be described using Fig 1.

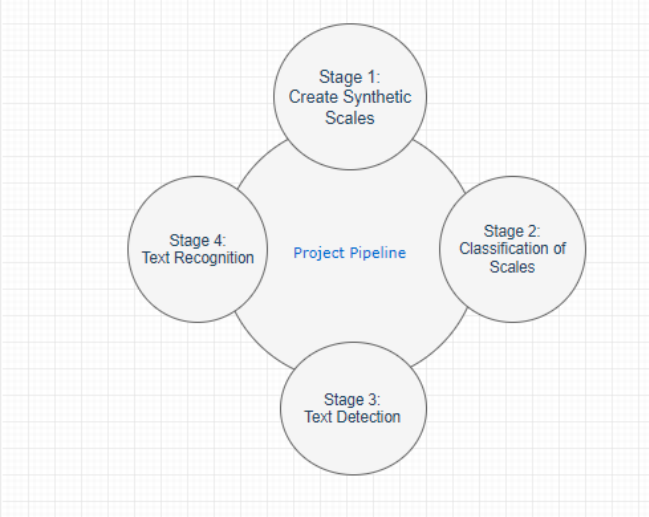


Fig 1. Project Pipeline

Due to absence of real-world data, the motivation was to create synthetic weighing scales in order to have larger volumes of data which can be processed for all types of deep learning algorithms.

Next, we dealt with the problem of image classification to classify the images between Analog and Digital Weighing Scales. Current approaches to image classification make essential use of Machine Learning methods. A very popular approach is to treat image classification on ImageNet [3] as a pre-trained network to help with the problem of limited data [4]. This practise of first training a Convolution Neural Network (CNN) to perform image classification on ImageNet (i.e. pre-training) and then adapting these features for a new task (i.e. fine tuning) has become de-facto standard for solving wide range of computer vision problems.

The next step in the pipeline was to detect the weights in these images. While the recognition of text within scanned documents is well studied and there are existing OCR systems that perform very well, these methods do not

translate to the high variable domain of scene text recognition. Thus, these OCR systems fail when applied to Natural Images as they are predominantly designed to be suited for black-and-white images, printed documents or line-based environment, while the text existing in Natural images suffer from image distortions, inconsistent lighting conditions and background noise.

When it comes to Effective Text Recognition, the task is generally divided into two stages: Text Detection and Word Recognition. The Text Detection stage involves the generation of bounded boxes around the text using various methods. Detecting text in constrained, controlled environments can typically be accomplished by using heuristic-based approaches, such as exploiting gradient information or the fact that text is typically grouped into paragraphs and characters appear on a straight line.

With the proliferation of smartphones with cameras, we need to be highly concerned with the condition of captured images and furthermore, the assumptions we can and cannot make. A detailed study conducted by [5] highlights some important natural scene text detection challenges:

- **Image/sensor noise:** A handheld camera has Sensor noise higher than that of a traditional scanner.
- **Viewing angles:** The text in Natural Scene is harder to recognize as it can have viewing angles that are not parallel to text.
- **Blurring:** Smartphone that does not have some form of stabilization tend to have blur in the images.
- **Lighting conditions:** It is not possible to make any assumptions of our lighting conditions in natural scene images. There might be situations where it may be near dark, the flash on the camera may be on, or the sun may be shining brightly, saturating the entire image.
- **Resolution:** We might have to deal with cameras with sub-par resolution.
- **Non-paper objects:** Text in natural scenes may be reflective, including logos, signs, etc.
- **Non-planar objects:** There might be cases where text is present on non-planar objects. For example: Bottle. While it is easy for humans to "detect" this, our algorithms will struggle.
- **Unknown layout:** It is not possible to give priori information to our algorithms as to where text resides.

Thus, it is important to have a robust text-detection algorithm which can encompass all these challenges.

In this paper we provide the approach to complete all these stages of recognition of weighing scales. Through our approach we aim to democratize the use of connected health services making it accessible to all sections of society.

## II. RELATED WORK

In last few years, there has been a significant surge in the market for connected health services. People suffering with conditions such as obesity and diabetes are advised to keep their weight in check and the advantages/disadvantages of these tools are studied in detail by [6][7][8]. Almost all these tools make use of Wireless or Bluetooth weighing scales for capturing the weight of patients. This technology is expensive thus cannot be accessible by everyone. In today's world, almost every household owns a smart phone [9]. Therefore, we propose this alternative to use images of weighing scale to record weights.

