

Vol. 33, No. 2, 2024

Machine
GRAPHICS & VISION

International Journal

Published by
The Institute of Information Technology
Warsaw University of Life Sciences – SGGW
Nowoursynowska 159, 02-776 Warsaw, Poland

in cooperation with
The Association for Image Processing, Poland – TPO

VERIFICATION OF DATA COMPRESSION FOCUSING ON CONTINUITY IN 3D PRINTING

Satoshi Kodama 

*Department of Information Technology, Faculty of Technology
International Professional University of Technology in Tokyo, Japan
kodama.satoshi@iput.ac.jp*

Abstract Recently, 3D printers have become capable of producing relatively large, high-resolution models. Unlike simple shapes, it is becoming possible to print large complex shapes with high accuracy. However, the data size of complex models is also large, and the slice data required for printing is also large. Thus, in this study, we investigated reducing the data size by focusing on the characteristics of the slice data required for 3D printing. The proposed method focuses on the continuity of each layer and the top/bottom layers of the cross-section used to print the 3D model. Preliminary experiments were conducted to determine whether the data size could be reduced by applying the difference method. Here, the results obtained from the continuity were output as text data, and various metadata, e.g., lamination pitch data, required for printing were ZIP compressed. Then, we compared conventional file formats as a format that can be converted as a printable file as lossless compression. The results demonstrated that the file size can be reduced for 3D complex shapes with a large number of vertices, which are difficult to handle. We found that the proposed difference method was effective for relatively large files that require a general-purpose graphics processing unit to create slice data.

Keywords: 3D printing, data compression, stereolithography, additive manufacturing, sliced data, image storage

1. Introduction

With the ongoing development of three-dimensional (3D) printers, it has become relatively easy to output high-definition, large-scale fabrication objects with complex shapes, which is difficult to do using conventional fabrication methods. Currently, 3D printers are used in many fields [2, 44, 52]. However, the need to handle a wide range of high-definition data has resulted in increased data sizes and the emergence of various file formats depending on the printer used for output, thereby increasing the difficulty of data handling. In addition, when creating cross-sectional views of each layer required for printing from 3D model data, the load for analysis increases [30, 34, 58, 59].

Generally, when printing objects with a 3D printer, the 3D data are decomposed into two-dimensional (2D) data for each layer, and then laminated based on the stacked 2D data [30, 31, 49, 56]. In this study, we conducted an initial verification to determine whether it is possible to reduce the data size using lossless compression based on the continuity of the sliced data required for 3D printing output. As a result, although this was preliminary experiment, we found that combining the proposed method and ZIP compression resulted in a data size that was smaller than that of the conventional large

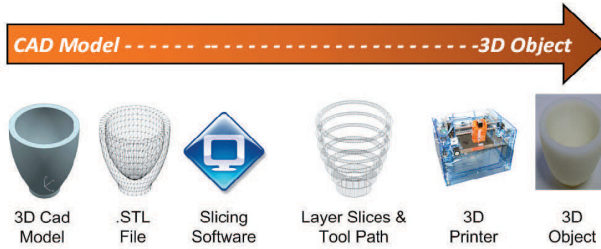


Fig. 1. The 3D Printing process (from [56], license: CC BY 3.0).

data format. In particular, the effectiveness of the proposed method was demonstrated for geometries with a large number of vertices, which tend to be complex geometries. On the other hand, we could not demonstrate the usefulness of the proposed method for 3D models with a small number of vertices. Therefore, the proposed method is effective for compression of 3D model data that requires time for shape analysis necessary for printing. In the future, based on this result, it is thought that even higher compression will be possible by considering such as the symmetry of slice data.

2. Related Work

Printing high-definition, relatively large-sized objects has become easier; thus, 3D printers are widely used in a variety of fields [2, 44, 52]. In addition, 3D printers with various output formats are currently available from multiple manufacturers, and even inexpensive consumer grade printers are increasing in terms of both size and print definition [1, 17].

However, it is becoming increasingly difficult to handle the various manufacturer-specific data formats and the large amounts of data required for 3D printing [3, 12, 21, 24, 38, 46].

Therefore, in this section, we introduce the main 3D printing methods and the methods used to create the data required for printing and file formats. In addition, we identify known problems with these existing methods.

2.1. Types of 3D printers

Various output methods are being researched and developed for 3D printers; however, the most common method is to transform a 3D model as a layer of 2D data and then laminate these layers in sequence (Fig. 1 [56]). Here, we introduce two common lamination methods, e.g., the fused deposition modeling (FDM) and resin printing technology methods.

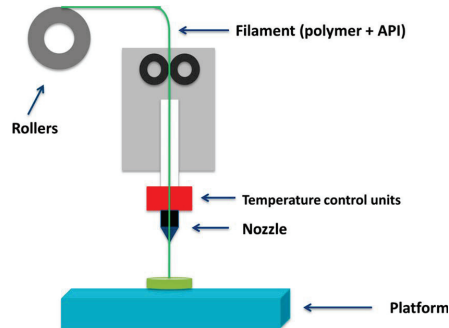


Fig. 2. FDM printing method (from [32], license: CC BY 4.0).

2.1.1. Production of 3D objects by heat

Generally, the well-known FDM method realizes 3D printing by melting a filament in a solid state at a high temperature and stacking it layer-by-layer through a nozzle (Fig. 2) [15, 32]. The FDM method is commonly used, including by individual users, and it has become increasingly affordable since the patent expired in 2009 [35, 50]. Currently, filaments of various materials are readily available, and it is possible to change the hardness and the like according to the use purpose [15]. However, although the structure is very simple, this method is susceptible to various output errors, e.g., heat shrinkage and prints detaching from the build platform, depending on the filament material and the ambient temperature of the print environment [42].

2.1.2. Production of 3D objects by ultraviolet light

Here, ultraviolet-curable resin is cured in a layer-by-layer manner by surface irradiation of ultraviolet light to produce a modeled object. 3D printing using resin primarily includes stereolithography apparatus (SLA), digital light processing (DLP), and liquid crystal display (LCD) depending on the ultraviolet irradiation method (Fig. 3) [25]. With the SLA method, an ultraviolet laser is applied from the bottom or top of a tank filled with photocurable resin for curing. The SLA involves tracing and curing a layer based on the 2D image as a point, and after creating a surface, the surface of the next layer is created in the same way (Fig. 4a) [53]. The DLP method replaces the laser with a projector and applies ultraviolet rays as a digital image. Unlike lasers, the DLP method can irradiate a wide range of ultraviolet rays simultaneously; thus, high-speed output is possible (Fig. 4b) [53]. The LCD method cures using an ultraviolet light as the backlight of an LCD panel, and this method can handle more precise modeling by using a high-resolution panel [19, 37].

A primary characteristic of these methods is that they require cleaning and secondary curing as postprocessing due to the resin characteristics; however, they can achieve a

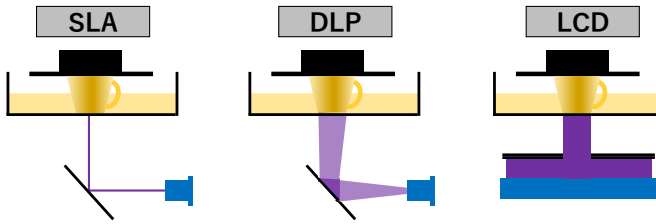


Fig. 3. FDM printing method (created by the author on the basis of [25]).

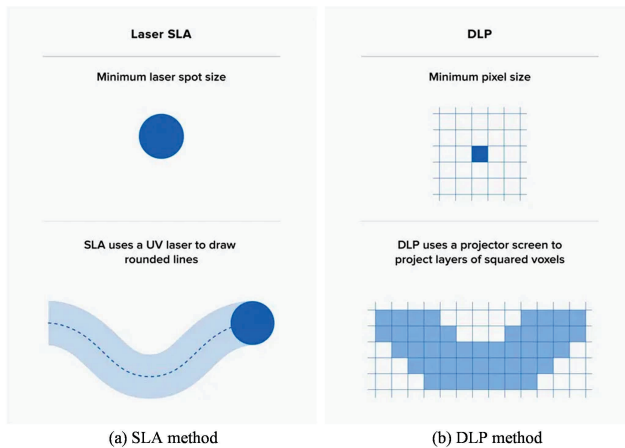


Fig. 4. Differences between (a) SLA and (b) DLC curing methods (from [53], license: CC BY 4.0).

cleaner finish than the FDM method without leaving stacking marks (Fig. 5) [13]. Note that the LCD method is frequently used due to its ability to print at high speed from surface emitting and low cost due to its simple structure. In addition, the LCD system has advanced to higher resolutions, and affordable consumer grade 4K and 8K devices are available.

2.2. Generating printing data

Generally, in the 3D printing process, a cross-sectional view is first created from the model data using slicer software [20, 34, 59]. In the following, we describe the 3D data file format required for printing and how to create 2D data using slicer software.

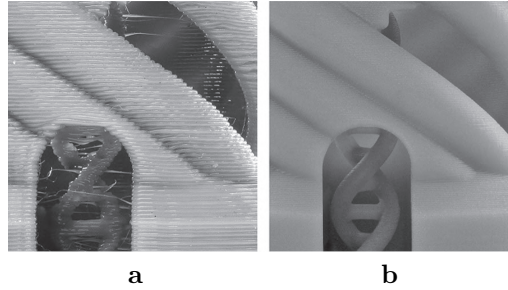


Fig. 5. Differences in output results between (a) FDM and (b) SLA methods (from [13]; available also from a number of other blogs).

```

solid name
{
  facet normal ninjnk
  outer loop
    vertex v1x v1y v1z
    vertex v2x v2y v2z
    vertex v3x v3y v3z
  endloop
endfacet
endsolid name

```

Fig. 6. STL file specification (created by the author on the basis of [16]).

2.2.1. Production of 3D objects by ultraviolet light

To create a 3D object, it is first necessary to model the 3D data. Various software tools are available for 3D modeling, from freely available software to high-performance commercial software. In addition, there are various file formats for the 3D data; however, the STL and OBJ formats are widely used because they are highly versatile [11, 40].

The STL and OBJ formats are based on vector information, e.g., vertex information; thus, the data size is not related to the size of the object (Fig. 6). However, the amount of data increases for precise shapes, which inevitably leads to increased data sizes [29, 30]. In other words, due to the high-definition of 3D printers, their use has increased, and the data size tends to increase even with this format, which is vector information [27].

The size of the data to be printed varies greatly depending on the size and lamination pitch of the production (Fig. 7). For example, if the stacking width is 100 μm , which is common, 1000 stacks are required to print a 10 cm object, which means that 1000 cross-sectional views must be generated. In recent years, it has become possible to achieve a lamination process of 25 μm or less, which inevitably increases the amount of data [26].

2.2.2. Creating slice data for printing

As mentioned previously, 3D printers create 3D objects by stacking layers; thus, it is necessary to create cross-sectional views corresponding to each layer from the model data

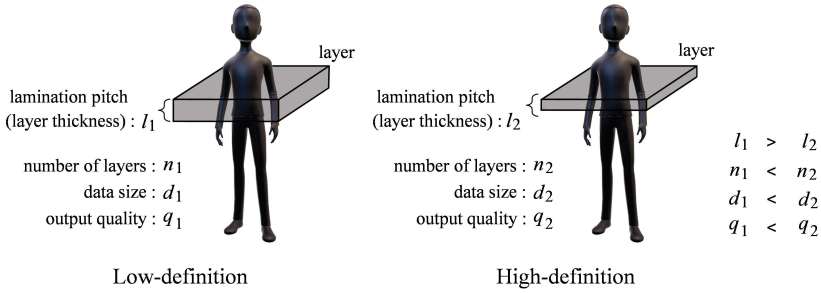


Fig. 7. Lamination pitch.

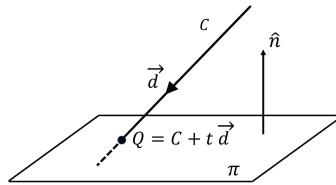


Fig. 8. Intersection between a straight line and a plane (from [23], license: CC BY 4.0).

(Fig. 1) [34,56,59]. Generally, slicer software is used to create a cross-sectional view that is compatible with a given 3D printer; however, some modeling software has a slicing function that creates a cross-sectional view [9].

Various methods can be used to create a cross section for 3D printing; however, the most common method is to use the intersection of a plane and a straight line (Fig. 8) [23]. In addition, the slicing algorithm using the intercept theorem for triangular mesh location information has also been studied [51]. Furthermore, a similar study is developing slice algorithm that utilizes the geometric topology information of the STEP model conforming to ISO10303 [54].

However, it has been demonstrated that the process of creating cross sections is generally time consuming. Recently, as the resolution has increased, converting 3D data to 2D data incurs an extremely long processing time; thus, methods that use a general-purpose graphics processing unit (GPGPU) are being actively investigated [20,34,59].

2.2.3. Slice data file format

The model data are converted into multiple pieces of 2D image data by the slicer software; however, when outputting, it is necessary to select a file format that is suitable for the given 3D printer. Common slicer software used for conversion, e.g., CHITUBOX, supports a wide range of formats, including PWS and CWS, which are used by several

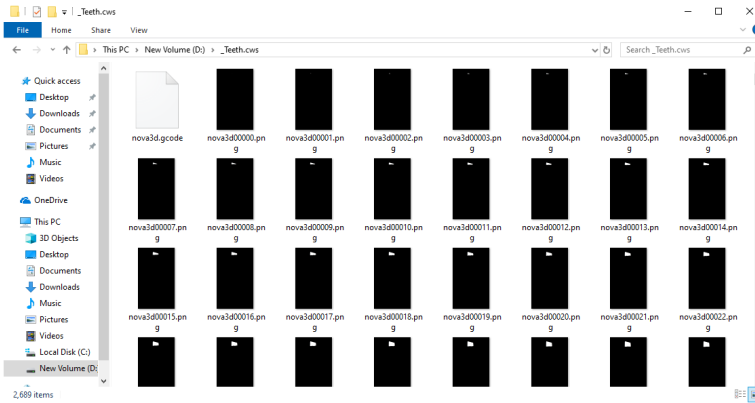


Fig. 9. CWS file format in which the image file and print settings file are ZIP compressed.

common 3D printers, e.g., ELEGOO, Anycube, and Flashforge printers, as well as the more generic SLC [12,43]. There is also a general-purpose storage method in which sliced images are compressed in ZIP format with various information included, e.g., lamination pitch [4]. Here, the 2D data can be saved as a raster format image, e.g., CWS format (Fig. 9), or saved by internal and external boundary polylines, e.g., SLC files [10].

Generally, when printing a 3D object from multiple images, specific information is required, e.g., lamination pitch; thus, a metadata file containing the information required for printing is frequently added (Fig 10). However, some devices can print 3D objects directly from multiple 2D image data by setting them at the time of printing [41,57].

2.3. Template Matching

Template Matching is one of the famous and fundamental Computer Vision technique for Object Detection/Recognition. Via Template Matching, the computer figures out whether the test image contains a given template in it or not [36]. In most situations the sum of absolute differences (SAD) and Sum of squared differences (SSD) are used as similarity measures to find the best similar block [48].

2.3.1. Pixel Differencing method using SAD

SAD is a technique for evaluating the similarity between two same size regions, and widely used in stereovision, optical flow, motion estimation and so on [45]. The total difference between the two signatures is calculated by adding the absolute value of differences between the samples. The match with the smallest total difference is taken as

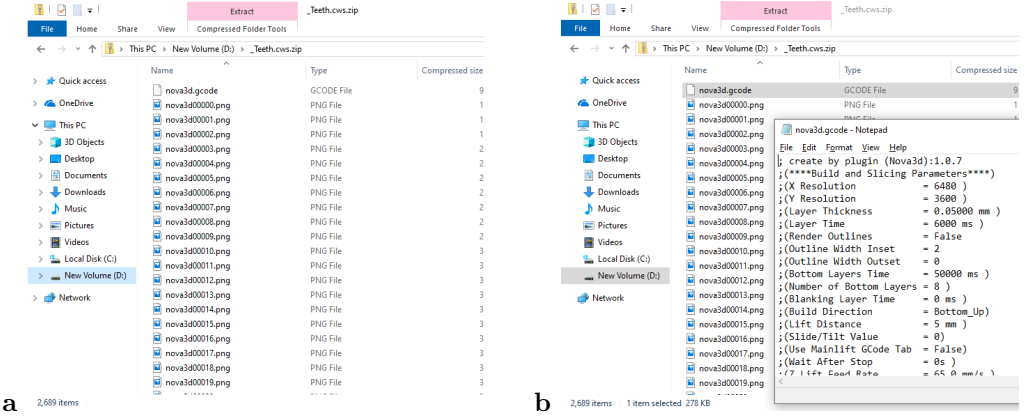


Fig. 10. Content of files required for printing (created by the slicer from model images). (a) Image data and metadata in print files. (b) Content of the metadata generated by the slicer.

best [14]. This method can be expressed as follows:

$$S_{\text{SAD}}(x, y) = \sum_{x=1}^M \sum_{y=1}^N \left| T(x, y) - I(x + u, y + v) \right|,$$

where M is size of rows in reference image and N is size of column while u and v are variable, shift component along x -direction and y -direction, respectively. $T(\cdot, \cdot)$ and $I(\cdot, \cdot)$ represent the value of the pixel at a location. If the images exactly match, the result will be zero [33].

2.3.2. Pixel differencing method using SSD

SSD is one of measures of match that is based on pixel by pixel intensity differences between the two images [47]. It calculates the summation of squared for the product of pixels subtraction between two images [55]. This method can be expressed as follows:

$$S_{\text{SSD}}(x, y) = \sum_{x=1}^M \sum_{y=1}^N (T(x, y) - I(x + u, y + v))^2.$$

The SSD score, like the SAD score, must be zero if the images are the same, pixel by pixel.

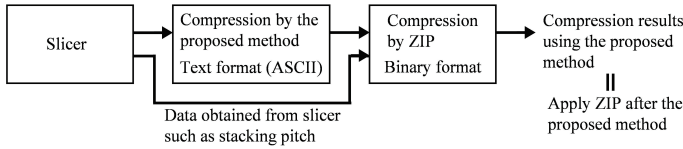


Fig. 11. Process flow of the proposed method.

3. Proposed Method

In this study, as an initial verification, we perform a data compression method that focuses on continuous data unique to 3D printers. Specifically, we target stereolithography, which can high-resolution products, and we attempted to perform lossless compression that can be converted into printable data.

Generally, the data printed by a 3D printer are characterized by relatively similar upper and lower layers. In addition, the objects floating in the air cannot be directly printed by a 3D printer because they must be suspended from another object that serves as a base. Therefore, due to these characteristics, as an initial experiment, we verify whether it is possible to reduce the data size of the 3D model required for printing.

Here, in consideration of versatility, the input data were 2D slice image data and the metadata required for printing obtained from a slicer. Thus, the slice data and metadata were converted from the 3D model by the slicer in advance. This was a preliminary experiment; thus, we output the intermediate results as text data to facilitate verification of the proposed method. Finally, the results of the proposed method, which were compressed in ZIP format including metadata, were compared with other file formats (Fig. 11).

3.1. Creating difference data from sliced data

The slicer converted the 3D data into 2D data for each layer, and scanned the generated 2D data for each XY coordinate (Fig. 12). Here, the scanned image data contained binary information; thus, it was possible to easily obtain internal and external information scanned (Fig. 13). In other words, it was possible to identify positions that change from outside to inside or inside to outside; thus, the differences in positions were stored sequentially. In this verification, the difference of each X coordinate was obtained in order along the y-axis (Fig. 14), and the same process was performed for all layers to obtain the difference data at each layer.

As mentioned previously, objects fabricated using a 3D printer frequently have approximate top and bottom layers, and it is impossible to print objects that are suspended in air that do not touch the print surface or are not connected to the object being printed.

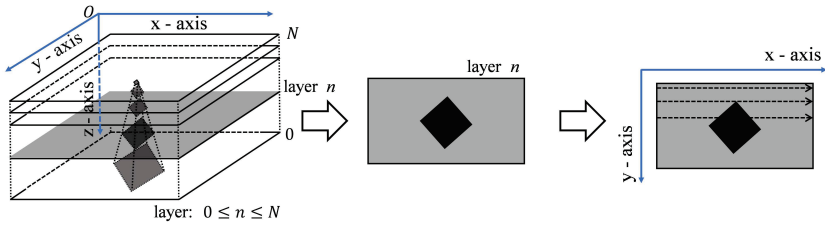


Fig. 12. Scanning method for internal and external determination.

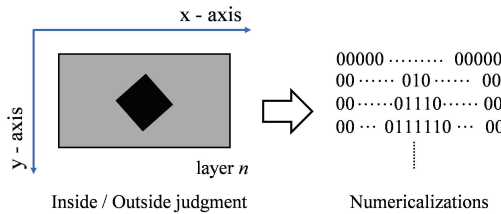


Fig. 13. Judgment by binarization.

Thus, some data will be continuous even in the upper and lower layers. Therefore, after obtaining the difference data in each layer, the difference based on the Z-axis was considered. Note, as this study is an initial validation, it was decided to use the index of the data in the upper layer if it is equal to the upper layer (Fig. 15 and Fig. 16). In other words, this is the case of perfect matching in SSD, and this method is an extended solution.

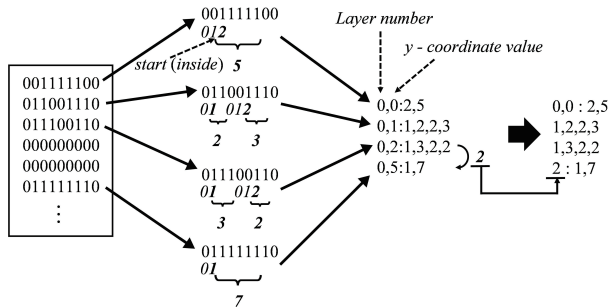


Fig. 14. Numericalization of continuous data for each layer.

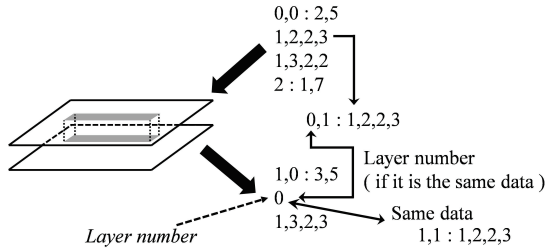


Fig. 15. If the *y*-axis values are the same data.

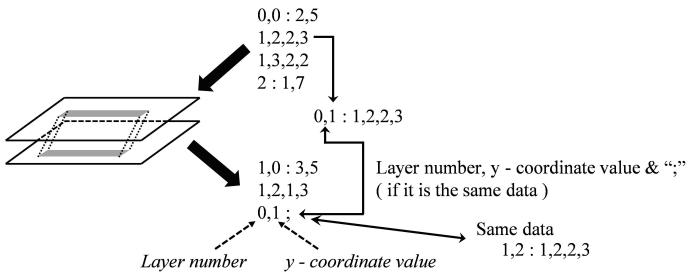


Fig. 16. If the *y*-axis values are different data.

3.2. Compression of numerical data by proposed method including metadata

For the final comparison, the text data created by the proposed method and other data, e.g., laminate pitch data obtained by the slicer, were combined and compressed together in ZIP format, and compared with each file format (Fig. 11). Note that this propose method utilizes lossless compression; thus, it can be changed to a format that can be printed with a 3D printer, e.g., the CWS format.

4. Experiments

Based on the proposed method, we verified whether there any difference could be observed in the data compression results compared with other formats. Here, the experiments were conducted assuming a large print size (298 mm × 164 mm × 30 mm) that can be output by a conventional optical 3D printer. Thus, the size of each experimental 3D model was increased or decreased to the maximum size within the above range. In addition, the lamination pitch was set to 50 μm. Note that the size of the STL and OBJ files used in the experiment could be changed easily because the data are based on coordinate information.

Tab. 1. Specifications of experimental hardware and software.

Specification	Value
OS	Windows 11 Professional (Build 22621)
Compiler	Javac 19.0.1 (Oracle JDK 19)
Slicer	CHITUBOX V1.8.0 Beta Nova3D Plugin 1.07 anycubic_plugin
CPU	Intel Core i9 - 12900H (2.5 GHz)
GPU	GeForce RTX 3070 Ti Laptop
Memory	64 GB (DDR5)

As mentioned earlier, printing requires a sliced 2D image of each layer. Generally, there are several software programs that convert such 3D data into 2D images, but this time, we decided to use CHITUBOX, which is freely available and compatible with many 3D printers [12]. CHITUBOX, like a general slicer, also generates configuration files necessary for printing. Configuration files other than images created with the CHITUBOX are also saved together with the data in the proposed method during ZIP compression. Therefore, in the evaluation, we will compare proposed file format with other file formats as printable files. The file formats to be compared were OBJ files or STL files that are model data, and the formats used by general stereolithography 3D printers Anycube, Phrozen, NOVA3D, and general-purpose SLC formats.

In this experiment, compression was applied and validated on eight shapes. The experimental hardware and software are described in Tab. 1. In the following results, the results of the proposed method are referred to as apply ZIP after the proposed method. As mentioned earlier, the result of the ZIPping process applied after the proposed method can be easily decompressed and converted to a printable file because it contains the control file for the 3D printer.

4.1. King kong bust

The complex shape model created in OBJ format shown in Fig. 17 was used for verification, and a portion of the cross-sectional view of Fig. 17 is shown in Fig. 18. The difference in data size for each file format is shown in Tab. 2. Note that this model contained 7,560,074 vertices and 2,985 layers.

4.2. Triceratops

The complex shape model created in STL format shown in Fig. 19 was used for verification, and a portion of the cross-sectional view of Fig. 19 is shown in Fig. 20. The



Fig. 17. Complex King kong bust 3D model [28] (Royalty Free No Ai License from “cgtrader | General Terms and Conditions | 21A. Royalty Free License”).

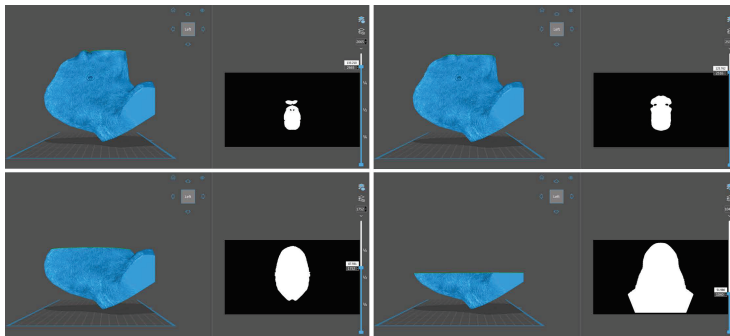


Fig. 18. Part of the cross-sectional view of the King kong bust 3D model.

difference in data size for each file format is shown in Tab. 3. Note that this model contained 1 278 259 vertices and 2 198 layers.

Tab. 2. Data size of various file formats and data size after compression by the proposed method (King kong bust: 7 560 074 vertices and 2 985 layers).

File type	File size [bytes]
OBJ (original) [28]	529 351 777
OBJ (after ZIP compression)	158 387 269
PWS	562 164 661
Photon	566 937 601
ZIP (2D image layers + metadata)	60 117 977
slc	204 691 406
phz	611 932 743
cws	56 439 318
pwx	56 940 340
svgx	381 507 904
Apply ZIP after the proposed method	14 730 722

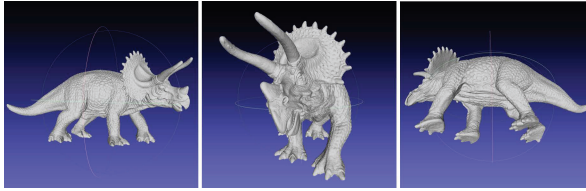


Fig. 19. Complex triceratops 3D model [18] (Royalty Free No Ai License from “cgtrader | General Terms and Conditions | 21A. Royalty Free License”).

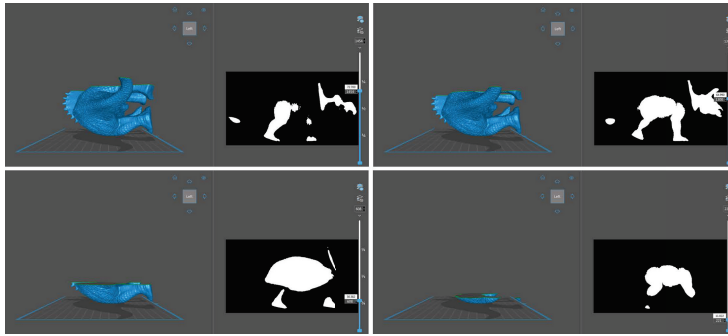


Fig. 20. Part of the cross-sectional view of the triceratops 3D model.

Tab. 3. Data size of various file formats and data size after compression by the proposed method (Triceratops: 1278 259 vertices and 2 198 layers).

File type	File size [bytes]
STL (original) [18]	127 829 284
STL (after ZIP compression)	54 745 386
PWS	414 868 236
Photon	418 408 591
ZIP (2D image layers + metadata)	54 211 661
slc	69 585 774
phz	453 295 532
cws	51 863 803
pwx	46 609 868
svgx	129 712 549
Apply ZIP after the proposed method	20 520 640

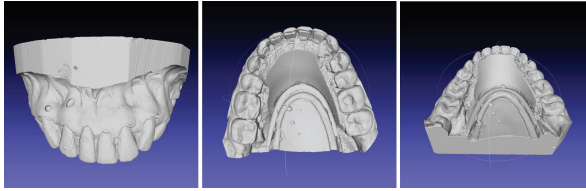


Fig. 21. Complex Plaster cast of teeth 3D model [7] (license: CC BY 4.0, by Artec 3D).

