Vol. 31, No. 1/4, 2022

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

Attention-based U-Net for Image Demoiréing

Tomasz M. Lehmann
Warsaw University of Technology, Warsaw, Poland
tomasz.lehmann.dokt@pw.edu.pl

Abstract. Image demoiréing is a particular example of a picture restoration problem. Moiré is an interference pattern generated by overlaying similar but slightly offset templates.

In this paper, we present a *deep learning* based algorithm to reduce moiré disruptions. The proposed solution contains an explanation of the *cross-sampling* procedure – the training dataset management method which was optimized according to limited computing resources.

Suggested neural network architecture is based on *Attention U-Net* structure. It is an exceptionally effective model which was not proposed before in image demoiréing systems. The greatest improvement of this model in comparison to *U-Net* network is the implementation of *attention gates*. These additional computing operations make the algorithm more focused on target structures.

We also examined three MSE and SSIM based loss functions. The SSIM index is used to predict the perceived quality of digital images and videos. A similar approach was applied in various computer vision areas.

The author's main contributions to the image demoiréing problem contain the use of the novel architecture for this task, innovative two-part loss function, and the untypical use of the *cross-sampling* training procedure.

Key words: image demoiréing, computer vision, attention U-Net, cross-sampling.

1. Introduction

Image demoiréing is a relatively new issue in the field of Computer Vision (CV). Essentially, it is a specific case of picture restoration. The problem of moiré fringes appears when an opaque ruled pattern with transparent gaps is overlaid on another similar pattern. For this kind of interference pattern to emerge, the two patterns must not be completely identical, but one should be slightly geometrically transformed with respect to the other, or should have a slightly dissimilar pitch. Moiré patterns appear e.g. in digital photography and television. In the first example, it occurs when a pattern on an object being photographed interferes with the shape of the light sensors to generate undesirable artifacts. The term originates from the French noun *moiré*, a type of textile, traditionally made of silk (now also cotton or synthetic material) with a rippled appearance. Examples of images from *ultra-high-definition demoiréing dataset* (UHDM) [27] with moiré fringes are depicted in Fig. 1.

Nowadays, taking photos of electronic displays is a common way of transferring image data and it is widely practiced in industries and everyday life. Because of signal interference between the pixel matrix of the display screen and the *Bayer filters* (color filter arrays (CFA) for arranging RGB color filters on a square grid of photosensors)

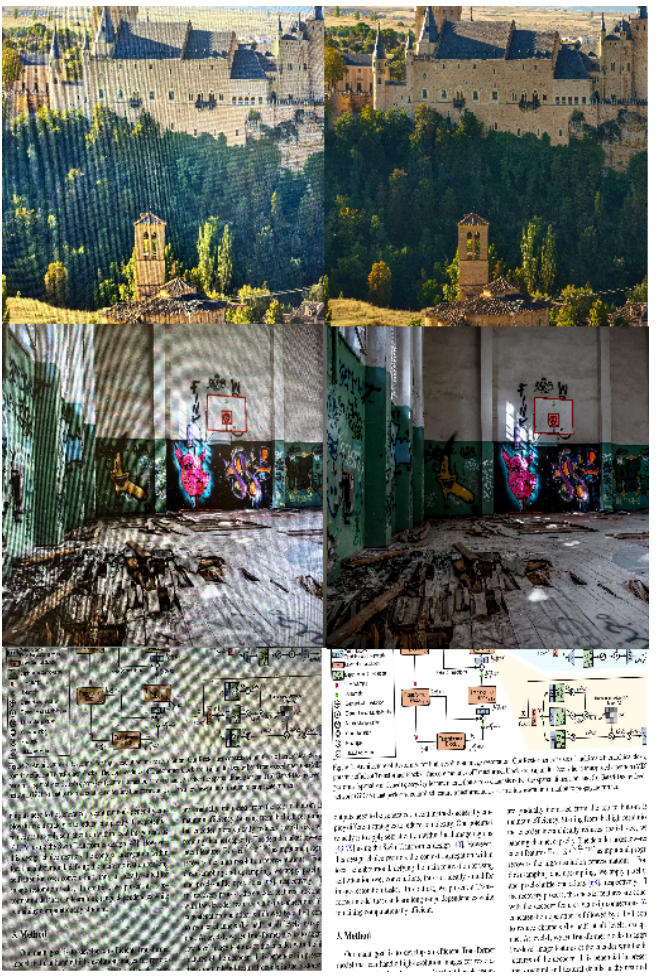


Fig. 1. Moiré examples from UHDM database [27]. Images with moiré patterns are presented on the left. On the right images without disruption are shown.

in a camera sensor objectionable moiré patterns significantly disrupt captured screen images. In this work, we tackle the problem of deep learning-based image demoiréing capable of improving the quality of such images.

Recently, deep neural network-based solutions achieve great results in many CV-related problems. Various image restoration and image deblurring issues were resolved with the help of these machine learning algorithms. The presented approach is based

on a type of *U-Net* convolutional neural network (CNN) [18]. The network consists of two main paths (a contracting path and an expansive path) which gives it the u-shaped architecture. Each contracting path is followed by blocks of activation functions and max pooling operations. As a result, the spatial information is reduced while the feature information is increased. The second path is intended to reverse this state. It is a particular example of the symmetrical *autoencoder* architecture.

The author's main contributions to the image demoiréing problem are the proposal of an unused before and highly efficient architecture, the presentation of a loss function implementation which is innovative for this area, and the untypical use of the *cross-sampling* training procedure.

1.1. Related Works

Leading architectures dedicated to image deblurring and image restoration are built on straight deep convolutional neural networks (DCNNs) [2, 13], generative adversarial networks (GANs) [11, 23], Transformer-based blocks [21, 25, 28] and U-Net-shaped hierarchical structures [22].

In the context of real-world (nonsynthetic) moiré removal, the number of studies is meaningfully lower. The first real-world committed dataset was proposed in [26] (the dataset is available from [20]) where authors used a multiresolution fully convolutional neural network. In other papers, a dynamic feature encoding module [8] and a novel multiscale bandpass convolutional neural networks were also suggested [29]. The algorithms strictly dedicated to high-resolution images analysis are also available – the multi-stage framework FHDe2Net [4] was applied to 1080p resolution image demoiréing on the FHDMi database and the ESDNet [27] architecture was Performed well trained and performed well on UHDM dataset. The first mentioned framework consists of two branches. The global to local cascades branch removes moiré patterns from the picture while the other part of the structure conserves high-resolution details. The second solution is based on a semantic-aligned scale-aware module to address the scale variation of moiré patterns. Most of the proposed deep learning algorithms have high computational complexity. We focused on developing a lightweight and effective model on lower resolution images ($512 \cdot 512$) to balance computational costs.

To solve the considered problem we propose a special variant of the U-Net architecture [18]. It is a convolutional network originally designed for biomedical image segmentation. The network is fast and it was successfully adopted in different areas of CV, like super-resolution [9, 12], depth estimation [1, 6] and image denoising [5]. In our research we used the *Attention U-Net* [14] – the U-Net-type architecture with *attention gates* applied. These extra operations automatically learn to focus on target structures without additional supervision. The concept comes from Natural Language Processing (NLP) for image captioning [3]. In this paper, we present our training procedure methods and our results achieved on the test datasets.

2. Experimental setup

2.1. Datasets

As there are few papers addressing the image demoiréing problem, there are also not too many datasets for this task. Most of them are generated by moiré promoted software. However, for some time real-world data is also available.

2.1.1. UHDM

The UHDM dataset [27] is a collection of ultra-high-definition images. It contains 5,000 pairs of images in resolution $4032 \cdot 3024$ and $4624 \cdot 3274$. The dataset was collected using various mobile phones which affects resolution and quality diversity. To produce realistic moiré images, authors shoot clean pictures displayed on the screen with a phone camera , and the phone was fixed on a smartphone gimbal, which allows them to conveniently and flexibly adjust the camera view through its control button. UHDM is randomly split into 4,500 pairs for the training procedure and 500 pairs for validation. In our research we used the same distribution of subsets.

2.1.2. TIP2018

Authors of the frequently cited publication [26] created a benchmark of 135 000 image pairs available from [20]. Collected images have a wide variety of moiré effects. Each pair contains an image contaminated with a moiré pattern and its corresponding uncontaminated reference image. Image references are enhanced with a black border which is explained by the observation that dark colors are least affected by the moiré effect. Displayed images were captured using a mobile phone. 90% of images are used as the training set and 10% are used for validation and testing. During the validation, we used these data to tune the parameters of a classifier when the test is executed to assess the final performance [17]. We kept these proportions.

2.2. Proposed methods

2.2.1. Attention U-Net

In our research, we proposed widely respected *Attention U-Net* architecture [14]. The algorithm uses self-attention gating modules that can be utilized in CNN-based standard image analysis models for dense label predictions.

Mentioned gates are located in the standard U-Net architecture to highlight salient features that are passed through the skip connections. Attention gates filter the neuron activation during backward and forward passes to down-weighted background regions and up-weighted spatial regions which are more relevant for a given problem.

This kind of neural network architecture has been never successfully used before in image demoiréing or image restoration tasks.

2.2.2. Loss function

In this paper, three loss functions are considered. First, we applied the *mean squared error* (MSE) function which measures the average squared difference between the estimated values and the actual value. Later, we experimented with combinations of MSE loss and *the structural similarity index measure* (SSIM) loss [24]. We defined the second one as the SSIM value according to (1). The SSIM measure is used to assess the perceived quality of digital images and videos. The SSIM value remains between -1 and 1. A value closer to 1 indicates better image quality. A similar approach was used in depth estimation models [1] where authors empirically found and set 0.1 as a reasonable weight for MSE part in the loss formula. We decided to start from the same value which might be understood as a significant approximation. Achieved results were compared also with the ten times bigger MSE-part multiplier. To determine the most accurate value the remarkably wider range should be considered. Because of technical limitations, we decided to investigate just these two cases. In [15] the author compares similarity and distance measures. The author showed and explained the theoretical similarities between SSIM and MSE functions which depend on the same parameters. Nevertheless, the experimental simulations showed a great difference between these two metrics. It can be observed that MSE is quite insensitive to all types of distortions while SSIM responds considerably to even minor blurriness and noise changes. By combining these two methods we can control more output image parameters.

The equations crucial for our computations are presented below.