### Creating Synthetic Images

A lot of work has been done in the field of Image Processing and synthetic imaging using the two important python libraries: Ski-kit learn and Pillow. [10] These foundations helped to further the research in the field to develop deep learning mechanisms.

When it comes to Synthetic datasets, they provide easier ground truth annotations which is a cheap and scalable alternative compared to manual annotations. It's helpful in application of large CNN models. [11][12][13]make the use of synthetic data to predict text in the natural sceneries and wildlife.

When applying deep learning techniques, researchers tend to augment their data due to the absence of variety and quantity. [14] tries to render objects using lighting and inferring 3D structures. Augmentor is a popular library used specifically for Image Augmentations on a pipeline of images [15].

### Image Classification

Initial approaches to image classification make essential use of machine learning methods. To improve the performance, they collect larger datasets and learn more complex models. The most commonly used technique in these scenarios is the use of Convolutional Neural Networks. [16][17] make the use of deep learning models for classification of images.

However, when the data is scarce these models are not efficient. Oversampling and Augmentation can help to resolve these issues, but it ends up causing overfitting of data. Also, computational cost for these operations was high due to large datasets.

This gave rise to the use of pre-trained models for image-classification problems. The ImageNet competition [3] brought the use of Inception Architecture [18] in picture which focused on these issues and brought in the concept of Transfer learning[19] where pre-trained models could be used to classify images efficiently. The computational

cost was low, and the accuracy of these models was higher than the previous models. Hence, Transfer learning has become the de-facto standard for image classification.

**Text Detection and Text Recognition**

Text detection has been an active research in the past few years which has inspired numerous approaches in this field [20]. Optical Character Recognition (OCR) has been a popular approach to deal with text detection. [21] [22] A common approach is the use of Open-CV EAST detector which is published by [23]. It makes the use of Tesseract 4.0 and OCR method to detect and predict text in the images. OCR approaches have a few disadvantages when it comes to bad lighting conditions and partial-blurry images.

Another approach is to make the use of CNN based sliding window classifier which identifies characters but is not able to deal with noise in the image. [24]

Recently, the use of Connectionist Text Proposal Network (CTPN) that directly localizes text sequences in convolutional layers has achieved significant results in this area. [25]

## III. CREATING SYNTHETIC IMAGES

At the beginning, we started collecting data manually. It was a tedious process and indexing had to be done manually. We were able to collect around 600 images which even with augmentation process were not enough to be trained on a deep learning model.

We figured that this problem however could be resolved with the creation of synthetic weighing scales.

The construction of Synthetic images was completed using Pillow Library of Python.

First step was to create a random scale generator. The purpose of this was to create a random digital display of weight. From PIL we imported important libraries namely *Image, ImageFont, ImageDraw, Image*. We also imported *pandas* for creation of Index.

We created a method "*createRandomScaleImage*" which took a blank scale image as an input that acts as a constant. A random float number between 40 and 110 was generated using the random method. We used "*DS-DIGI.ttf*" font -type to display the weights on the scale. The result can be seen in Fig2.


Fig2. Scale Creation

For the next step, we used Generative Adversarial Networks (GANs) generated bedroom images by which we created a database consisting of Mobile Bedroom Images. Since there were no existing Bathroom Image GANs available, we figured bedroom images would be closest resemblance to use as a background to the display. Fig 3 represents the outcome.


Fig 3. Synthetic Images

In addition to this, we indexed these images in three major categories namely "*Weight-Category*" which is 5kg category scale, "*Weight*" depicting exact weight and "*Type*" representing the scale being Digital or Analog, since we plan to implement Analog Scales as an extension. The data is stored in a .csv file against the created name of each image. The index table with ground truth values is represented as Table 1:

| Index | Weight | Weight -Category | Type |
|---|---|---|---|
| synImage1 | 46 | 45-50 | Digital |
| synImage2 | 46.87 | 45-50 | Digital |
| synImage3 | 46.68 | 45-50 | Digital |
| synImage4 | 82.85 | 80-85 | Digital |
| synImage5 | 73.95 | 70-75 | Digital |
| synImage6 | 108.34 | 105-110 | Digital |
| synImage7 | 57.25 | 55-60 | Digital |
| synImage8 | 51.24 | 50-55 | Digital |
| synImage9 | 109.21 | 105-110 | Digital |
| synImage10 | 92.8 | 90-95 | Digital |
| synImage11 | 103.08 | 100-105 | Digital |
| synImage12 | 104.56 | 100-105 | Digital |
| synImage13 | 84.89 | 80-85 | Digital |

| | | | |
|---|---|---|---|
| synImage14 | 109.9 | 105-110 | Digital |

Table 1. Index table with ground truth values

### Providing Affine transformations

To replicate the real-world scenarios of crappy images i.e. distortions, blur or bad light, we went ahead to provide Affine Transformations to the created synthetic images. Affine transformations are used to create Augmentations of the normal images.