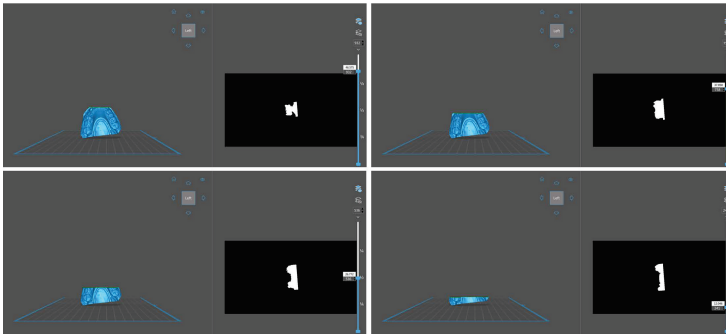


Fig. 22. Part of the cross-sectional view of the Plaster cast of teeth 3D model.

4.3. Plaster cast of teeth

The complex shape model created in OBJ format shown in Fig. 21 was used for verification, and a portion of the cross-sectional view of Fig. 21 is shown in Fig. 22. The difference in data size for each file format is shown in Tab. 4. Note that this model contained 999 998 vertices and 2 688 layers.

4.4. Motorcycle engine HD

The complex shape model created in OBJ format shown in Fig. 23 was used for verification, and a portion of the cross-sectional view of Fig. 23 is shown in Fig. 24. The difference in data size for each file format is shown in Tab. 5. Note that this model contained 999 808 vertices and 3 422 layers.

4.5. Giraffe skull

The complex shape model created in OBJ format shown in Fig. 25 was used for verification, and a portion of the cross-sectional view of Fig. 25 is shown in Fig. 26. The difference in data size for each file format is shown in Tab. 6. Note that this model contained 744 647 vertices and 2 540 layers.

Tab. 4. Data size of various file formats and data size after compression by the proposed method (Plaster cast of teeth: 999 998 vertices and 2 688 layers).

File type	File size [bytes]
OBJ (original) [7]	81 400 987
OBJ (after ZIP compression)	23 484 543
PWS	505 540 271
Photon	509 593 239
ZIP (2D image layers + metadata)	38 397 319
slc	66 177 342
phz	550 203 080
cws	34 151 966
pwx	52 675 512
svgx	120 725 818
Apply ZIP after the proposed method	7 493 959

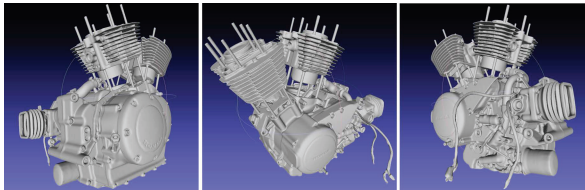


Fig. 23. Complex Motorcycle engine HD 3D model [6] (license: CC BY 4.0, by Artec 3D).

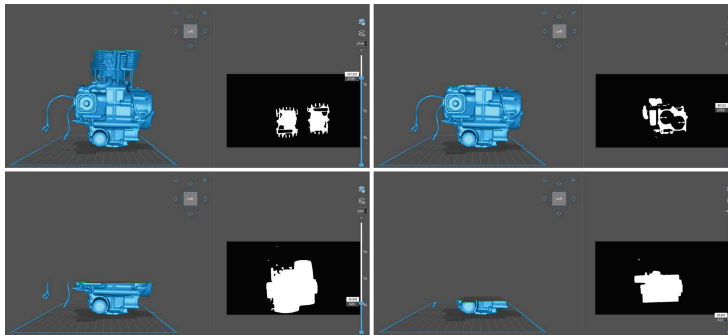


Fig. 24. Part of the cross-sectional view of the motorcycle engine HD 3D model.

Tab. 5. Data size of various file formats and data size after compression by the proposed method (Motorcycle engine HD: 999 808 vertices and 3 422 layers).

File type	File size [bytes]
midrule OBJ (original) [6]	83 283 059
OBJ (after ZIP compression)	30 086 007
PWS	648 374 184
Photon	653 719 793
ZIP (2D image layers + metadata)	63 941 331
slc	89 001 582
phz	712 843 758
cws	62 687 086
pwx	85 320 032
svgx	164 447 945
Apply ZIP after the proposed method	26 801 339

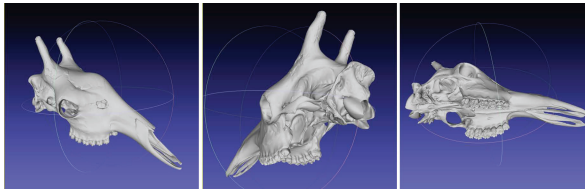


Fig. 25. Complex giraffe skull 3D model [5] (license: CC BY 4.0, by Artec 3D).

4.6. Model house for train free 3D print model

The complex shape model created in OBJ format shown in Fig. 27 was used for verification, and a portion of the cross-sectional view of Fig. 27 is shown in Fig. 28. The

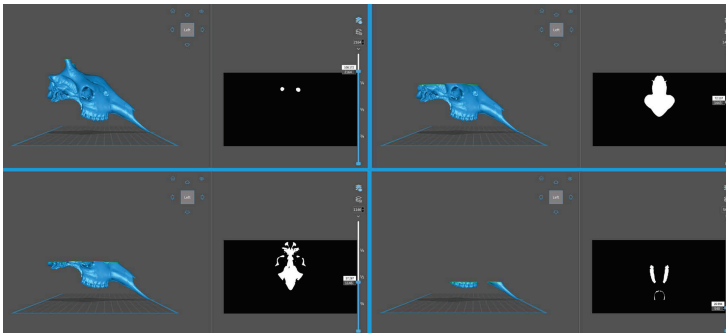


Fig. 26. Part of the cross-sectional view of the giraffe skull 3D model.

Tab. 6. Data size of various file formats and data size after compression by the proposed method (Giraffe skull: 744647 vertices and 2540 layers).

File type	File size [bytes]
OBJ (original) [5]	61 013 447
OBJ (after ZIP compression)	20 083 573
PWS	475 426 789
Photon	479 369 665
ZIP (2D image layers + metadata)	29 136 537
slc	45 616 214
phz	510 800 838
cws	27 715 959
pwx	44 495 436
svgx	81 846 149
Apply ZIP after the proposed method	7 718 206

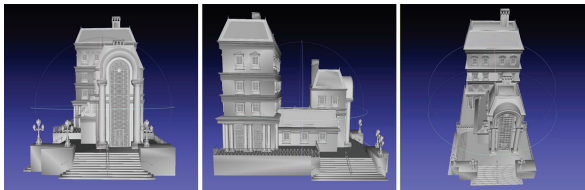


Fig. 27. Complex model house for train free 3D print model [39] (Royalty Free No Ai License from “cgtrader | [General Terms and Conditions](#) | 21A. Royalty Free License”).

difference in data size for each file format is shown in Tab. 7. Note that this model contained 339834 vertices and 3371 layers.

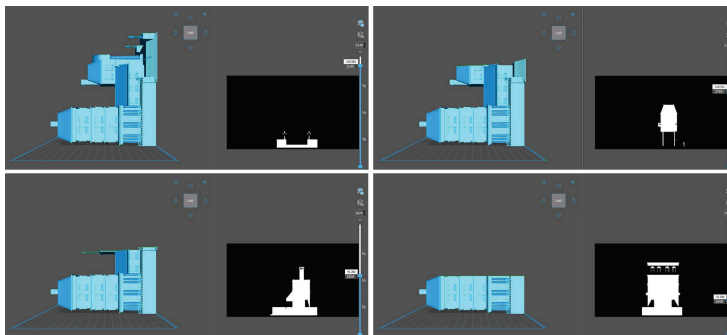


Fig. 28. Part of the cross-sectional view of the model house for train free 3D print model.

Tab. 7. Data size of various file formats and data size after compression by the proposed method (Model house for train free 3D print model: 339 834 vertices and 3371 layers).

File type	File size [bytes]
OBJ (original) [39]	55 109 933
OBJ (after ZIP compression)	14 153 127
PWS	634 917 533
Photon	639 893 543
ZIP (2D image layers + metadata)	20 279 624
slc	24 930 638
phz	688 854 398
cws	19 942 696
pwx	68 048 184
svgx	47 438 712
Apply ZIP after the proposed method	1 184 268

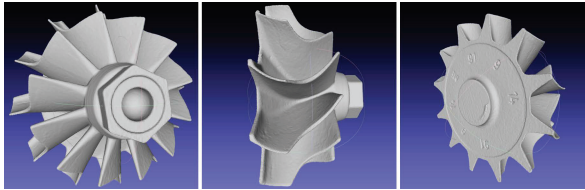


Fig. 29. Complex turbine 3D model [8] (license: CC BY 4.0, by Artec 3D).

4.7. Turbine

The complex shape model created in OBJ format shown in Fig. 29 was used for verification, and a portion of the cross-sectional view of Fig. 29 is shown in Fig. 30. The difference in data size for each file format is shown in Tab. 8. Note that this model contained 150 002 vertices and 1 856 layers.

4.8. Stool

The complex model created in OBJ format shown in Fig. 31 was used for verification, and a portion of the cross-sectional view of Fig. 31 is shown in Fig. 32. The difference in data size for each file format is given in Tab. 9. Note that this model contained 106 702 vertices and 4 237 layers.

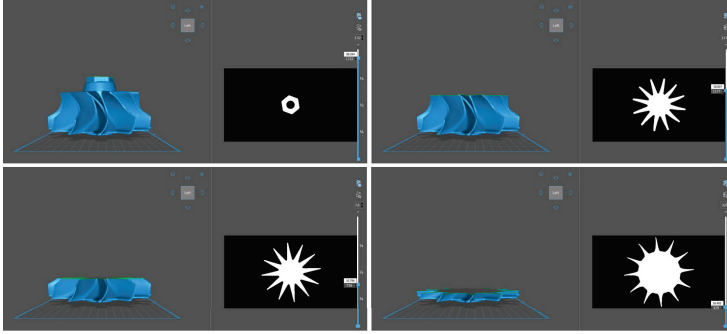


Fig. 30. Part of the cross-sectional view of the turbine 3D model.

Tab. 8. Data size of various file formats and data size after compression by the proposed method (Turbine: 150 002 vertices and 1 856 layers).

File type	File size [bytes]
OBJ (original) [8]	15 657 744
OBJ (after ZIP compression)	5 564 081
PWS	352 112 095
Photon	355 137 383
ZIP (2D image layers + metadata)	55 480 606
slc	27 272 182
phz	389 967 957
cws	53 156 143
pwx	50 088 630
svgx	50 166 142
Apply ZIP after the proposed method	22 291 844

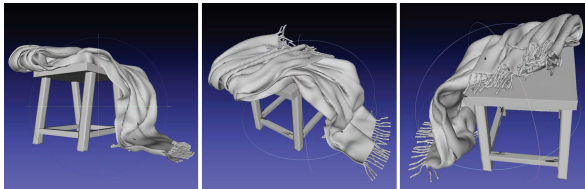


Fig. 31. Complex stool 3D model [22] (Royalty Free No Ai License from “cgtrader | General Terms and Conditions | 21A. Royalty Free License”).

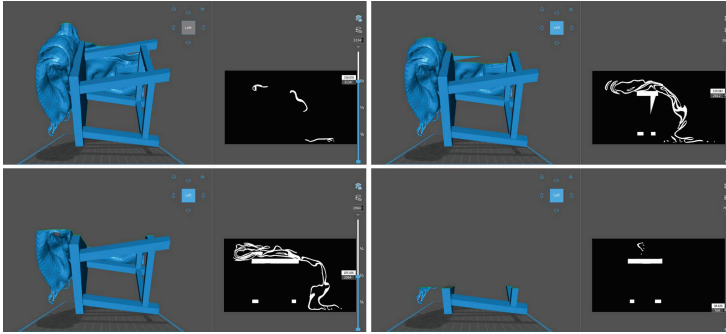


Fig. 32. Part of the cross-sectional of the stool 3D model.

Tab. 9. Data size of various file formats and data size after compression by the proposed method (Stool: 106 702 vertices and 4 237 layers).

File type	File size [bytes]
OBJ (original) [22]	15 616 745
OBJ (after ZIP compression)	4 096 359
PWS	810 316 247
Photon	816 776 523
ZIP (2D image layers + metadata)	109 642 159
slc	45 733 614
phz	901 592 107
cws	115 805 120
pwx	131 373 624
svgx	87 396 109
Apply ZIP after the proposed method	50 387 383

5. Discussion

For complex shapes with a large number of vertices, the proposed method obtained a higher compression ratio than the compared file formats. This is due to the large number of vertices required to print complex shapes, resulting in large original file sizes. However, we found that the proposed method was not necessarily effective for shapes with a small number of vertices. For files such as OBJ and STL, which create models based on vertex information, the file size is generally smaller when the number of vertices is small. Therefore, in the case of a file in a general printing format that is saved as a raster image for each layer, the file size will be larger than data with a small number of vertices. In other words, in such a file with a small number of vertices, the proposed method using the difference method for slice images inevitably results in a large file size.

On the other hand, as the 3D model becomes more complex, the number of vertices increases and the original file size also increases. In such cases, the proposed method using the difference method based on slice data would have been more effective. Thus, the proposed method is more effective for compressing large files that require a GPGPU when creating slice data.

Since this is an initial experiment, the slice data are output in ASCII format like OBJ files, but it is thought that the compression ratio can be further improved by using a binary format. In addition, this method does not consider the features of the image itself. It is considered that it is possible to further increase the compression ratio by utilizing shape similarity such as symmetry.

6. Conclusion

With the recent advancements in the case of LCD 3D printers, it has become possible to print extremely precise objects due to the high-resolution of the liquid crystal panels. As a result, the amount of data required for 3D printing has increased significantly. In contrast, 2D slice data, i.e., a cross-sectional view used for printing, tends to have continuity in the data (unlike general images). Thus, by focusing on such features, a high compression ratio can be obtained even with a method that uses a relatively simple difference.

In this study, we performed a preliminary verification experiment, where the compression results obtained by the proposed method were output as text data. In addition, to convert the data to a printable file format, the data obtained from the slicer were used without modification, and the final result summarized by ZIP was obtained as the compression ratio. We found that the proposed difference method is effective; thus, it is considered possible to improve the compression ratio further by applying a new compression method, e.g., a binary format.

Currently, the proposed method is effective for data with a large number of vertices that may require GPGPU processing for analysis. In other words, it is effective for large files; however, in future, we would like to improve the compression ratio for small files with a relatively small number of vertices.

References

- [1] 3DWithUs. The Best Resin 3D Printers 2023 – Comparison & Innovations, 2023. <https://3dwithus.com/best-resin-3d-printers>, [Accessed: 17 Mar 2023].
- [2] A. Ambrosi and M. Pumera. 3D-printing technologies for electrochemical applications. *Chemical Society Reviews*, 45(1):2740–2755, 2016. doi:10.1039/C5CS00714C.
- [3] F. Arceo. The 8 best slicers for resin 3D printers (SLA, LCD, DLP). In: 3D Solved. <https://3dsolved.com/best-slicers-for-resin-3d-printers-sla-lcd-dlp/>, [Accessed: 11 Mar 2023].

- [4] Area 515. Photonic3D. In: GitHub. <https://github.com/area515/Photonic3D>, [Accessed: 11 Mar 2023].
- [5] Artec 3D. Giraffe skull. <https://www.artec3d.com/3d-models/giraffe-skull>, [Accessed: 11 Mar 2023].
- [6] Artec 3D. Motorcycle engine HD. <https://www.artec3d.com/3d-models/motorcycle-engine-hd>, [Accessed: 11 Mar 2023].
- [7] Artec 3D. Plaster cast of teeth. <https://www.artec3d.com/3d-models/plaster-cast-teeth>, [Accessed: 10 Mar 2023].
- [8] Artec 3D. Turbine. <https://www.artec3d.com/3d-models/turbine>, [Accessed: 11 Mar 2023].
- [9] AUTODESK Support. How to create a sketch of a cross-section of a 3D model in Fusion 360. AUTODESK, 2020. <https://www.autodesk.com/support/technical/article/caas/sfdcarticles/sfdcarticles/How-to-create-a-sketch-of-a-cross-section-of-a-3D-Model-in-Fusion-360.html>, [Accessed 15 Mar 2023].
- [10] P. Bourke. SLC – SLiCe format. In: Paul Bourke personal/professional web pages. <http://paulbourke.net/dataformats/slc/>, [Accessed: 11 Mar 2023].
- [11] E. Champion and H. Rahaman. Survey of 3D digital heritage repositories and platforms. *Virtual Archaeology Review*, 11(23):1–15, 2020. doi:10.4995/var.2020.13226.
- [12] CHITUBOX. All-in-one SLA/DLP/LCD preprocessing slicer. In: CHITUBOX Docs. <https://www.chitubox.com/en/page/chitubox-vs>, [Accessed: 12 Mar 2024].
- [13] A. O. Demirel. FDM vs SLA: 3D baskıda hangi teknolojiyi seçmelisiniz? In: 3dörtgen Blog, 2023. <https://blog.3dortgen.com/fdm-vs-sla-3d-baskida-hangi-teknolojiyi-secmelisiniz/>, [Accessed: 15 Jan 2024].
- [14] S. Divakar and R. V. S. Satyanarayana. Comparison of matching algorithms for MST radar data. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, 2(12):6271–6278, 2013. https://www.ijareeie.com/upload/2013/december/46_Comparison.pdf.
- [15] P. Dudek. FDM 3D printing technology in manufacturing composite elements. *Archives of Metallurgy and Materials*, 58(4):1415–1418, 2013. doi:10.2478/amm-2013-0186.
- [16] Ebrary.net. Additive manufacturing and 3D printing technology: Principles and applications, chapter: Format specifications. https://ebrary.net/158117/engineering/format_specifications, [Accessed: 11 Mar 2023].
- [17] R. A. I. Elzain. Large-scale 3D printing – Market analysis. Technical Report 47, Constructor University Technical Reports, 9 Dec 2022. <http://nbn-resolving.org/urn:nbn:de:gbv:579-opus-1011273>.
- [18] fabelar. Triceratops FREE printable Free 3D print model. In: cgtrader. <https://www.cgtrader.com/free-3d-print-models/science/biology/triceratops-free-printable>, [Accessed: 11 Mar 2023].
- [19] FLASHFORGE. Explaining the types and features of stereolithography 3d printers! (in Japanese), 23 Jun 2022. https://flashforge.jp/benefit_list/about-stereolithography/, [Accessed: 11 Mar 2023].
- [20] FormWare. Formware 3D: 3D printing slicing algorithms, their speed and applications, 2022. <https://www.formware.co/article/SlicingSpeed>, [Accessed: 11 Mar 2023].
- [21] FormWare. Formware 3D: Powerful 3D Printing slice and support generation software, 2022. <https://www.formware.co/slicer>, [Accessed: 10 Mar 2023].

- [22] giuliazocca. Stool with blanket 3D model – seat free 3D model. In: cgtrader. <https://www.cgtrader.com/free-3d-models/furniture/chair/stool-with-blanket-3d-model-seat>, [Accessed: 11 Mar 2023].
- [23] X. Han, Z. Zhan, X. Song, and L. Cui. An additive manufacturing direct slicing algorithm based on a step model. *Electronics*, 11(10):1582, 2022. doi:10.3390/electronics11101582.
- [24] S. Hill. How to slice STL files in Cura | Step by step guide. In: fabheads, 2022. <https://www.crealitycloud.com/blog/tutorials/how-to-slice-stl-files-in-cura>, Tutorial. [Accessed: 17 Mar 2023].
- [25] Home3Ddo. Difference of stereolithography method “SLA, DLP, LCD, LFS, CLIP” 3D printer (in Japanese), 5 May 2023. <https://home3ddo.blog.jp/3dprinter-sla>, [Accessed: 14 Aug 2023].
- [26] i-MAKER. What is the level of stacking for 3D printer optical fabrication? (in Japanese), 2021. <https://i-maker.jp/blog/sla-stacking-pitch-16374.html>, [Accessed: 11 Mar 2023].
- [27] L. F. J. Hao and R. E. Williams. An efficient curvature-based partitioning of large-scale STL models. *Rapid Prototyping Journal*, 17(2):116–127, 2011. doi:10.1108/13552541111113862.
- [28] D. Joldoshbek. King kong bust Free 3D print model. In: cgtrader. <https://www.cgtrader.com/free-3d-print-models/games-toys/toys/king-kong-f0ab707c-9617-4d32-9f56-5f287f40bcd1>, [Accessed: 10 Mar 2023].
- [29] J. H. Kim, O. H. P. Baptista, A. P. Ayres, R. L. B. da Silva, J. F. Lima, et al. Accuracy comparison among 3D-printing technologies to produce dental models. *Applied Sciences*, 12(17):8425, 2022. doi:10.3390/app12178425.
- [30] B. King, A. Rennie, and G. Bennett. An efficient triangle mesh slicing algorithm for all topologies in additive manufacturing. *The International Journal of Advanced Manufacturing Technology*, 112(3):1023–1033, 2018. doi:10.1007/s00170-020-06396-2.
- [31] M. Kolla. What is the role of slicing in 3D printing? In: fabheads, 2021. <https://fabheads.com/blogs/what-is-the-role-of-slicing-in-3d-printing/>, [Accessed: 10 Mar 2023].
- [32] A. A. Konta, M. Garcia-Piña, and D. R. Serrano. Personalised 3D printed medicines: Which techniques and polymers are more successful? *Bioengineering*, 4(4):79, 2017. doi:10.3390/bioengineering4040079.
- [33] A. Kumar, A. Joshi, A. Kumar, and A. Mittal. Template matching application in geo-referencing of remote sensing temporal image. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 7(2):201–210, 2014. doi:10.14257/IJSIP.2014.7.2.19.
- [34] X. Lai and Z. Wei. Slicing algorithm and partition scanning strategy for 3D printing based on GPU parallel computing. *Materials*, 14(15):4297, 2021. doi:10.3390/ma14154297.
- [35] A. O. Laplume, B. Petersen, and J. M. Pearce. Global value chains from a 3D printing perspective. *Journal of International Business Studies*, 47:595–609, 2016. doi:10.1057/jibs.2015.47.
- [36] D. Lee. Template Matching. In: velog, as ldw200012.log, 12 Jul 2021. <https://velog.io/@ldw200012/Template-Matching>, [Accessed: 3 Apr 2023].
- [37] Y. Lim. *Influence of Different Postprocessing Rinsing Agents on the Manufacturing Accuracy of Dental Models Printed by LCD Resin 3D Printer*. Master’s thesis, Texas A & M University, 2022. [Accessed: 11 Mar 2023]. <https://hdl.handle.net/1969.1/197435>.
- [38] M. Long. How to slice 3D printable files for filament and resin printers. In: electro{maker}, 2 Sep 2019. <https://www.electromaker.io/tutorial/blog/how-to-slice-3d-printable-files-for-filament-and-resin-printers>, Tutorial. [Accessed: 11 Mar 2023].

- [39] T. Ly. Model house for train Free 3D print model. In: cgtrader. <https://www.cgtrader.com/free-3d-print-models/hobby-diy/other/model-house-for-train>, [Accessed: 11 Mar 2023].
- [40] K. McHenry and P. Bajcsy. An overview of 3D data content, file formats and viewers. Tech. Rep. isda08-002, National Center for Supercomputing Applications, Image Spatial Data Analysis Group, 31 Oct 2008. <https://isda.ncsa.illinois.edu/peter/publications/techreports/2008/NCSA-ISDA-2008-002.pdf>.
- [41] MiiCraft. User manual: MiiCraft 125. In: “MiiCraft | Support | User Manual | MiiCraft 125” <https://miicraft.com/user-manual/>. [Accessed: 11 Mar 2023].
- [42] F. M. Mwema and E. T. Akinlabi. Basics of fused deposition modelling (FDM). In: *Fused Deposition Modeling: Strategies for Quality Enhancement*, pp. 1–15. Springer International Publishing, Cham, 2020. doi:10.1007/978-3-030-48259-6_1.
- [43] R. K. Müller. MSLA Anycubic Photon Mono 4K. XYZdims.com, 2023. <https://xyzdims.com/3d-printers/msla-anycubic-photon-mono-4k/>, [Accessed: 12 Aug 2023].
- [44] T. Ngo, A. Kashani, G. Imbalzano, K. Nguyen, and D. Hui. Additive manufacturing (3D printing): A review of materials, methods, applications and challenges. *Composites Part B: Engineering*, 143:172–196, 2018. doi:10.1016/j.compositesb.2018.02.012.
- [45] H. Niitsuma and T. Maruyama. Sum of absolute difference implementations for image processing on FPGAs. In: *Proc. 2010 International Conference on Field Programmable Logic and Applications*, pp. 167–170. Milan, Italy, 31 Aug – 02 Sep 2010. doi:10.1109/FPL.2010.40.
- [46] ocf81. Open Source 3D Printing – Deel 3, 2021. https://gathering.tweakers.net/forum/list_messages/2059884, [Accessed: 17 Mar 2023].
- [47] S. Ourselin, R. Stefanescu, and P. Xavier. Robust registration of multi-modal images: Towards real-time clinical applications. In: T. Dohi and R. Kikinis, eds., *Medical Image Computing and Computer-Assisted Intervention – Proc. MICCAI 2002*, pp. 140–147. Springer Berlin Heidelberg, Tokyo, Japan, 25–28 Sep 2002. doi:10.1007/3-540-45787-9_18.
- [48] A. V. Paramkusam and V. S. K. Reddy. An efficient fast full search block matching algorithm with SSD criterion. In: *Proc. 2011 Annual IEEE India Conference*, pp. 1–6. Hyderabad, India, 16–18 Dec 2011. doi:10.1109/INDCON.2011.6139485.
- [49] G. N. Pham, S.-H. Lee, O.-H. Kwon, and K.-R. Kwon. A 3D printing model watermarking algorithm based on 3D slicing and feature points. *Electronics*, 7(2):23, 2018. doi:10.3390/electronics7020023.
- [50] T. Pohlmann. Patent and litigation trends for 3D printing technologies. In: IAM, Law Business Research, 12 Mar 2019. <https://www.iam-media.com/article/patent-and-litigation-trends-3d-printing-technologies>, [Accessed: 11 Mar 2023].
- [51] F. Schurig. Slicing Algorithms for 3D-Printing. In: *Proseminar: H4ck3r’s D3light*. Technische Universität München, TUM School of Computation, Information and Technology, Chair for Decentralized Information Systems and Data Management, 2015. https://blog.hk-fs.de/wp-content/uploads/2015/08/Paper_Fabian_Schurig_3D_Printing.pdf.
- [52] N. Shahrubudin, T. Lee, and R. Ramlan. An overview on 3D printing technology: Technological, materials, and applications. *Procedia Manufacturing*, 35:1286–1296, 2019. doi:10.1016/J.PROMFG.2019.06.089.
- [53] J. Slaymaker, J. Woolley, and S. Hirani. 3D printing in orthodontics: An introduction. *SVOA Dentistry*, 4(6):229–241, 2023. doi:10.58624/SVOADE.2023.04.0155.
- [54] P. Stavropoulos, K. Tzimanis, T. Souflas, and H. Bikas. Knowledge-based manufacturability assessment for optimization of additive manufacturing processes based on automated feature recognition from CAD models. *The International Journal of Advanced Manufacturing Technology*, 122(2):993–1007, 2022. doi:10.1007/s00170-022-09948-w.

- [55] L. D. Stefano and S. Mattoccia. Fast template matching using bounded partial correlation. *Machine Vision and Applications*, 13:213–221, 2003. doi:10.1007/s00138-002-0070-5.
- [56] E. Taneva, B. Kusnoto, and C. A. Evans. 3D scanning, imaging, and printing in orthodontics. In: F. Bourzgui, ed., *Issues in Contemporary Orthodontics*. IntechOpen, Rijeka, 2015. doi:10.5772/60010.
- [57] Tiger 3D. Tiger3D series: Operators manual, 2017. <https://www.romanoff.com/magenew/media/manuals/Tiger3D-Operators-Manual.pdf>, [Accessed: 11 Mar 2023].
- [58] A. Wang, C. Zhou, Z. Jin, and W. Xu. Towards scalable and efficient GPU-enabled slicing acceleration in continuous 3D printing. In: *Proc. 2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 623–628. Chiba, Japan, 16-19 Jan 2017. doi:10.1109/ASPDAC.2017.7858393.
- [59] X. Zhang, G. Xiong, Z. Shen, Y. Zhao, C. Guo, et al. A GPU-based parallel slicer for 3D printing. In: *Proc. 2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, pp. 55–60. Xi'an, China, 20-23 Aug 2017. doi:10.1109/coase.2017.8256075.

DETERMINATION OF SPHERICAL COORDINATES OF SAMPLED COSMIC RAY FLUX DISTRIBUTION USING PRINCIPAL COMPONENTS ANALYSIS AND DEEP ENCODER-DECODER NETWORK

Tomasz Hachaj¹, Marcin Piekarczyk¹, Łukasz Bibrzycki² and Jarosław Wąs¹

¹*Faculty of Electrical Engineering, Automatics, Computer Science and Biomedical Engineering,
AGH University of Krakow, Kraków, Poland*

²*Faculty of Physics and Applied Computer Science,
AGH University of Krakow, Kraków, Poland*

**Corresponding author: Tomasz Hachaj thachaj@agh.edu.pl*

Abstract In this paper we propose a novel algorithm based on the use of Principal Components Analysis for the determination of spherical coordinates of sampled cosmic ray flux distribution. We have also applied a deep neural network with encoder-decoder (E-D) architecture in order to filter-off variance noises introduced by sampling. We conducted a series of experiments testing the effectiveness of our estimations. The training set consisted of 92 250 images and validation set of 37 800 images. We have calculated mean absolute error (MAE) between real values and estimations. When E-D is applied, the number of cases (estimations) where $MAE < 10$ increases from 48% to 79% for θ and from 62% to 65% for ϕ , $MAE < 5$ increases from 24% to 45% for θ and from 47% to 52% for ϕ , $MAE < 1$ increases from 6% to 9% for θ and from 12% to 16% for ϕ , where θ is the zenith angle, and ϕ is the azimuthal angle. This is a significant change and it demonstrates the high utility of the E-D network use and shows the accuracy of the PCA-based algorithm. We also publish the source code used in our research in order to make it reproducible.