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + C_1) + (2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}\,, \tag{1}$$

where:

μ_x – the pixel sample mean of x,
μ_y – the pixel sample mean of y,
σ_x^2 – the variance of x,
σ_y^2 – the variance of y,
σ_{xy} – the variance of x and y,
C_1, C_2 – two variables to stabilize the division with weak denominator.

$$\mathrm{MSE}(x,y) = \frac{1}{n}\sum_n (x_n - y_n)^2\,, \tag{2}$$

where:

n – number of pixels,
x_n – ground truth pixel,
y_n – predicted pixel.

$$L_{\mathrm{SSIM}}(x, y) = 1 - \mathrm{SSIM}(x, y)\,. \tag{3}$$

During experiments we applied and analysed three loss functions described by (4-6).

$$L_1(x, y) = \mathrm{MSE}(x, y)\ , \tag{4}$$

$$L_2(x, y) = 0.1 \cdot \mathrm{MSE}(x, y) + L_{\mathrm{SSIM}}(x, y)\ , \tag{5}$$

$$L_3(x, y) = \mathrm{MSE}(x, y) + L_{\mathrm{SSIM}}(x, y)\ . \tag{6}$$

2.2.3. Training procedures

To train our implemented solutions we used ADAM optimizer [10]. It is an algorithm for first-order gradient-based optimization of stochastic objective functions. The method is efficient and requires less memory than fundamental stochastic gradient descent. The training lasted 100 epochs and every single training epoch was followed by validation. During the validation SSIM and *peak signal-to-noise ratio* (PSNR) metrics were monitored. The equation for PSNR is shown below.

$$\mathrm{PSNR}(x, y) = 20 \cdot \log_{10}(I_{\max}) - 10 \cdot 10 \log_{10}(\mathrm{MSE}(x, y)) \tag{7}$$

where:

$I_{\max}$ – the maximum possible pixel value of the image. When the pixels are represented using 8 bits per sample, this is 255.

To reduce computational costs we decided to use a *cross-sampling* technique. It is a method used for balancing uneven or extensive datasets. During the training, we used all elements from the UHDM training set and only 15 500 elements from TIP2018 training set. After every epoch, the elements from the second-mentioned set were randomly drawn. For comparison, we also trained one particular model on the whole training dataset.

As a result of time-consuming prevention, the resolution of the whole collected database (UHDM and TIP2018) was reduced to a dimension of $512 \cdot 512$ pixels. Tests and validations were made on the UHDM and the TIP2018 datasets separately. During the examination of our methods, the testing resolution also decreased to $512 \cdot 512$. The final comparison with other algorithms was made on the images with original resolution.

We implemented our proposed network using PyTorch [16] and it was trained on the NVIDIA TITAN V100 GPU with 32 GB memory.

3. Experimental results

3.1. Loss functions comparison

During epochs validating we monitored two main metrics: SSIM and PSNR. These two measuring tools are widely used in image quality assessment. To minimize computational costs we decided to use the *cross-sampling* method in experiments with loss functions.

In Figs. 2a and b we can see how the validation metrics change in the training time for the UHDM validation dataset.

We can observe that by optimizing SSIM value – PSNR value increases slower than without using SSIM-related loss function. It is explained by strong PSNR-MSE relation which is presented in [15]. The two-part loss function minimizes MSE less efficiently which is a necessary compromise if we want to maximize the SSIM measure. It is a newer measurement tool that is designed based on three main factors, i.e., luminance, contrast, and structure to better suit the operation of the human visual system [19].

Validation measures in function of epochs for the TIP2018 validation set are depicted in Figs. 3a and b.

A similar tendency of the SSIM-PSNR dependence can be observed but the proportions are slightly different. It is observed that with decreasing MSE part in the MSE – SSIM combinational function – PSNR increase.

All results for test datasets are located in the Table 1. According these results, the most proficient loss function is $0.1 \cdot \mathrm{MSE} + \mathrm{SSIM}$ which achieves the best results with SSIM measure for both datasets. On the other hand, the PSNR was not so undervalued as using MSE+SSIM minimizing. Choosing the most accurate loss function should depend on task specification but we present the pros and cons of these calculations.

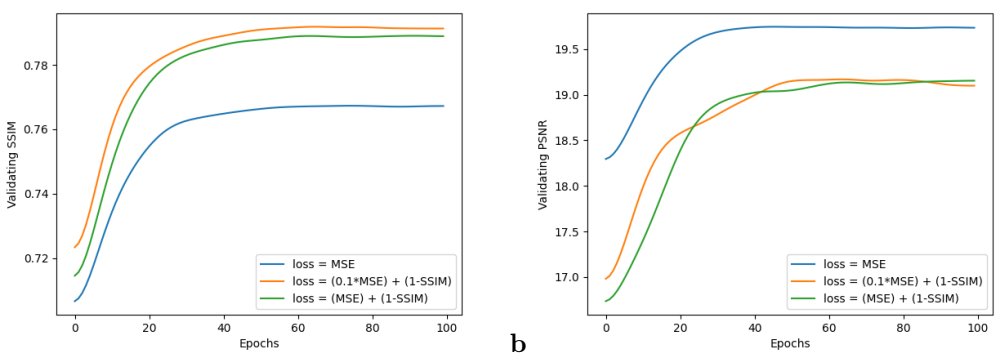
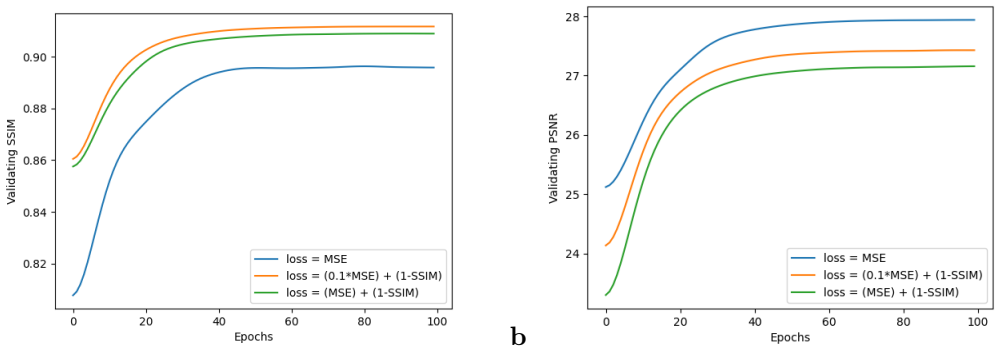


Fig. 2. Validating with two measures in function of epochs for the UHDM validation dataset: (**a**) with SSIM; (**b**) with PSNR. Graph color represents the loss function applied.



a **b**

Fig. 3. Validating with two measures in function of epochs for the TIP2018 validation dataset: (**a**) with SSIM; (**b**) with PSNR. Graph color represents the loss function applied.

The presented SSIM-related loss function includes point-wise differences but also optimizes the process of distortion removal by looking at regions around each point. For the optimization of the algorithm to be better suited to the operation of the human visual system, we recommend the proposed cost function.

3.2. Influence of the cross-sampling implementation

We examined the influence of the *cross-sampling* method which reduced the training time almost eight times. Because of a lack of resources was decided to train the model on full TIP2018 and UHDM datasets just once. We made decision to use $0.1 \cdot \mathrm{MSE} + \mathrm{SSIM}$ loss function to perform it. Later we compared the achieved measures with the cross-sampling method.

Tab. 1. Final metrics for different datasets and loss functions.

DATASET	METRIC	LOSS FUNCTION	VALUE
UHDM	SSIM	MSE	0.77
		$0.1 \cdot \mathrm{MSE} + L_{\mathrm{SSIM}}$	0.80
		$\mathrm{MSE} + L_{\mathrm{SSIM}}$	0.79
	PSNR	MSE	19.80
		$0.1 \cdot \mathrm{MSE} + L_{\mathrm{SSIM}}$	19.31
		$\mathrm{MSE} + L_{\mathrm{SSIM}}$	19.23
TIP2018	SSIM	MSE	0.90
		$0.1 \cdot \mathrm{MSE} + L_{\mathrm{SSIM}}$	0.91
		$\mathrm{MSE} + L_{\mathrm{SSIM}}$	0.91
	PSNR	MSE	27.94
		$0.1 \cdot \mathrm{MSE} + L_{\mathrm{SSIM}}$	27.46
		$\mathrm{MSE} + L_{\mathrm{SSIM}}$	27.18

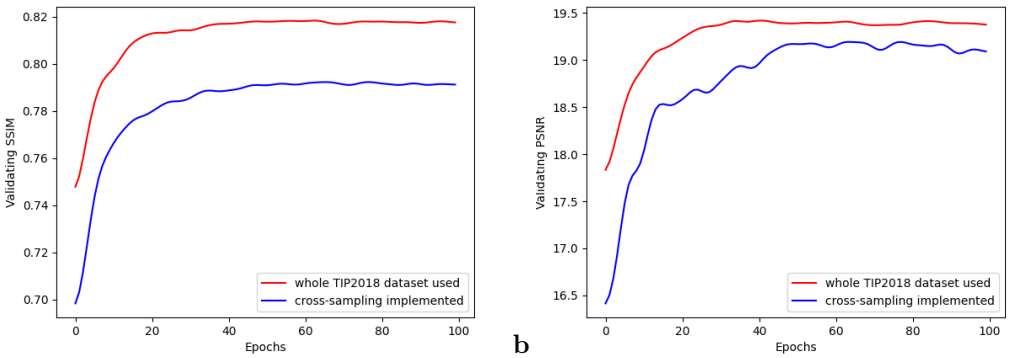


Fig. 4. Validating with two measures in function of epochs for the UHDM validation dataset: (**a**) with SSIM; (**b**) with PSNR. Graph color represents the loss function applied.

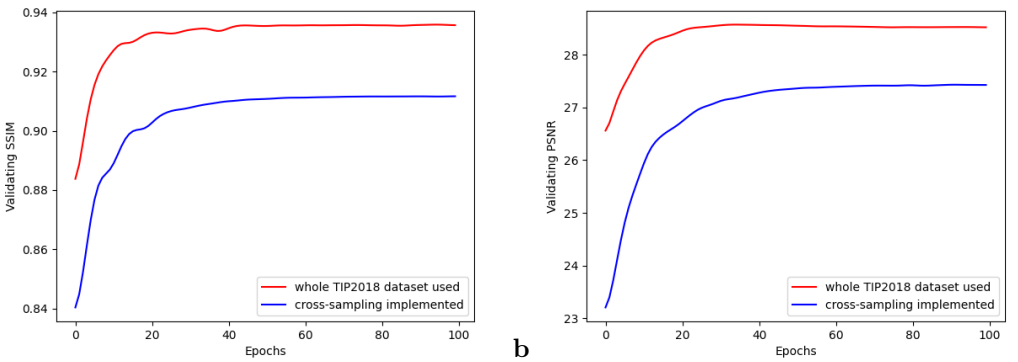


Fig. 5. Validating with two measures in function of epochs for the TIP2018 validation dataset: (**a**) with SSIM; (**b**) with PSNR. Graph color represents the loss function applied.