To apply Affine transformations, we used ImgAug Library of Python.[26] It provides a lot of Augmentation techniques such as perspective transformations, contrast changes, gaussian noise, dropout of regions, hue/saturation changes, cropping/padding, blurring etc.

Firstly, we imported the important libraries namely *imgaug, numpy and glob*. Next the data was stored in glob.

For the next step, we created a complex method which provided a series of distortions to the Image based on random probabilities.
It starts with basic cropping and padding and next it provides some physical transformations like Scaling, Translation, Rotation. It moves on to Gaussian Blur, Average Blur and Median Blur for covering the aspect of Blurry images. It also adds Gaussian Noise and Simplex Alpha noise to cover the scenarios where the real-world images are affected by noise. To cover the aspects of lighting i.e. real-world images being taken in bad light or excessive light we provided the aspects of Hue and Saturation and Contrast Normalization. Finally, we added Elastic Transformation, Grayscale and some other transformations to make these images more challenging to deal with.

Below present are the transformations as a part of Fig 4.



Fig 4. Augmentations and Transformations

## IV. Classification of Scales

Since the data was limited, the classification problem was approached as Transfer Learning problem. Transfer learning allows us to train deep networks in a situation where data is less and thus it is not possible to train data from scratch. With the use of this approach, it can transfer the "knowledge" that a model has learned from the previous task, to the current one. The idea is that the two tasks are not totally different, and thus the network parameters that first model has learned can be leveraged, without having to do it in the second model. This is represented by Fig 5.

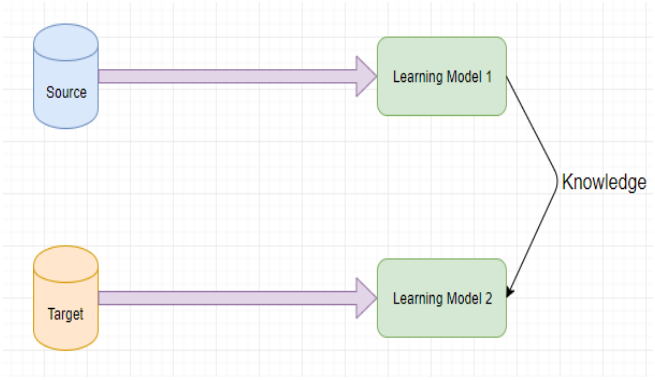


Fig 5. Learning Process of Transfer Learning

The first thing that was required was to prepare the data. We used the manually collected images for this model since we didn't create Analog Scales. For image classification in Keras, the best way to do it was to separate the data into folders for each class. For example, the dataset had two classes: Analog and Digital, so we had two folders namely Analog and Digital. Each of the folders should contain images for that class.

Once the base data was prepared, we needed to split some data for testing and validation, moving images to the train and test folders. We use the *train_test_split()* function from scikit-learn to build these two sets of data. Therefore, we created a structure with training and testing data, and a directory for each target class. This is the way a common folder is structured to use for training a custom image classifier — with any number of classes — with Keras. In the end, folder structure looks like:

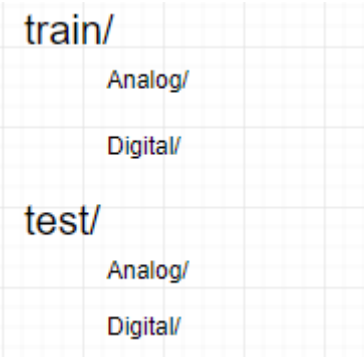


Fig 6

Next, we defined our network. We used two networks or models for classification namely InceptionV3 and ResNet to compare performance. We instantiated both networks from the *keras.applications* module, but keeping the flag *include_top* = False to load the model and their weights but leaving out the last fully connected layer, since that is particular to the ImageNet competition.