Keywords: cosmic-ray shower, spherical coordinates, detector grid, Principal Component Analysis, Encoder-Decoder network.

1. Introduction

The ultra-high-energy cosmic radiation reaching the Earth's atmosphere is extensively studied because of its still unknown sources and mechanisms of acceleration as well as because of the implications for the dynamics of the atmosphere, life on Earth, interferences with electronic systems and even possible correlations with seismic phenomena [25], to name just a few [5]. Practical exploration of these phenomena is based on observations obtained from specialized detectors capable of detecting secondary jets produced in the atmosphere and reaching the Earth's surface. These jets can arise due to atmospheric collisions of either single primary high-energy particles or cosmic ray ensembles (CRE), i.e., groups of cosmic rays generated in outer space.

Such observations are made primarily by large-scale infrastructure detectors in projects such as Pierre Auger Observatory in Argentina [48], IceCube in Antarctica [1, 2] or Baikal-GVD at Lake Baikal in Russia [6, 44]. Due to their fixed location, such installations have a limited detection area. To expand the possibilities in this regard,

observational structures involving small-scale detectors distributed around the world have been proposed. Projects such as CRAYFIS [32], DECO [51], CREDO [9, 24] incorporate widely available mobile devices like smartphones and webcams into the citizen science paradigm. Projects allied within the CREDO consortium aggregate observations obtained from a variety of simple and low-cost detectors that can be densely distributed over a local area [29]. The novel image processing algorithms make it possible to detect potential cosmic ray events using even off-the-shelves CMOS cameras [21].

Recently, advanced AI methods have been widely used to analyze such data. The potential of such techniques is used both for low-level recognition of detector signals [8, 19, 37, 53] and globally to detect features and correlations for surface or distributed detector networks [13, 28, 31, 45, 49]. The latest scientific and implementation research allows for real-time detection of potentially anomalous particle tracks and similar particle tracks detection in big data image datasets acquired by CMOS sensors [22, 38]. We need simulations to understand spatial distribution of showers [16, 34]. In this paper, we propose an AI-based method to disentangle the directional information from sparse lateral distributions.

AI methods used in detection of ultra-high-energy particles are basically determined by the types of measurement sensors that are used to detect cosmic rays. Stationary observatories such as Pierre Auger, IceCube or Baikal-GVD use a well-defined spatial arrangement of homogeneous detectors. For this reason, this is a fundamentally different research problem than the one considered in our work, which is a feasibility study aimed at proving that the jet geometry can be reconstructed with non-homogeneous but very flexible cosmic rays detectors set-up. This set-up may consist of various types of detectors, both industrial made and amateur constructions, with almost arbitrary localizations which fit into the general notion of Citizen Science paradigm. The paper proves that it is possible and our method is directly useful in the design of small-scale complex cosmic-ray exposure (CRE) secondary flux detection systems which can be a part of distributed cosmic ray observatories like CREDO.

To the best of our knowledge, the results presented in this work are pioneering in the design of small-scale complex CRE secondary flux detection systems, and it is difficult to point out research results with which to contrast our proposed method.

2. Material and methods

2.1. Muon lateral distribution

In order to generate the simulated cosmic shower, we used the approach previously described in the paper [20]. The equation describing the muon distribution is shown in equation (1) (muon is an elementary particle similar to the electron but with a much greater mass). According to this equation the distribution is singular at $r = 0$ for typical

values of the age parameter s , thus needs to be truncated for distances smaller than r_{\min} [12, 17].

$$\rho_{\mu}(r) = \begin{cases} \rho_{\mu}(r_{\min}), & r < r_{\min}, \\ \frac{N_{\mu}}{2\pi r_0^2} \frac{\Gamma(4.5-s)}{\Gamma(s)\Gamma(4.5-2s)} \left(\frac{r}{r_0}\right)^{s-2} \left(1 + \frac{r}{r_0}\right)^{s-4.5}, & r \geq r_{\min}, \end{cases} \quad (1)$$

where s is the age parameter [52], N_{μ} is the shower size parameter, and r_0 describes the characteristic size of the shower.

It should also be taken into account that the cosmic shower can hit the ground at different angles, which can be defined using the spherical coordinates $\theta \in [0, \pi/2]$ and $\phi \in [0, \pi]$ (r is constant), θ is the zenith angle, and ϕ is the azimuthal angle [12]. There is no need to consider a larger range of angles, since the distributions in them at the ground are periodic (see visualizations in [20]). Example distributions depending on the angles of (θ, ϕ) can be seen later in this paper (Figures 6, 7); a detailed analysis can be found in [20]. The paper [20] also presents the relationship between spherical coordinates θ, ϕ and the distribution of projection of cosmic ray shower registered on ground.

Finding a pair of angles (θ, ϕ) on the basis of statistical analysis of the distribution of cosmic shower particles on the earth's surface makes it possible to determine the direction from which the cosmic shower came. The proposition and evaluation of the algorithm for determination of spherical coordinates of sampled cosmic ray flux distribution using data acquired by grid of detectors in the presence of background noise is the main objective of this paper.

2.2. Calculation of angles (θ, ϕ) based on analysis of cosmic ray particle distribution using Principal Components Analysis

Determining the (θ, ϕ) pair of angles allows us to determine from which direction the cosmic rays came.

An Img matrix of size $n \times m$ is given, which represents a discrete measurement grid of cosmic rays. The value in each grid field is equal to the number of particles that have been recorded in that grid field. Suppose there is a recorded cosmic shower inside the grid with a distribution consistent with (1) but with unknown values of (θ, ϕ) . The Algorithm 1 allows us to estimate these unknown angles using PCA [27]. PCA is a popular and proven technique, which in different variants allows analyzing the distribution of data depending on angular parameters for example in geodesic [18, 33, 41, 42] or climatological data [40]. Paper [47] reports successful application of PCA-based analysis of cosmic-ray data for extraction of Hale Cycle. In [43] authors perform fully empirical atmospheric correction of cosmic ray data using PCA. Because of those facts, PCA seems to be promising for analysis of another natural phenomena like analysis of spherical coordinates of cosmic ray flux distribution.

```

Data: Input: Img – input image with dimensions  $n \times m$ 
Result:  $(\theta, \phi)$  – a pair of spherical coordinates.
// Resample image to range [0,255] making values discrete
 = Integer(255 · (Img - min(Img)) / max(Img));
// Create empty list of points
Points ← ∅;
// Iterate through all grid points and add as many times a point with the
    given coordinates as the number of times a particle has been registered
    in it
for  $x = 0; x < n; x ++$  do
    | for  $y = 0; y < m; y ++$  do
    | | for  $c = 0; c < \text{Img}[x, y]; c ++$  do
    | | | Points.Append( $(x, y)$ );
    | | end
    | end
end
// mean1, mean2 - mean points value; v1, v2 - PCA components; exv1, exv2 -
    explained variance
[mean, v0, v1, exv0, exv1] = PCA(Points);
// modify the first vector in order to have first coordinate positive
if  $v1[0] < 0$  then
    |  $v1 = -1 * v1$ ;
end
 $\theta = \arccos(\text{exv1}/\text{exv0})$ ;
 $\phi = \pi - \arctan2(v1[0], v1[1])$ ;
return  $(\theta, \phi)$ ;

```

Algorithm 1: Estimating the angles (θ, ϕ) of the distribution (1).

In practice, however, we will not have such a dense measurement grid to be able to measure the distribution of particles at discrete points in contact. Suppose we have a square grid in which the distance between the particle detectors horizontally and vertically is constant at d (see Figure 1. Let us denote the sampled *Img* grid as Img_d .

Analysis of variance based on a set of relatively distant samples might be biased. To increase the spatial density of the samples, we can use a convolution with a Gaussian kernel with a size proportional to the sampling d [26]. In our case, we proposed a filter size equal to $4 \cdot d + 1$ with $\sigma = 0$, which has a large enough diameter to cancel out the sampling “holes” (2):

$$\text{Img}_{d,\text{Gauss}} = \text{Img} \otimes \text{GaussianKernel}_{(4 \cdot d + 1, 4 \cdot d + 1)}, \quad (2)$$

where \otimes denotes convolution. For the purpose of performing variance analysis with the Algorithm 1, it is not necessary to use a high-resolution grid. If the distribution is inside

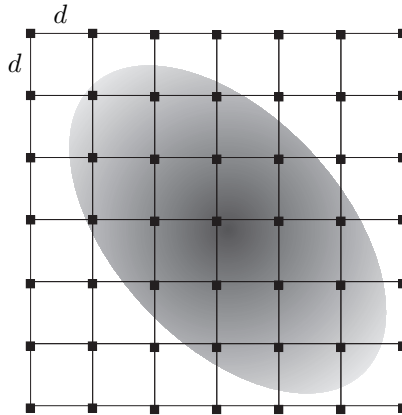


Fig. 1. An illustrative drawing showing a 7×7 grid of detectors, spaced by d in both directions. The detectors are black squares that are equally spaced along the x and y axes. A hypothetical histogram of the density of the particle distribution is shown in the background.

the sampled area, which preserves its spatial contour, it may be possible to resampling the original grid to a given lower resolution. This can be done using, for example, the following proposed Algorithm 2. It allows us to create fewer samples while preserving the sum of the detected particles.

2.3. Enhancing sampled distribution image by deep convolutional encoder-decoder

The method proposed in the previous section for determining the parameters of (θ, ϕ) consists of three steps: registration of radiation samples, Gaussian filtering (2), resampling with the Algorithm 2, and estimation of angles using the Algorithm 1. We can try to improve the reconstruction of the original particle distribution by using an encoder-decoder (E-D) neural network [14, 15, 23, 46]. The role of this network will be to reconstruct the original pre-sampled signal, but after applying Gaussian filtering (2) and rescaling with Algorithm 2. For this purpose, we used the following network:

- Encoder:
 - Convolution layer with 16 3×3 filters followed by ReLU activation and max pooling 2×2 ;
 - Convolution layer with 8 3×3 filters followed by ReLU activation and max pooling 2×2 ;
 - Convolution layer with 8 3×3 filters followed by ReLU activation and max pooling 2×2 ;
- Decoder:
 - Convolution layer with 8 3×3 filters followed by ReLU activation and up-sampling 2×2 ;
 - Convolution layer with 8 3×3 filters followed by ReLU activation and up-sampling 2×2 ;
 - Convolution layer with 16 3×3 filters followed by ReLU activation and up-sampling 2×2 ;
 - Convolution layer with 1 3×3 filter followed by sigmoid activation.

Data: Input: Img – input image with dimensions $n \times m$; scale – resampling factor ($\text{scale} > 1$)

Result: Img_{res} – resampled image.

```
// calculate the size of the resampled image
xSize = Integer(n / scale);
ySize = Integer(m / scale);
// initialize the resulting matrix with zeros
 $\text{Img}_{\text{res}}$  = zeros(xSize, ySize);
// iterate through all grid points
for  $x = 0; x < \text{xSize}; x ++$  do
    for  $y = 0; y < \text{ySize}; y ++$  do
        sum = 0;
        for  $a = 0; a < \text{scale}; a ++$  do
            for  $b = 0; b < \text{scale}; b ++$  do
                sum = sum +  $\text{Img}[(\text{scale} * x) + a, (\text{scale} * y) + b]$ ;
            end
        end
         $\text{Img}_{\text{res}}[x, y] = \text{sum}$ ;
    end
end
return  $\text{Img}_{\text{res}}$ ;
```

Algorithm 2: Resampling with preservation of the sum of detected particles.

In training, we will use the binary cross entropy loss function. The training data consisted of pairs:

- as an input Img filtered by (2), resampled by Algorithm 2 and scaled to discrete (integer) range $[0, 255]$
- as an output Img_d filtered by (2), resampled by Algorithm 2 and scaled to discrete (integer) range $[0, 255]$

The resulting image generated by the encoder-decoder described in this section is then subjected to the Algorithm 1 to estimate the (θ, ϕ) angle pair. In Figure 2 we present a diagram that explains the proposed method.

3. Results

In this section, we will describe the validation tests of our PCA-based Algorithm 1 and how the use of E-D network affects the resulting angle pair estimates (θ, ϕ) .

To test the performance of our method, we specified the following particle flux parameters (1): $N_\mu = 10^6$ (corresponds to a primary particle energy of more than 10^{16} eV [39]), $r_0 = 100$ (this value is compatible with the Molière radius in the Earth’s atmosphere at

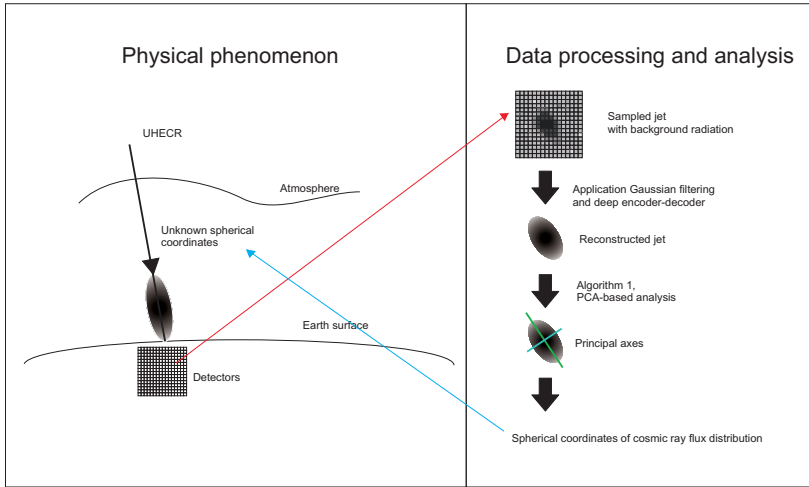


Fig. 2. Diagram that explains the proposed method. Ultra-high energy cosmic ray (UHECR) with unknown spherical coordinates generates a jet that is observed on Earth surface by the detectors. Data registered by detectors is sampled and mixed with background radiation (noise). After applying Gaussian filtering, deep encoder-decoder network reconstructs the original jet and Algorithm 1 is used to calculate spherical coordinates of the cosmic ray flux distribution.

the ground level [3, 4]), $s = 1.3$ [20]. We assumed that the particle flux is recorded over an area of 800×800 cm. The area is divided by a 1×1 cm grid. This means that the initial Img image has a resolution of 800×800 ($n = m = 800$). Let us assume that in the 800×800 cm area at intervals of $d = 25$ cm there are detectors equally spaced horizontally and vertically, each with an area of 1×1 cm. This means that we sample Img with $32 \times 32 = 1024$ samples (detectors) thus obtaining Img_d . We assume that each detector is capable of recording all the radiation particles that hit it during a single event. We have assumed the value of background radiation according to [35, 36], where the muon flux density at the earth's surface is $1 \frac{\mu\text{ons}}{\text{cm}^2 \cdot \text{min}}$. Thus, we can assume that over a period of 1 second, the background radiation density at 1 cm^2 of the earth's surface averages $\rho = \frac{1}{60} \frac{\mu\text{ons}}{\text{s} \cdot \text{cm}^2} = 0.01(6) \frac{\mu\text{ons}}{\text{s} \cdot \text{cm}^2}$.

We conducted a series of experiments testing the effectiveness of the Algorithm 1.

1. Angle estimation based on Img with resolution 800×800 ;
2. Angle estimation based on Img filtered by (2) and resampled with Algorithm 2 to a resolution of 80×80 pixels.
3. Angle estimation based on Img_d filtered by (2) and resampled with the Algorithm 2 to a resolution of 80×80 pixels.
4. Angle estimation based on Img_d filtered by (2) and resampled with Algorithm 2 to a resolution of 80×80 pixels and reconstructed by E-D network.

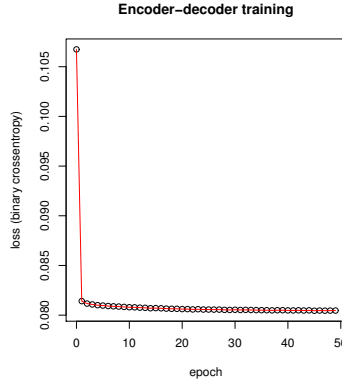


Fig. 3. A plot of loss function value during training.

The last two cases in the list above (3 and 4) are real-life scenarios. Cases 1 and 2 are used to check the validity of the Algorithm 1 assumptions, since in practice in these cases we have a very densely sampled distribution, which is not very realistic.

In order to train the E-D network, we generated distributions (1) in which the angle of $\theta \in \{0, 2, 4, \dots, 80\}$, $\phi \in \{0, 2, 4, \dots, 178\}$ (a total of 3690 distributions). We then discretized the distributions to an 800×800 grid by adding offsets along the x and y axes of $\{0, 3, 6, 9, 12\}$ cm. Thus, the final training set consisted of 92 250 discrete images. We filtered the 800×800 image (2) and resampled with the algorithm 2 to resolution 80×80 . These images were the input to the E-D network. In the output we used the same data Img_d where $d = 25$. We filtered Img_d by (2) and resampled with Algorithm 2 to resolution 80×80 . In this case, the validation set was not needed because the validation was done as part of the validation of the entire Algorithm 1 (see discussion below). We used the optimization algorithm Adam [30] with learning rate = 0.001. The training lasted 50 epochs. A plot of loss function is presented in Figure 3.

We implemented our approach in Python 3.8 using Keras/Tensorflow 2.8, scipy 1.8 and opencv-python 4.5 libraries. Significant speed-ups in generating distributions (1) were achieved using the numba 0.56 library. The entire experiment including data generation, network training and validation Algorithm 1 on a PC computer with Intel Core i7 3.00 Ghz; 64 GB RAM, Windows 10 OS took more than 3 days to execute. Some of the figures were made in R 3.6 language using dplyr 0.8 library and ggplot2 3.4.

In order to test the performance of our method, we generated a validation set of distributions (1) in which the angles $\theta \in \{1, 3, 5, \dots, 83\}$, $\phi \in \{1, 3, 5, \dots, 179\}$. In addition, we introduced a random offset of the distribution along the x and y axes in the range of values $[0, 12]$ cm which corresponds to half the distance between the positions of the simulated particle detectors. For each test configuration of the (θ, ϕ) pair, we performed 10 independent simulations (1) and background radiation with random offset values.

There were 37 800 images in total. We then applied the Algorithm 1 to each test case 1, 2, 3 and 4. The results for each pair (θ, ϕ) were averaged and the estimated angles (θ, ϕ) are shown in Figures 4 and 5. All the results are shown in degrees.

The source codes we have written can be downloaded from [10]. That online repository contains all source codes that are required to replicate the study including data generation script, network training, evaluation and plotting the results. All calculations included in this work were made using source codes from that repository.

The examples of results form the presented analysis made with the Algorithm 1 are shown in Figs. 6 and 7.

4. Discussion

The training of the E-D network, which changes in the loss function in successive iterations can be seen in Figure 3 was stable. After 50 epochs, loss had a binary cross entropy value of 0.080, which has remained virtually unchanged since epoch 40.

As can be seen in the Figures 6 and 7, the Algorithm 1 using PCA to detect the directions along which the largest variance is found works as expected. The axes are found with relatively small error, so that the estimation of the angle θ , which is calculated directly from the first axis of the PCA, is precise. In the case of the angle ϕ , for the calculation of which the variance ratio along the PCA axis is used, images that have more noise such as *Img+sampled+Gauss* and *Img+sampled+Gauss+resampled* have a less precise estimate of the ϕ angle than the *img+Gauss+resampled* and *Img+sampled+Gauss+resampled+E-D* example. Figures 6 and 7 also show that the proposed E-D is effective in cleaning *Img+sampled+Gauss+resampled* from measurements that cause disturbances in the variance estimation, which directly translates into improved estimation of θ .

These conclusions are supported by detailed statistic studies, the results of which we present in Figures 4 and 5. These figures show mean absolute error (MAE) estimates of (θ, ϕ) . In the case of the full *Img* image, the θ angle estimate is very accurate and decreases due to resampling. The largest error in determining the angle θ is in the area of small value of this angle (less than 10 to 20 degrees) because then the change in the statistical distribution of radiation particles is not large enough to be accurately estimated by PCA. A similar phenomenon also occurs in the case of the angle ϕ , when the largest estimation error is in the interval $[0, 10]$ and $[170, 180]$. This is due to trigonometric periodicity. Both of the phenomena discussed above are expected and easily explained, which proves the stability of our proposed method. As can be seen in Figure 5, the use of the E-D network significantly improves the accuracy of the angle θ estimate, which is calculated based on the variance along the PCA axis sometimes by as much as 20 degrees on average, compared to the estimate without the E-D architecture. The larger the actual θ angle is, the more the particle distribution is “stretched” in space

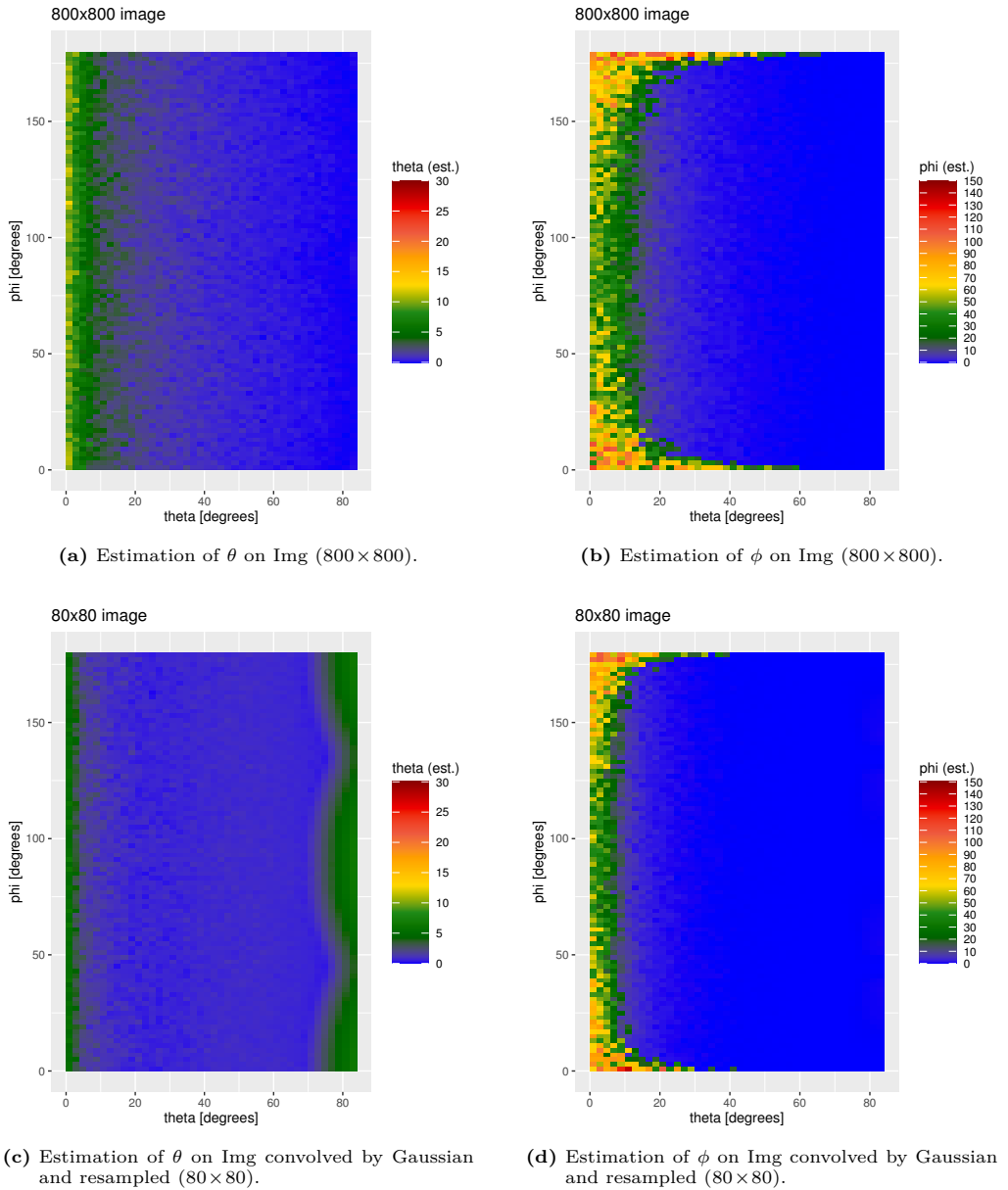


Fig. 4. MAE of estimations of (θ, ϕ) from Algorithm 1. Each measurement is averaged over 10 trials.

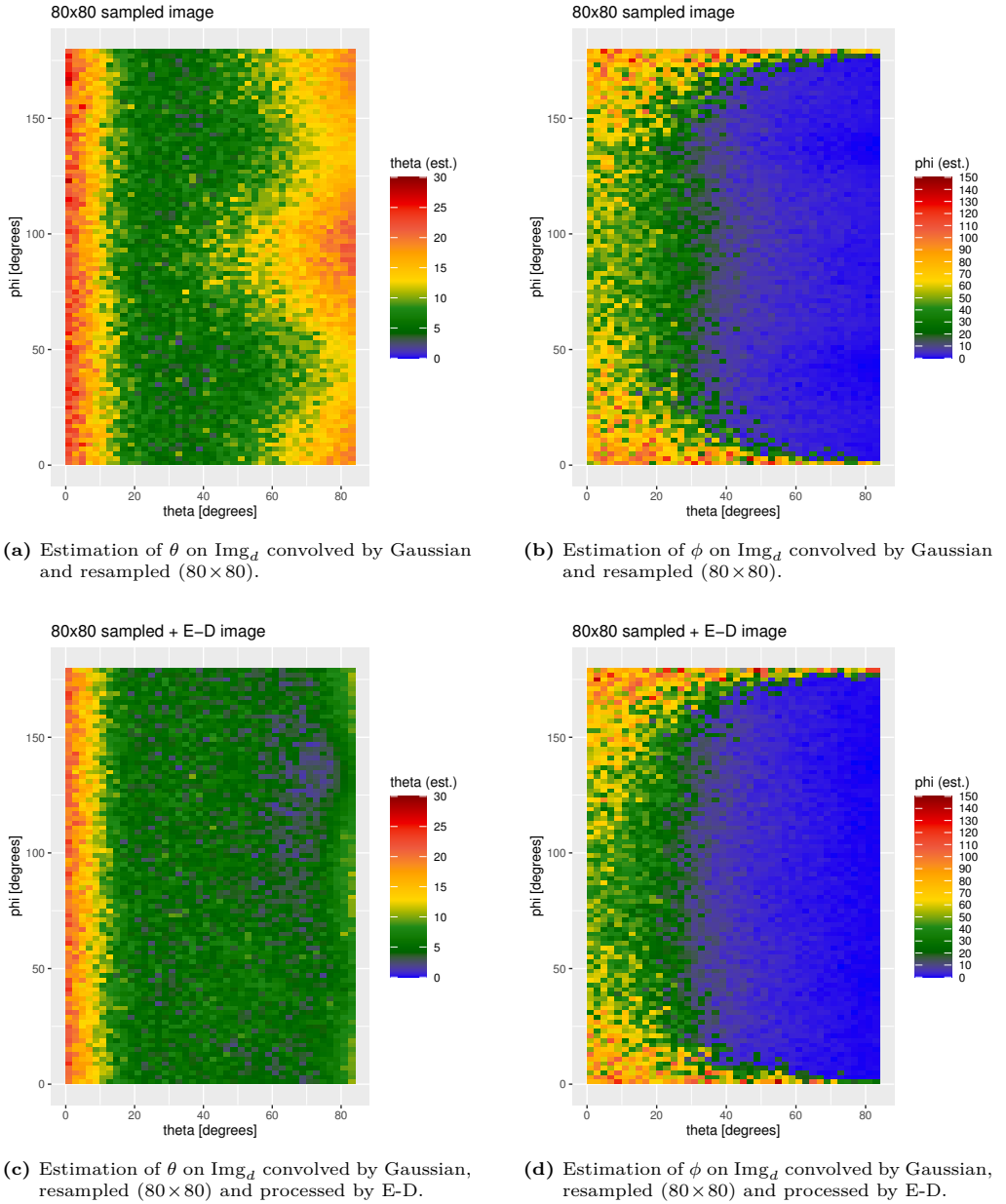


Fig. 5. MAE of estimations of (θ, ϕ) from Algorithm 1. Each measurement is averaged over 10 trials.