Charts with measures in the function of epoch iterations for the UHDM dataset are presented in Figs. 4a and b.

It can be noticed that differentials and rate of increase of the SSIM measure are similar for both methods. The correlation between PSNR measures is shaped differently and obtained efficiency gap is narrowed.

The corresponding charts for the TIP2018 dataset are shown in Figs. 5a and b.

Accurate results for both discussed training methods are presented in the Tab. 2.

We provided two versions of our model: *cross-sampled Attention U-net for image demoiréing (cs-AUid)* and *Attention U-net for image demoiréing (AUid)*. The first solution is much less time-consuming and easier to train while the second one achieves better results.

Tab. 2. Comparison of the *cross-sampling* method and full-dataset training metrics.

DATASET	METRIC	CROSS-SAMPLING	VALUE
UHDM	SSIM	YES	0.80
		NO	0.82
	PSNR	YES	19.31
		NO	19.48
TIP2018	SSIM	YES	0.91
		NO	0.94
	PSNR	YES	27.46
		NO	28.58

3.3. Comparison with other algorithms

In Tab. 3 we presented our results in comparison with the best available and documented solutions. Based on the following outcome, we can conclude that our method outperforms most of the other techniques in SSIM metric. We can suspect that it is an advantage of the relatively innovative loss function and efficient *Attention U-Net* algorithm. The results of *cs-AUid* model might be considered as a satisfying replacement that can easily be trained even with second-rate computational resources. It is less time-consuming and based on effective training procedures.

Notwithstanding, we need to remember that results presented in the previous chapter were obtained for a resolution $512 \cdot 512$. For scientific cases, we tested our trained model also on the benchmark images with a much higher number of image pixels.

In Figs. 6 and 7 the examples of the model outcome are shown. The images were made with our *cs-AUid* approach. We can observe that moiré is barely visible.

4. Conclusions

In this work, we proposed a convolutional neural network based *Attention U-Net* for image demoiréing. We presented two training procedures and we analyzed three loss functions. Our innovation lies in efficient datasets management and proper architecture choice. Our solution might be interpreted as an efficient alternative for more complex and time-consuming models.

Tab. 3. Comparison with other *state-of-the-art* algorithms. *cs-AUid* corresponds to cross-sampling method while *AUid* means large dataset used in the training. Both models were trained with $0.1 \cdot \mathrm{MSE} + L_{\mathrm{SSIM}}$ loss function. Compared solutions: TIP2018 [20], MopNet [7], MBCNN [29], FHDe2Net [4], ESDNet-L [27].

DATASET	MEASURE	INPUT	TIP2018	MopNet	MBCNN	FHDe2Net	ESDNet-L	cs-AUid	AUid
UHDM	SSIM	0.51	0.76	0.76	0.79	0.75	0.80	0.76	0.78
	PSNR	17.12	19.91	19.49	21.41	20.34	22.42	18.20	18.25
TIP2018	SSIM	0.74	0.87	0.89	0.89	0.90	0.92	0.88	0.90
	PSNR	20.30	26.77	27.75	30.03	27.78	30.11	26.03	26.82
Number of parameters (millions)			1.4	58.6	15.2	13.6	10.6	6.4	6.4

Fig. 6. Examples of images from UHDM dataset. From left: moiréd image, original image, demoiréd image (cs-AUid output).

Fig. 7. Examples of images from TIP2018 dataset. Image references are enhanced with a black border which is explained by the observation that dark colors are least affected by the moiré effect. From left: moiréd image, original image, demoiréd image (cs-AUid output).

The next part of the research should be adapting models for higher resolutions. Presented models were trained and validated on the $512 \cdot 512$ pixel maps. Basing on the results collected in Tab. 3 we can notice that with the increase in image resolution the quality of the images indicated with the respective quality measures significantly decreased. We also could study the loss function in the wider range of SSIM-MSE relation in the measure L_2 according to (5) to optimize the model parameters. In the next steps, the PSNR also should be maximized. In [15] its author presented a new similarity measure denoted there as CMSC which can be examined as a loss function in our further research.

References

[1] I. Alhashim and P. Wonka. High quality monocular depth estimation via transfer learning. *arXiv*, 2018. doi:10.48550/arXiv.1812.11941.

[2] Y. Zhang an Y. Tian, Y. Kong, B. Zhong, and Y. Fu. Residual dense network for image restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(7):2480–2495, 2020. doi:10.1109/tpami.2020.2968521.

[3] P. Anderson, X. He, C. Buehler, et al. Bottom-up and top-down attention for image captioning and visual question answering. In *Proc. 2018 IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 6077–6086, Salt Lake City, Utah, 18-23 Jun 2018. IEEE. doi:10.1109/CVPR.2018.00636.

[4] X. Cheng, Z. Fu, and J. Yang. Multi-scale dynamic feature encoding network for image demoiréing. In *Proc. 2019 IEEE/CVF Int. Conf. Computer Vision Workshops (ICCVW)*, pages 3486–3493, Seoul, Korea, 27 Oct – 2 Nov 2019. IEEE. doi:10.1109/ICCVW.2019.00432.

[5] J. Gurrola-Ramos, O. Dalmau, and T. E. Alarcon. A residual dense U-Net neural network for image denoising. *IEEE Access*, 9:31742–31754, 2021. doi:10.1109/ACCESS.2021.3061062.

[6] C. Wang K. Batmanghelich D. Tao H. Fu, M. Gong. Deep ordinal regression network for monocular depth estimation. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. doi:10.1109/cvpr.2018.00214.

[7] B. He, C. Wang, B. Shi, and L. Y. Duan. Mop moiré patterns using MopNet. In *Proc. 2019 IEEE/CVF Int. Conf. Computer Vision (ICCV)*, pages 2424–2432, Seoul, Korea, 27 Oct – 2 Nov 2019. IEEE. doi:10.1109/ICCV.2019.00251.

[8] B. He, C. Wang, B. Shi, and L.-Y. Duan. FHDe^{2}Net: Full high definition demoireing network. In A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, editors, *Computer Vision – Proc. ECCV 2020*, volume 12367 of *Lecture Notes in Computer Science*, pages 713–729, Glasgow, UK, 23–28 Aug 2020. Springer International Publishing. doi:10.1007/978-3-030-58542-6_43.

[9] X. Hu, M. A. Naiel, A. Wong, et al. RUNet: A robust UNet architecture for image super-resolution. In *Proc. 2019 IEEE/CVF Conf. Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 505–507, Long Beach, California, 16-20 Jun 2019. IEEE. doi:10.1109/CVPRW.2019.00073.

[10] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *Proc. 3rd Int. Conf. Learning Representations, ICLR 2015*, San Diego, CA, 7-9 May 2015. Accessible in arXiv. doi:10.48550/arXiv.1412.6980.

[11] O. Kupyn, T. Martyniuk, J. Wu, and Z. Wang. DeblurGAN-v2: Deblurring (orders-of-magnitude)

faster and better. In *Proc. 2019 IEEE/CVF Int. Conf. Computer Vision (ICCV)*, pages 8877–8886, Seoul, Korea, 27 Oct – 2 Nov 2019. IEEE. doi:10.1109/ICCV.2019.00897.

[12] Z. Lu and Y. Chen. Single image super resolution based on a modified U-net with mixed gradient loss. *Signal, Image and Video Processing*, 16(5):1143–1151, 2022. doi:10.1007/s11760-021-02063-5.

[13] 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)*, pages 257–265, Honolulu, Hawaii, 21-26 Jul 2017. IEEE. doi:10.1109/CVPR.2017.35.

[14] O. Oktay, J. Schlemper, L. L. Folgoc, et al. Attention U-Net: Learning where to look for the pancreas. *arXiv*, 2018. doi:10.48550/arXiv.1804.03999.

[15] G. Palubinskas. Image similarity/distance measures: what is really behind MSE and SSIM? *International Journal of Image and Data Fusion*, 8(1):32–53, 2016. doi:10.1080/19479832.2016.1273259.

[16] A. Paszke, S. Gross, F. Massa, et al. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32 – Proc. 33rd Conf. Neural Information Processing Systems (NeurIPS 2019)*, volume 11, pages 8024–8035, Vancouver, Canada, 8-14 Dec 2019. Accessible in arXiv. doi:10.48550/arXiv.1912.01703.

[17] B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996. doi:10.1017/CBO9780511812651.

[18] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In N. Navab, J. Hornegger, W. M. Wells, et al., editors, *Medical Image Computing and Computer-Assisted Intervention – Proc. MICCAI 2015*, volume 9351 of *Lecture Notes in Computer Science*, pages 234–241, Munich, Germany, 5-9 Oct 2015. Springer International Publishing. doi:10.1007/978-3-319-24574-4_28.

[19] De R. I. M. Setiadi. PSNR vs SSIM: imperceptibility quality assessment for image steganography. *Multimedia Tools and Applications*, 80(6):8423–8444, 2021. doi:10.1007/s11042-020-10035-z.

[20] Y. Sun, Y. Yu, and W. Wang. Moiré photo restoration using multiresolution convolutional neural networks, 8 May 2018. `https://paperswithcode.com/paper/moire-photo-restoration-using-multiresolution`. [Accessed: May, 2022].

[21] F.-J Tsai, Y.-T. Peng, Y.-Y. Lin, et al. Stripformer: Strip transformer for fast image deblurring. In S. Avidan, G. Brostow, M. Cissé, et al., editors, *Computer Vision – Proc. ECCV 2022*, volume 13679, Part XIX of *Lecture Notes in Computer Science*, pages 146–162, Tel Aviv, Israel, 23-27 Oct 2022. Springer Nature Switzerland. doi:10.1007/978-3-031-19800-7_9.

[22] Z. Tu, H. Talebi, H. Zhang, et al. MAXIM: Multi-Axis MLP for image processing. In *Proc. 2022 IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 5759–5770, New Orleans, Louisiana, 18-24 Jun 2022. IEEE. doi:10.1109/CVPR52688.2022.00568.

[23] X. Wang, Y. Li, H. Zhang, and Y. Shan. Towards real-world blind face restoration with generative facial prior. In *Proc. 2021 IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 9164–9174, Virtual conference, 20-25 Jun 2021. IEEE. doi:10.1109/CVPR46437.2021.00905.