*base_model = ResNet(weights='imagenet', include_top=False)*

Next, we added custom classification layer. We preserved the original InceptionV3 architecture but adapted the output to our number of classes. We used a GlobalAveragePooling2D preceding the fully-connected Dense layer of 2 outputs.

Now we needed to freeze all our *base_model* layers and train the last ones.

Finally, we compiled the model selecting the optimizer, the loss function, and the metric. We used *RMSProp* optimizer with the default learning rate of 0.001, and a *categorical_crossentropy* — used in multiclass classification tasks — as loss function.

**Data Augmentation:**

Data Augmentation is a step used for increasing the size of dataset and the model generalizability. It is essentially the process of artificially increasing the size of dataset via transformations i.e. rotation, flipping, stretching, cropping etc.

With Keras, we used the class *ImageDataGenerator()* for data augmentation. This class can be parametrized to implement several transformations, and our task was to decide which transformations make sense for our data. Images were directly taken from our defined folder structure using the method *flow_from_directory()*.

**Transfer Learning:**

Lastly, we trained our custom classifier using the fit_generator method for transfer learning. In this example, we used five epochs as it gave us good accuracy.

We have defined three values:EPOCHS, STEPS_PER_EPOCH, and BATCH_SIZE. These values are important since we cannot pass the data to computer to the system at once due to memory limitations. So, to overcome this problem we divided the dataset into smaller batches and passed it to memory iteratively, updating the weights of neural network at every end of each iteration to fit in the data given.

We defined a BATCH_SIZE of 32 images, which is the number of training examples present in a single iteration or step. And 320 STEPS_PER_EPOCH as the number of iterations or batches needed to complete one epoch. Learning is an iterative process in this case and each epoch is when an entire dataset is passed through the neural network.

**Prediction**s:

Now that we had trained the model and saved it in MODEL_FILE, we used it to predict the class of an image file. The accuracy and loss of two Models of two models is described by two graphs Fig 7 and Fig 8.

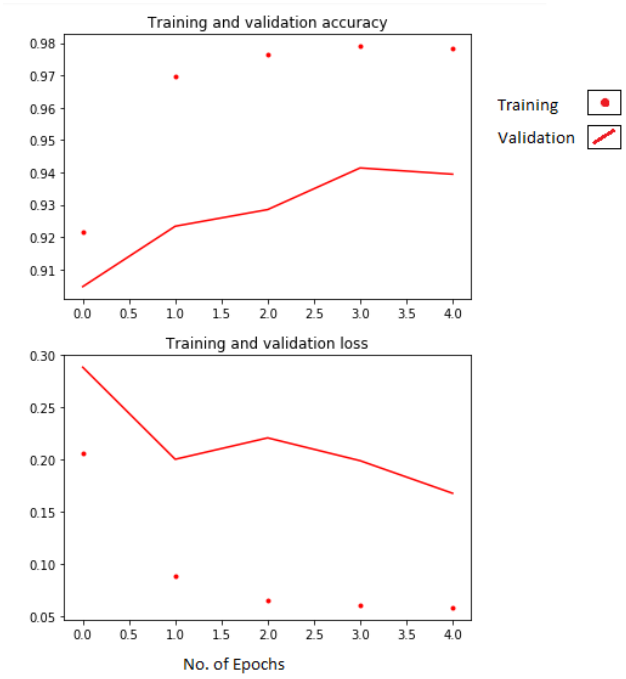


Fig 7. ImageNet (Accuracy and Loss)