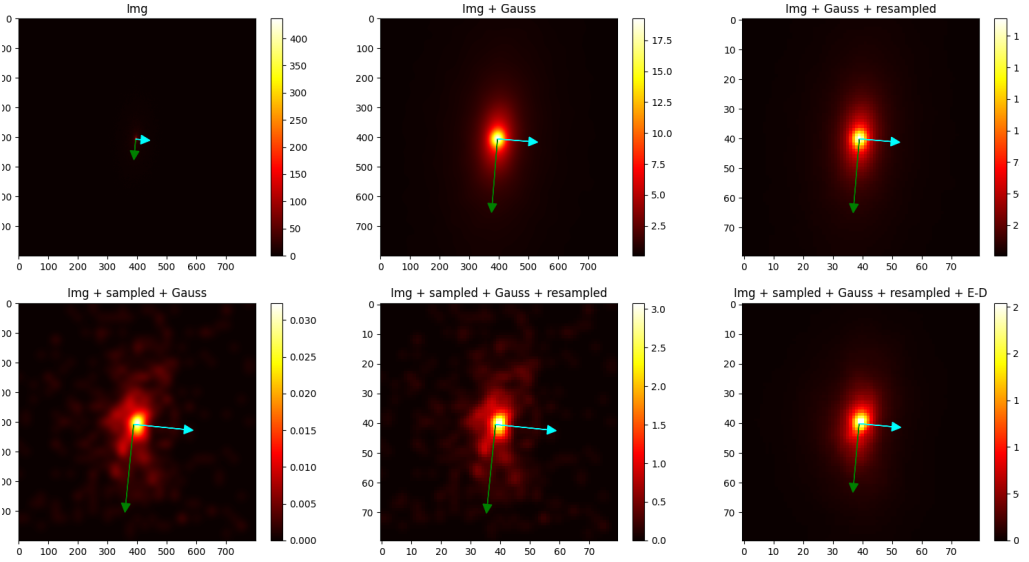


Fig. 6. Examples of results of Algorithm 1 for $(\theta = 62, \phi = 85)$. From left to right, from top to bottom (see also the titles of images): estimation of these angles for Img (800×800) equals $(\theta = 62.64, \phi = 85.15)$, for Img convolved by Gaussian and resampled to 80×80 it is $(\theta = 60.96, \phi = 85.38)$, for Img convolved by Gaussian (800×800) it is $(\theta = 50.17, \phi = 84.34)$, for sampled Img_d convolved by Gaussian and resampled to 80×80 it is $(\theta = 49.43, \phi = 84.19)$, for sampled Img_d convolved by Gaussian, resampled to 80×80 and processed by E-D it is $(\theta = 56.55, \phi = 84.84)$.

and more noise appears on Img_d through the sampling process. The E-D network does an excellent job of reducing this unfavorable phenomenon. As in the previously discussed cases, this phenomenon is expected and easily explained, which proves the stability of Algorithm 1. When E-D is applied for Img_d convolved and resampled, the number of cases (estimations) where $\text{MAE} < 10$ increases from 48% to 79% for θ and from 62% to 65% for ϕ , $\text{MAE} < 5$ increases from 24% to 45% for θ and from 47% to 52% for ϕ , $\text{MAE} < 1$ increases from 6% to 9% for θ and from 12% to 16% for ϕ . This is a significant change and demonstrates the high utility of the E-D network used.

5. Conclusion

The proposed algorithm based on the use of PCA for the determination of spherical coordinates of sampled cosmic ray flux distribution proved to be an effective and precise method in the experiment we conducted. The additional use of a deep neural network with an encoder-decoder architecture significantly increases its efficiency in the area of

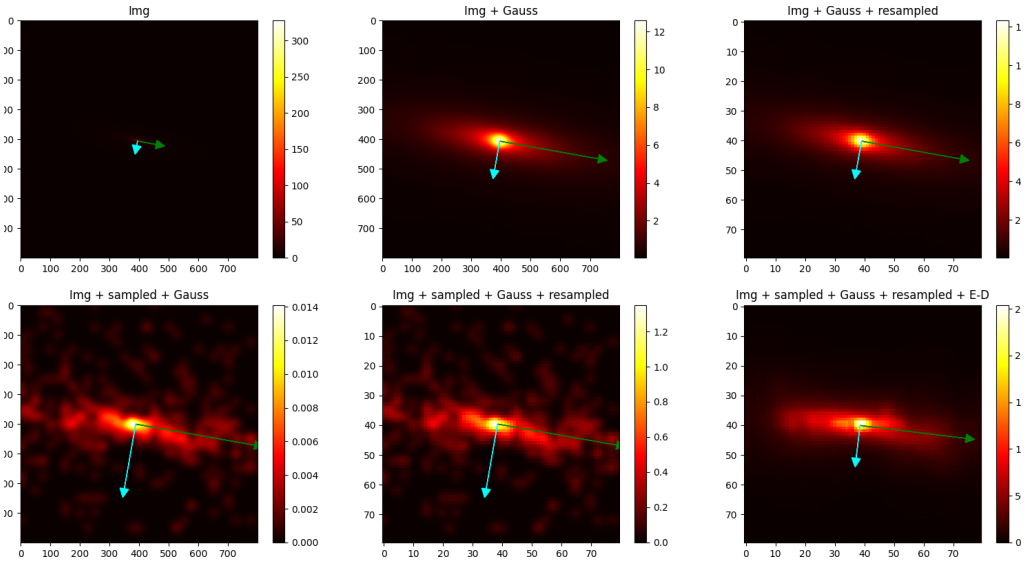


Fig. 7. Examples of results of Algorithm 1 for $(\theta = 76, \phi = 170)$. From left to right, from top to bottom (see also the titles of images): estimation of these angles for Img (800×800) equals $(\theta = 75.87, \phi = 169.54)$, for Img convolved by Gaussian and resampled to 80×80 it is $(\theta = 72.85, \phi = 169.81)$, for sampled Img_d convolved by Gaussian (800×800) it is $(\theta = 57.13, \phi = 169.97)$, for sampled Img_d convolved by Gaussian and resampled to 80×80 it is $(\theta = 57.04, \phi = 170.01)$, for sampled Img_d convolved by Gaussian, resampled to 80×80 and processed by E-D it is $(\theta = 72.00, \phi = 173.06)$.

high values of angles (θ, ϕ) making the proposed approach even more effective. Our Algorithm 1, together with the E-D network, is a very important method that will find its application in the research related to physical observations of fundamental astronomical processes. In particular, the introduced scheme can be directly useful in the design of small-scale complex CRE secondary flux detection systems. As we mentioned earlier, to the best of our knowledge, the results presented in this paper are pioneering in the field of small-scale complex CRE secondary flux detection systems, and it is difficult to point out research for the direct comparison. However, based on published research describing the use of a deep encoder-decoder for image denoising and original probabilistic distribution reconstruction [7, 11, 50, 54] we obtained the expected results, that is, the removal of unwanted noise at various frequencies while enhancing the signal with the desired statistical distribution. The effect of this denoising was an increased accuracy in estimating the rotation angles of the particle distribution described by the equation (1).

Several research problems have arisen in the preparation of this study that need to be addressed in future work. These include the effect of the topology of the detector grid

on the efficiency of estimating the angles (θ, ϕ) , the estimation of the appropriate grid density on the correctness of the estimate, and the dependence of the estimate on the energy of the particle flux and the offset of the flux center with respect to the detector grid center. These issues will be the subject of future research; nevertheless, it can already be summarized that our proposed method is a very effective approach.

References








- [1] M. G. Aartsen, M. Ackermann, J. Adams, J. Aguilar, M. Ahlers, et al. The IceCube Neutrino Observatory: instrumentation and online systems. *Journal of Instrumentation*, 12(03):P03012, 2017. doi:[10.1088/1748-0221/12/03/P03012](https://doi.org/10.1088/1748-0221/12/03/P03012).
- [2] M. G. Aartsen, M. Ackermann, J. Adams, J. Aguilar, M. Ahlers, et al. Erratum: The IceCube Neutrino Observatory: instrumentation and online systems. *Journal of Instrumentation*, 19(05):E05001, 2024. doi:[10.1088/1748-0221/19/05/E05001](https://doi.org/10.1088/1748-0221/19/05/E05001).
- [3] J. Abraham, P. Abreu, M. Aglietta, C. Aguirre, E. Ahn, et al. Atmospheric effects on extensive air showers observed with the surface detector of the Pierre Auger observatory. *Astroparticle Physics*, 32(2):89–99, 2009. doi:[10.1016/j.astropartphys.2009.06.004](https://doi.org/10.1016/j.astropartphys.2009.06.004).
- [4] J. Abraham, P. Abreu, M. Aglietta, C. Aguirre, E. Ahn, et al. Erratum to “Atmospheric effects on extensive air showers observed with the surface detector of the Pierre Auger observatory” [Astroparticle Physics 32(2) (2009), 89–99]. *Astroparticle Physics*, 33(1):65–67, 2010. doi:[10.1016/j.astropartphys.2009.10.005](https://doi.org/10.1016/j.astropartphys.2009.10.005).
- [5] R. Aloisio. Ultra High Energy Cosmic Rays an overview. *Journal of Physics: Conference Series*, 2429(1):012008, 2023. Proc. 12th Cosmic Ray International Seminar (CRIS 2022), 12-16 Sep 2022, Napoli, Italy. doi:[10.1088/1742-6596/2429/1/012008](https://doi.org/10.1088/1742-6596/2429/1/012008).
- [6] A. D. Avrorin, A. V. Avrorin, V. M. Aynutdinov, R. Bannash, I. A. Belolaptikov, et al. Baikal-GVD. *EPJ Web of Conferences*, 136:04007, 2017. Proc. 6th Roma International Conference on Astroparticle Physics (RICAP16), 21-24 Jun, Roma, Italy. doi:[10.1051/epjconf/201713604007](https://doi.org/10.1051/epjconf/201713604007).
- [7] K. Bajaj, D. K. Singh, and M. A. Ansari. Autoencoders based deep learner for image denoising. *Procedia Computer Science*, 171:1535–1541, 2020. Proc. 3rd International Conference on Computing and Network Communications (CoCoNet’19), 18-21 Dec, Trivandrum, Kerala, India. doi:[10.1016/j.procs.2020.04.164](https://doi.org/10.1016/j.procs.2020.04.164).
- [8] O. Bar, Ł. Bibrzycki, M. Niedźwiecki, M. Piekarczyk, K. Rzecki, et al. Zernike moment based classification of cosmic ray candidate hits from CMOS sensors. *Sensors*, 21(22):7718, 2021. doi:[10.3390/s21227718](https://doi.org/10.3390/s21227718).
- [9] Ł. Bibrzycki, D. Burakowski, P. Homola, M. Piekarczyk, M. Niedźwiecki, et al. Towards a global cosmic ray sensor network: CREDO detector as the first open-source mobile application enabling detection of penetrating radiation. *Symmetry*, 12(11):1802, 2020. doi:[10.3390/sym12111802](https://doi.org/10.3390/sym12111802).
- [10] browarsoftware. `cosmic_ray_spherical`. GitHub. https://github.com/browarsoftware/cosmic_ray_spherical.
- [11] Y. Choi, S. Park, and S. Kim. Development of point cloud data-denoising technology for earthwork sites using encoder-decoder network. *KSCE Journal of Civil Engineering*, 26(11):4380–4389, 2022. doi:[10.1007/s12205-022-0407-8](https://doi.org/10.1007/s12205-022-0407-8).
- [12] M. T. Dova, L. N. Epele, and A. G. Mariazzi. Particle density distributions of inclined air showers. *Nuovo Cimento C*, 24(4–5):745–750, 2001. <https://www.sif.it/riviste/sif/ncc/econtents/2001/024/04-05/article/5>.

- [13] D. Droz, A. Tykhonov, X. Wu, and M. Deliyergiyev. Neural networks for TeV cosmic electrons identification on the DAMPE experiment. In: *Proc. The European Physical Society Conference on High Energy Physics PoS(EPS-HEP2021)*, vol. 398 of *Proceedings of Science*, p. 045. Online – Hamburg, Germany, 26-30 Jul 2021, published in 2022. doi:10.22323/1.398.0045.
- [14] V. Dutta, M. Choraś, M. Pawlicki, and R. Kozik. A deep learning ensemble for network anomaly and cyber-attack detection. *Sensors*, 20(16):4583, 2020. doi:10.3390/s20164583.
- [15] V. Dutta, M. Pawlicki, R. Kozik, and M. Choraś. Unsupervised network traffic anomaly detection with deep autoencoders. *Logic Journal of the IGPL*, 30(6):912–925, 2022. doi:10.1093/jigpal/jzac002.
- [16] J. Glombitza, M. Erdmann, M. Vieweg, and M. Dohmen. Deep learning based air shower reconstruction at the Pierre Auger Observatory. In: *Proc. DPG Spring meeting 2019*, vol. Aachen 2019 issue of *Verhandlungen der Deutschen Physikalischen Gesellschaft*. Aachen, Germany, 25-29 Mar 2019. <https://inis.iaea.org/records/dwnb-hhf07>.
- [17] K. Greisen. Cosmic ray showers. *Annual Review of Nuclear Science*, 10(1):63–108, 1960. doi:10.1146/annurev.ns.10.120160.000431.
- [18] Y. Guo and B. W.-K. Ling. Spherical coordinate-based kernel principal component analysis. *Signal, Image and Video Processing*, 15(3):511–518, 2021. doi:10.1007/s11760-020-01771-8.
- [19] T. Hachaj, Ł. Bibrzycki, and M. Piekarczyk. Recognition of cosmic ray images obtained from CMOS sensors used in mobile phones by approximation of uncertain class assignment with deep convolutional neural network. *Sensors*, 21(6):1963, 2021. doi:10.3390/s21061963.
- [20] T. Hachaj, Ł. Bibrzycki, and M. Piekarczyk. Fast training data generation for machine learning analysis of cosmic ray showers. *IEEE Access*, 11:7410–7419, 2023. doi:10.1109/ACCESS.2023.3237800.
- [21] T. Hachaj and M. Piekarczyk. The practice of detecting potential cosmic rays using CMOS cameras: Hardware and algorithms. *Sensors*, 23(10):4858, 2023. doi:10.3390/s23104858.
- [22] T. Hachaj, M. Piekarczyk, and J. Wąs. Searching of potentially anomalous signals in cosmic-ray particle tracks images using rough k-means clustering combined with eigendecomposition-derived embedding. In: *Proc. International Joint Conference on Rough Sets (IJCRS 2023)*, vol. 14481 of *Lecture Notes in Computer Science*, pp. 431–445. Springer, Kraków, Poland, 5-8 Oct 2023. doi:10.1007/978-3-031-50959-9_30.
- [23] P. Hasić, A. Świtoński, H. Josiński, and K. Wojciechowski. Anomaly detection of motion capture data based on the autoencoder approach. In: *Proc. International Conference on Computational Science (ICCS 2023 2023)*, vol. 14074 of *Lecture Notes in Computer Science*, pp. 611–622. Springer, 3-5 Jul 2023. doi:10.1007/978-3-031-36021-3_59.
- [24] P. Homola, D. Beznosko, G. Bhatta, Ł. Bibrzycki, M. Borczyńska, et al. Cosmic-ray extremely distributed observatory. *Symmetry*, 12(11):1835, 2020. doi:10.3390/sym12111835.
- [25] P. Homola, V. Marchenko, A. Napolitano, R. Damian, R. Guzik, et al. Observation of large scale precursor correlations between cosmic rays and earthquakes with a periodicity similar to the solar cycle. *Journal of Atmospheric and Solar-Terrestrial Physics*, 247:106068, 2023. doi:10.1016/j.jastp.2023.106068.
- [26] J. Jaworek-Korjakowska and R. Tadeusiewicz. Determination of border irregularity in dermoscopic color images of pigmented skin lesions. In: *Proc. 2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 6459–6462. IEEE, Chicago, IL, USA, 26-30 Aug 2014. doi:10.1109/EMBC.2014.6945107.
- [27] H. Josiński, D. Kostrzewa, A. Michalczuk, A. Świtoński, and K. Wojciechowski. Feature extraction and HMM-based classification of gait video sequences for the purpose of human identification. In:

- A. Nawrat and Z. Kuś, eds., *Vision Based Systems for UAV Applications*, vol. 481 of *Studies in Computational Intelligence*, pp. 233–245. Springer, 2013. doi:10.1007/978-3-319-00369-6_15.
- [28] O. Kalashev, I. Kharuk, G. Rubtsov, on behalf of the Baikal-GVD Collaboration, et al. Machine learning based background rejection for Baikal-GVD neutrino telescope. *Journal of Physics: Conference Series*, 2438:012099, 2023. Proc. 20th International Workshop on Advanced Computing and Analysis Techniques in Physics Research, 29 Oct – 3 Dec 2021, Virtual and Daejeon, South Korea. doi:10.1088/1742-6596/2438/1/012099.
- [29] M. Karbowski, M. Orzechowski, T. Wibig, Ł. Bibrzycki, P. Kovacs, et al. Small shower array for education purposes—the CREDO-Maze Project. In: *Proc. 37th International Cosmic Ray Conference PoS(ICRC2021)*, vol. 395 of *Proceedings of Science*, p. 219. Online – Berlin, Germany, 21–23 Jul 2021, published in 2022. doi:10.22323/1.395.0219.
- [30] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In: *Proc. 3rd Int. Conf. Learning Representations, ICLR 2015*. San Diego, CA, 7–9 May 2015. Accessible in arXiv. doi:10.48550/arXiv.1412.6980.
- [31] P. Koundal, R. Abbasi, M. Ackermann, J. Adams, IceCube Collaboration, et al. Composition analysis of cosmic-rays at IceCube Observatory using graph neural networks. In: *Proc. 27th European Cosmic Ray Symposium – PoS(ECRS)*, vol. 423 of *Proceedings of Science*, p. 085. Nijmegen, The Netherlands, 25–29 Jul, 2022, published in 2023. doi:10.22323/1.423.0085.
- [32] R. Kumar. Tracking cosmic rays by crayfis (cosmic rays found in smartphones) global detector. In: *Proc. The 34th International Cosmic Ray Conference PoS(ICRC2015)*, vol. 236 of *Proceedings of Science*, p. 1234. The Hague, The Netherlands, 30 Jul – 6 Aug 2015, published in 2016. doi:10.22323/1.236.1234.
- [33] R. Nirwan and N. Bertschinger. Rotation invariant householder parameterization for Bayesian PCA. In: K. Chaudhuri and R. Salakhutdinov, eds., *Proc. 36th International Conference on Machine Learning*, vol. 97 of *Proceedings of Machine Learning Research*, pp. 4820–4828. PMLR, Long Beach, CA, USA, 9–15 Jun 2019. <https://proceedings.mlr.press/v97/nirwan19a.html>.
- [34] R. D. Parsons and S. Ohm. Background rejection in atmospheric Cherenkov telescopes using recurrent convolutional neural networks. *The European Physical Journal C*, 80(5):363, 2020. doi:10.1140/epjc/s10052-020-7953-3.
- [35] Particle Data Group, R. L. Workman, V. D. Burkert, V. Crede, E. Klempt, et al. Review of particle physics. *Progress of Theoretical and Experimental Physics*, 2022(8):083C01, 2022. doi:10.1093/ptep/ptac097.
- [36] Particle Data Group, P. A. Zyla, R. M. Barnett, J. Beringer, O. Dahl, et al. Review of Particle Physics. *Progress of Theoretical and Experimental Physics*, 2020(8):083C01, 2020. doi:10.1093/ptep/ptaa104.
- [37] M. Piekarczyk, O. Bar, Ł. Bibrzycki, M. Niedźwiecki, K. Rzecki, et al. CNN-based classifier as an offline trigger for the CREDO experiment. *Sensors*, 21(14):4804, 2021. doi:10.3390/s21144804.
- [38] M. Piekarczyk and T. Hachaj. On the search for potentially anomalous traces of cosmic ray particles in images acquired by CMOS detectors for a continuous stream of emerging observational data. *Sensors*, 24(6):1835, 2024. doi:10.3390/s24061835.
- [39] J. S. Pryga, W. Stanek, K. W. Woźniak, P. Homola, K. Almeida Cheminant, et al. Analysis of the capability of detection of extensive air showers by simple scintillator detectors. *Universe*, 8(8):425, 2022. doi:10.3390/universe8080425.
- [40] D. B. Reusch, R. B. Alley, and B. C. Hewitson. Relative performance of self-organizing maps and principal component analysis in pattern extraction from synthetic climatological data. *Polar Geography*, 29(3):188–212, 2005. doi:10.1080/789610199.

- [41] K. Sargsyan, J. Wright, and C. Lim. GeoPCA: a new tool for multivariate analysis of dihedral angles based on principal component geodesics. *Nucleic Acids Research*, 40(3):e25–e25, 2012. doi:[10.1093/nar/gkr1069](https://doi.org/10.1093/nar/gkr1069).
- [42] K. Sargsyan, J. Wright, and C. Lim. GeoPCA: a new tool for multivariate analysis of dihedral angles based on principal component geodesics. *Nucleic Acids Research*, 43(21):10571–10572, 2015. (This is a correction to [41]). doi:[10.1093/nar/gkv1000](https://doi.org/10.1093/nar/gkv1000).
- [43] M. Savić, A. Dragić, D. Maletić, N. Veselinović, R. Banjanac, et al. A novel method for atmospheric correction of cosmic-ray data based on principal component analysis. *Astroparticle Physics*, 109:1–11, 2019. doi:[10.1016/j.astropartphys.2019.01.006](https://doi.org/10.1016/j.astropartphys.2019.01.006).
- [44] J. Stasielak, P. Malecki, D. Naumov, V. Allakhverdian, on behalf of the Baikal-GVD Collaboration, et al. High-energy neutrino astronomy—Baikal-GVD neutrino telescope in Lake Baikal. *Symmetry*, 13(3):377, 2021. doi:[10.3390/sym13030377](https://doi.org/10.3390/sym13030377).
- [45] Z. Szadkowski and K. Pytel. Trigger based on a fuzzy logic for a detection of very inclined cosmic rays in the surface detector of the Pierre Auger Observatory. In: *Proc. 2019 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, pp. 1–5. IEEE, Auckland, New Zealand, 20-23 May 2019. doi:[10.1109/I2MTC.2019.8827075](https://doi.org/10.1109/I2MTC.2019.8827075).
- [46] R. Tadeusiewicz, R. Chaki, and N. Chaki. *Exploring neural networks with C#*. CRC Press, Boca Raton, 2015. doi:[10.1201/b17332](https://doi.org/10.1201/b17332).
- [47] J. Takalo. Extracting hale cycle related components from cosmic-ray data using principal component analysis. *Solar Physics*, 297(9):113, 2022. doi:[10.1007/s11207-022-02048-8](https://doi.org/10.1007/s11207-022-02048-8).
- [48] The Pierre Auger Collaboration. The Pierre Auger Cosmic Ray Observatory. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 798:172–213, 2015. doi:[10.1016/j.nima.2015.06.058](https://doi.org/10.1016/j.nima.2015.06.058).
- [49] The Pierre Auger Collaboration, A. Aab, P. Abreu, M. Aglietta, J. Albury, et al. Extraction of the muon signals recorded with the surface detector of the Pierre Auger Observatory using recurrent neural networks. *Journal of Instrumentation*, 16(07):P07016, 2021. doi:[10.1088/1748-0221/16/07/P07016](https://doi.org/10.1088/1748-0221/16/07/P07016).
- [50] C. Tian, L. Fei, W. Zheng, Y. Xu, W. Zuo, et al. Deep learning on image denoising: An overview. *Neural Networks*, 131:251–275, 2020. doi:[10.1016/j.neunet.2020.07.025](https://doi.org/10.1016/j.neunet.2020.07.025).
- [51] J. Vandenbroucke, S. BenZvi, S. Bravo, K. Jensen, P. Karn, et al. Measurement of cosmic-ray muons with the distributed electronic cosmic-ray observatory, a network of smartphones. *Journal of Instrumentation*, 11(04):P04019, 2016. doi:[10.1088/1748-0221/11/04/p04019](https://doi.org/10.1088/1748-0221/11/04/p04019).
- [52] T. Wibig. Small shower CORSIKA simulations. *Chinese Physics C*, 45(8):085001, 2021. doi:[10.1088/1674-1137/ac0099](https://doi.org/10.1088/1674-1137/ac0099).
- [53] M. Yu, T. B. Anderson, Y. Chen, S. Coutu, T. LaBree, et al. Machine learning applications on event reconstruction and identification for ISS-CREAM. In: *Proc. 37th International Cosmic Ray Conference PoS(ICRC2021)*, vol. 395, p. 061. Proceedings of Science, Online – Berlin, Germany, 21-23 Jul 2021, published in 2022. doi:[10.22323/1.395.0061](https://doi.org/10.22323/1.395.0061).
- [54] D. Zheng, S. H. Tan, X. Zhang, Z. Shi, K. Ma, et al. An unsupervised deep learning approach for real-world image denoising. In: *Proc. 8th International Conference on Learning Representations (ICLR)*. Virtual, 26 Apr – 1 May 2020. Published in OpenReview. <https://openreview.net/forum?id=tIjRAiFmU3y>.

MACHINE VISION FOR AUTOMATED MATURITY GRADING OF OIL PALM FRUITS: A SYSTEMATIC REVIEW

Afsar Kamal^{1,*} , Nur Diyana Kamarudin^{1,2,*} , Khairol Amali Bin Ahmad^{1,2} ,
Syarifah Bahiyah Rahayu^{1,2} , Mohd Rizal Mohd Isa¹ ,
Siti Noormiza Makhtar² , Zulkifli Yaakub³ 

¹*Faculty of Defence Science and Technology, National Defence University of Malaysia, Kuala Lumpur, Malaysia*

²*Centre for Cyber Security and Digital Industrial Revolution, National Defence University of Malaysia, Kuala Lumpur, Malaysia*

³*Malaysia Palm Oil Board, Kajang, Selangor, Malaysia*

*Corresponding authors: Nur Diyana Kamarudin (nurdiyana@upnm.edu.my),
Afsar Kamal (kamal786lari@gmail.com)

Abstract The maturity of oil palm fruits is a very crucial factor for oil extraction industry in Indonesia, Malaysia, Thailand, and other countries to ensure the oil quality and increase productivity. This literature review examines the various machine learning techniques, especially the deep learning techniques used to automate the maturity grading process of oil palm fresh fruit bunches. The crucial advantages of using machine learning approaches were highlighted, and the limitations and prospects of each research article were discussed. This review describes the various image pre-processing techniques utilized to prepare images for model training. CNN is identified as the dominant over all classification techniques of machine learning to classify the oil palm fruits images based on maturity level, due to its ability of learning complex features.

Keywords: machine Learning, deep Learning, CNN, feature extraction, Computer Vision, maturity grading.

1. Introduction

The maturity grading of oil palm fruits plays a pivotal role in the overall success and sustainability of the palm industry. To find out the accurate estimation of fruit maturity is a central key point for improving agricultural practices, ensuring oil quality, and maximizing the palm fruit production. In the recent era, the implementation of machine learning algorithm has become a transformative way to increase the efficiency and precision of maturity grading process. This literature review is based on to explore the existing body of knowledge on the uses of machine learning techniques in the field of maturity grading of palm fruit, focussing on to provide the comprehensive overview of the current research, methodologies used and the key findings.

Computer Vision and Image Processing techniques are the two correlated and interconnected fields of machine learning in Computer Science. The potentiality of these techniques is to derive and conclude the information from the visual data and train the machines with the ability to interpret like human visual perception. There are

many applications based on Computer Vision and Image Processing across various domains, including fruit categorization, surveillance system, healthcare, object detection, and more [113].

Computer Vision techniques are utilized to develop the algorithms and models to impart the systems with the ability to make decisions from the visual data and interpret into the human demand. Many tasks, such as recognition, detection, segmentation, and visual understanding are being performed within the range of Computer Vision Technique. The decisive goal is to enhance machine capability to be able to comprehend and make interaction with the visual world [37]. Image Processing as the name suggests, it has the role of manipulate and analyse the images to boost their qualities, extract information and perform a subsequent analysis. It has basic operations such as compression, enhancement, filtering, and restoration among the others. It plays a vital and potential role in pre-process data before being applied to higher tasks in Computer Vision [41].

The grading system of oil palm fruits has its own potential role by categorising them based on the parameters such as color, bunches, and shape to enhance the efficiency and improve the productivity. The Machine Learning approach has the great impact of automating the maturity grading process. The Machine Learning approach can analyse the images captured in RGB format, and it can extract the morphological and color features. There are many Machine Learning techniques, such as CNN, KNN, fuzzy logic, and other that are employed to detect the fruits accurately and classify them accordingly [52].

2. Literature survey

Palm oil is an important commodity that is widely used in the food industry, biofuels, and various other industries [96]. The maturity of palm fruits influences the quality of palm oil, with different grades of oil produced from fruits at different stages of maturity [133]. Therefore, accurate detection and grading of palm fruits is crucial to ensure the quality of the oil produced.

Traditional methods of palm fruit detection and grading are time-consuming, labor-intensive, and prone to errors. In recent years, machine learning techniques, particularly deep learning, have presented hopeful outcomes in numerous computer vision tasks, containing object detection and classification [70]. However, by incorporating quantum computing techniques, researchers can further improve the performance of these methods.

Quantum deep learning (QDL) is an emerging field that combines quantum computing and deep learning to achieve better performance in machine learning tasks [39]. QDL uses quantum circuits to enhance the performance of classical deep learning models. The potential of QDL in enhancing machine learning tasks has been demonstrated in various applications, such as image classification and natural language processing.

Zheng et al. (2020) proposed a QDL-based approach, Quantum PalmNet, for palm tree detection, which outperformed classical deep learning models in terms of accuracy and speed [5]. This approach can be extended to palm fruit detection and maturity grading.

In the agriculture sector, computer vision techniques have been used to analyze various crops, including fruits and vegetables. Machine learning algorithms have been used to detect and classify different types of fruits and vegetables based on their color and shape features [103]. QDL has the potential to improve the performance of these algorithms by leveraging the power of quantum computing. Sabri et al. (2017) used deep learning techniques to classify palm fruits based on color features. The study showed promising results in identifying fruits at different maturity stages [98]. However, incorporating QDL techniques can improve the performance of the model.