[24] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Processing*, 13(4):600–612, Apr 2004. doi:10.1109/TIP.2003.819861.

[25] Z. Wang, X. Cun, J. Bao, et al. Uformer: A general U-shaped transformer for image restoration. In *Proc. 2022 IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 17662–17672, New Orleans, Louisiana, 18-24 Jun 2022. IEEE. doi:10.1109/CVPR52688.2022.01716.

[26] W. Wang Y. Sun, Y. Yu. Moiré photo restoration using multiresolution convolutional neural networks. *IEEE Trans. Image Processing*, 27(8):4160–4172, 2018. doi:10.1109/tip.2018.2834737.

[27] X. Yu, P. Dai, W. Li, L. Ma, et al. Towards efficient and scale-robust ultra-high-definition image demoiréing. In S. Avidan, G. Brostow, M. Cissé, et al., editors, *Computer Vision – Proc. ECCV 2022*, volume 13678, Part XVIII of *Lecture Notes in Computer Science*, pages 646–662, Tel Aviv, Israel, 23-27 Oct 2022. Springer Nature Switzerland. doi:10.1007/978-3-031-19797-0_37.

[28] S. W. Zamir, A. Arora, S. Khan, et al. Restormer: Efficient transformer for high-resolution image restoration. In *Proc. 2022 IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 5718–5729, New Orleans, Louisiana, 18-24 Jun 2022. IEEE. doi:10.1109/CVPR52688.2022.00564.

[29] B. Zheng, S. Yuan, G. Slabaugh, and A. Leonardis. Image demoireing with learnable bandpass filters. In *Proc. 2020 IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 3633–3642, Virtual conference, 14-19 Jun 2020. IEEE. doi:10.1109/CVPR42600.2020.00369.

ForestTaxator: A Tool for Detection and Approximation of Cross-Sectional Area of Trees in a Cloud of 3D Points

Maciej Małaszek, Andrzej Zembrzuski, Krzysztof Gajowniczek
Institute of Information Technology
Warsaw University of Life Sciences – SGGW, Warsaw, Poland
krzysztof_gajowniczek@sggw.edu.pl

Abstract. In this paper we propose a novel software, named *ForestTaxator*, supporting terrestrial laser scanning data processing, which for dendrometric tree analysis can be divided into two main processes: tree detection in the point cloud and development of three-dimensional models of individual trees. The usage of genetic algorithms to solve the problem of tree detection in 3D point cloud and its cross-sectional area approximation with ellipse-based model is also presented. The detection and approximation algorithms are proposed and tested using various variants of genetic algorithms. The work proves that the genetic algorithms work very well: the obtained results are consistent with the reference data to a large extent, and the time of genetic calculations is very short. The attractiveness of the presented software is due to the fact that it provides all necessary functionalities used in the forest inventory field. The software is written in C# and runs on the .NET Core platform, which ensures its full portability between Windows, MacOS and Linux. It provides a number of interfaces thus ensuring a high level of modularity. The software and its code are made freely available.

Key words: point cloud, genetic algorithms, trees, 3D scan.

1. Introduction

A point cloud is an unordered collection of points in the R^3 space, which is in fact a three-dimensional image (most often of a real-life, scanned space). Due to the different parameters of devices scanning their surroundings, each point may contain additional information, such as the strength of light reflection or the color of a given point. Such information are especially useful when the cloud is used to create a three-dimensional model of a real object along with its colors.

Working with a 3D point cloud very often can be difficult, because it contains even billions of spatial points and for this reason, at a given time one often works with a small segment of the cloud [1]. The point cloud is often stored as a text file in ASCII format, due to ease of reading. In such a file each point is stored as three Cartesian coordinates (X, Y and Z) [2]. One effective way to manage such a cloud is to use the octal tree algorithm. Appropriate implementation makes it possible to obtain not only the faster access to a group of points located in the vicinity, but also the data compression even up to 10% of the uncompressed data size [1]. The potential uses of such an image are very wide as it provides relatively accurate information about the shape and size of a given object or environment. Examples of areas in which the point cloud is used

are industrial automation, architecture, agriculture, construction and maintenance of tunnels and mines, urban space planning, as well as inventory of forest resources [1, 2].

Inventory of forest resources is usually performed using a terrestrial laser scan (TLS). The use of this technique allows for quick, effective and automatic determination of the basic parameters of the stand, i.e., the number and location of trees, their diameter at a height of about 1.3 m (the so-called *diameter at breast height*, DBH) and the shape of the trunk and crown [2]. By applying the selected algorithm of matching a circle or an ellipse to the cross-section of a tree, it is possible not only to detect the tree, but also to determine the volume of the trunk, which is an important information in forest management [1].

In this work, the problem of 3D point cloud analysis in searching for information about trees will be presented. The technical part will be focused on solving the problem of detecting and modelling a tree using genetic algorithms. Both the tree detection algorithm and the algorithm for modeling a tree with ellipses will be described. The tree detection algorithms can be interpreted as a filtering algorithm, and by detection is meant identifying the points that make up the tree trunk and discarding those points which make up branches, leaves and other objects. The tree model will consist of a collection of ellipses saved in a tree structure. The ellipses will inform about the shape of the trunk, its location and diameter. The use of ellipses instead of circles is a non-standard approach, due to the lack of a formula that would allow analytical determination of the distance of any point from the circumference of the ellipse, but but the model based on ellipses will achieve much better results than the model based on circles in the case of tilted trees.

To date, there is no commercial software for working with TLS data that covers all aspects of data processing for forestry purposes [2]. Two companies have developed such software, i.e., Treemetrics (operating in Ireland) has developed the *AutoStem* software [3], and Taxus IT (operating in Poland) has developed the *tScan* software [4]. However, in both cases, they have not yet been included in their companies' commercial offerings. Nevertheless, there are several free solutions (Table 2 in Section 8) that enable such TLS data processing, mainly developed by research centers. These include standalone desktop applications and libraries of specialized tools used in software development environments such as Matlab, R or Python. Unfortunately, none of them provides all necessary functionalities in one place.

To fill this gap, in this paper we present a novel software called *ForestTaxator* [5]. The software supports tree detection in the point cloud and the development of three-dimensional models of individual trees. The software and its code are made freely available. A library providing ready-made solutions for creating genetic algorithms is also available. This library was entirely designed, constructed and distributed under the MIT license by the author of this work and is his sole intellectual property. The main contributions of this paper can be summarized as follows.

1. It provides all necessary functionalities in one place compared to other freely available software.
2. It provides innovative noise removal filters, for both non-tree elements and noise resulting from scanner error.
3. It employs genetic algorithms to solve the complex problem of tree detection and its cross-sectional area approximation with ellipse-based modelling 3D point cloud.
4. In both the detection and approximation algorithms various version of genetic algorithms are applied and numerically tested, achieving very good accuracy in relatively short time.
5. The analysis is performed using our software, which is made freely available.

The remainder of this paper is organized as follows. Section 2 provides an overview of the similar research for detection and approximation of cross-sectional area of trees in a cloud of 3D points. In Section 3, the theoretical framework of the proposed algorithms is presented. In Sections 4 and 5 the detection and approximation algorithms are described in detail. Section 6 presents the architecture, the functionalities of the software along with the illustrative example. Section 7 outlines the numerical experiments and presents the discussion of the results. Section 8 outlines the impact of the software together with a comparison between *ForestTaxator* and other free tools. The paper ends with concluding remarks in Section 9.

2. Literature review

TLS scanners work by measuring the distance and the horizontal and vertical angles between the device and the object under examination usinglaser light beams emitted by the device [2]. Nowadays, there are several models of TLS on the market, which can be divided into two main groups: phase shift scanners and time-of-flight (ToF) scanners. The main distinguishing feature between these two scanners' types is the distance measurement technology [6]. ToF scanners measure the distance less accurately than phase-shift scanners; nevertheless, the data obtained with phase-shift scanners are subject to noise. ToF scanners tend to have a greater data measurement range compared to phase-shift scanners [7]. A main superiority of ToF scanners is the ability to record many reflections of the laser beam. This is especially important when scanning objects near vegetation [2, 6].

The TLS data acquisition in the forest environment can be performed on three different levels, i.e., sample plot, individual tree and whole stand. Sample plots are usually performed in single scan (SS) or multi scan (MS) data acquisition mode [8], where the latter one is much faster. The main disadvantage of the SS mode data acquisition is the high likelihood of the so-called 'occlusion effect' [9], i.e., some trees to be omitted because trees in the same azimuth relative to the plot center obscure each other [10].

TLS data gathered based on individual trees are particularly useful in improving or developing allometric equations for traits such as whole-tree volume or biomass [2]. The main pros is the speed, non-invasiveness, and precision of obtaining stem morphological curve information. This kind of scanning is usually done in MS mode, where scanner locations are placed around the tree from at least three positions [11]. Nevertheless, this number should be selected depending on the size of the object being scanned and the planned level of data detail [12]. The MS mode is usually employed for the whole stand scanning, but is should be remembered that this mode is usually more complicated due to the need of repositioning locations of the scanning positions [13].

When planning data acquisition with TLS, in addition to technical aspects such as determining scanning parameters and selecting the appropriate data acquisition mode, external environmental factors must be considered, such as weather conditions and vegetation period. Optimal conditions for TLS are windless days without precipitation, moderate temperatures, and low humidity [2, 14]. Wind is a factor that can significantly affect the quality of data collected (occur when the average wind speed does not exceed 5 m/s [15]), especially in the tree canopy. Scanning can be driven in light rain or fog, but is also not recommended, as well when there is snow cover [16]. For surveys designed to obtain the most accurate data on the morphology of woody parts of trees and to estimate their volume or biomass, the best time to scan is in early spring or late fall [2].