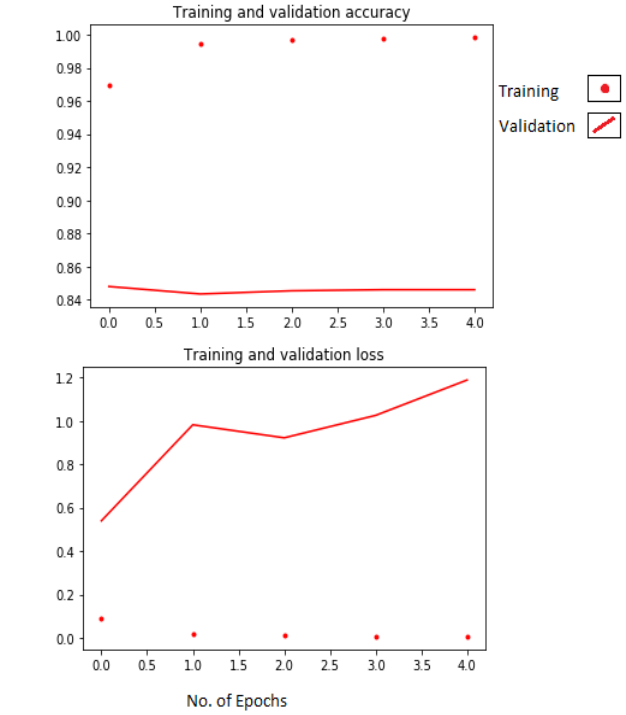


Fig 8. ResNet (Accuracy and Loss)

The prediction probabilities are present in Table 2.

As we can see ResNet based model seems to perform better than Inception V3. The prediction probabilities of Analog Scales are significantly less due to smaller volume of Analog Data.

| | Analog | Digital |
|---|---|---|
| Inception V3 | 0.808 | 0.852 |
| Resnet | 0.871 | 0.936 |

Table 2. Predictions (in Probabilities)

## V. TEXT DETECTION AND RECOGNITION

With the release of OpenCV 4 and OpenCV 3.4.2, we used a deep learning text detector called **E**fficient and **A**ccurate **S**cene **T**ext detection pipeline (EAST).

The EAST pipeline is said to predict words and lines of text at arbitrary orientations on 720p images, and furthermore, can run at 13 FPS, according to the authors.

To build and train such a deep learning model, the EAST method utilizes novel, carefully designed loss functions.

To begin, we imported our required packages and modules. Notably, we import NumPy and OpenCV. We have five command line arguments:

- input: The path to our input image.
- model: The EAST scene text detector model file path.
- min-confidence: Probability threshold to determine text. Optional with default=0.5.
- width: Image width resized — must be multiple of 32. Optional with default=320.
- height: Image height resized— must be multiple of 32. Optional with default=320.

In order to perform text detection, we needed to extract output feature maps of two layers.

1. Sigmoid Activation layer which gives us the probability of a region containing text or not.
2. The output feature map that represents the "geometry" of the image which is used to derive bounding box coordinates of the text in the input image.

The neural network in the memory was loaded by *cv2.dnn.readNet* by passing the path to EAST detector. Then we prepared our image by converting into a blob.

Next, OpenCV is instructed to return two feature maps:

- Geometry, to derive the bounding box coordinates for the text in images.
- Scores, which contain the probability of a given region containing text.

Then we run the loops to identify the rectangle with most "Scores". Based on that we keep updating our original parameters.

Finally, Non-Maxima Suppression implementation present in *imutils* is used on the bounding boxes to suppress weak overlapping bounding boxes.

**Results:**

We checked the algorithm on Images with Scales and Images without scales to validate the algorithm. Here are the results:


Fig 9. Images Without Scales: Note that no rectangle boxes are created,



Fig 10. Text Detection in Images with Weighing Scales:

For Image Recognition we used Tesseract V4 OCR. However, the results are not satisfactory and further research is ongoing. Fig 11 represents few examples:


Fig 11. Text Recognition

## VI. Conclusion and Future Work

In this research paper, we have intended to explain and create a possible alternative in the world of Connected Health services. As per the project pipeline we were able to successfully accomplish three out of four stages while work on Text-Recognition is still in progress and yet to give some concrete results.

We created a synthetic scale database along with the indexes which I will make public post the completion of project allowing analysts to work on the possible applications in the areas of Health and Text-Detections. This also helps analysts in case of GDPR issues where collection of personal data is not possible.

We successfully created a classifier on collected images to identify the difference between Analog and Digital Scale. The classifiers can be further extended to identify if the image contains a weighing scale or not.

We were able to test a Text-Detection algorithm which locates the presence of Weights in the image. A Text-Recognition model is in progress which can identify the exact weights on the scale.

As a part of future work, we plan to move to the creation and recognition of Analog weighing scales. Also, we want to extend this task to Blood Pressure monitors and other monitoring devices.

Finally, as per the original idea, we plan to create an app which will record and maintain the data, create dashboards and share it with the General Practitioners for further advice.

## VII. Acknowledgements

I would like to thank my supervisor, Dr. Tomas Ward, for his constant support, guidance and the belief he showed in this project.

## References

[1] W. H. Organization, “Obesity and overweight 16,” no. February 2018, pp. 2–7, 2019.