2.1. Pre-processing

As the name suggests, the pre-processing refers to the stage of images before going to be processed. It plays a crucial role to prepare and transform the image data for machine learning tasks such as recognition, classification, prediction, analysis, detection, and segmentation by using the image acquisition, enhancement, resizing, normalization, extracting relevant features and others [117].

2.1.1. Image acquisition

The first step in pre-processing is to collect the image data using the high quality of input devices. To ensure the extensive acquisition of image characteristics, you must capture the images on different angles under the predefined lighting conditions. The classical spectrum imaging technique captures images using cameras based on the RGB wavelength. These images can be utilized to provide color information for assistance in maturity grading [31]. Unlike the visible spectrum imaging technique where only three or four bands (e.g., RGB and sometimes nearer to Infrared) are used to capture the image data, the HSI (Hyperspectral Imaging) technique works on minor continuous spectral bands to capture images to improve the image feasibility. The HSI has the great impact on maturity grading of palm fruits based on the spectral characteristics [26]. While the NIR (Near-Infrared) Imaging technique captures electromagnetic radiation within the range of wavelength 700-2500 nanometers. This feature offers the insights to ponder into the pulp of fruit and produces feature extraction such as biochemical composition and moisture amount. This technique is useful to assess the fruit quality [114].

2.1.2. Image enhancement

Image enhancement is an important pre-processing step for the computer vision task, employed to enhance the image quality and extract the relevance information from the acquired images. This step may consist of adjusting sharpness, maintaining brightness, and increasing the contrast to reduce the noise and enhance the feature selection of fruit

images. In the current era, machine learning techniques significantly have advancement and improvement to the image enhancement methods [12].

The classical image enhancement approaches consist of the techniques like contrast stretching [125], histogram equalization [129], and spatial filtering [1]. These techniques are simple, easy to use and computationally efficient, but commonly they don't produce satisfactory outputs, especially in the noise and lightening conditions. To deal with such type of conditions, deep learning techniques are utilized, especially CNNs (Convolutional Neural Network) have gained the popularity for image enhancement due to their capability of learning from complex features directly or indirectly from image data. There are many methodologies proposed in this field, such as SISR [126] (Single Image Super Resolution) has the potential goal to enhance the quality of low-resolution images. The other approaches such as SRGAN [72] (Super-Resolution Generative Adversarial Network), and ESRGAN [121] (Enhanced Super-Resolution Generative Adversarial Networks) and SRCNN [29] (Super-Resolution Convolutional Neural Network) have produced the satisfactory results in preparing the high-resolution images from the lower resolution input images. Enhancing and converting the images captured in the low-light situation is a tedious job due to the poor visibility and external noises. The Retinex-based model [74], Fusion-based approach, and CNN-based technique [23] offer solutions to handle this type of situations.

To fulfil the criteria of image enhancement is impossible without dealing with the unwanted noise. The deep learning-based Noise reduction methods, such as RIDNet [8] (Real Image Denoising Network), CBDNet [42] (Convolutional Blind Denoising Network), and DnCNN [130] (Denoising Convolutional Neural Network), have the superior priority over classical denoising techniques. To deal with the blurred images is also a critical task in accomplishing the image enhancement process. Removing the blurriness of the images becomes essential while working on medical imaging or surveillance applications. The most used CNN-based deblurring models like DeblurGAN [66] (Deblurring Generative Adversarial Network) and SRN-DeblurNet [81] (Spatial Recurrent Network for Image Deblurring) have presented the satisfactory result to restore the sharpness to the blurred images.

GANs (Generative Adversarial Networks) have emerged as the effective frameworks to satisfy the image enhancement tasks. They follow the process of adversarial training to generate the visual and realistic required images. The task of image enhancement is accomplished by several models, including Pix2Pix [53], SRGAN [72], and CycleGAN [132], which utilize Generative Adversarial Networks. These models enable image-to-image translation, higher resolution generation, and enhancement of unpaired images, respectively.

2.1.3. Image resizing

Image resizing is a crucial task in image pre-processing and computer vision. Image resampling or image scaling is another name for it. In the machine learning context, researchers have implemented many techniques to resize the images while maintaining and preserving the actual visual effects of images. Bicubic interpolation [94] is a commonly used methodology to resize the images. It produces the computationally efficient and smooth results. The recent advancement in deep learning technology has led the CNN model to do the image resizing tasks as well. Kim et al. (2016) proposed a model called ESPCN (Efficient Sub-Pixel Convolutional Neural Network) which modifies and scales the images from lower resolution to higher resolution. It effectively has the job for single image super resolution [62]. Content-Aware image resizing technique is also a powerful approach that preserves the visual content and minimizes the distortion in the region of interest.

Avidan and Shamir (2007) introduced Seam carving algorithm, a standard content-aware image resizing technique that inserts or eliminates the seams of the lower resolution pixel to resize the images [10]. Although it plays an important role in the image resizing while preserving the actual visual content, it may produce unwanted results when dealing images of high complexity. The other technique to resize the images is Learned Image Resizing, that adopts the machine learning algorithms to resize the images and produce the required result based on their content. Danon et al. (2021) proposed a learned image resizing approach to resize the images using a deep neural network being trained on the large dataset of paired images [27]. This method achieves the better response compared to a classical image resizing technique, especially in complex data with challenging images. The implementation of the multi-scale approaches to resize the images and segmented them based on the resolutions, then finally merged them to generate the required resized output image. He et al. (2016) proposed deep network architecture for the multi-scale high-resolution images. This approach progressively scales the images on various measures to get the best result [46].

Table 1 presents a comprehensive overview of color parameters and features used in various studies, including color spaces, color values, statistical color features, color histograms, and texture analysis, with references to relevant literature.

2.1.4. Color normalization

Color Normalization is a critical image pre-processing step to ensure the color persistent across various imaging stages to perform accurate analysis and interpretation of visual image data in the computer vision applications. It ensures the reliability and robustness in numerous machine learning applications. Numerous techniques have been proposed to normalize the color distribution and minimize the changing effects caused by lightening, sensors, or other input devices. The Table 2 lists various techniques for image processing and color normalization, including Histogram Equalization for enhancing contrast, Deep

Tab. 1. Comparison of color features for oil palm fruits analysis.

Ref.	Color parameters & features	Color Space
[54]	Photogrammetric, L*a*b color space	RGB
[5]	Color Values, Statistical Color Features, Color Histogram	
[112]	Uniformity, inverse difference, homogeneity, & Outer Color	
[78]	Visual Inspection, black/dark purple color,	
[45]	Surface Color, Dark purple, orange red	
[2]	Color histogram & Statistical color, UV, RGB, & NIR	
[9]	Color composition, Red & orange skin color, blackish brown, black skin	RGB, HSV
[80]	TCS3200, RGB	
[4]	Statistical color features & color histograms	RGB, HSV, L*a*b
[3]	Statistical color features & color histograms	
[58]	Random brightness ranges from -40% to +60%, RGB	RGB, L*a*b
[59]	Gray-scale threshold, RGB	HSI, L*a*b
[110]	Histogram Analysis, RGB & L*a*b	CIE L*a*b
[40]	Hue value, Hue measurement	RGB, HSI, HSV
[49]	HOG & FREAK, RGB color channels	RGB, HSI
[124]	Hue, Saturation & Value conversion from RGB to HSV	HSV
[92]	FRedS4, IRedS4, etc., Blue, Green, Amber, Red, Deep Red	Far-Red band
[116]	Mesocarp color, Orange for ripe, yellowish/yellow for unripe	Wavelength
[13]	Spectral bands & reflectance values, Carotenoids and Chlorophylls	Spectral Bands
[77]	Surface Color, Dark Purple, Red Orange	Yellowish Red

Learning-based Methods like CNN for color normalization, Batch Renormalization to standardize image activations, Gray-Level Co-Occurrence Matrix (GLCM) for texture analysis, Retinex Algorithm for improving color consistency, Instance Normalization for per-pixel color adjustments, Color Transfer for matching color distributions between images, and Color Constancy Algorithms for maintaining consistent colors despite lighting variations.

2.1.5. Image segmentation

The primary goal of this fundamental task is to partition an image into meaningful pixel or segments. These segments correspond to the region of interests in an image. Image Segmentation approach is used in various machine learning tasks such as image classification, object detection and semantic segmentation. The Tab 3 outlines various segmentation techniques, including Thresholding-Based Methods, like Otsu's method for optimal threshold selection, Edge-Based Segmentation techniques, such as Canny Edge

Tab. 2. Various techniques used in color normalization for image processing.

Techniques	Description
Histogram Equalization	This technique is used to enhance the contrast of an image by adjusting the intensity distribution of pixels [89].
Deep Learning based Methods	In the deep learning-based approach, CNN methodology is used to normalize the color directly from the image data [131].
Batch Renormalization	Batch Renormalization is an extended version of Batch Normalization technique, fundamentally used in deep learning to normalize the activations of mini batches in the neural network. It transforms the color distribution of images into a standard form, also reduces the generated color variations because of lightening situations, cameras, sensors, and other devices that affect the image appearances [50].
Gray Level Co-Occurrence Matrix (GLCM)	GLCM is a technique, particularly used in texture analysis. It also employed as the feature selection method to quantify the texture of an image, which enhances the robustness and performance of a machine learning model on trained images [36].
Retinex Algorithm	It is a very powerful tool used in the process of image pre-processing to accomplish the task of color normalization. It manages to reestablish the gaps due to the lightening variations, sensors, or other imaging input devices, eventually the image color looks more realistic and natural. Basically, it improves the consistency of color, enables robust feature selection, and enhances the interoperability and visual quality of the images [68].
Instance Normalization	Instance Normalization is a technique, specially used to normalize the colors and features across the various instances. It performs the normalization task separately on each pixel values of each individual image. It also computes the mean value and standard deviation of each color (Red, Green, and Blue) of an image [119].
Color Transfer	Color transfer is a machine learning approach used in color normalization as it provides the consistency and accuracy to image analysis and the computer vision tasks. This approach tries to match the color distribution from source image to targeted image. It transfers the color features from the target to source image while preserving the dimensional information [95].
Color Constancy Algorithms	This algorithm has a crucial role in a condition in which lightening may vary on images due to any reason. The goal is to keep the image color consistent irrespective of any lightening situation. This is helpful in the object detection process where variations in lightening may affect the object appearance [35].

Detection and Sobel Operator, Region-Based Segmentation for partitioning images based on attributes, Deep Learning-Based Methods utilizing CNN architectures like U-Net, and Clustering-Based Methods, like K-means and Fuzzy C-means for grouping similar pixels.

These are many image segmentation techniques; each has its own advantages and disadvantages. The better selection of image segmentation techniques depends on various factors like image nature, computational resources, objects complexity, etc. Table 4 shows that various segmentation techniques, including K-means Clustering, Gray-scale thresholding, and Histogram-based Analysis, have been applied in maturity detection, ripeness detection, and weight prediction of agricultural products between 2009 and 2023.

Tab. 3. Techniques used in image segmentation for image processing.

Techniques	Description
Thresholding Based Methods	This method uses the gray-level histogram analysis to determine the optimal threshold value for an image segmentation. The method works on to increase the inter-class variance of pixel intensities in an image. Otsu (1979) implemented a threshold selection method, that is very effective in object detection, image analysis in the medical field, document processing and other applications [85].
Edge Based Segmentation	<p>This technique identifies the boundaries/edges of an object within an image. These edges refer to the transitions in color, texture and intensity that is useful for object detection, feature selection and image pre-processing. Here is an overview of edge-based segmentation in image pre-processing:</p> <ol style="list-style-type: none"> 1. Canny Edge Detection: The Canny edge detection method is common to identify the edges in an image. This method follows the different steps like gradient computation to detect the edge strength, hysteresis thresholding to recognise and interconnect the edges, non-maxima suppression to decrease the edge thickness, Gaussian smoothing for the noise reduction and others. It minimizes the false detection and maximizes the true edge detection and can robust the noises [18]. 2. Sobel Operator: It is a convolutional based gradient method to detect the simple edges of an image. It involves with separate kernels to detect horizontal and vertical edges of an image by calculating the approximate gradient magnitude [108]. 3. Laplacian of Gaussian (LoG): LoG is the combination of Laplacian edge detection and Gaussian smoothing to enhance edge clarity and to reduce the noise as well [76]. 4. Gradient-Based Methods: This method extracts gradient magnitude and the direction of pixels in an image for edge detection. Techniques like Scharr operator [17], Prewitt operator [21], Robert Cross operator [82] are mostly used in gradient-based method for edge detection [91].
Region Based Segmentation	This segmentation technique is used to partition an image based on its color intensity, pixel resolution, texture, or other attributes. It is also employed to combine pixels that have same features into an identical region. It provides the meaningful and more obvious result in the presence of unwanted background and noise, compared to other methods like pixel-wise segmentation technique. The watershed algorithm is also used for image segmentation based on image gradient [120].
Deep Learning Based Methods	CNNs have achieved the incredible success in the field of image segmentation of machine learning tasks. It works with different architectures like SegNet [60], FCN (Fully Convolutional Network) [57] and U-Net [6] and uses coder decoder structures to produce safeguard for spatial information [97].
Clustering Based Methods	<p>Clustering based methodology partitions an image into different segments/regions based on the pixels' similarity and try to combine them that have common features like similar color, intensity, resolution, spatial proximity, texture, and others.</p> <ol style="list-style-type: none"> 1. K-mean clustering methods partition image pixels into the number of K clusters based on their similar features. It also involves in color based (RGB) segmentation or LAB (Lightness, Green-Red, Blue yellow) spaces [44]. 2. Fuzzy C-means clustering method is an extended version of K-mean clustering method, in which every pixel is assigned to a degree in each cluster to accomplish the task of image segmentation [15].

Tab. 4. Descriptions of segmentation techniques used for palm fruits analysis.

Year	Segmentation Techniques	Applications	Reference
2009	K-means Clustering Algorithm	Maturity Detection	[54]
2009	Gray-scale thresholding	Maturity & Weight Prediction	[59]
2015	Histogram-based Analysis	Maturity Detection	[102]
2019	K-means Clustering Algorithm	Maturity Detection	[40]
2023	K-means Clustering Algorithm	Ripeness Detection	[58]

Tab. 5. Various methodologies of feature extraction in image pre-processing

Methods	Description
Histogram based	<p>This method is a graphical representation to analyse the distribution of pixel intensities in an image. By analysing the histogram, many features like skewness, entropy and uniformity can be extracted to deliver the detailed information about an image [11]. Popular histogram methods are:</p> <ol style="list-style-type: none"> 1. Histogram Equalization: It improves the contrast of an image by applying the redistribution task on pixel intensities in an image [85]. 2. Local Binary Patterns (LBP): Extracting the features of an image by performing the comparison on each pixel with its nearest pixels [84] 3. This method introduced by Dalal and Triggs in 2005 to represent the edge characteristics, local gradient texture and shape information of an image. In this method, an image is divided into small regions that are called cells, then the histogram orientation for each cell is computed [25].
Transform based	<p>Mathematical transformation is applied in this method to extract the image features from the specified domain. It transforms the basic image attributes into other domain where features extraction can be performed easily. In this method, Principal Component Analysis (PCA) technique is used to reduce dimension and Discrete Wavelet Transform (DWT) to acquire the information about an image with multi-resolution [61]. Common Transformed-based methods are:</p> <ol style="list-style-type: none"> 1. Wavelet Transformation: It transforms the images into different frequency patterns and captures each minor details of an image [75]. 2. Discrete Fourier Transform (DFT): This transformation method is used to compress and filter the image. It transforms the images into different frequency domains [55].
Deep Learning Based Methods	<p>The Deep Learning-based method involves the Convolutional Neural Network (CNNs) for features selection in the image pre-processing to automate the learning of hierarchical demonstration from raw image data [24].</p>
Texture Analysis	<p>This technique is used to collect the information about the correlation between image pixels based on its texture. It describes and categorize the repetitive patterns found in an image. The methods like Gabor Filter and Local Binary Patterns (LBP) are common to provide the information about patterns and texture properties within an image [48]. Gabor Filters named after a physicist Dennis Gabor, generally used for texture analysis. They are the linear filters employed on multiple orientations and scales [73]. The other method used in Texture Analysis is Gray-Level Co-occurrence Matrix (GLCM) to capture the statistical features of image textures by performing analysis on pixel correlations [51].</p>
Statistical Features	<p>These methods used to capture the statistical features of an image, such as kurtosis, skewness, mean and standard deviation for each color pixel or the entire image. These methods are very efficient to provide information about the distribution of an image intensity [28].</p>

2.1.6. Feature extraction

Feature extraction in image pre-processing involves determining and identify the color features from the segmented image. Color features have the attributes of shape, texture patterns and histogram which distinguish one image from others. The goal of this task is to acquire the most relevant features from an image to satisfy the criteria of machine learning models. The Tab 5 summarizes various image feature extraction methods, including histogram-based, transform-based, deep learning-based, texture analysis, and statistical features, providing a comprehensive overview of techniques used to analyze and understand image content.

Tab. 6. Techniques used in feature scaling for image processing.

Techniques	Description
Min-Max Scaling	This is a most common technique to scale the pixel values into a fixed range, usually between 0 and 1. This is a widely used technique due to its effectiveness and simplicity [107].
Z-Score Normalization	It is also known as the Standardization, to transform the pixel values into the mean value of 0 and the standard deviation with 1. This technique is best suitable and most effective in the case where the pixel values vary significantly across the images [22].
Robust Scaling	This technique is used to scale the image according to the model needs. The Median absolute technique (MAD) method is generally used in this technique to reduce the effect of outliers. It is a more robust method compared to other classical mean and standard deviation-based normalization techniques [19].
Histogram Equalization	This technique is commonly used to redistribute the intensities of the pixel to boost the image's contrast. It modifies and improves the visual view of images and make it beneficial for the task in segmentation and object detection [38].
PCA-Based Methods	It stands for Principal Component Analysis (PCA), used for feature extraction and to decrease the dimensionality of image data. It converts the original pixel space into a lower dimensional space while maintaining the relevant information about an image. It minimizes the computational complexity and improves the performance of the model [105].
Adaptive Histogram Equalization (AHE)	It is a standard version of Histogram Equalization technique that employed on only small regions of an image to adapt the variations of local contrast. This technique is highly recommended when lightening conditions are not uniform [89].
Contrast Limited Adaptive Histogram Equalization(CLAHE)	It is an extension of AHE, widely used in processing of medical imaging and satellite images. It limits the contrast amplification and minimizes the over noise in an image [88].

2.1.7. Feature scaling

The term Feature Scaling in image pre-processing of machine learning refers to the standardization and normalization of image pixels before utilizing in machine learning model. Image is represented in the form of array with pixel values, where each pixel in the array corresponds to a numerical value that represents the color or intensity of an image. Pixel values depend on many factors such as image capturing devices, sensors, or the lightening conditions. Therefore, Feature Scaling technique is applied to bring them with common pixel values to form a similar scale and improve the performance. The table 6 provides a comprehensive overview of various image preprocessing techniques, including normalization methods (Min-Max Scaling, Z-Score Normalization, Robust Scaling), histogram equalization techniques (Histogram Equalization, Adaptive Histogram Equalization, Contrast Limited Adaptive Histogram Equalization), and feature extraction methods (PCA-Based Methods), highlighting their applications and benefits in image analysis tasks.

Tab. 7. Techniques used in feature scaling for image processing

Data Types	Description
Image	Image data augmentation technique is fundamentally used to increase the size and diversity of original image datasets by applying various transformation methods. Some common functionalities for image data augmentation are: Flipping, Cropping, Addition of Gaussian noise, Adjustment of contrast, Elastic transformation, Rotation, Scaling, Translation, Adjustment of brightness, Color jittering and so on. These techniques can be applied independently or in the combined to generate the augmented image datasets. The selection criteria for image augmentation techniques depends on various factors such as dataset features, task consideration and the level of augmentation required [106].
Textual	Textual data augmentation technique is employed to improve the performance and robustness of NLP, especially with limited training datasets. This technique generates the new synthetic data points by applying numerous transformations to the existing text data. Some functionalities in textual data augmentation technique are Addition, deletion, and word swapping, paraphrasing and replacement of synonyms. These are generally used in NLP tasks such as sentiment analysis, textual classification and to make the model capable to generate a wide range of training datasets [122].
Audio	Audio data augmentation technique is basically used in background sound classification, speaker identification and automatic speech recognition (ASR). This technique is involved in generating the new audio samples from the existing one to increase the size of datasets. The tasks involve in this technique are Time stretching, addition of background noise, Speed setup, shifting of pitch and the time warping [87].
Tabular	Tabular data augmentation technique is used to address the challenges faced with imbalanced datasets and to increase the size of tabular datasets from the existing one. The tabular data are organised in rows and columns, typically found in databases or spreadsheets. The tasks involve in tabular data augmentation techniques are: Synthetic Minority Over-Sampling Technique (SMOTE), Noise injection and Random sampling etc [34].
Video	Video augmentation technique is aimed to generate the new samples while preserving the significant content of the original videos. The involved in these techniques are: Temporal cropping, Frame sampling and Frame changing, etc. These tasks may be applied individually or in a combined mode to enhance the diversity of training datasets [20].

2.1.8. Data augmentation

Data augmentation technique aimed to improve the robustness and model generalization by increasing the training size of datasets. It typically generates the new datasets from the existing ones. The Tab 7 provides a comprehensive overview of data augmentation techniques for various data types, including image, textual, audio, tabular, and video data, highlighting the methods used to increase dataset size and diversity while preserving data integrity and enhancing model performance.

2.2. Classification methods

Classification is very essential task in the context of machine learning, used to categorize the data into predefined labels or classes based on their features. It is a supervised learning technique in which model is trained with the labelled data and each data point is assigned with a class label. The main goal of the classification task is to perform

a mapping from input features to class labels, then can be utilized later to predict the class label of unseen, new data. There are many classification algorithms, each has its own application, pros, and cons, etc. The common classification algorithms will be presented in the following sections.

2.2.1. Traditional Classification Methods

The classification methods used to categorize the data based on features have been used for several decades. They are the main foundations for modern classification algorithms in the current era. However, due to the advancement in the field of deep learning and neural networks, the modern classification methods have become a popular choice for larger and complex datasets. The common traditional classification methods are:

1. **Decision Trees:** As the name suggests, it is a hierarchical or tree like structure in which leaf node represents the class label, branches to decision rule and internal node represents the features. It is the most famous supervised machine learning algorithm known for its simplicity, easiness for implementation and interpretability. It is used for classification and regression to handle both numerical and categorical data [63].
2. **k-Nearest Neighbors (k -NN or KNN):** KNN is a very simple and effective algorithm for the image classification task. In this method, the class label of the new dataset is basically determined by the class label of its nearest neighbors. Because of the non-parametric method, it classifies instances based on the majority class labels of its nearest neighbors. It is widely used for both classification and regression tasks [64].
3. **Support Vector Machines (SVM):** SVM is a very powerful, effective, and popular supervised machine learning algorithm for image classification, especially with small to medium size datasets where interpretability for the model is required. It constructs an optimal hyperplane or a set of hyperplanes to separate the different classes while minimizing the classification errors and maximizing the margin between classes. It is designed to work very well as a linear and nonlinear classification in high-dimensional space [47].
4. **Logistic Regression:** Logistic Regression is widely used linear model for binary classification in machine learning, also applied for image classification. It uses the logistic function to estimate the probability that given instances belong to a specific class label or not [69].

2.2.2. Deep Learning-Based Classification Methods

Deep learning-based image classification is significantly a revolutionized advancement in the field of computer vision, capable of achieving a high level of accuracy and scalability across numerous applications. This classification method takes the advantages of deep neural networks, particularly CNNs, to automatically learn hierarchical representations from raw image datasets.

Convolutional Neural Networks (CNN): CNN is a deep learning model, especially designed for analysing and processing the grid data, such as images. It has revolutionized the deep learning-based image classification method. It consists of different types of layers, and each layer plays an important role in the predefined network to automatically learn the hierarchical representation from the raw image data. There are mainly three types of layers found in CNN, such as convolutional layers, pooling layers, and fully connected layers [86].

Recurrent Neural Networks (RNN): It is a type of artificial neural network, primarily designed for time-series and sequential data to be processed. It comes under the category of supervised machine learning techniques based on neuron with at least one feedback loop. It is capable of handling a sequence of random length, unlike the feedforward neural network, which can only process the fixed size of input vectors. Backpropagation through time (BPTT) or its variant, truncated backpropagation through time is used to train RNN for the classification tasks. It has been applied to various classification tasks, such as speech recognition, NLP, sentiment analysis, and others. However, it has the limitation of a gradient problem that makes it unable to train on long range dependencies. To solve this problem, researchers have proposed many advanced RNN architectures, such as GRUs and LSTM. Overall, it is a very powerful tool for performing classification tasks [101].

Generative Adversarial Networks (GANs): GANs primarily focus on image generation tasks because of being a generative model, but may also be used for image classification tasks. A single GAN is designed with two neural networks: generator and discriminator. The generator is responsible for producing fake images while discriminator makes attempts to differentiate between the fake and real images. By using the continuous learning and training process, the generative enhances its capability to create seemingly actual images. They may be utilized to perform the image classification by using data augmentation technique, features selection and other semi-supervised learning model [128].

2.2.3. Ensemble learning methods

Ensemble learning methods are emerging powerful machine learning techniques, especially to perform the image classification tasks. The methods are employed to make improvements in model performance by achieving the goal of combining multiple base learners to get a stronger robust predictor. These methods are extensively effective, proven to be robustness and explore the generalization ability in image classification.

However, training of ensemble methods requires some attentions during the process of main learner selection and diversity to improve the efficiency. Here's an overview of the ensemble learning methods used for image classification in machine learning.

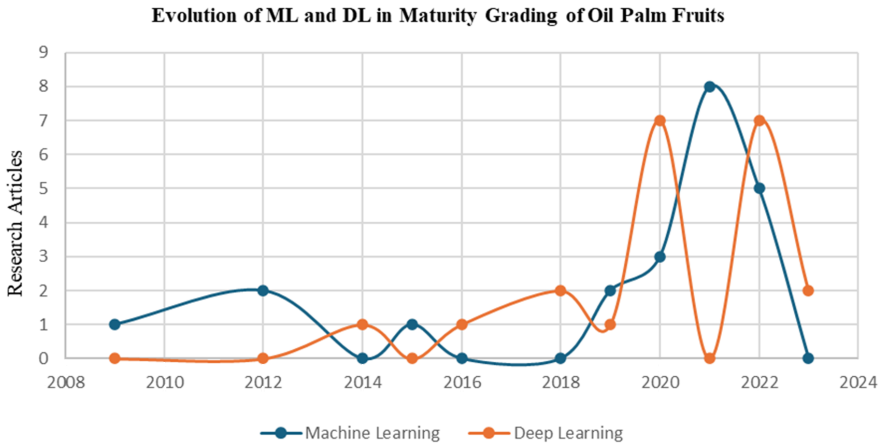


Fig. 1. Research publications (2009-2023) for Maturity Grading of Oil Palm Fruits.

Random Forest: Random forest is an ensemble method that can achieve the required performance with limited computational instances, or in a case where interpretability is needed. It is not normally used for the image classification tasks, but it can be served as a baseline model in the ensemble method, as it constructs the multiple decision tree during model training and predicts the class labels. It is capable of enhancing accuracy and reduce the overfitting [16].

Gradient Boosting Machines (GBM): GBM is not the first choice for image classification, but still can be considerable as an alternative where robustness, interpretability and effectiveness is required. It is a powerful classification method associated with tabular and structured datasets. It can be used for image classification with some applicable features and pre-processing techniques. The fundamental work of GBM is to build a series of weak learners, usually in the form of decision tree to form a stronger learner [14].

Fig. 1 illustrates the evolution of the number of research articles on machine learning and deep learning in the context of oil palm fruit maturity grading over the period from 2008 to 2023, showing a significant increase in the use of deep learning techniques in recent years.

Table 8 provides a comprehensive overview of research on oil palm fruit maturity grading using various machine learning and deep learning classifiers, including PCA, MLP, QDA, ANN, CNN, SVM, ELM, FCM Clustering, Simple Logistic, Lazy KStar, LDA, MDA, YOLO, and YOLOv4, with varying accuracies and dataset sizes.

Tab. 8. Comparative analysis of classification techniques used for maturity grading of oil palm fruits with their accuracy. Tr: training set, Ts: testing set, Val: validation set.