TLS data processing for dendrometric tree analysis can be divided into two main processes: detection of trees in the point cloud and creation of 3D models of individual trees [2]. The first process usually consists of two stages, where the first phase involves placing a thin horizontal slice of a 3D point cloud on a horizontal plane. During the next phase, trees are detected using clustering algorithms grouping the points and next by searching for some geometric shapes, such as circles or semicircle [17, 18]. Unfortunately, this assumption does not hold for complex stands with high tree density or undergrowth [19]. To address this issue, practitioners have developed methods for accurately identifying stems or woody parts of trees directly in the point cloud. Authors in [20] presumed that woody parts have a higher reflectance intensity (normalizing of this value is time-consuming and complicated [19]) than leaves. Authors in [23] used a different approach by iteratively employing the local geometric features of point clouds, where the nearest groups of points, defined by the radius of the sphere or the number of nearest neighbors around the focal point, are selected by the principal component analysis (PCA). Geometric features of point clouds like verticality, flatness and linearity were used for trunk detection [6, 24]. Finally, machine learning algorithms were used to classify the trunk as well [25]. The second process usually uses the quantitative structure model (QSM). Such a model is assumed to represent tree morphology as accurately as possible and is fully measurable since these properties allow for accurate determination

of the thickness and volume of aboveground woody components [2]. Authors in [6] distinguish five levels of detail of digital tree models that allow for different characterizations of the modelled trees [2].

3. Theoretical framework of the proposed method

3.1. LiDAR

LiDAR is a mechanism that makes it possible to accurately record the elements of the environment. The coordinates of the points representing the individual details of the scanned objects are recorded in the Cartesian system and conventionally referred to as X, Y and Z coordinates. Each recorded point represents a point in real space from which the laser beam was reflected. The coordinates are computed from the distance measurement and the vertical and horizontal angle measurements at which the laser beam is emitted. Each LiDAR laser scanner has two degrees of freedom that describe the azimuth and elevation of the scanner at a given moment. With this information, the scanner records the reflection of the beam and can record the exact Cartesian coordinates of the detected opaque point. LiDAR scanners record data at the speed of thousands of points per second, achieving an accuracy of the order of millimeters.

There are two methods of measuring the distance. The first is the method of measuring the phase shift of the electromagnetic wave. The phase shift of the wave is proportional to the distance traveled by the beam [26] and can be expressed by the formula:

$$d = \frac{\frac{c}{2f}\phi}{2\pi}\,, \tag{1}$$

where d – distance of the reflection point from the emitter, ϕ – phase shift (delay), f – amplitude modulation frequency of the emitted beam, c – speed of light. The intensity of the wave, in other words – the brightness of the light emitted by the laser, is modulated. The length of the electromagnetic wave may differ between scanner models – there are models that emit waves in the visible light range as well as in the near infrared range. Most often, however, it is in the range from 600 nm to 800 nm. This method is characterized by high accuracy, high number of scanned points per second and an average range of effective measurements (up to about 150 m).

The assumptions of the second method are much simpler because it consists in measuring the time between sending the laser pulse and receiving its reflection, and the distance can be determined by the formula:

$$d = \frac{\frac{1}{2}c}{t}\,. \tag{2}$$

Unlike the previous method, the emission of the wave is not continuous, but is a single

pulse. The implementation of these assumptions requires the use of components operating at very high frequencies, in the order of tens of GHz, and this greatly complicates the achievement of a precise device. This method allows for measurements at a maximum range of up to 2000 m.

3.2. Genetic algorithms

Genetic algorithms are a subset of evolutionary algorithms. They are inspired by natural evolution and follow its principles by implementing mechanisms of natural selection, mutation and inheritance. The idea of evolutionary algorithms has been around for almost 60 years, but its foundations remain unchanged. There is a large class of optimization problems for which no specialized algorithms have been developed. In the case of simpler problems, the solution space of which is small, iterative search can be used. In the case of huge spaces, the evolutionary solutions are usually applied. Genetic optimization is a stochastic approach in which the way of searching the solution space is modeled by genetic inheritance and Darwinian competition for survival.

The feature that distinguishes the evolutionary approach from other algorithms is its genetic resistance to the phenomena affecting the operation of traditional analytical optimization methods, i.e. the lack of of necessity to use derivatives, insensitivity to the presence of discontinuities in the solution space, or ability to step over local extrema. Of course, this does not mean that the evolutionary algorithm ensure that a globally optimal solution is found. The robustness of these algorithms results from the combination of four features: encoding the parameters, operating on populations, introducing the element of randomness and using the minimum necessary information about the problem.

As mentioned before, when describing genetic algorithms, concepts derived from natural genetics are used. The meaning of all of these concepts should therefore be clearly stated. An individual represents exactly one element in the search (or solutions) space of a given problem. A population is a collection of individuals or solutions. Fitness (or objective) function is a measure of the fit of a specific individual to the solution sought. Selection is a method of stochastic or deterministic selection of individuals which will pass on their features in the form of genetic material to the next generation. The foundations of the genetic algorithms have been described for example in [21, 22].

4. Detection algorithm

4.1. Genetically distinguishing the arrow of the tree from branches, leaves and noise

The first stage of the analysis, after cleaning the data, is to find the points shaping the tree's arrow. This can be done in various ways, including the Hough transform, but we decided to use a hybrid genetic algorithm for this. In order to solve the problem with

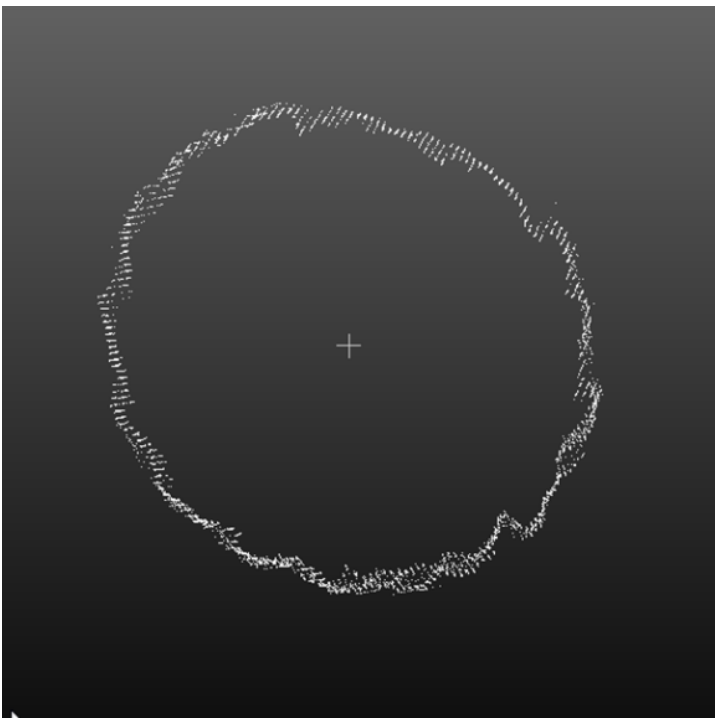


Fig. 1. Vertical projection of the cross-section of the tree scan at a height of about 1.5 m.

the help of an evolutionary algorithm, a simplified model is needed. Once a model is identified, an evaluation function should be designed that clearly identifies which of the two potential solutions is better. It is also necessary to define an encoding method that will make it possible to use the selected crossover operators.

4.2. The perfect arrow model

As has been said before, the cross section of the tree arrow can be approximated with a circle. The data, from the RemBioFor project [28], were made in a multi-station mode, i.e. from three stations located at the vertices of the triangle. As a result, most of the scanned tree lacks places shaded by the tree itself. This is illustrated in Figure 1.

The density of the points remains approximately the same. Therefore, in a circle there would be a relationship in which the distribution of points in any chosen axis would resemble a symmetrical bimodal distribution, which would be similar to the sketch presented in Figure 2.

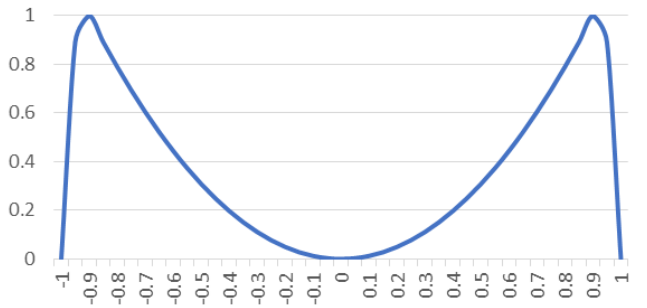


Fig. 2. Sketch of a 3D point distribution for a circle.

Since the distribution will be identical regardless of the selected axis, one can select coordinate axes lying in the plane of the cross-section. Thanks to this, it will be possible to precisely define the shape of the points. The exact dimensions of the point group are also known, which makes it possible to normalize the domain of the function that will be used to approximate the distribution. To calculate the distribution, the entire group should be divided into equal fragments, and then it should be counted how many points are there in the given fragment of the group. The denser the split, the smoother the distribution will be, but at the same time there will be larger deviations due to the structure of the tree. In our solution, we decided to divide the groups into 32 equal parts. For a tree with a diameter of 50 cm, the width of one group will be 1.5625 cm. In addition, we decided to omit the first two and last two fragments in the analysis. This allows us to approximate the distribution using the quadratic function.

It should be emphasized that the above simplified approach is used only to detect trees, while the shape and size of the cross-section of a detected tree will be later much more accurately approximated using ellipses.

In summary, in the ideal tree arrow model it is assumed that it is a circle with an even distribution of points on the tree surface. This model is represented by two bimodal distributions that will be approximated by quadratic functions. The parameters of the entire model are the parameters of both square functions, i.e. six floating point values. This is a case in which the application of genetic algorithms is simple and at the same time can give very good results.

4.3. Fitness function

The selection of the evaluation function should reflect the properties of the analyzed problem. The function needs not be differentiable, but should be continuous as far as possible in the domain of individuals that may arise. The problem of discontinuity could

be avoided, for example, by introducing the lethality of some mutations, but this would introduce an additional performance overhead.

In the adopted model, the fitness function is the similarity of the distribution of points in perpendicular axes to the quadratic function. Similarity is best determined by calculating the differences for each of the 28 fragments. The requirement for the fitness function is that it must return one floating point number, which forces additional operations on the calculated differences. A frequently used aggregation function is the arithmetic mean, but it is not immune to outliers. Also, when some values are greater than the model values, and others are smaller, the average difference may be 0, and in fact the error will be large and the analyzed group of points will not be a circle. Therefore, it is also worth using the standard deviation, which measures how far individual values are from the mean value.

Both metrics should be used to form a single number, ideally this number should be 0. We have arbitrarily chosen the sum of the absolute values of the mean and the standard deviation. Thus, the following model arises:

$$\mathrm{avg}_i = \frac{\sum_{x=3}^{30} \mathrm{diff}_i[x]}{28}\,, \tag{3}$$

$$\mathrm{stdDev}_i = \sqrt{\frac{\sum_{x=3}^{30} (\mathrm{diff}_i[x] - \mathrm{avg}_i)^2}{28}}\,, \tag{4}$$

$$f_i = |\mathrm{avg}_i| + |\mathrm{stdDev}_i|\,, \tag{5}$$

$$f = \max_i f_i\,, \tag{6}$$

where $\mathrm{diff}_i[x]$ is the difference between the model and the individual for the fragment with index x in the i-th axis, avg_i is the mean value of the difference for the i-th axis, stdDev_i is the population standard deviation for the i-th axis, f_i is the value of the fitness function on the i-th axis, and f is the value of the fitness function for the individual.