[2] W. H. Organization and R. Office, “Ireland - WHO Country Profile,” 2013.

[3] O. Russakovsky *et al.*, “ImageNet Large Scale Visual Recognition Challenge,” *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.

[4] A. Karpathy, H. Zhiheng, M. Bernstein, and A. Khosla, “ImageNet Classification with Deep Convolutional Neural Networks,” *Handb. Approx. Algorithms Metaheuristics*, pp. 60-1-60–16, 2007.

[5] C. Mancas and B. Gosseli, “Natural Scene Text Understanding,” *Vis. Syst. Segmentation Pattern Recognit.*, no. June, 2012.

[6] R. Dobson, R. Whittaker, R. Murphy, M. Khanolkar, and S. Miller, “The Use of Mobile Health to Deliver Self-Management Support to Young People With Type 1 Diabetes : A Cross-Sectional Survey Corresponding Author :,” vol. 2, no. 1.

[7] K. Lithgow, A. Edwards, and D. Rabi, “Smartphone App Use for Diabetes Management : Evaluating Patient Perspectives Corresponding Author :,” vol. 2, pp. 1–5, 1820.

[8] L. Kelly, C. Jenkinson, and D. Morley, “Experiences of using web-based and mobile technologies to support self-management of type 2 diabetes: Qualitative study,” *J. Med. Internet Res.*, vol. 20, no. 5, 2018.

[9] Pew Research Center, “Smartphone Ownership Is Growing Rapidly Around the World, but Not Always Equally,” *Pew Res. Cent.*, no. February, p. 47, 2019.

[10] S. van der Walt *et al.*, “scikit-image: image processing in Python,” *PeerJ*, vol. 2, p. e453, 2014.

[11] C. V Dec *et al.*, “Reading Text in the Wild with Convolutional Neural Networks,” vol. 1, pp. 1–10.

[12] A. Vedaldi, “Synthetic Data for Text Localisation in Natural Images Ankush Gupta,” vol. 1.

[13] C. V Dec and K. Simonyan, “Natural Scene Text Recognition,” pp. 1–10.

[14] K. Karsch and D. Forsyth, “Rendering Synthetic Objects into Legacy Photographs,” 2011.

[15] M. D Bloice, C. Stocker, and A. Holzinger, “Augmentor: An Image Augmentation Library for Machine Learning,” *J. Open Source Softw.*, vol. 2, no. 19, p. 432, 2017.

[16] Y. Seo and K. shik Shin, “Hierarchical convolutional neural networks for fashion image classification,” *Expert Syst. Appl.*, vol. 116, pp. 328–339, 2019.

[17] X. Yang, Y. Ye, X. Li, R. Y. K. Lau, X. Zhang, and X. Huang, “Hyperspectral image classification with deep learning models,” *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 9, pp. 5408–5423, 2018.

[18] C. Szegedy, V. Vanhoucke, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” 2014.

[19] L. Shao, F. Zhu, and X. Li, “Transfer learning for visual categorization: A survey,” *IEEE Trans. Neural Networks Learn. Syst.*, vol. 26, no. 5, pp. 1019–1034, 2015.

[20] Y. Zhu, C. Yao, and X. Bai, “Scene text detection and recognition: recent advances and future trends,” *Front. Comput. Sci.*, vol. 10, no. 1, pp. 19–36, 2016.

[21] A. Veit, T. Matera, L. Neumann, J. Matas, and S. Belongie, “COCO-Text: Dataset and Benchmark for Text Detection and Recognition in Natural Images,” 2016.

[22] Q. Ye and D. Doermann, “Text Detection and Recognition in Imagery: A Survey,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 7, pp. 1480–1500, 2015.

[23] X. Zhou *et al.*, “EAST: An efficient and accurate scene text detector,” *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 2642–2651, 2017.

[24] W. Huang, Y. Qiao, and X. Tang, “Robust scene text detection with convolution neural network induced MSER trees,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8692 LNCS, no. PART 4, pp. 497–511, 2014.

[25] Z. Tian, W. Huang, T. He, P. He, and Y. Qiao, “Detecting text in natural image with connectionist text proposal network,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9912 LNCS, pp. 56–72, 2016.

[26] Alexander B. Jung, “imgaug,” *https://github.com/aleju/imgaug*. [Online]. Available: https://github.com/aleju/imgaug.