Year	Classifiers	Classification Classes	Data-set Size	Dataset Partition	Accuracy	Ref.
2009	PCA	Unripe, Under-Ripe, Ripe, Over-Ripe	48	Tr: 32, Ts: 16	99.2%	[59]
2012	MLP	Unripe, Under-Ripe, Ripe, Over-Ripe	n/a	n/a	93% (reduced features), 91.67% (full feat.)	[33]
	QDA	Unripe, Ripe, Over-Ripe	120	Tr: 90, Ts: 30	85%	[99]
2014	ANN	Under-Ripe, Ripe, Over-Ripe	469	Tr: 439, Ts: 30	95%	[13]
2015	Ripeness Index derived from GA	Under-Ripe, Ripe	76	Tr: 40, Ts: 36	67.10%	[102]
2016	ANN	Under-Ripe, Ripe	60	Tr: 40, Ts: 20	70%	[104]
2018	CNN	Unripe, Under-Ripe, Ripe, Over-Ripe	120	Tr: 96, Ts: 24	100%	[98]
	ANN	Under-Ripe, Ripe, Over-Ripe	180	Tr: 120, Ts: 60	93%	[4]
2019	SVM	Unripe, Under-Ripe, Ripe, Over-Ripe	400	Tr: 360, Ts: 40	57% by color features, 70% by Bag of Visual Words	[40]
	ELM	Very Good, Good, Quite Good and Poor	297	n/a	MAPE:20-50%	[115]
	CNN	Young Trees Mature Trees	284 244	Tr: 199, Ts: 85 Tr: 159, Ts: 85	95.11% (Young) 92.96% (Mature)	[79]
2020	CNN	Unripe, Under-Ripe, Ripe	200	n/a	85%	[124]
	ANN	Under-Ripe, Ripe, Over-Ripe	450	Tr: 180, Ts: 270	94%	[5]
	FCM Clustering	Unripe, Ripe, Over-Ripe	n/a	n/a	Tr: 73.07%, Ts: 71.04%	[112]
	Simple Logistic	Unripe, Ripe, Over-Ripe	30	n/a	83.8% by OPRiD, 86.8% by NLI	[92]
	Lazy KStar	Unripe, Under-Ripe, Ripe, Over-Ripe	106	Tr: 95, Ts: 11	63%	[116]
	CNN	Unripe and Ripe	n/a	n/a	96% (training)	[100]
	CNN	Unripe, Ripening, Less-Ripe, Almost-Ripe, Ripe, Perfect-Ripe, Over-Ripe	400	Tr: 253, Ts: 77, Val: 80	69% (DenseNet Sigmoid), 69% (ResAtt DenseNet), 64% (DenseNet+SE Layer), 60% (AlexNet)	[111]
	CNN	Unripe (Full), Unripe, Almost-Ripe, Ripe, Ripe (Full), Over-Ripe, Over-Ripe (Full)	400	Tr: 240, Ts: 160	71.34%	[43]
	R-CNN	FFB Detection & Counting	100	Tr: 80, Val: 20	80%	[90]
CNN	Ripe, Unripe	628	n/a	95.6%	[32]	

to be continued in the next page

Tab. 8. Comparative analysis of classification techniques... (continued)

Year	Classifiers	Classification Classes	Data-set Size	Dataset Partition	Accuracy	Ref.
2021	Fine KNN	Under-Ripe, Ripe, Over-Ripe	46	n/a	100%	[93]
	Weighted KNN				91.3%	
	SVM fine Gaussian kernel				80.4%	
	SVM medium Gaussian				97.8%	
	SVM medium Gaussian kernel				91.3%	
	SVM Quadratic kernel				95.7%	
	SVM Cubic kernel				97.8%	
SVM Quadratic discriminant	97.8%					
2022	CNN	Crude, Ripe, Rotten	400	Tr: 320, Ts: 80	Tr: 98%, Ts: 76%	[9]
	ANN	Under-Ripe, Ripe, Over-Ripe	270	90 (ambiguity)	93% (Using BGLAM, ROI2, & ROI3), 93% (statistical color features)	[2]
	KNN SVM				93% 92%	
	LDA MDA ANN KNN	Under-Ripe, Ripe, Over-Ripe	297	n/a	Tr: 86%, Ts: 85.4% Tr: 86.7%, Ts: 81.8% Tr: 99.1%, Ts: 92.5% Tr: 82%, Ts: 74.2%	[134]
	ANN	Under-Ripe, Ripe, Over-Ripe	52	Tr: 33, Ts: 14	97.90%	[118]
	DNN	Bare soil, built-up area, forest, water, immature and mature oil palm	13218	Tr: 10574, Ts: 2644	99.35% (Overall Accuracy), 98.49% (Kappa Accuracy)	[56]
	CNN YOLO	Unripe, Ripe Oil palm tree or Not	490 3100	Tr: 430, Ts: 60 n/a	87.9% 85.6% [83]	[67]
2023	YOLOv4-Tiny 3L (Model_16)	Unripe, Under-ripe, Ripe, Over-Ripe, Abnormal, Empty	57	Tr: 47, Ts: 10	mAP of 90.56% (SingleClass category)	[58]
	CNN	Unripe, Under-ripe, Ripe, Over-Ripe, Abnormal, Empty	4160	Tr: 2908, Ts: 417, Val: 835	88.01% (YOLOv4-320), 88.27% (YOLOv4-416), 88.94% (YOLOv4-512)	[109]

3. Deep learning models

Deep learning models have emerged as the revolutionary in various fields such as the healthcare, NLP, and Computer Vision, due to their ability to learn automatically from hierarchical data. These models are inspired by the human brain in functionality and structures, consisting of multiple layers of interconnected AI neurons that make classification and prediction process a success. CNN is very useful and well suited for the tasks involving spatial data and images. AlexNet [65] and LeNet-5 [71] are the pioneer architecture of CNN that demonstrated their effectiveness in image classification tasks. Alfatni et al. (2018) proposed a real-time maturity grading system for oil palm FFBs using SVM, ANN and KNN classifiers [4]. Table 9 presents a comprehensive overview of deep learning models used in oil palm fruit maturity grading, including AlexNet, LeNet, DenseNet, ResNet, Inception, CNN, RCANet, VGG-16, MLP, ANN, YOLO, and YOLOv4, with their corresponding accuracies and references.

Tab. 9. Summary of deep learning models applied in palm fruit detection.

Year	Deep Learning Model	Accuracy	Reference
2018	AlexNet	100%	[49]
2019	LeNet	95.11% (Young), 92.96% (Mature)	[79]
2020	AlexNet	85%	[124]
	DenseNet Sigmoid	69%	
	ResAtt DenseNet	69%	[111]
	DenseNet + SE Layer	64%	
	AlexNet	60%	
	ResNet152	71.34%	[43]
	ResNet-50	86%	
	ResNet-101	82%	
	Inception V2	85%	[90]
	Inception ResNet V2	82%	
2021	CNN	95.6%	[32]
	RCANet	96.88%	[30]
2021	ResNet-50	91.24%	[127]
	VGG-16	98.13%	
2022	CNN	Tr: 98%, Ts: 76%	[9]
	Multilayer Perceptron (MLP)	92.5%	[134]
	ANN	97.90%	[118]
	DNN	99.35%	[56]
	YOLOv4	70.19%	[67]
	YOLO	85.6%	[83]
	YOLOv3	97.28%	
	YOLOv4	97.74%	[123]
	YOLOv5m	94.94%	
DNN	91%	[7]	

to be continued in the next page

Tab. 9. Summary of deep learning models... (continued)

Year	Deep Learning Model	Accuracy	Reference
2023	YOLOv4-CSPDarknet53	mAP 97.64%	
	YOLOv4-Tiny	mAP 83.57%	[58]
	YOLOv4-Tiny 3L	mAP 90.56%	
	YOLOv4-320	88.01%	[109]

4. Conclusion

This review paper explores and examines the various classical methods and DL methods used for the maturity grading of oil palm FFBS. Each journal paper has been analyzed with its merits and demerits. It contains the exploration of deep knowledge regarding image analysis techniques, deep learning algorithms and spectral imaging techniques. This review highlights the advancements of existing approaches, including enhanced efficiency, improved accuracy, and reliable consistency. It also observes the efficacy of various color features used for image analysis. Additionally, the challenges or limitations associated with each paper were also mentioned after implementing these techniques, such as limited dataset size, the impact of environmental and regional variations. The review also focuses on the future prospectives for further research and development in the field of maturity grading of oil palm FFBS. This systematic review presents a valuable remark for researchers to optimize the maturity assessment of oil palm fruits.

Acknowledgement

Authors gratefully acknowledge the National Defence University, Malaysia for financial assistance under research Grant J0333 – UPM/2023/GPJP/ICT/1 (Research Title: *Modelling of Visual-Spectral Approach Using Quantum-inspired Deep Learning for Oil Palm Fruit Maturity Precision*). Authors also confirm that no external dataset is used in this article.

References

- [1] S. Abdul Saleem and T. Abdul Razak. Survey on color image enhancement techniques using spatial filtering. *International Journal of Computer Applications*, 94(9):39–45, 2014. doi:10.5120/16374-5837.
- [2] M. S. M. Alfatni, S. Khairunniza-Bejo, M. H. B. Marhaban, O. M. B. Saeed, A. Mustapha, et al. Towards a real-time oil palm fruit maturity system using supervised classifiers based on feature analysis. *Agriculture*, 12(9):1461, 2022. doi:10.3390/agriculture12091461.
- [3] M. S. M. Alfatni, A. R. M. Shariff, M. Z. Abdullah, M. H. Marhaban, S. B. Shafie, et al. Oil palm fresh fruit bunch ripeness classification based on rule-based expert system of roi image processing technique results. In: *Proc. 7th IGRSM Int. Conf. and Exhibition on Geospatial & Remote Sensing*, vol. 20 of *IOP Conference Series: Earth and Environmental Science*, p. 012018.

- Institute of Physics Publishing, Kuala Lumpur, Malaysia, 22-23 Apr 2014. doi:10.1088/1755-1315/20/1/012018.
- [4] M. S. M. Alfatni, A. R. M. Shariff, S. K. Bejo, O. M. B. Saaed, and A. Mustapha. Real-time oil palm FFB ripeness grading system based on ANN, KNN and SVM classifiers. In: *Proc. 9th IGRSM Int. Conf. and Exhibition on Geospatial & Remote Sensing*, vol. 169 of *IOP Conference Series: Earth and Environmental Science*, p. 012067. Institute of Physics Publishing, Kuala Lumpur, Malaysia, 24-25 Apr 2018. doi:10.1088/1755-1315/169/1/012067.
 - [5] M. S. M. Alfatni, A. R. M. Shariff, O. M. Ben Saaed, A. M. Albhah, and A. Mustapha. Colour feature extraction techniques for real time system of oil palm fresh fruit bunch maturity grading. In: *Proc. 10th IGRSM Int. Conf. and Exhibition on Geospatial & Remote Sensing*, vol. 540 of *IOP Conference Series: Earth and Environmental Science*, p. 012092. Institute of Physics Publishing, Kuala Lumpur, Malaysia, 20-21 Oct 2020. doi:10.1088/1755-1315/540/1/012092.
 - [6] Z. Alom, M. Hasan, C. Yakopcic, T. M. Taha, and V. K. Asari. Recurrent residual convolutional neural network based on U-Net (R2U-Net) for medical image segmentation. *arXiv*, 2018. ArXiv:1802.06955. doi:10.48550/arXiv.1802.06955.
 - [7] Y. Ang, H. Z. M. Shafri, Y. P. Lee, S. A. Bakar, H. Abidin, et al. Oil palm yield prediction across blocks from multi-source data using machine learning and deep learning. *Earth Science Informatics*, 15(4):2349–2367, 2022. doi:10.1007/s12145-022-00882-9.
 - [8] S. Anwar and N. Barnes. Real image denoising with feature attention. In: *Proc. 2019 IEEE/CVF Int. Conf. Computer Vision (ICCV)*, pp. 3155–3164. Seoul, Korea (South), 27 Oct – 02 Nov 2019. doi:10.1109/ICCV.2019.00325.
 - [9] S. Ashari, G. J. Yanris, and I. Purnama. Oil palm fruit ripeness detection using deep learning. *Sinkron*, 6(2):649–656, 2022. doi:10.33395/sinkron.v7i2.11420.
 - [10] S. Avidan and A. Shamir. Seam carving for content-aware image resizing. *ACM Transactions on Graphics*, 26(3):10-1–10-9, July 2007. doi:10.1145/1239451.1239461.
 - [11] M. Bagheri and M. H. Sedaaghi. A new method for detecting jittered PRI in histogram-based methods. *Turkish Journal of Electrical Engineering and Computer Sciences*, 26(3):1214–1224, 2018. doi:10.3906/elk-1710-169.
 - [12] A. Bansal and N. Singh. Image enhancement techniques: A review. *Asian Journal For Convergence In Technology (AJCT)*, 6(2):07–11, 2020. doi:10.33130/AJCT.2020v06i02.002.
 - [13] O. M. Bensaeed, A. M. Shariff, A. B. Mahmud, H. Shafri, and M. Alfatni. Oil palm fruit grading using a hyperspectral device and machine learning algorithm. In: *Proc. 7th IGRSM Int. Conf. and Exhibition on Geospatial & Remote Sensing*, vol. 20 of *IOP Conference Series: Earth and Environmental Science*, p. 012017. Institute of Physics Publishing, Kuala Lumpur, Malaysia, 22-23 Apr 2014. doi:10.1088/1755-1315/20/1/012017.
 - [14] C. Bentéjac, A. Csörgő, and G. Martínez-Muñoz. A comparative analysis of gradient boosting algorithms. *Artificial Intelligence Review*, 54(3):1937–1967, 2021. doi:10.1007/s10462-020-09896-5.
 - [15] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Springer, New York, USA, 1981. doi:10.1007/978-1-4757-0450-1.
 - [16] G. Biau and E. Scornet. A random forest guided tour. *Test*, 25(2):197–227, 2016. doi:10.1007/s11749-016-0481-7.
 - [17] N. Bibi and H. Dawood. SEBR: Schar edge-based regularization method for blind image deblurring. *Arabian Journal for Science and Engineering*, 49(3):3435–3451, 2024. doi:10.1007/s13369-023-07986-4.

- [18] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986. doi:10.1109/ASICON.2011.6157287.
- [19] X. H. Cao, I. Stojkovic, and Z. Obradovic. A robust data scaling algorithm to improve classification accuracies in biomedical data. *BMC Bioinformatics*, 17:359, 2016. doi:10.1186/s12859-016-1236-x.
- [20] N. Cauli and D. Reforgiato Recupero. Survey on videos data augmentation for deep learning models. *Future Internet*, 14(3):93, 2022. doi:10.3390/FI14030093.
- [21] G. N. Chaple, R. D. Daruwala, and M. S. Gofane. Comparisons of Robert, Prewitt, Sobel operator based edge detection methods for real time uses on FPGA. In: *Proc. Int. Conf. Technologies for Sustainable Development (ICTSD 2015)*, pp. 1–4. Mumbai, India, 04-06 Feb 2015. doi:10.1109/ICTSD.2015.7095920.
- [22] C. Cheadle, M. P. Vawter, W. J. Freed, and K. G. Becker. Analysis of microarray data using Z Score transformation. *The Journal of Molecular Diagnostics*, 5(2):73–81, 2003. doi:10.1016/S1525-1578(10)60455-2.
- [23] C. Chen, Q. Chen, J. Xu, and V. Koltun. Learning to see in the dark. In: *Proc. 2018 IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pp. 3291–3300. Salt Lake City, UT, USA, 18-23 Jun 2018. doi:10.1109/CVPR.2018.00347.
- [24] Y. Chen, H. Jiang, C. Li, X. Jia, and P. Ghamisi. Deep feature extraction and classification of hyperspectral images based on convolutional neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 54(10):6232–6251, 2016. doi:10.1109/TGRS.2016.2584107.
- [25] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In: *Proc. 2005 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'05)*, pp. 886–893. San Diego, CA, USA, 20-25 Jun 2005. doi:10.1109/CVPR.2005.177.
- [26] L. M. Dale, A. Thewis, C. Boudry, I. Rotar, P. Dardenne, et al. Hyperspectral imaging applications in agriculture and agro-food product quality and safety control: A review. *Applied Spectroscopy Reviews*, 48(2):142–159, 2013. doi:10.1080/05704928.2012.705800.
- [27] D. Danon, M. Arar, D. Cohen-Or, and A. Shamir. Image resizing by reconstruction from deep features. *Computational Visual Media*, 7(4):453–466, 2021. doi:10.1007/s41095-021-0216-x.
- [28] S. Ding, W. Jia, C. Su, F. Jin, and Z. Shi. A survey on statistical pattern feature extraction. In: *Proc. Conf. Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence (ICIC 2008)*, vol. 5227 of *Lecture Notes in Computer Science*, pp. 701–708. Shanghai, China, 15-18 Sep 2008. doi:10.1007/978-3-540-85984-0_84.
- [29] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In: *Proc. European Conf. Computer Vision (ECCV 2014)*, vol. 8692 of *Lecture Notes in Computer Science*, pp. 184–199. Zurich, Switzerland, 6-12 Sep 2014. doi:10.1007/978-3-319-10593-2_13.
- [30] R. Dong, W. Li, H. Fu, L. Gan, L. Yu, et al. Oil palm plantation mapping from high-resolution remote sensing images using deep learning. *International Journal of Remote Sensing*, 41(5):2022–2046, 2020. doi:10.1080/01431161.2019.1681604.
- [31] S. R. Dubey and A. S. Jalal. Application of image processing in fruit and vegetable analysis: A review. *Journal of Intelligent Systems*, 24(4):405–424, 2015. doi:10.1515/jisys-2014-0079.
- [32] P. Dutta. Palm oil classification using deep learning. *International Journal of Advance Research, Ideas and Innovations in Technology*, 6(5):V6I5-1379, 2020. <https://www.ijariit.com/manuscript/palm-oil-classification-using-deep-learning/>.
- [33] N. Fadilah, J. Mohamad-Saleh, Z. A. Halim, H. Ibrahim, and S. S. S. Ali. Intelligent color vision system for ripeness classification of oil palm fresh fruit bunch. *Sensors*, 12(10):14179–14195, 2012. doi:10.3390/s121014179.

- [34] J. Fang, C. Tang, Q. Cui, F. Zhu, L. Li, et al. Semi-supervised learning with data augmentation for tabular data. In: *Proc. 31st ACM Int. Conf. Information & Knowledge Management (CIKM '22)*, p. 3928–3932. Atlanta, GA, USA, 2022. doi:[10.1145/3511808.3557699](https://doi.org/10.1145/3511808.3557699).
- [35] G. D. Finlayson, M. S. Drew, and B. V. Funt. Color constancy: generalized diagonal transforms suffice. *Journal of the Optical Society of America A*, 11(11):3011–3019, 1994. doi:[10.1364/josaa.11.003011](https://doi.org/10.1364/josaa.11.003011).
- [36] G. D. Finlayson, S. D. Hordley, and P. M. Hubel. Color by correlation: A simple, unifying framework for color constancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1209–1221, 2001. doi:[10.1109/34.969113](https://doi.org/10.1109/34.969113).
- [37] D. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2003.
- [38] P. Garg and T. Jain. A comparative study on histogram equalization and cumulative histogram equalization. *International Journal of New Technology and Research*, 3(9):73–81, 2003. doi:[10.1016/S1525-1578\(10\)60455-2](https://doi.org/10.1016/S1525-1578(10)60455-2).
- [39] S. Garg and G. Ramakrishnan. Advances in quantum deep learning: An overview. *arXiv*, 2020. ArXiv:2005.04316. doi:[10.48550/arXiv.2005.04316](https://doi.org/10.48550/arXiv.2005.04316).
- [40] S. A. Ghazalli, H. Selamat, Z. Omar, and R. Yusof. Image analysis techniques for ripeness detection of palm oil fresh fruit bunches. *ELEKTRIKA—Journal of Electrical Engineering*, 18(3):57–62, 2019. doi:[10.11113/elektrika.v18n3.192](https://doi.org/10.11113/elektrika.v18n3.192).
- [41] R. C. Gonzalez, R. E. Woods, and S. L. Eddins. *Digital Image Processing Using MATLAB*. Gatesmark, 2020. <https://www.imageprocessingplace.com>.
- [42] S. Guo, Z. Yan, K. Zhang, W. Zuo, and L. Zhang. Toward convolutional blind denoising of real photographs. In: *Proc. 2019 IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, pp. 1712–1722. Long Beach, CA, USA, 15–20 Jun 2019. doi:[10.1109/CVPR.2019.00181](https://doi.org/10.1109/CVPR.2019.00181).
- [43] Harsawardana, R. Rahutomo, B. Mahesworo, T. W. Cenggoro, A. Budiarto, et al. AI-based ripeness grading for oil palm fresh fruit bunch in smart crane grabber. In: *Proc. 3rd Int. Conf. Eco Engineering Development*, vol. 426 of *IOP Conference Series: Earth and Environmental Science*, p. 012147. Institute of Physics Publishing, 13–14 Nov 2020. doi:[10.1088/1755-1315/426/1/012147](https://doi.org/10.1088/1755-1315/426/1/012147).
- [44] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979. doi:[10.2307/2346830](https://doi.org/10.2307/2346830).
- [45] N. H. Harun, N. Mison, R. M. Sidek, I. Aris, D. Ahmad, et al. Investigations on a novel inductive concept frequency technique for the grading of oil palm fresh fruit bunches. *Sensors (Switzerland)*, 13(2):2254–2266, 2013. doi:[10.3390/s130202254](https://doi.org/10.3390/s130202254).
- [46] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In: *Proc. 2016 IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778. Las Vegas, NV, USA, 27–30 Jun 2016. doi:[10.1109/CVPR.2016.90](https://doi.org/10.1109/CVPR.2016.90).
- [47] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf. Support vector machines. *IEEE Intelligent Systems and their Applications*, 13(4):18–28, 1998. doi:[10.1109/5254.708428](https://doi.org/10.1109/5254.708428).
- [48] A. Humeau-Heurtier. Texture feature extraction methods: A survey. *IEEE Access*, 7:8975–9000, 2019. doi:[10.1109/ACCESS.2018.2890743](https://doi.org/10.1109/ACCESS.2018.2890743).
- [49] Z. Ibrahim, N. Sabri, and D. Isa. Palm oil fresh fruit bunch ripeness grading recognition using convolutional neural network. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, 10(3-2):109–113, 2018. <https://jt.ec.utem.edu.my/jt/ec/article/view/4720>.
- [50] S. Ioffe. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. In: *Advances in Neural Information Processing Systems 30 – Proc. 30th Conf. Neural*

Information Processing Systems (NIPS 2017), vol. 30, pp. 1945–1953. Long Beach, CA, USA, 4-9 Dec 2017. <http://papers.nips.cc/paper/by-source-2017-1198/>.

- [51] N. Iqbal, R. Mumtaz, U. Shafi, and S. M. H. Zaidi. Gray level co-occurrence matrix (GLCM) texture based crop classification using low altitude remote sensing platforms. *PeerJ Computer Science*, 7:e536, 2021. doi:10.7717/PEERJ-CS.536.
- [52] N. Ismail and O. A. Malik. Real-time visual inspection system for grading fruits using computer vision and deep learning techniques. *Information Processing in Agriculture*, 9(1):24–37, March 2022. doi:10.1016/j.inpa.2021.01.005.
- [53] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with Conditional Adversarial Networks. In: *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pp. 5967–5976. Honolulu, HI, USA, 2017. doi:10.1109/CVPR.2017.632.
- [54] A. Jaffar, R. Jaafar, N. Jamil, C. Y. Low, and B. Abdullah. Photogrammetric grading of oil palm fresh fruit bunches. *International Journal of Mechanical & Mechatronics Engineering*, 9(10):7–13, 2009.
- [55] A. K. Jain. *Fundamentals of digital image processing*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1989.
- [56] A. N. Jarayee, H. Z. M. Shafri, Y. Ang, Y. P. Lee, S. A. Bakar, et al. Oil palm plantation land cover and age mapping using Sentinel-2 satellite imagery and machine learning algorithms. In: *Proc. 8th Int. Conf. Geomatics and Geospatial Technology (GGT 2022)*, vol. 1051 of *IOP Conference Series: Earth and Environmental Science*, p. 012024. Institute of Physics Publishing, Online, 25-26 May 2022. doi:10.1088/1755-1315/1051/1/012024.
- [57] X. Jiang, Y. Wang, W. Liu, S. Li, and J. Liu. CapsNet, CNN, FCN: Comparative performance evaluation for image classification. *International Journal of Machine Learning*, 9(6):840–848, 2019. doi:10.18178/ijmlc.2019.9.6.881.
- [58] F. A. Junior and Suharjito. Video based oil palm ripeness detection model using deep learning. *Heliyon*, 9(1):e13036, 2023. doi:10.1016/j.heliyon.2023.e13036.
- [59] P. Junkwon, T. Takigawa, H. Okamoto, H. Hasegawa, M. Koike, et al. Potential application of color and hyperspectral images for estimation of weight and ripeness of oil palm (*Elaeis guineensis* Jacq. var. tenera). *Agricultural Information Research*, 18(2):72–81, 2009. doi:10.3173/air.18.72.
- [60] B. Khagi and G. R. Kwon. Pixel-label-based segmentation of cross-sectional brain MRI using simplified SegNet architecture-based CNN. *Journal of Healthcare Engineering*, 2018:3640705, 2018. doi:10.1155/2018/3640705.
- [61] S. Khalid, T. Khalil, and S. Nasreen. A survey of feature selection and feature extraction techniques in machine learning. In: *Proc. 2014 Science and Information Conference (SAI 2014)*, pp. 372–378. IEEE, London, UK, 27-29 Aug 2014. doi:10.1109/SAI.2014.6918213.
- [62] J. Kim, J. K. Lee, and K. M. Lee. Accurate image super-resolution using very deep convolutional networks. In: *Proc. 2016 IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pp. 1646–1654. Las Vegas, NV, USA, 27-30 Jun 2016. doi:10.1109/CVPR.2016.182.
- [63] S. B. Kotsiantis. Decision trees: A recent overview. *Artificial Intelligence Review*, 39(4):261–283, 2013. doi:10.1007/s10462-011-9272-4.
- [64] O. Kramer. K-nearest neighbors. In: *Dimensionality Reduction with Unsupervised Nearest Neighbors*, vol. 51 of *Intelligent Systems Reference Library*, pp. 13–23. Springer, Berlin, Heidelberg, 2013. doi:10.1007/978-3-642-38652-7_2.

- [65] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In: *Proc. 25th Int. Conf. Neural Information Processing Systems (NIPS)*, pp. 1097–1105. Curran Associates, Lake Tahoe, NV, USA, 3-8 Dec 2012. https://proceedings.neurips.cc/paper_files/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html.
- [66] O. Kupyn, V. Budzan, M. Mykhailych, D. Mishkin, and J. Matas. DeblurGAN: Blind motion deblurring using conditional adversarial networks. In: *Proc. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8183–8192. Salt Lake City, UT, USA, 18-23 Jun 2018. doi:10.1109/CVPR.2018.00854.
- [67] J. W. Lai, H. R. Ramli, L. I. Ismail, and W. Z. W. Hasan. Real-time detection of ripe oil palm fresh fruit bunch based on yolov4. *IEEE Access*, 10:95763–95770, 2022. doi:10.1109/ACCESS.2022.3204762.
- [68] E. Land and J. McCann. Lightness and retinex theory. *Journal of the Optical Society of America*, 61(1):1–11, 1971. doi:10.1364/JOSA.61.000001.
- [69] M. P. LaValley. Logistic regression. *Circulation*, 117(18):2395–2399, May 2008. doi:10.1161/CIRCULATIONAHA.106.682658.
- [70] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. doi:10.1038/nature14539.
- [71] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2323, 1998. doi:10.1109/5.726791.
- [72] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, et al. Photo-realistic single image super-resolution using a generative adversarial network. In: *Proc. 2017 IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pp. 105–114. Honolulu, HI, USA, 21–26 Jul 2017. doi:10.1109/CVPR.2017.19.
- [73] W. Li, K. Z. Mao, H. Zhang, and T. Chai. Selection of Gabor filters for improved texture feature extraction. In: *Proc. Int. Conf. Image Processing (ICIP)*, pp. 361–364. Hong Kong, China, 26-29 Sep 2010. doi:10.1109/ICIP.2010.5653278.
- [74] Q. Ma, Y. Wang, and T. Zeng. Retinex-based variational framework for low-light image enhancement and denoising. *IEEE Transactions on Multimedia*, 25:5580–5588, 2023. doi:10.1109/TMM.2022.3194993.
- [75] S. G. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, 1989. doi:10.1109/34.192463.
- [76] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings of the Royal Society B Biological Sciences*, 207(1167):187–217, 1980. doi:10.1098/RSPB.1980.0020.
- [77] N. Mison, N. A. Aliteh, N. H. Harun, K. Tashiro, T. Sato, et al. Relative estimation of water content for flat-type inductive-based oil palm fruit maturity sensor. *Sensors (Switzerland)*, 17(1):52, 2017. doi:10.3390/s17010052.
- [78] N. Mison, N. S. K. Azhar, M. N. Hamidon, I. Aris, K. Tashiro, et al. Fruit battery with charging concept for oil palm maturity sensor. *Sensors*, 20(1):52, 2020. doi:10.3390/s20010226.
- [79] N. A. Mubin, E. Nadarajoo, H. Z. M. Shafri, and A. Hamedianfar. Young and mature oil palm tree detection and counting using convolutional neural network deep learning method. *International Journal of Remote Sensing*, 40(19):7500–7515, 2019. doi:10.1080/01431161.2019.1569282.
- [80] A. Mustaffa, F. Arith, N. I. F. Peong, N. R. Jaffar, E. L. Linggie, et al. Segregation of oil palm fruit ripeness using color sensor. *Indonesian Journal of Electrical Engineering and Computer Science*, 25(1), 2022. doi:10.11591/ijeecs.v25.i1.pp130-137.