The above function will make it possible to clearly define which individual is closer to the model and which is farther.

4.4. Genetic representation

The quadratic function will be used in the general form: $y = ax^2 + bx + c$. It is known that the second derivative must be positive. It follows that the parameter a must be positive. It is also known that the number of points in the distribution will not be lower than 0, so parameter c must also be non-negative. The model takes into account normalization of the values so that the values are in the range $[0; 1]$. This gives information in what interval the values of the quadratic function should be, and indirectly – in what interval the parameter values should be. If the maximum value of the function exceeds 1.5, it will be known that the given group is noise. Therefore, it can be assumed that the

range within which the parameter a will change should be equal to the range $[0.01; 1.5]$. The remaining parameters should oscillate close to zero, but in order to introduce some flexibility we chose the interval $[0; 0.5]$.

The scope of parameter changes has been limited. This eliminates the need to use larger variable types such as `double` or `int32`. Our solution assumes operating on an array of six `unsigned short` variables, i.e. 16 bit integers. This will give 65 536 possible values for each parameter. For the interval $[0.01; 1.5]$ it gives the realtive accuracy of 0.00002.

The solution will therefore be represented as a 96-byte string. The `CollectiveGenotype` class from the GeneticToolkit library [30] will be used for this purpose. As a parameterizing type, a special class `ParabolicParameters` will be used, containing a total of 6 floating-point variables (3 for each axis) as well as serializing and deserializing functions. The representation introduces no domain restrictions, so there is no need for a lethal mutation mechanism.

5. Approximation algorithm

5.1. Model for the approximating algorithm

Most often, circles are used to approximate the cross-section of a tree. It may happen that the cross-section of the tree is closer to the ellipse. The most important reason for this is that the tree can grow at a certain angle that is not taken into account when splitting the point cloud into layers. Therefore, in this work, we will use an ellipse as a model.

The ellipse-based model has the benefit of being able to obtain better results in the fitness function compared to the circular-based model. A significant disadvantage of this solution is the introduction of a greater number of degrees of freedom, which increase the computational complexity of the problem. In a two-dimensional space, a circle has three degrees of freedom: two center coordinates and a radius. An ellipse in the same space has five degrees of freedom: two coordinates of the first focus, two coordinates of the second focus, and the length of the major semi-axis. The model we have adopted will consist of three variables:

1. Two-dimensional coordinates of the first focus.
2. Two-dimensional coordinates of the second focus.
3. Length of the great driveshaft.

5.2. Fitness function

The fitness function will take into account the distance of points from the set under consideration to the ellipse. The distance of a point from the ellipse is meant as the Euclidean distance of a point in two-dimensional space from the nearest point on the

circumference of the ellipse. If we designate the focal points of the ellipse as F_1 and F_2, and any point lying on its circumference as P, then the major axis marked as a has the following relationship:

$$|PF_1| + |PF_2| = 2a\,. \tag{7}$$

Unfortunately, calculating the distance of any point from the circumference of the ellipse is not a trivial task, let alone a task with low computational complexity. However, as mentioned earlier, the evaluation function is primarily used to determine which of the two individuals is better, that is, closer to the expected solution. For this reason, we decided to introduce the following relationship for any point P:

$$d \sim |PF_1| + |PF_2| - 2a. \tag{8}$$

Such a relationship is not linear – the same increase in the distance of the point to the ellipse results in a different increase in the value of d depending on how far P was before moving away. This is advantageous because the farthest points will penalize the subject much greater than points close to the ellipse. As a result, minor distortions, caused for example by the shape of the cortex, will not have as large an effect as if the distance between the point and the ellipse were correctly measured. Eventually, we are going to use the following value as the fitness function:

$$f = \sum_i \frac{|P_iF_1| + |P_iF_2| - 2a}{n}\,, \tag{9}$$

where n stands for the number of points.

5.3. Genetic representation

The genotype must encode five floating point variables. The coordinates of the foci can basically have any values, therefore it is impossible to narrow down the range in which they should be searched for. In the case of the large driveshaft, it is known that it must be a value greater than 5 cm, and it will also be much smaller than 5 m. For all variables, differences not exceeding 0.01 mm will not seriously affect the results. Assuming a very rough approximation that the tree is a cylinder 30 m high, the accuracy in the calculated volume will be about $0.000000942\,\mathrm{m}^3$. This is a completely negligible error, and adopting such precision will allow for the use of single precision floating-point numbers.

The above assumptions indicate that it will be efficient to use the `CollectiveGenotype` class with the `EllipticParameters` parameterizing type. The `EllipticParameters` class will work analogously to the `ParabolicParameters` class, that is, it will contain five floating point variables as well as serialization and deserialization functions, taking into account the assumptions made above regarding the search ranges.

6. Software Description

6.1. Software architecture

The software is written in C# and runs on the .NET Core platform, which ensures its full portability between Windows, MacOS and Linux. It provides a number of interfaces thus ensuring a high level of modularity. The software development is supported by the use of the Azure CI/CD (Continous Integration – Continous Delivery) system, which allows to design the procedure for testing and implementing the solution. The quality of the code from the beginning of the project has been maintained at the highest A level in terms of ease of maintenance, reliability and security. In order to work efficiently the *ForestTaxator* requires some 3D point cloud and mesh processing software such as *CloudCompare* [31].

The software is available at the GitHub repository and after downloading the source files it can be compiled using the following command:

```
dotnet build -c Release forest-taxator
```

ForestTaxator provides all functionalities in user friendly console application, i.e. it takes input and displays output at a command line console with access to three basic data streams: standard input, standard output and standard error. By calling the following command

```
./ForestTaxator.Application --help
```

the user sees that the software is composed of two main functionalities.

```
ForestTaxator.Application 1.0.0
Copyright (C) 2021 ForestTaxator.Application
analyze
convert
help      Display more information on a specific command.
version   Display version information.
```

The convert functionality allows to transform one file structure into another, such as, PCD (Point Cloud Data) file into XYZ or GPD (Generic Printer Description) file into XYZ. The analyze functionality aggregates all necessary functions allowing to conduct the entire study step by step.

```
ForestTaxator.Application 1.0.0
Copyright (C) 2021 ForestTaxator.Application
approximate-trees Approximates tree trunk using ellipses
detect-trees
filter
slice
```

```
terrain
tree-height
help              Display more information on a specific command.
version           Display version information.
```

6.2. Illustrative example

As an input ForestTaxator accepts flat files such as .txt, .csv or binary compressed PCD file format. In this section we present an illustrative example performed on 3D Forest dataset [32] stored as a flat files. Dataset consists of multiple layers constituting the base cloud where all points of plot are present. Below we present all necessary commands along with the required parameters (and their short description).

1. Extract terrain height map from `example.xyz` file into `terrain.thf`.

   ```
   Input: Source point cloud file
   -o -> path & filename where output file has to be saved
   ./ForestTaxator.Application analyze example.xyz
   -o output/terrain
   ```

2. Extract tree height map from `example.xyz` file into tree-height – it takes highest point from each square in a grid.

   ```
   Input: Source point cloud file
   -o -> path & filename where output file has to be saved
   ./ForestTaxator.Application analyze tree-height example.xyz
   -o output/tree-height
   ```

3. Split `example.xyz` file into collection of slices and save this collection to format that supports grouping, -t parameter specifies path to terrain height map, which is used to normalize height of each point in cloud.

   ```
   Input: Source point cloud file; Terrain heightmap
   -t / --terrain -> Path to terrain heightmap file
   -o -> path & filename where output file has to be saved
   ./ForestTaxator.Application analyze slice example.xyz
   -t output/terrain -o output/sliced
   ```

4. Create small groups of point based on their neighborhood – points that are located near each other form a group of points. Based on properties of each group, application filters out those groups that are considered to be *noise* (leaves, small branches etc.). Result is stored in filtered directory. All necessary parameters are stored and defined by the user in the configuration `.json` file (please see Listing 1 for the structure of the file).

```
Input: Sliced point cloud from slice step; Terrain height map
-c -> Path to FiltersConfiguration.json file
-o -> path & filename where output file has to be saved
./ForestTaxator.Application analyze filter output/sliced
-c Configs/FiltersConfiguration.json -o output/filtered.gpd
```

5. Convert `gdp` file into `XYZ` file.

```
./ForestTaxator.Application convert -i GPD
-o XYZ output/filtered.gpd output/filtered.xyz
```

6. Create collection of point groups. Each collection is considered to be a tree candidate. Collections are created based on XY location for each slice. Tree candidates that do not meet requirements are removed, and only remaining ones are exported to detected directory.

```
Input: GPD file from filter step of collection of XYZ files
from detect-trees step
--export-preview -> for each exported GPD file, XYZ file is created
so it can be viewed in 3d cloud viewer
-o -> path to DIRECTORY where output file has to be saved
./ForestTaxator.Application analyze detect-trees output/filtered.gpd
-o output/detected --export-preview
```

7. Create approximation of tree trunk using genetic algorithm.

```
Input: GPD file from detect-trees step; Tree height heightmap
--export-preview -> exports collection of approximating ellipses
into XYZ file. 4th value in point represents error equal to mean
distance between ellipse border and cloud point
--smooth - Smooth tree using regression on all nodes. It replaces
ellipses with circles
-h / --node-height -> float number representing height of single
slice. Required if file was sliced with different height than
default 0.1 m
-t / --tree-heightmap -> Path to tree heightmap file
-o -> path to DIRECTORY where output file has to be saved
./ForestTaxator.Application analyze approximate-trees output/detected
-c Configs/ApproximationConfiguration.json -o output/approx
-t output/tree-height --export-preview
```

Listing 1. Structure of the configuration file `FiltersConfiguration.json`.

```
{
  "LargeGroupsFilter": {
    "Order": 1,
    "LargeGroupsMaxSize": {
      "variables": [
        {
          "name": "height",
          "type": "System.Double",
          "order": 1
        }
      ],
      "expression": "Max(0.1f, 0.75f - 0.01f * height)"
    }
  },
  "SmallGroupsFilter": null,
  "AspectRatioFilter": null,
  "EllipsisMatchFilterConfiguration": {
    "FitnessThreshold": 0.1,
    "MatchEccentricityThreshold": 0.80,
    "BufferWidth": 0.002,
    "InvalidEccentricityThreshold": 0.85,
    "GeneticAlgorithmConfigurationFile":
      "Configs/DetectionEllipsisMatch.GeneticAlgorithm.json",
    "Order": 5
  },
  "GeneticDistributionFilterConfiguration": {
    "GeneticAlgorithmConfigurationFile":
      "Configs/Distribution.GeneticAlgorithm.json",
    "DistributionResolution": 32,
    "TrunkThreshold": {
      "variables": [
        {
          "name": "height",
          "type": "System.Double",
          "order": 1
        }
      ],
      "expression": "0.3"
    },
    "Order": 3
  }
}
```

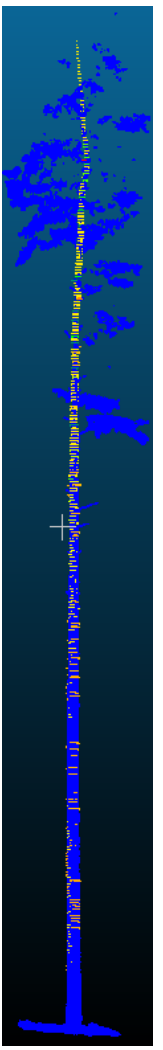


Fig. 3. Final output presented by CloudCompare.

After calling application in the last 7th step *ForestTaxator* creates `XYZ` file and `json` file. Both files consist of a collection of approximating ellipses (or circles) in a given format. After combining aforementioned `XYZ` file with the input `example.xyz` file, the final output is presented in the Figure 3, where blue dots denote the tree while yellow dotes represent the collection of ellipses.

7. Research framework and results

7.1. Experiment design

The data-set used in this article was provided by the Department of Geomatics of the Forest Research Institute with headquarters in Sękocin Stary as part of the RemBioFor project [28]. The test data includes scans of seven Scots pine trees, i.e. Hajnówka, Jedwabno, Nowe Ramuki, R_ilaw, R_lidz, Wejherowo, Wipsowo. All scans were made in a multi-station mode with the use of three stations in triangular layout. The data was manually trimmed by the employees of the Institute so that the cloud contained only the model tree. The cloud has been normalized to so that the lowest point is at height 0. All scans were made with the Faro Focus3D scanner.

Despite the high quality of the data, the scans do not show all trees due to auto-occlusion. The scans contain a lot of noise caused by branches and leaves; also, there are discontinuities in the tree trunks. This has a negative effect on the functionality of the algorithms. Halfway up the tree one can compensate for gaps by applying linear regression on correctly approximated fragments. However, the most important problem is the misrepresentation of the actual height information on the trees. This information is critical, mainly because in the formula by which it is calculated the diameters of the tree at different heights are used. In addition, a distortion in the height of the tree leads to a large error in estimating the volume of wood that can be harvested from a given single tree.

All simulations were repeated 10 times. After 10 repetitions, the mean and standard deviation were calculated. To make it justifiable to aggregate the results for all trees (comparing the results for all trees simultaneously), the results cannot be given depending on the height at which the analyzed group is located. Instead, the groups, the tree slices, were grouped by height as a percentage. This means that the height of each group was divided by the total height of the tree and multiplied by 100. This makes sense because each tree reaches a different height, so its crown begins at a different height, which strongly influences the results. Moreover, a different height could cause the last rows of the table to contain information obtained from only one tree, and the previous rows would have a very large standard deviation, precisely because of differences in the crown boundary.

In the case of the detection algorithm, the most important element of the analysis is the error matrix and the three derived measures: sensitivity, specificity, precision and negative predictive value. As previously stated, large gaps in the obtained tree trunk slices can be easily compensated by linear regression. It is therefore essential to obtain as high specificity as possible in order to filter out the noise. All tested groups were manually divided into noise and trunk fragments. The program had access to this information to generate the error matrix. The stop conditions were as follows:

1. Sufficient solution was found (fitness function value 0.22 or less).
2. Calculation time elapsed (2 s per group).
3. Maximum level of genetic homogeneity set at 90% is exceeded.

In the case of the approximating algorithm, the most important thing is to compare the results and the data on the cross-section of the tree. For this reason, the results will contain tree dimensions (defined every 1 m of height) and the corresponding values of the model will be approximated by genetic algorithms (and linear regression). The results may differ from the values indicated in the test data because the measurements were made by the authors of the test data so that the axes of the ellipses coincide with the north-south and east-west directions. It was not possible to calculate the direction in our program. It can be assumed that a correct fit will force all algorithms to indicate very similar values. The alignment results are also presented graphically by point clouds representing ellipses created by the best genetic algorithm (with the closest results). All algorithms should give very similar results because maximum available information about the problem is used. We predict that significant errors may occur in data containing trees whose cross-section is irregular in shape. Especially the "Hajnowka" set, which was prepared with an error consisting in incorrect combination of partial scans. The stop conditions were as follows:

1. Sufficient solution was found (fitness function value 0.015 or less).
2. Calculation time elapsed (1 s per group).
3. The maximum level of genetic homogeneity set at 90% is exceeded.

It is worth noting that for a genetic algorithm these are very extreme working conditions. The computation time of the genetic algorithm can be counted even in hours for complex problems. Moreover, the determination of the value of the fitness function of a sufficiently fit individual enables the quality of the obtained results to be controlled. Decreasing this value will be connected with the extension of the computation time. Without defining the maximum computation time and imposing an exact match, the algorithm could get stuck trying to find a perfectly matched individual when it might be impossible. An error of less than 1.5 cm seems to be a good compromise for the experimental conditions.

The experiments will cover selection methods, crossing methods, mutation methods and mutation policies. Among the selection methods, these will be:

1. Roulette wheel selection.
2. Ranked roulette with the mapping i_i^3.
3. A tournament with a size of 1% of the population, but not less than 2 individuals.

Crossbreeding methods:

1. Arithmetic crossover (2 parents, 2 children, 1 byte resolution).
2. One-point crossing (2 parents, 2 children).
3. Multipoint crossing (2 parents, 2 children, 3 crossing points).

4. Uniform crossing (2 parents, 2 children).

Mutation methods:

1. Arithmetic mutation (range $[-10; 10]$ in 1 byte resolution).
2. Negation of bits.

Mutation Policies:

1. Simple mutation (3% probability of mutation and maximum 10% mutated genes).
2. Backa method (maximum 10% mutated genes).
3. Hesser-Manner method ($\alpha = 1$, $\beta = 1$, maximum 10% of mutated genes).

7.2. Determination of the value of the parameter ϵ

The analysis was carried out with the accuracy of a single group of points. Each group was assigned the following information: a) the height of the point group; b) the value of the fitness function; c) the classification according to the threshold $\epsilon = 0.22$; d) the manual classification using CloudCompare; e) whether the algorithm made a mistake; f) does the group belong to the tree trunk (not noise). On their basis, we counted the number of errors to determine the approximate effectiveness of the algorithm, and also determined two values: a) the smallest fitness function value achieved for the noise group; and b) the highest value of the fitness function achieved for the group being a fragment of the trunk. We used these two values to calculate the mean value, we got the values presented in Table 1.

From the above table, we finally got an average value of $\epsilon = 0.24$ with a standard deviation of 0.02. The relatively low standard deviation led to the conclusion that this value would give the best results. It is a threshold that is equidistant from the values for the groups forming the trunk and from the noise groups.

Noteworthy are the graphs (in Supplementary Materials [29]) illustrating the values of the fitness function depending on the height for individual trees. The gray line in the graphs described as "EPSILON" represents the threshold value previously calculated as the mean value between the noise group with the lowest score and the trunk group with

Tab. 1. Mean value of ϵ for each dataset.

Dataset	Mean value of ϵ
Hajnówka	0.286
Jedwabno	0.233
Nowe Ramuki	0.235
R_ilaw	0.217
R_lidz	0.245
Wejherowo	0.201
Wipsowo	0.237

the highest score. When analyzing the above graphs, it can be seen that in five out of seven cases, the sets of the trunk-forming and noise groups are approximately linearly separable.

7.3. Analysis of results of the detection algorithm

A summary of all results is presented in the Supplementary Materials (in the file `DetailedResults.xlsx` [29]). A satisfactory level of specificity was achieved in each of the 72 cases, i.e. the percentage of noise groups that were rejected. The specificity of the algorithms ranges from 96.57% to 98.95%, so the difference between the best and worst algorithms is less than 2.4 percentage points (sheet `Detection_Results` in `DetailedResults.xlsx`.).

A much more pronounced difference is for sensitivity, which should be interpreted as the percentage of groups that form the trunk that are not removed by the algorithm. The worst algorithm, which turned out to be a combination of ranked roulette, arithmetic cross, arithmetic mutation, and the Hesser-Manner mutation policy, was only 67.19%. For comparison, the best algorithm, the classic two-player tournament, one-point crossover at the bit level and a constant probability of mutation involving the negation of a random bit, achieved a sensitivity of 84.43% with a specificity of 96.58%. This gives almost the lowest precision in understanding the error matrix, but at the same time the highest negative predictive value. Precision should be interpreted as the probability that the groups classified as constituting the trunk actually constitute the trunk. Likewise, a negative predictive value represents the probability that only noise is discarded.

Of course, in the case of the worst algorithm, more than $2/3$ of the information about the tree has been preserved, which allows quite effective use of linear regression on the basis of groups that are correctly approximated at the approximation stage. However, low sensitivity is also associated with a higher probability of losing information about higher parts of the tree, which are very important for obtaining a reliable tree model – without them, linear regression may be biased with greater errors.