- [81] S. Nah, T. H. Kim, and K. M. Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In: *Proc. 2017 IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pp. 257–265. Honolulu, HI, USA, 21–26 Jul 2017. doi:[10.1109/CVPR.2017.35](https://doi.org/10.1109/CVPR.2017.35).
- [82] B. R. Naidu, P. L. Rao, P. Babu, and K. V. L. Bhavani. Efficient case study for image edge gradient based detectors – Sobel, Robert Cross, Prewitt and Canny. *International Journal of Electronics Communication and Computer Engineering*, 3(3):561–570, 2012. <https://www.ijecece.org/index.php/issues?view=publication&task=show&id=162>.
- [83] I. Nurhabib, K. B. Seminar, and Sudradjat. Recognition and counting of oil palm tree with deep learning using satellite image. In: *Proc. 2nd Int. Conf. Sustainable Plantation*, vol. 974 of *IOP Conference Series: Earth and Environmental Science*, p. 012058. Institute of Physics Publishing, Bogor, Indonesia, 2–3 Sep 2022. doi:[10.1088/1755-1315/974/1/012058](https://doi.org/10.1088/1755-1315/974/1/012058).
- [84] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002. doi:[10.1109/TPAMI.2002.1017623](https://doi.org/10.1109/TPAMI.2002.1017623).
- [85] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1):62–66, 1979. doi:[10.1109/TSMC.1979.4310076](https://doi.org/10.1109/TSMC.1979.4310076).
- [86] K. O’Shea and R. Nash. An introduction to convolutional neural networks. *arXiv*, 2015. ArXiv:08458v2. doi:[10.48550/arXiv.1511.08458](https://doi.org/10.48550/arXiv.1511.08458).
- [87] D. S. Park, W. Chan, Y. Zhang, C. C. Chiu, B. Zoph, et al. SpecAugment: A simple data augmentation method for automatic speech recognition, 15–19 Sep 2019. doi:[10.21437/Interspeech.2019-2680](https://doi.org/10.21437/Interspeech.2019-2680).
- [88] S. Pizer, R. Johnston, J. Ericksen, B. Yankaskas, and K. Muller. Contrast-limited adaptive histogram equalization: speed and effectiveness. In: *[1990] Proc. First Conference on Visualization in Biomedical Computing*, pp. 337–345. Atlanta, GA, USA, 22–25 May 1990. doi:[10.1109/VBC.1990.109340](https://doi.org/10.1109/VBC.1990.109340).
- [89] S. M. Pizer, E. P. Amburn, J. D. Austin, R. Cromartie, A. Geselowitz, et al. Adaptive histogram equalization and its variations. *Computer Vision, Graphics, and Image Processing*, 39(3):355–368, 1987. doi:[10.1016/S0734-189X\(87\)80186-X](https://doi.org/10.1016/S0734-189X(87)80186-X).
- [90] N. A. Prasetyo, Pranowo, and A. J. Santoso. Automatic detection and calculation of palm oil fresh fruit bunches using faster R-CNN. *International Journal of Applied Science and Engineering*, 17(2):121–134, 2020. doi:[10.6703/IJASE.202005.17\(2\).121](https://doi.org/10.6703/IJASE.202005.17(2).121).
- [91] J. M. S. Prewitt. Object Enhancement and Extraction. In: B. S. Lipkin and A. Rozenfeld, eds., *Picture Processing and Psychopictorics*, pp. 75–150. Academic Press, New York, London, 1970.
- [92] G. J. Quan, A. R. B. M. Shariff, and N. M. Nawi. Grading of maturity of oil palm fruit based on visible and NIR band. In: *Proc. Asian Conference on Remote Sensing (ACRS 2020)*, 2020. <https://acrs-aars.org/proceeding/ACRS2020/35k13w.pdf>.
- [93] T. Raj, F. H. Hashim, A. B. Huddin, A. Hussain, M. F. Ibrahim, et al. Classification of oil palm fresh fruit maturity based on carotene content from raman spectra. *Scientific Reports*, 11(1):18315, 2021. doi:[10.1038/s41598-021-97857-5](https://doi.org/10.1038/s41598-021-97857-5).
- [94] P. R. Rajarapolu and V. R. Mankar. Bicubic interpolation algorithm implementation for image appearance enhancement. *International Journal of Computer Science and Technology*, 8(2):23–26, 2017. <https://www.ijcst.com/vol8/8.2/4-prachi-r-rajarapolu.pdf>.
- [95] E. Reinhard, M. Ashikhmin, B. Gooch, and P. Shirley. Color transfer between images. *IEEE Computer Graphics and Applications*, 21(5):34–41, 2001. doi:[10.1109/38.946629](https://doi.org/10.1109/38.946629).

- [96] Y. D. Rivera-Mendes, J. C. Cuenca, and H. M. Romero. Physiological responses of oil palm (*Elaeis guineensis* Jacq.) seedlings under different water soil conditions. *Agronomia Colombiana*, 34(2):163–171, 2016. doi:10.15446/agron.colomb.v34n2.55568.
- [97] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In: *Medical Image Computing and Computer-Assisted Intervention – Proc. MIC-CAI 2015*, vol. 9351 of *Lecture Notes in Computer Science*, p. 234–241. Springer Verlag, Munich, Germany, 5-9 Oct 2015. doi:10.1007/978-3-319-24574-4_28.
- [98] N. Sabri, Z. Ibrahim, S. Syahlan, N. Jamil, and N. N. A. Mangshor. Palm oil fresh fruit bunch ripeness grading identification using color features. *Journal of Fundamental and Applied Sciences*, 9(4S):563–579, 2018. doi:10.4314/JFAS.V9I4S.32.
- [99] O. M. B. Saeed, S. Sankaran, A. R. M. Shariff, H. Z. M. Shafri, R. Ehsani, et al. Classification of oil palm fresh fruit bunches based on their maturity using portable four-band sensor system. *Computers and Electronics in Agriculture*, 82:55–60, 2012. doi:10.1016/j.compag.2011.12.010.
- [100] A. Y. Saleh and E. Liansitim. Palm oil classification using deep learning. *Science in Information Technology Letters*, 1(1):1–8, 2020. doi:10.31763/sitech.v1i1.1.
- [101] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee. Recent advances in recurrent neural networks. *arXiv*, 2017. ArXiv:01078v3. doi:10.48550/arXiv.1801.01078.
- [102] M. I. Sameen and A. R. b. Mohamed Shariff. The use of genetic algorithm for palm oil fruit maturity detection. In: *Proc. The 36th Asian Conf. Remote Sensing (ACRS 2015)*, pp. 3552–3556. Curran Associates, Inc., 2016, Quezón City, Metro Manila, Philippines, 24-28 Oct 2015.
- [103] Y. D. Sean, D. D. Smith, V. S. P. Bitra, V. Bera, and S. N. Umar. Development of computer vision system for fruits. *Current Journal of Applied Science and Technology*, 40(36):1–11, 2021. doi:10.9734/cjast/2021/v40i3631576.
- [104] M. K. Shabdin, A. R. M. Shariff, M. N. A. Johari, N. K. Saat, and Z. Abbas. A study on the oil palm fresh fruit bunch (FFB) ripeness detection by using hue, saturation and intensity (HSI) approach. In: *Proc. 8th IGRSM Int. Conf. and Exhibition on Geospatial & Remote Sensing*, vol. 37 of *IOP Conference Series: Earth and Environmental Science*, p. 012039. Institute of Physics Publishing, Kuala Lumpur, Malaysia, 13-14 Apr 2014. doi:10.1088/1755-1315/37/1/012039.
- [105] J. H. Shah, M. Sharif, M. Raza, and A. Azeem. A survey: Linear and nonlinear PCA based face recognition techniques. *The International Arab Journal of Information Technology*, 10(6):536–545, 2013. <https://iajit.org/paper/3370/A-Survey-Linear-and-Nonlinear-PCA-Based>.
- [106] C. Shorten and T. M. Khoshgoftaar. A survey on image data augmentation for deep learning. *J Big Data*, 6(1):60, 2019. doi:10.1186/s40537-019-0197-0.
- [107] S. Sinsomboonthong. Performance comparison of new adjusted min-max with decimal scaling and statistical column normalization methods for artificial neural network classification. *International Journal of Mathematics and Mathematical Sciences*, 2022:3584406, 2022. doi:10.1155/2022/3584406.
- [108] I. E. Sobel. *Camera Models and Machine Perception*. Ph.D. thesis, Stanford University, Stanford, CA, 1970.
- [109] Suharjito, F. A. Junior, Y. P. Koeswandy, Debi, P. W. Nurhayati, et al. Annotated datasets of oil palm fruit bunch piles for ripeness grading using deep learning. *Scientific Data*, 10(1):72, 2023. doi:10.1038/s41597-023-01958-x.
- [110] K. Sunilkumar and D. Babu. Surface color based prediction of oil content in oil palm (*Elaeis guineensis* Jacq.) fresh fruit bunch. *African Journal of Agricultural Research*, 8(6):564–569, 2013. doi:10.5897/AJAR12.1789.

- [111] A. Susanto, T. W. Cenggoro, and B. Pardamean. Oil palm fruit image ripeness classification with computer vision using deep learning and visual attention. *Journal of Telecommunication, Electronic and Computer Engineering*, 12(2):21–27, 2020. <https://jtec.utem.edu.my/jtec/article/view/5543>.
- [112] A. Syaifuddin, L. N. A. Mualifah, L. Hidayat, and A. M. Abadi. Detection of palm fruit maturity level in the grading process through image recognition and fuzzy inference system to improve quality and productivity of crude palm oil (CPO). In: *Proc. 3rd Int. Seminar on Innovation in Mathematics and Mathematics Education (ISIMMED 2019)*, vol. 1581 of *Journal of Physics: Conference Series*, p. 012003. Institute of Physics Publishing, Yogyakarta, Indonesia, 3-4 Oct 2020. doi:10.1088/1742-6596/1581/1/012003.
- [113] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer Nature, 2022. doi:10.1007/978-3-030-34372-9.
- [114] E. Teye, C. L. Y. Amuah, T. S. Yeh, and R. Nyorkeh. Nondestructive detection of moisture content in palm oil by using portable vibrational spectroscopy and optimal prediction algorithms. *Journal of Analytical Methods in Chemistry*, 2023:3364720, 2023. doi:10.1155/2023/3364720.
- [115] Y. Triyanto, R. Watrianthos, Y. Sepriani, and K. Rizal. Palm oil prediction production using extreme learning machine. *International Journal of Scientific & Technology Research*, 8:8, 2019. <https://www.ijstr.org/final-print/aug2019/Palm-Oil-Prediction-Production-Using-Extreme-Learning-Machine.pdf>.
- [116] A. Tuerxun, A. R. M. Shariff, R. Janius, Z. Abbas, and G. A. Mahdiraji. Oil palm fresh fruit bunches maturity prediction by using optical spectrometer. In: *Proc. 10th IGRSM Int. Conf. and Exhibition on Geospatial & Remote Sensing*, vol. 540 of *IOP Conference Series: Earth and Environmental Science*, p. 012085. Institute of Physics Publishing, Kuala Lumpur, Malaysia, 20-21 Oct 2020. doi:10.1088/1755-1315/540/1/012085.
- [117] V. Tyagi. *Understanding Digital Image Processing*. CRC Press, Boca Raton, 2018. doi:10.1201/9781315123905.
- [118] G. T. H. Tzuan, F. H. Hashim, T. Raj, A. B. Huddin, and M. S. Sajab. Oil palm fruits ripeness classification based on the characteristics of protein, lipid, carotene, and guanine/cytosine from the Raman spectra. *Plants*, 11(15):1936, 2022. doi:10.3390/plants11151936.
- [119] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv*, 2016. ArXiv:1607.08022. doi:10.48550/arXiv.1607.08022.
- [120] L. Vincent and P. Soille. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(06):583–598, 1991. doi:10.1109/34.87344.
- [121] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, et al. ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks. In: *Proc. European Conf. Computer Vision (ECCV) Workshops*, pp. 63–79. Springer, 2018, Munich, Germany, 8-14 Sep 2018. doi:10.1007/978-3-030-11021-5_5.
- [122] J. Wei and K. Zou. EDA: Easy Data Augmentation techniques for boosting performance on text classification tasks. In: *Proc. ICLR 2019 Workshop on Learning from Limited Labeled Data*. New Orleans, LA, United States, 6-9 May 2019. Accessible on OpenReview. <https://openreview.net/forum?id=BJeIsDvo84>.
- [123] H. Wibowo, I. S. Sitanggang, M. Mushtofa, and H. A. Adrianto. Large-scale oil palm trees detection from high-resolution remote sensing images using deep learning. *Big Data and Cognitive Computing*, 6(3):89, 2022. doi:10.3390/bdcc6030089.

- [124] Z. Y. Wong, W. J. Chew, and S. K. Phang. Computer vision algorithm development for classification of palm fruit ripeness. In: *Proc. 13th. Int. Engineering Research Conference (13TH EURECA 2019)*, vol. 2233 of *AIP Conference Proceedings*, p. 030012, 2020. doi:[10.1063/5.0002188](https://doi.org/10.1063/5.0002188).
- [125] B. Xu, Y. Zhuang, H. Tang, and L. Zhang. Object-based multilevel contrast stretching method for image enhancement. *IEEE Transactions on Consumer Electronics*, 56(3):1746–1754, 2010. doi:[10.1109/TCE.2010.5606321](https://doi.org/10.1109/TCE.2010.5606321).
- [126] W. Yang, X. Zhang, Y. Tian, W. Wang, J. H. Xue, et al. Deep learning for single image super-resolution: A brief review. *IEEE Transactions on Multimedia*, 21(12):3106–3121, 2019. doi:[10.1109/TMM.2019.2919431](https://doi.org/10.1109/TMM.2019.2919431).
- [127] K. Yarak, A. Witayangkurn, K. Kritiyutanont, C. Arunplod, and R. Shibasaki. Oil palm tree detection and health classification on high-resolution imagery using deep learning. *Agriculture*, 11(2):183, 2021. doi:[10.3390/agriculture11020183](https://doi.org/10.3390/agriculture11020183).
- [128] Y. Zhan, D. Hu, Y. Wang, and X. Yu. Semisupervised hyperspectral image classification based on generative adversarial networks. *IEEE Geoscience and Remote Sensing Letters*, 15(2):212–216, 2018. doi:[10.1109/LGRS.2017.2780890](https://doi.org/10.1109/LGRS.2017.2780890).
- [129] F. Zhang, Y. Shao, Y. Sun, K. Zhu, C. Gao, et al. Unsupervised low-light image enhancement via histogram equalization prior. *arXiv*, 2021. ArXiv:2112.01766. doi:[10.48550/arXiv.2112.01766](https://doi.org/10.48550/arXiv.2112.01766).
- [130] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017. doi:[10.1109/TIP.2017.2662206](https://doi.org/10.1109/TIP.2017.2662206).
- [131] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In: *Proc. European Conf. Computer Vision (ECCV 2016)*, vol. 9907 of *Lecture Notes in Computer Science*, pp. 649–666. Springer International Publishing, Amsterdam, The Netherlands, 11-14 Oct 2016. doi:[10.1007/978-3-319-46487-9_40](https://doi.org/10.1007/978-3-319-46487-9_40).
- [132] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In: *Proc. 2017 IEEE Int. Conf. Computer Vision (ICCV)*, pp. 2242–2251. Venice, Italy, 22-29 Oct 2017. doi:[10.1109/ICCV.2017.244](https://doi.org/10.1109/ICCV.2017.244).
- [133] X. Zhu, D. Shen, R. Wang, Y. Zheng, S. Su, et al. Maturity grading and identification of *Camellia oleifera* fruit based on unsupervised image clustering. *Foods*, 11(23):3800, 2022. doi:[10.3390/FOODS11233800](https://doi.org/10.3390/FOODS11233800).
- [134] S. Zolfagharnassab, A. R. B. M. Shariff, R. Ehsani, H. Z. Jaafar, and I. B. Aris. Classification of oil palm fresh fruit bunches based on their maturity using thermal imaging technique. *Agriculture*, 12(11):1779, 2022. doi:[10.3390/agriculture12111779](https://doi.org/10.3390/agriculture12111779).

Afsar Kamal originally from India, is currently pursuing a Ph.D. at the National Defence University of Malaysia, specializing in Artificial Intelligence and Quantum Deep Learning. His research, titled *Maturity Grading of Oil Palm Fruit using Quantum Deep Learning for Precision Farming and Food Security*, focuses on leveraging advanced technologies for agricultural sustainability.

He holds a Master of Technology (M.Tech) and Master of Computer Applications (MCA) from Maulana Azad National Urdu University, Hyderabad, India, and a Bachelor of Computer Applications (BCA) from Baba Saheb Bhimrao Ambedkar University, Bihar.

Afsar has a keen interest in innovative applications of AI, machine learning, and quantum computing. He has contributed significantly to academic research and is dedicated to enhancing knowledge in his field through publications and teaching.

Nur Diyana Kamarudin is a Senior Lecturer and Research Fellow in Faculty of Defence Science & Technology (FSTP) and Cyber Security Centre, National Defence University of Malaysia (UPNM). She obtained the Ph.D. degree in Image Processing and Artificial Intelligence in 2017 and has been appointed as the Deputy Dean (Research and Postgraduate) in FSTP, UPNM from 2022 until now. Her research interests include (but not limited to) image processing, Artificial Intelligence, IoT and big data analytics. Nur Diyana serves as a Web of Science and Scopus journal reviewer for IEEE Access and Scientific Reports and subject matter expert for National AI group in Malaysia.

Khairol Amali bin Ahmad obtained a B.Sc. in Electrical Engineering in 1992 from the United States Military Academy, West Point, and an M.Sc. in Military Electronic Systems Engineering in 1999 from Cranfield University, UK. He also holds a Master of Military Arts & Science from the US Army Command and General Staff College, Fort Leavenworth, USA, and an M.Sc. in Aeronautical and Space Systems from ISAE-SUPAERO, France. He then joined the National Defence University of Malaysia (NDUM) and completed his Ph.D. from ISAE-SUPAERO, France, in 2015. While serving in the Malaysian Army, he was positioned in various posts and units including those dealing with engineering maintenance and procurement consultation of equipment, especially electronics related systems such as radar, electro-optics, early warning, command & control, and so forth.

He is now retired from the Army and currently, serving as the Deputy Vice-Chancellor (Student Affairs & Alumni). He was the Assistant Vice-Chancellor (Research & Innovation) and an associate to Centre of Defence Research and Technology (CODRAT) and Centre of Cybersecurity and Industrial Digital Revolution (PKS&RID) in NDUM. He is also a member of the Institution of Engineering and Technology (IET), Board of Engineer Malaysia (BEM) and Malaysia Board of Technologists (MBOT).

He has served as Organizing Chair of several international conferences such as the DSTC and IntCET series. He co-edited 2 books entitled *Machine Learning and Big Data: Concepts, Algorithms, Tools and Applications*, and *Functional Encryption*. Among his current research interests include signal processing related to signal propagation in Tropical area, mesh networks for IoT applications, Delay Disruption Tolerant Network (DTN) and machine learning applications. He is involved in several research and development projects such as Wave Energy Converter, Advance Aircraft Satellite Tracking System, Cloud Seeding Rocket, Smart Approach in Explosive Material Characterization, etc.

Syarifah Bahiyah Rahayu Ph.D. is a fellow researcher at Cyber Security and Digital Industry Revolution Centre, National Defence University Malaysia. She has vast experience in IT industry and academia. She has obtained her BSc in Computer Information System (1999) Northern Arizona University (NAU), Arizona, USA; MSc in Information Technology (2001) Queensland University Technology (QUT), Brisbane, Australia and PhD in Information Science (2014) Universiti Kebangsaan Malaysia (UKM), Bangi, Malaysia. Earlier working experience as a COBOL tutor in NAU (1997-1999), then joined Universiti Teknologi PETRONAS (UTP) as Academic Trainee (1999-2000) and later been promoted as a Lecturer (2001-2008). From 2008-2018, she has been working as IT Consultant, before joining NDUM as Fellow Researcher in Cyber Security and Digital Industry Revolution Centre then later appointed as Senior Lecturer in Faculty of Defence Science and Technology (FSTP). She is currently Head, Department of Defence Science, FSTP. She is actively giving talks related to cybersecurity and blockchain. Her research interests include (but not limited to) blockchain, artificial intelligence and big data.

Mohd Rizal Mohd Isa received the B.Sc. in Data Communication and Networking and M.Sc. in Information Technology degrees respectively, from the Universiti Teknologi MARA (UiTM), Malaysia. He received his Ph.D. degree at the University of Portsmouth, United Kingdom in Computer Engineering. His research interests include cyber security, biometric systems and information hiding.

Siti Noormiza Makhtar is a Lecturer in the Dept of Electrical & Electronic Engineering, Universiti Pertahanan Nasional Malaysia. Siti graduated in 2006 with B.Eng. in Biomedical Engineering from University of Malaya, Malaysia. She was then awarded with M.Eng. in Electrical & Electronic Engineering (2009) from The University of Adelaide, Australia, and Ph.D. in Electronic Engineering from The University of York, UK in 2018. Her research interest includes statistical signals processing, machine learning, instrumentations, control systems and computational neuro-engineering.

Zulkifli Yaakub is a senior oil palm breeder with expertise in both conventional and molecular breeding techniques. Since beginning his research in 2005, he has focused on evaluating field experiments and applying molecular tools to develop new oil palm seed varieties and elite palms for cloning. Dr. Zulkifli earned his Ph.D. in Genetics from Universiti Kebangsaan Malaysia and currently serves as the Head of the Breeding & Tissue Culture Unit at the Malaysian Palm Oil Board (MPOB). With over a decade of experience, his research encompasses the evaluation and diversity analysis of oil palm germplasm, clonal seed production, genetic mapping, and quantitative trait locus (QTL) studies.

A COMPARATIVE STUDY OF DEEPLABCUT AND OTHER OPEN-SOURCE PUPILLOMETRY DATA ANALYSIS ALGORITHMS – WHICH TO CHOOSE?

Amitesh Badkul^{1,3} , Sonakshi Mishra¹ , and Srinivasa P. Kommajosyula^{2,*} 

¹*Department of Electrical & Electronics Engineering,
Birla Institute of Technology and Science,
Hyderabad Campus, Jawahar Nagar, Hyderabad, India*

²*Department of Pharmacy, Birla Institute of Technology and Science,
Hyderabad Campus, Jawahar Nagar, Hyderabad, India*

³*Ph.D. Program in Computer Science, The Graduate Center,
The City University of New York, New York, NY, United States of America*

*Corresponding author: Srinivasa P. Kommajosyula (ksprasad@hyderabad.bits-pilani.ac.in)

Abstract Pupillometry measures pupil size, and several open-source algorithms are available to analyse pupillometry data. However, only a few studies compared these algorithms' accuracy and computational resources. This study aims to compare the accuracy of computer vision-based algorithms (Swirski, Starburst, PuRe, ElSe, ExCuSe algorithms) and the machine learning algorithm, DeepLabCut, to the double-blinded human examiners (gold-standard). Training of DeepLabCut with different architectures and a variable number of markers (2-9 markers) was done on an open-source dataset. The duration of training was statistically longer for the ResNet152 model compared to the MobileNet model. The pupil diameters in computer vision-based software such as PuRe, Starburst, and Swirski were statistically different from human measurements. MobileNet 2 and 3 marker models were the closest to the human measurements. In conclusion, this work highlights the efficiency of lower marker models based on MobileNet architecture in DeepLabCut, which consumes fewer computational resources and is more accurate.

Keywords: machine learning, deep learning, pupillometry, DeepLabCut, MobileNet, computer vision.

1. Introduction

Pupillometry measures pupil size changes in response to external stimuli or internal states [8, 33]. Pupil size changes with bright light, cognitive load, attention, memory, internal state, emotional and neuromodulatory changes [15, 16, 19]. Pupillometry is used both clinically and in basic science research to evaluate neurological function and in the diagnosis of attentional disorders [10, 30]. Neuroscience research in both animal models and humans has identified that an increase in activity at *locus coeruleus* and release of norepinephrine are causal in pupil diameter changes. Some researchers even use pupil diameter as a surrogate for *locus coeruleus* activity [6, 20].

Most methods of pupillometry use conventional image processing-based techniques like segmentation, edge detection, and ellipse fitting with thresholding followed by contour fitting. Recent studies, however, have employed sophisticated machine learning-based algorithms such as convolutional neural networks or generative adversarial networks, as these machine learning algorithms give superior accuracy values. [2, 4, 14, 21, 28].

Some studies have analysed the efficacy of these analytical techniques to determine the computational demand as well [21]. Many open-source pupillometry software are available that use various mechanisms to analyse pupil diameter, such as PupilEXT, Starburst, ExCuSe, ElSe, PuRe, and PuReST [7, 35]. A recent study validated this software and found that the ExCuSe, ElSe, PuReST, and PuRe algorithms attained adequate accuracy for pupil diameter measurement [35]. The Starburst algorithm detected many false peaks and produced highly variable results. The Swirski algorithm failed to detect the pupil in the 630nm spectrum.

Among machine learning applications, both supervised and unsupervised learning approaches are present [13]. A recent experiment using a deep learning algorithm called DeepLabCut garnered interest due to its applicability in a variety of experimental variables, broad user base with active software development, and ease of use due to its graphical user interface based on Python [3]. Here, the experimenter manually places the markers on the pupil diameter. It employs a transfer learning approach and requires less data (30 frames) compared to other approaches requiring thousands of frames for training, making it an attractive option for pupillometry data analyses [24]. DeepLabCut was initially developed as a pose estimation software in biology/behavioral/neuroscience research. It has also been used to detect animal behaviour and movement [18]. It has been applied to pupillometry research only recently [24]. Privitera et al. developed a low-cost (approximately 300 euros) Raspberry Pi setup and pupillometry software to analyse pupillometry data. They trained a machine learning model (Deep LabCut with 11 markers) using ResNet to quantify pupil diameter utilizing the DeepLabCut library. However, the reliability of DeepLabCut compared to other open-source software, the impact of the number of markers on the accuracy of the measurements, and the usage of computational resources are not known. Hence, this study is designed to address the following issues.

- A.** To assess the computational efficiency of DeepLabCut architectures and models.
- B.** To evaluate the accuracy of the DeepLabCut models (ResNet, and MobileNet architecture-based models with various markers ranging from 2-9 markers) in comparison to other open-source pupillometry software and human examiners measuring the pupil (considered as the gold-standard).
- C.** To benchmark current open-source algorithms against the gold-standard.

2. Methodology

2.1. Experimental design

We have used data from an open-source dataset published in a previous publication by Privitera et al. [24]. In brief, mice on C57 background ($n = 17$) of age 2-4 months were head restrained, and changes in pupil diameter were recorded using a Raspberry Pi

NoIR V2 camera under dark conditions with IR and UV lights. In the current manuscript, the DeepLabCut models were trained using 30 data frames and tested them on 20 different frames snipped randomly from different videos made by Privitera et al. Random test frames were selected to avoid temporal bias and capture different phases of pupil dilations. However, the same test frames were used to test all the models evaluated in this study. In a previous study employing DeepLabCut, only 30 frames or 150 frames of data were used [22, 24]. During the training phase, the computational resources being consumed were measured using the weights and biases tool (WandB [32]). After training DeepLabCut models based on various deep convolutional neural architectures like ResNet 50, ResNet 152, and MobileNetV2. Later, compared the pupil measurements of these DeepLabCut models to open-source algorithms as well as human examiner measurements of pupil diameter for accuracy check. The distance between two points on the pupil was measured to calculate the diameter, followed by inference of radii, and an average was taken in cases where multiple markers were used. Human examiners measured the pupil diameter in the same frames used for testing DeepLabCut and other open-source software using ImageJ version 1.53 (a National Institute of Health algorithm). The distance was measured in pixels and converted to millimeters using the ground-truth values derived by Privitera et al.

For measurement purposes, the data were measured in pixels and converted to mm. The distance between the two tracked calibration points in pixels was calculated using the formula:

$$d^{\text{px}} = \sqrt{(x_{P_{c1}}^{\text{px}} - x_{P_{c2}}^{\text{px}})^2 + (y_{P_{c1}}^{\text{px}} - y_{P_{c2}}^{\text{px}})^2}, \quad (1)$$

where $x_{P_{ci}}^{\text{px}}, y_{P_{ci}}^{\text{px}}, i = 1, 2$, are x and y coordinates of the calibration points P_{c1}, P_{c2} , respectively, in pixels. Privitera et al. advise using median values for these calculations. The absolute dimension of the calibration object in mm (d^{mm}) has to be divided by d^{px} to obtain the pixel-to-mm conversion ratio:

$$\text{ratio}^{\text{mm/px}} = \frac{d^{\text{mm}}}{d^{\text{px}}}, \quad (2)$$

and x and y coordinates of all tracked points P_j at each frame are multiplied by the conversion ratio, resulting in a metric description of the tracked points:

$$\begin{aligned} x_{P_j}^{\text{mm}} &= x_{P_j}^{\text{px}} \times \text{ratio}^{\text{mm/px}}, \\ y_{P_j}^{\text{mm}} &= y_{P_j}^{\text{px}} \times \text{ratio}^{\text{mm/px}}. \end{aligned} \quad (3)$$

All the data measured in pixels by human examiners, as well as the algorithms, were individually converted to millimeters using the above formulae.

2.2. DeepLabCut and deep convolutional neural networks (DCNNs)

DeepLabCut, a Python-based framework, was used to create the DCNN models. A variable number of markers from 2 to 9 were used to generate the training dataset for the DCNN models. The principle of this method is based on the concept of transfer learning, that is, training pre-trained models for a different task. The following models were used here: ResNet50, ResNet152, and MobileNet V2 [9, 12, 26]. After training, the DCNN models render the markers and output the location of each marker on each frame, from which the pupil radius is calculated. The hyperparameters used are mentioned below.

Markers Markers were used on the pupil to train the DCNN models, ranging from 2 markers to 9 markers. An additional two markers on the ends of the eyes were used to establish the ground truth. The reason for the usage of different types of labeling is to improve the accuracy while calculating the pupil radii.

Iterations The number of training iterations is 30,000.

Learning rate scheduler The learning rate used in training approach follows a multi-step schedule. The rate is adjusted at specific iterations, a strategy that enhances the training process's efficacy and performance. The learning rates were 0.005 and 0.020, while the respective iterations were 10000 and 30000.

Batch size The number of samples the model processes before each update, known as the batch size, is set to 1. Learning rate decay gradually diminishes the learning rate, preventing the model from overshooting the loss function's minima. In this case, for every 30,000 iterations, the decay is utilized.

Loss Lastly, the loss function used in this model is the Huber loss function, which integrates the properties of mean squared error loss and mean absolute error loss.

2.3. Other open-source software

Most prominent open-source softwares, such as Swirski, Starburst, PuRe, ElSe, and ExCuSe were used to compare their accuracy to that of the DeepLabCut models. These algorithms use template matching, edge detection, thresholding, or best-fit approaches.

Swirski This algorithm detects the light reflection and uses template matching followed by thresholding and edge detection techniques to estimate the size of the pupil. Readers are directed to the manuscript by Zandi et al. for a more detailed review [35]. The parameters used in this study for Swirski are mentioned here: Minimum radius: 20; Maximum radius: 140; Canny blur: 1.6; Canny threshold 1: 15; Canny threshold 2: 45; Perc inliners: 20; Inliner iterations: 2; Image Aware RANSAC: Yes.

Starburst This algorithm first detects the edges using Canny edge detection, followed by interpolation of the center from the detected edges. Readers are directed to the manuscript by Zandi et al. for a more detailed review [35]. The parameters used in this study for Starburst are mentioned here: Edge threshold: 21; Number of rays:

32; Minimum feature candidates: 7; CR Ratio (to image height): 10; CR window (px): 433.

PuRe This algorithm fits a model to detect pupil diameter. Readers are directed to the manuscript by Zandi et al. for a more detailed review [35]. The parameters used in this study for PuRe are mentioned here: Image width (downscaling): 320; Image height (downscaling): 240; Mean Canthi distance: 27.6; Maximum pupil size: 8; Minimum pupil size: 2; Minimum radius: 50.

ElSe This algorithm uses edge detection and thresholding techniques followed by ellipse fitting to identify the pupil diameter. Readers are directed to the manuscript by Zandi et al. for a more detailed review [35]. The parameters used in this study for ElSe are mentioned here: Minimum area (%): 0.005; Maximum area(%): 0.2.

ExCuSe The algorithm uses edge detection followed by mathematical estimations to calculate the pupil diameter. Readers are directed to the manuscript by Zandi et al. for a more detailed review [35]. The parameters used in this study for ExCuSe are mentioned here: Ellipse Goodness threshold: 15; Maximum radius: 50.

Non-deep learning algorithms are based on mathematical approaches and don't need any training. Testing time has been reported to be under 100 milliseconds per frame previously, and is similar to our anecdotal observations. Since comparing the accuracy of the test was the prime objective, testing times were not compared and this is a caveat that could be addressed in the future [27, 35]. In the current study, a standard PC was used (RAM: 8 GB DDR4-3200 MHz; CPU type AMD Ryzen™ 5 5600H. Graphics card: NVIDIA GeForce RTX 3060, Laptop GPU with 6 GB GDDR6 VRAM, storage: SSD/HD of 512 GB M.2 NVMe SSD).

ImageJ Examiners used ImageJ to measure the pupil diameter. Using line tool, and functions: analyse and measure, the pupil diameters were measured.

WandB WandB has been a powerful tool in logging the hyper-parameters involved in training a model over a dataset and storing the visualizations of a training run, analysing the different metrics that come into play while developing a model through training mainly when a graphics processing unit (GPU) is involved. This gives the developer an overall and comprehensive view of the process in play during training and also saves the training behaviour of GPU: power usage, memory allocated, time spent accessing memory, temperature and utilization for future references and comparisons. Training of DeepLabCut models was performed in Google Collab and recorded the GPU parameters using WandB as in the literature [1].

Statistical analyses A comparison of the accuracy of the detection of pupil diameter by various algorithms and human experimenters was done. A repeated measures ANOVA test followed by post-hoc Bonferroni correction for comparisons between the groups was used. T-tests and ANOVA were used to compare the computational resource usage. Data are represented with mean and standard error of the mean (SEM) values across this manuscript.

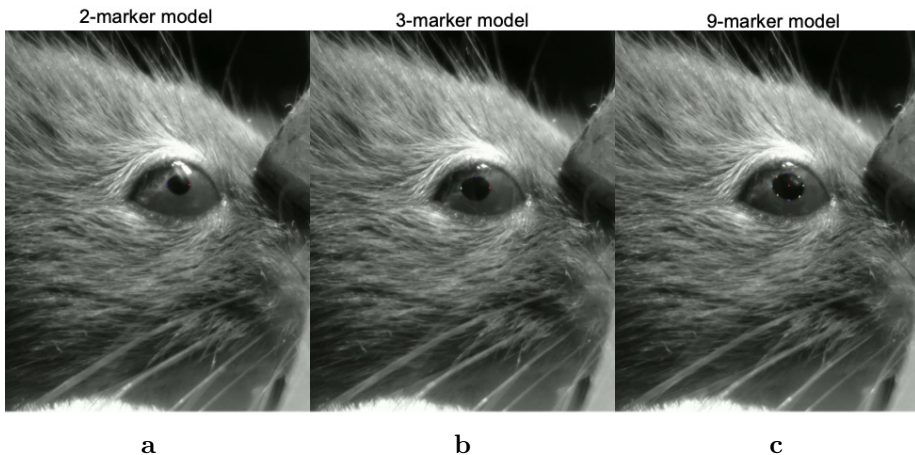


Fig. 1. Mouse pupils with overlaid markers for different architectures in an anesthetized and head-fixed mouse. (a) Pupil with two markers at both ends of the eye. (b) Pupil with three markers. (c) Pupil with nine markers placed around the eye.

3. Results

The training-related use of computational resources by these DCNN models in the DeepLabCut module was assessed. Firstly, one of the experimenters marked specific regions in the image as the borders of the pupils using a variable number of markers (Fig. 1).

The GPU use times, as a measure of computational resource usage, for ResNet50, ResNet152, and MobileNetV2 models were measured during training in DeepLabCut using WandB. In most cases, the GPU usage was within the range 80-95%; hence, this parameter was not statistically validated. There is a significant difference in GPU usage durations during training between all the network architectures (F -statistic = 846; $p < 0.001$). The post-hoc Bonferroni test showcased significant differences between all three network architectures in the usage of GPU resources during training. The GPU power usage duration was longer in ResNet152 models (range = $\langle 147.5, 153.5 \rangle$ min, mean = 151.3 min) vs. ResNet50 models (range = $\langle 61.5, 78 \rangle$ min, mean = 74.4 min). In comparison, MobileNet models were using the GPU resources for the least amount of time (range = $\langle 47, 70.5 \rangle$ min, mean = 59.7 min) (Fig. 2).

These results show that MobileNet models, regardless of the number of markers, consume GPU resources for lesser duration compared to other neural network architectures. An individual analysis of these different network architectures across different marker models showed similar GPU consumption in ResNet152 (147.5, 151, 150, 151.5, 152.5, 151, 153.5 and 153.5 min) and ResNet50 (61.5, 74.5, 76, 77, 75.5, 77, 78 and 75.5 min)

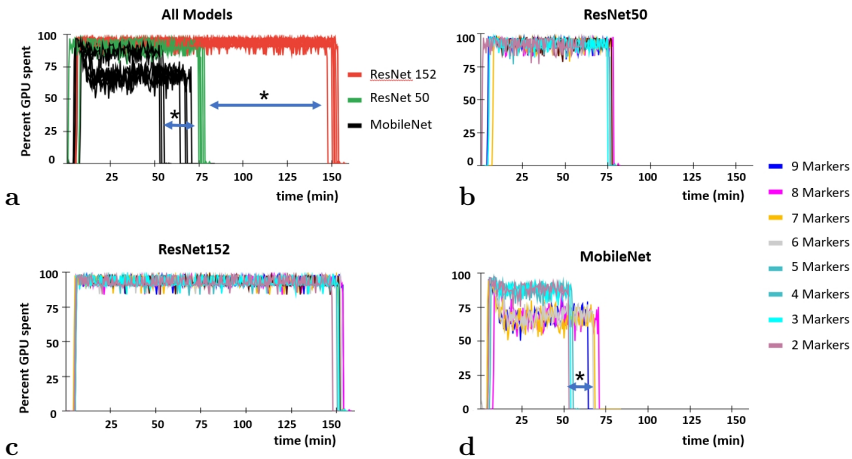


Fig. 2. Comparison of ResNet152, ResNet50 and MobileNet on the duration of use of graphics processing unit (GPU) during training in the DeepLabCut. (a) Three models together. MobileNet models utilize the least time, whereas the ResNet152 models the most and ResNet50 models are in between. (b) ResNet50 models showcase a similarity in GPU usage time across all models (2–9 markers). (c) ResNet152 models showcase a similarity in GPU usage time across all models (2–9 markers). (d) MobileNet models showcase a significant difference in GPU usage time for lower marker models (2–5 markers) vs. higher marker models (6–9 markers). Blue arrows with stars indicate the differences.

for different marker models (2-9 markers) (Fig. 2b and c). However, a decrease in GPU usage duration was noted for lower marker models (2, 3, 4 and 5: 52.5, 53.5, 47 and 55 min) vs. higher marker models (6, 7, 8, and 9: 67, 68, 70.5 and 64 min) in MobileNet architecture (Fig. 2d). The differences in GPU usage times were significantly different in MobileNet lower and higher models (Mean \pm SEM values, lower vs. higher: 52 \pm 1.74 min vs. 67.38 \pm 1.34 min, t -statistic = 5.12, $d_f = 3$, $p < 0.05$; d_f – number of degrees of freedom). This suggests that lower marker models of MobileNet architecture consume the least computational resources of all the DeepLabCut models.

After training the models, the accuracy of these models was tested on a set of 20 images from different mice (Fig. 3). As the gold standard for comparisons the human examiners' measurements of pupils were considered. The two examiners were double-blinded and weren't involved in any part of the experiments. Examiners used ImageJ software, and the values determined by the examiners were averaged to arrive at a single value and to enable statistical comparisons. The inter-examiner variability in measurements was less than 0.7% across all the frames. The mean value of pupil diameter found by the human examiners was 1.01 mm.

In addition to DeepLabCut models, the PuRe, Starburst, Swirski, ElSe, and ExCuSe

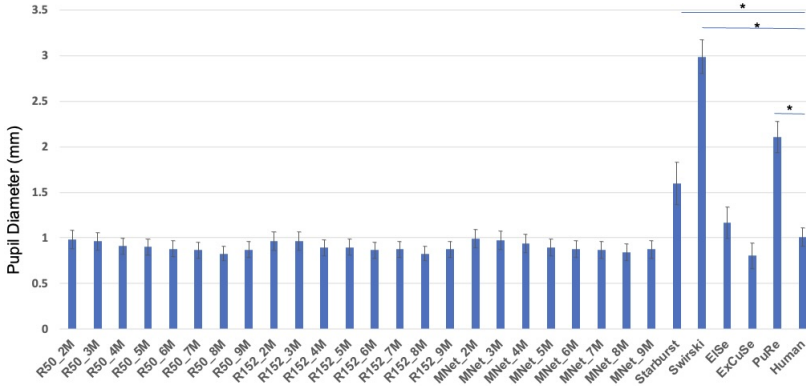


Fig. 3. Comparison of all DeepLabCut architecture models and computer vision-based models to the human examiner measures of pupil diameter. Three DeepLabCut-based architectures, including ResNet50, ResNet152, and MobileNet, were compared to computer vision-based models and human measurements. This includes: ResNet50 for 2–9 markers (R50.2M to R50.9M), ResNet152 for 2–9 markers (R152.2M to R152.9M), MobileNet for 2–9 markers (MNet.2M to MNet.9M), and human examiners (Human). Blue arrows with stars indicate significant differences between groups MobileNet vs. ResNet50; ResNet 50 vs. ResNet152 and lower marker models (2-5) vs. higher marker models (6-9) of MobileNet.

were also tested to compare these models' performance. The results of these models were significantly different from the human measurements of the pupil diameter ($p < 0.001$, Bonferroni post-hoc test, repeated measures ANOVA).

The repeated measures ANOVA showcased a significant difference between all the models (F -value = 34.857, $p < 0.001$). Post-hoc tests using Bonferroni corrections were performed for individual comparisons. All the DCNN-trained models were non-significantly different from the pupil diameter values measured by examiners. However, there was a significant difference in the pupil diameter measured using open-source algorithms such as PuRE, Starburst, and Swirski in comparison to examiners (Mean \pm SEM values: 2.104 \pm 0.17, 1.599 \pm 0.232, 2.987 \pm 0.185 vs. 1.01 \pm 0.1, $p < 0.001$, repeated measures ANOVA followed by post-hoc Bonferroni test). ExCuSe and ElSe were not significantly different in comparison to human examiners (Mean \pm SEM values: 0.805 \pm 0.141, 1.165 \pm 0.169 vs. 1.01 \pm 0.1). However, the mean values of ExCuSe and ElSe compared to the mean of examiners vary by 20.3% and 15.3%. All the machine learning models were non-significantly different from the examiner's measures. Among all the machine learning models, two marker models of ResNet50 (0.983 \pm 0.099), ResNet152 (0.964 \pm 0.098), as well as the MobileNet model (0.99 \pm 0.102), showcased the least variance from the examiners' mean pupil diameter (1.01 \pm 0.1) and their means varied by 2.6%, 4.5%, and 1.98%, respectively, from the mean measurement of examiners. Three marker models of

all architectures followed the two marker models closely (Mean \pm SEM values: ResNet152 – 3 marker: 0.966 ± 0.099 ; ResNet50 – 3 marker: 0.961 ± 0.097 ; MobileNet – 3 marker: 0.973 ± 0.104). The 2 and 3 marker models were more accurate in detecting the pupil diameter than models with a higher number of markers.

The measurements of memory consumption made with WandB indicated that the DeepLabCut models, MobileNet models, specifically the lower marker models from 2 to 5 markers, consume less memory resources than other models.

4. Discussion

The data showcased that among the DeepLabCut models, MobileNet models, specifically the lower marker models from 2 to 5 markers, consume less memory resources. Mainly, the accuracy of these models were compared to open-source pupil measurement software and human observers. The accuracy of the MobileNet 2 marker model is shown to be closest to that of human observers. All open-source pupillometry software tested here has either overvalued or undervalued the pupil diameter compared to human experimenters. ElSe and ExCuSe were the closest in terms of performance as compared to human observers. DeepLabCut toolbox has been used mostly in animal pose estimation, and only recently has it been used to analyse pupil data [18,24]. Results show that the error in pupil diameter increases with the increase in the number of markers across all three network architectures in the DeepLabCut. This error could be due to the intrinsic nature of deep learning models, where a balance between the number of markers (a surrogate for the learnable parameters), the use of definite architectures (a surrogate for the complexity of the models), and the number of frames used in training determine the learning efficiency and prevent over/underfitting. Since the amount of training data was the same in experiments for any architecture and number of markers (2 – 9), this could have impacted the model’s ability to learn and led to over/underfitting. These may cause a decrease in the performance of models using a higher number of markers [11]. There is a difference in the amounts of parameters used by ResNet 50 (23.5 million) vs. ResNet 152 (58.3 million) vs. MobileNetV2 (3.4 million). The lower number of parameters in MobileNetV2 decreases the training duration/computational resources compared to other deep learning models. The depthwise convolutions with fewer parameters in MobileNetV2 increase efficiency and decrease computational costs [17,26]. These findings suggest that while larger models (e.g., ResNet-152) offer a theoretical advantage in complex tasks due to their higher capacity and can handle vanishing gradient issues. While, the simpler MobileNetV2 architecture performed equally well for this specific task with much lower computational demand. This makes MobileNetV2 the most efficient choice for real-time pupillometry or scenarios where hardware resources are limited without a significant sacrifice in accuracy [29].

It is difficult to compare the mathematical superiority of DeepLabCut with other

techniques because each technique uses different mathematical approaches and principles. However, here is a comparison of some key features of DeepLabCut that make it stand out from the other techniques.

Deep learning Learning complex patterns and features from input data is common in DeepLabCut and other deep learning techniques. These traits increase the adaptability of DeepLabCut to different lighting conditions, pupil sizes, and head positions. Thus DeepLabCut is more robust and accurate than traditional computer vision algorithms.

Flexibility DeepLabCut provides flexibility and automatically extracts features that are unlike traditional computer vision algorithms, which often have fixed and predefined features.

Small training data DeepLabCut requires only a small amount of data for training as opposed to other machine learning algorithms, and it can also perform with better accuracy and robustness. This is because DeepLabCut can learn from a diverse set of examples and generalize to new conditions.

Real-time processing DeepLabCut can process images in real-time on a variety of platforms, making it suitable for applications that require accurate pupil detection [31].

Model architecture DeepLabCut models can be designed and optimized for specific tasks and data types. This allows for better performance and generalization compared to traditional computer vision algorithms, which often use generic and fixed models.

Adaptability DeepLabCut models can be re-trained and fine-tuned for new datasets or applications, making them adaptable to changing requirements and conditions.

Open source software tools tested here are based on traditional computer vision techniques such as template matching, thresholding, edge detection, and curve fitting. The pros and cons of using simple computer vision and deep learning techniques are detailed in the review by O'Mahony et al. [23]. Briefly, there are several advantages to using deep learning models, such as adaptability to lighting, movement artifacts, using transfer learning approach to train using fewer data points where less data is available, using lightweight architectures such as MobileNet could enable real-time calculations and making algorithms scalable/transferable between applications. Some disadvantages to deep learning models include high computational costs, long training times, inefficient real-time processing in some architectures, and overfitting in some architectures that require careful tuning of hyperparameters. Recently, a web application using novel convolutional neural networks and AdaBelief optimizer was launched, where the user needs to upload data on the website only and will be given the results [25]. This application was based on newer algorithms such as U-Net and achieved accuracy rates above 70-80% [5, 34]. However, the algorithm used a semantic segmentation method that takes into consideration the grayscale values to segment and falls short if the grayscale values are not very different [25].

In conclusion, results show that DeepLabCut, based on MobileNet architecture with a lower number of markers, consumes fewer computational resources during training. Also, the same DeepLabCut architecture with a lower number of markers (2 markers) is more accurate and closer to the values measured by humans than other architectures. This study establishes that with the least amount of training, which spans only an hour, using only a few frames, DeepLabCut can outperform current open-source software and its results are close to the values achieved by a human examiner.

5. Authors' declarations

5.1. Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

5.2. Declaration of generative AI in scientific writing

The authors declare that they have not used any type of AI while writing this manuscript

5.3. Data availability

The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

5.4. Funding

is work was supported by the Birla Institute of Technology and Science Pilani, Hyderabad Campus intramural funding schemes RIG and ACRG (Grant numbers: 1139 and 1346).

5.5. Author contributions

AB has conducted the experiments, and drafted the manuscript; SM has conducted the experiments and analysed the results; SPK has conceptualized the study, analysed the results, drafted the manuscript and supervised the work. The funding body has no involvement in any of the above-mentioned details.

Acknowledgement

The authors would like to thank Professor Venkateswaran Rajagopalan, Department of Electrical and Electronics Engineering, BITS-Pilani Hyderabad Campus for his insightful comments on the manuscript.

References

- [1] A. Biró, A. I. Cuesta-Vargas, J. Martín-Martín, L. Szilágyi, and S. M. Szilágyi. Synthesized multilanguage OCR using CRNN and SVTR models for realtime collaborative tools. *Applied Sciences*, 13(7):4419, 2023. doi:10.3390/app13074419.
- [2] F. Boutros, N. Damer, K. Raja, R. Ramachandra, F. Kirchbuchner, et al. Iris and periocular biometrics for head mounted displays: Segmentation, recognition, and synthetic data generation. *Image and Vision Computing*, 104:104007, 2020. doi:10.1016/j.imavis.2020.104007.
- [3] Brain Initiative Alliance, Neuroscience. DeepLabCut: Neuroscience research initiative. <https://www.braininitiative.org/toolmakers/resources/deeplabcut/>, [Accessed: 3 Nov 2021].
- [4] Y. Chen, M. Adjouadi, C. Han, J. Wang, A. Barreto, et al. A highly accurate and computationally efficient approach for unconstrained iris segmentation. *Image and Vision Computing*, 28(2):261–269, 2010. doi:10.1016/j.imavis.2009.04.017.
- [5] W. Chinsatit and T. Saitoh. CNN-based pupil center detection for wearable gaze estimation system. *Applied Computational Intelligence and Soft Computing*, 2017(1):8718956, 2017. doi:10.1155/2017/8718956.
- [6] V. D. Costa and P. H. Rudebeck. More than meets the eye: the relationship between pupil size and locus coeruleus activity. *Neuron*, 89(1):8–10, 2016. doi:10.1016/j.neuron.2015.12.031.
- [7] W. Fuhl, T. Santini, G. Kasneci, and E. Kasneci. Pupilnet: Convolutional neural networks for robust pupil detection. *arXiv*, 2016. ArXiv:1601.04902. doi:10.48550/arXiv.1601.04902.
- [8] S. D. Goldinger and M. H. Papesh. Pupil dilation reflects the creation and retrieval of memories. *Current Directions in Psychological Science*, 21(2):90–95, 2012. doi:10.1177/0963721412436811.
- [9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*, pp. 770–778. Las Vegas, NV, USA, 27–30 Jun 2016. doi:10.1109/CVPR.2016.90.
- [10] H. Himmel and T. E. Faelker. Pupillary function test in rat: Establishment of imaging setup and pharmacological validation within modified Irwin test. *Journal of Pharmacological and Toxicological Methods*, 99:106588, 2019. doi:10.1016/j.vascn.2019.106588.
- [11] J. P. Horwath, D. N. Zakharov, R. Mégret, and E. A. Stach. Understanding important features of deep learning models for segmentation of high-resolution transmission electron microscopy images. *npj Computational Materials*, 6(1):108, 2020. doi:10.1038/s41524-020-00363-x.
- [12] A. G. Howard. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv*, 2017. ArXiv:1704.04861. doi:10.48550/arXiv.1704.04861.
- [13] R. A. Jeyarani and R. Senthilkumar. Eye tracking biomarkers for autism spectrum disorder detection using machine learning and deep learning techniques. *Research in Autism Spectrum Disorders*, 108:102228, 2023. doi:10.1016/j.rasd.2023.102228.
- [14] N. Kondo, W. Chinsatit, and T. Saitoh. Pupil center detection for infrared irradiation eye image using CNN. In: *Proc. 2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, pp. 100–105. IEEE, Kanazawa, Japan, 19–22 Sep 2017. doi:10.23919/SICE.2017.8105630.
- [15] K. Kuraoka and K. Nakamura. Facial temperature and pupil size as indicators of internal state in primates. *Neuroscience Research*, 175:25–37, 2022. doi:10.1016/j.neures.2022.01.002.
- [16] R. S. Larsen and J. Waters. Neuromodulatory correlates of pupil dilation. *Frontiers in Neural Circuits*, 12:21, 2018. doi:10.3389/fncir.2018.00021.

- [17] M. C. Leong, D. K. Prasad, Y. T. Lee, and F. Lin. Semi-CNN architecture for effective spatio-temporal learning in action recognition. *Applied Sciences*, 10(2):557, 2020. doi:[10.3390/app10020557](https://doi.org/10.3390/app10020557).
- [18] M. W. Mathis and A. Mathis. Deep learning tools for the measurement of animal behavior in neuroscience. *Current Opinion in Neurobiology*, 60:1–11, 2020. doi:[10.1016/j.conb.2019.10.008](https://doi.org/10.1016/j.conb.2019.10.008).
- [19] S. Mathôt. Pupillometry: Psychology, physiology, and function. *Journal of Cognition*, 1(1), 2018. doi:[10.5334/joc.18](https://doi.org/10.5334/joc.18).
- [20] P. R. Murphy, R. G. O’Connell, M. O’Sullivan, I. H. Robertson, and J. H. Balsters. Pupil diameter covaries with BOLD activity in human locus coeruleus. *Human brain mapping*, 35(8):4140–4154, 2014. doi:[10.1002/hbm.22466](https://doi.org/10.1002/hbm.22466).
- [21] A. Nasim, A. Maqsood, and T. Saeed. Multicore and GPU based pupillometry using parabolic and elliptic Hough transform. *International Journal of Mechanical Engineering and Robotics Research*, 6(5), 2017. doi:[10.18178/ijmerr.6.5.425-433](https://doi.org/10.18178/ijmerr.6.5.425-433).
- [22] T. Nath, A. Mathis, A. C. Chen, A. Patel, M. Bethge, et al. Using DeepLabCut for 3D markerless pose estimation across species and behaviors. *Nature Protocols*, 14(7):2152–2176, 2019. doi:[10.1038/s41596-019-0176-0](https://doi.org/10.1038/s41596-019-0176-0).
- [23] N. O’Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. V. Hernandez, et al. Deep learning vs. traditional computer vision. In: *Advances in Computer Vision: Proc. 2019 Computer Vision Conference (CVC)*, vol. 943 of *Advances in Intelligent Systems and Computing*, pp. 128–144. Springer, LasVegas, USA, 2-3 May 2020. doi:[10.1007/978-3-030-17795-9](https://doi.org/10.1007/978-3-030-17795-9).
- [24] M. Privitera, K. D. Ferrari, L. M. von Ziegler, O. Sturman, S. N. Duss, et al. Author correction: A complete pupillometry toolbox for real-time monitoring of locus coeruleus activity in rodents. *Nature Protocols*, 16(8):4108, 2021. doi:[10.1038/s41596-021-00493-6](https://doi.org/10.1038/s41596-021-00493-6).
- [25] M. M. Raffaele, F. Carrara, A. Viglione, L. Lupori, L. L. Verde, et al. MEYE: Web-app for translational and real-time pupillometry. *ENEURO*, 8(5), 2021. doi:[10.1523/ENEURO.0122-21.2021](https://doi.org/10.1523/ENEURO.0122-21.2021).
- [26] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In: *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, pp. 4510–4520. Salt Lake City, UT, USA, 18-23 Jun 2018. doi:[10.1109/CVPR.2018.00474](https://doi.org/10.1109/CVPR.2018.00474).
- [27] T. Santini, W. Fuhl, and E. Kasneci. Pure: Robust pupil detection for real-time pervasive eye tracking. *Computer Vision and Image Understanding*, 170:40–50, 2018. doi:[10.1016/j.cviu.2018.02.002](https://doi.org/10.1016/j.cviu.2018.02.002).
- [28] T. Satriya, S. Wibirama, and I. Ardiyanto. Robust pupil tracking algorithm based on ellipse fitting. In: *Proc. 2016 International Symposium on Electronics and Smart Devices (ISESD)*, pp. 253–257. IEEE, Bandung, Indonesia, 29-30 Nov 2016. doi:[10.1109/ISESD.2016.7886728](https://doi.org/10.1109/ISESD.2016.7886728).
- [29] D. Sinha and M. El-Sharkawy. Thin MobileNet: An enhanced MobileNet architecture. In: *Proc. 2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pp. 0280–0285. IEEE, New York, NY, USA, 10-12 Oct 2019. doi:[10.1109/UEMCON47517.2019.8993089](https://doi.org/10.1109/UEMCON47517.2019.8993089).
- [30] S. Sirois and J. Brisson. Pupillometry. *WIREs Cognitive Science*, 5(6):679–692, 2014. doi:[10.1002/wcs.1323](https://doi.org/10.1002/wcs.1323).
- [31] M. E. Suryanto, F. Saputra, K. A. Kurnia, R. D. Vasquez, M. J. M. Roldan, et al. Using DeepLabCut as a real-time and markerless tool for cardiac physiology assessment in zebrafish. *Biology*, 11(8):1243, 2022. doi:[10.3390/biology11081243](https://doi.org/10.3390/biology11081243).
- [32] Weights & Biases. The AI developer platform to build AI applications and models with confidence. <https://wandb.ai>.

- [33] P. Van der Wel and H. Van Steenbergen. Pupil dilation as an index of effort in cognitive control tasks: A review. *Psychonomic Bulletin & Review*, 25:2005–2015, 2018. doi:10.3758/s13423-018-1432-y.
- [34] Y.-H. Yiu, M. Aboulatta, T. Raiser, L. Oprey, V. L. Flanagan, et al. DeepVOG: Open-source pupil segmentation and gaze estimation in neuroscience using deep learning. *Journal of Neuroscience Methods*, 324:108307, 2019. doi:10.1016/j.jneumeth.2019.05.016.
- [35] B. Zandi and M. Lode. PupilEXT: Flexible open-source platform for high-resolution pupillometry in vision research. *Frontiers in Neuroscience*, 15:676220, 2021. doi:10.3389/fnins.2021.676220.



Amitesh Badkul is currently a Ph.D. student at the Department of Computer Science at the City University of New York. He graduated from BITS-Pilani in 2023 with a Bachelor's degree in EEE and a Master's in Chemistry. His research focuses on developing advanced deep learning models for biological applications, particularly in drug discovery, with an emphasis on uncertainty quantification and out-of-distribution generalization in real-world chemical and biological settings.



Sonakshi Mishra graduated from BITS-Pilani with a Bachelor's degree in ECE and a Master's degree in Mathematics in 2024. She is currently working at Math AI as a software development engineer, focusing on enhancing asset workflows and caching strategies. Her work has achieved a 40% reduction in load times and a 50% improvement in server response times, leveraging Next.js and Node.js for scalable performance.



Srinivasa Prasad Kommajosyula is currently an assistant professor at the Department of Pharmacy at BITS-Pilani Hyderabad Campus. His research interests include hearing loss, depression, anxiety, stroke, and biomedical devices.