The analysis of the computation time shows some valuable facts. First, the mean standard deviation for the algorithms does not exceed 0.1 s, so it is reasonable to draw conclusions from the attached data. Secondly, the selection of the algorithm can significantly affect the results with a limited time – the best algorithm is able to remove noise from the tree scan on average in 0.12 s, and the worst one takes as long as 20.37 s for the same task. One can also see a very clear trend. Back's mutation policy causes even a hundredfold increase in computation time in relation to the constant mutation probability. The second important fact is that the fastest algorithms are those that use value roulette, and the slowest are the tournaments. This is due to the optimization of roulette calculations and the high computational complexity of the evaluation function. Value roulette optimization is based on recording the value of the evaluation function of

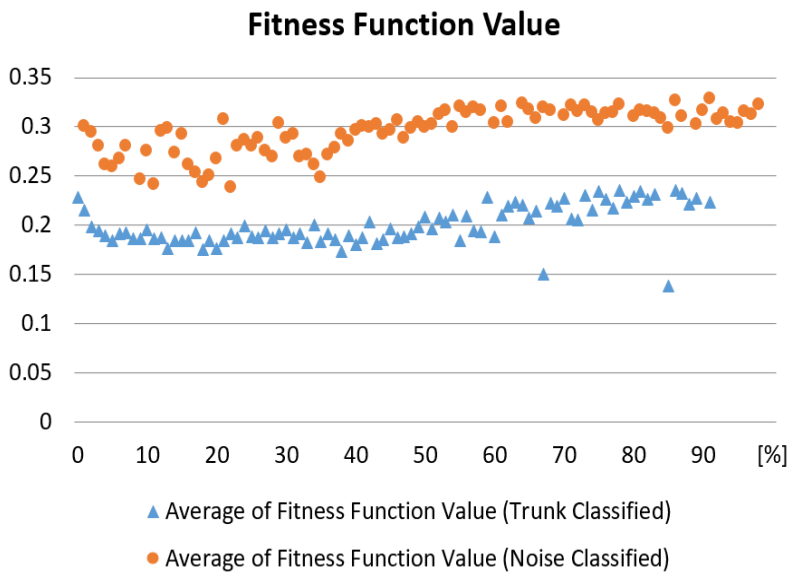


Fig. 4. Results of the best chosen algorithm called `Tournment-SinglePointCrossover-ArithmeticMutation-HesserMannerMutation-0`.

all elements every generation. As a result, the evaluation of the value of the evaluation function takes place exactly once per individual in a given generation.

Taking into account both factors – negative predictive value and average computation time – the most advantageous algorithm can be selected (`DetailedResults.xlsx` [29]). The quotient of the negative predictive value and the average calculation time shows the real profitability of using a given algorithm in a limited time. It turns out that the most effective way is to use a tournament with a single or even crossing. The method of mutation and the policy do not have much influence (except for the previously described Back mutation policy). The use of a tournament or ranked roulette is almost seven times more profitable than the use of value roulette.

The Figure 4 shows the values at which the fitness function is maintained. The difference between the groups classified as trunk and noise is large and remains stable regardless of height. The adopted threshold value of $\epsilon = 0.236214$ works very well at any height. However, it can be noticed that after exceeding 60% of the tree height, the values start to increase slightly. Increasing the threshold value to 0.25 from 60% of the tree height could positively affect the sensitivity of the algorithm. However, the current results are so satisfactory that no such test was performed.

This stage of the experiments was a definite success. The analysis showed that the noise filtering efficiency exceeds 95% while maintaining up to 84% information about the trunk. Taking into account the operating time of the algorithms, one can select one algorithm that is the fastest and the most effective and should be used as the basic one in this task. This algorithm uses a two-player tournament, binary single point crossing with an arithmetic mutation and a Hesser-Manner crossing policy.

7.4. Approximation algorithm

Due to the very low performance of Back's algorithm, we decided to exclude it from further testing. For this reason, 48 combinations will be compared instead of 72.

Analyzing results with an accuracy of 1 m of the tree does not make sense, because in practical applications the accuracy of estimating the volume of the tree as a whole matters. Therefore, we decided to present only generalized results. Moreover, in the analysis we did not perform approximation for the trunk height below or equal 1 m. Calculation close to the ground can cause large errors due to the influence of roots. The most important information is the measurement of DBH, i.e. the diameter of the tree at a height of 1.3 m, which is included in the analysis. In Fig. 5 we are presenting graphs for the `Tournament-UniformCrossover-ArithmeticMutation-HesserMannerMutation-0` algorithm from *ForestTaxator*.

As can be seen in the charts, the error usually does not exceed 5 cm. The graphs show a tendency to overestimate the trunk diameter in the lower parts of the tree and underestimate it in the higher parts. This is directly due to the dendrometric formula used in linear regression:

$$y^2 = px^r\,, \tag{10}$$

where y is the diameter of the tree, x is the distance from the top of the tree, p and r are parameters whose values vary from tree to tree. The formula in the above form is not linear, but it can be reduced to a linear form by taking a logarithm of the sides:

$$2\ln(y) = r\ln(x) + \ln(p)\,. \tag{11}$$

Since the value of x depends very much on knowledge of the tree's height, failure to do so results in an error in which the tree narrows at a different rate than it should. The lack of information about the height of the tree is inevitable with a laser scan. For this reason, this type of error will occur in any technique that uses this pattern.

It is also worth paying attention to the jump that occurs at the turn of 19 and 20 meters in the "Wejherowo" dataset. Such a jump from a negative value to a positive value means that the algorithm did not have even a single point above the height of 19.5 m that would allow to approximate the height of the tree more precisely and the diameter of the tree was approximated to the height of 19.5 m only. For the height above, 0 is

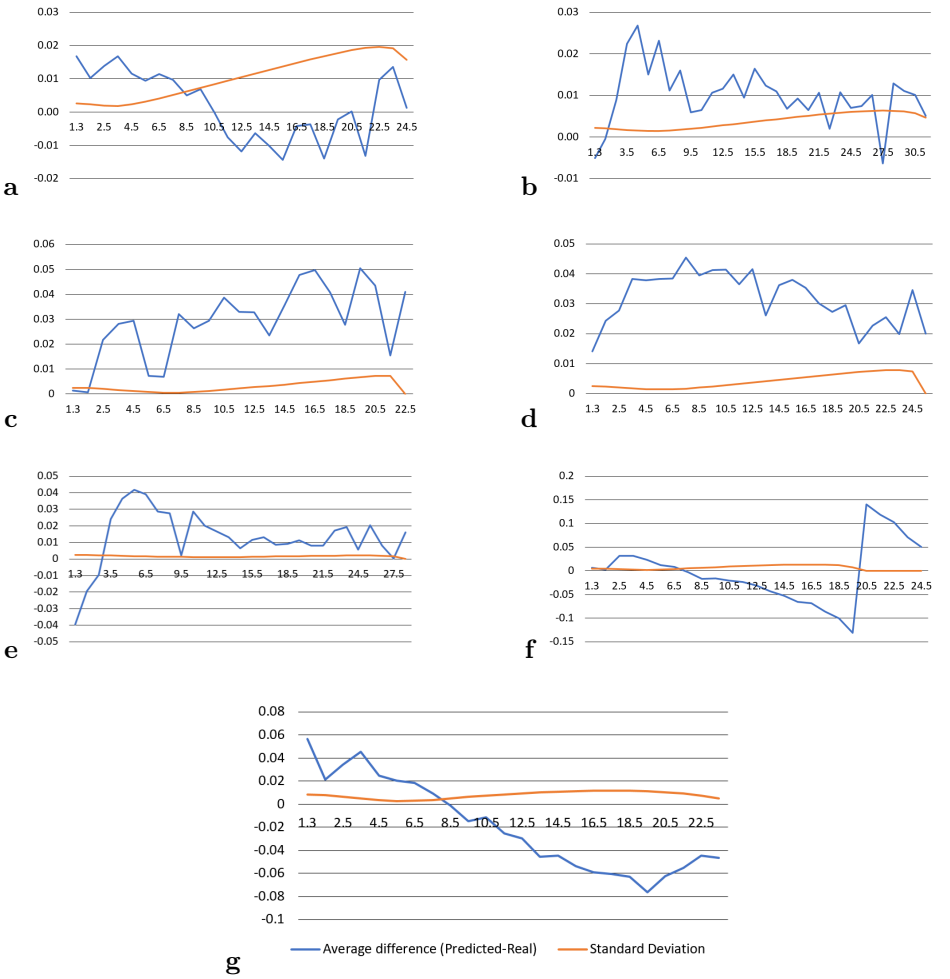


Fig. 5. Average estimation error graphs for datasets: (**a**) "Hajnówka", (**b**) "Jedwabno", (**c**) "Nowe Ramuki", (**d**) "R_ilaw", (**e**) "R_lidz", (**f**) "Wejherowo", (**g**) "Wipsowo". Legend shown below Fig. **g**.

returned. Positive values are therefore the tree diameter values that were given in the reference data.

When delving into the comparison of the results of the algorithms with the division into the measurement height, it can be noticed that the standard deviation for all algorithms does not exceed 2 cm and rather remains below 1 cm. This is illustrated in the table in `DetailedResults.xlsx`. As can be seen, the standard deviation for all heights,

except for the highest parts of trees, remains within a single millimeter, regardless of the algorithm used. This shows that, in fact, none of the tested parameters significantly affects the results obtained, which can be described as very good, considering that the error rarely exceeds the value of 1.5 cm, which is a value defined as good enough.

The final comparison of the algorithms takes into account the average results for all trees as a whole. These results are summarized in the table in the Supplementary Materials `DetailedResults.xlsx`. The table clearly shows that for the effectiveness of this algorithm, the choice of the selection, crossing and mutation methods used is irrelevant. The best algorithm achieves an average error of 1.23 cm, while the worst one does so with an error of 1.68 cm. Of course, these values differ by almost 0.5 cm, but it should be noted that the standard deviation for the error also reaches values close to 1.5 cm, so more than 3 times the difference between the best and worst algorithm.

In Fig. 6 there are pictures containing ellipses representing a given tree superimposed on the point cloud with the original data. These ellipses have been marked in blue and the points have been enlarged in CloudCompare to make them easier to see.

7.5. Discussion

Several conclusions can be drawn from the conducted experiments. The experiments carried out on the detection algorithm show that in some cases the choice of crossing and selection methods can significantly affect the quality of the obtained solution. The classic solution in the form of tournament selection with two individuals, one-point bit crossing combined with arithmetic mutation and the Hesser-Manner crossing policy gave the best results in the shortest time.

The tree detection experiment also showed a significant weakness in Back's mutation policy. Its performance is very dependent on the computational complexity of the fitness function, or at best, of the individuals comparison function. Each determination of the mutation probability requires finding the best individual in the population, which involves additional operations and a high time cost.

In the case of the second experiment, one of the most important features of genetic algorithms was shown – their resistance to deficiencies and errors in the input data. Despite the lack of several meters of tree in one set and errors in the preparation of another set, the algorithm did very well, reaching an accuracy of 2 cm. An important fact is that in this case the type of selection, crossing and mutation methods used did not have any significant impact on the quality of the solutions obtained. The differences were further diminished by the operation of linear regression, which completed the missing information and smoothed the entire tree model.

In both experiments, the genetic algorithms worked very well, achieving great results and satisfactory operation time.

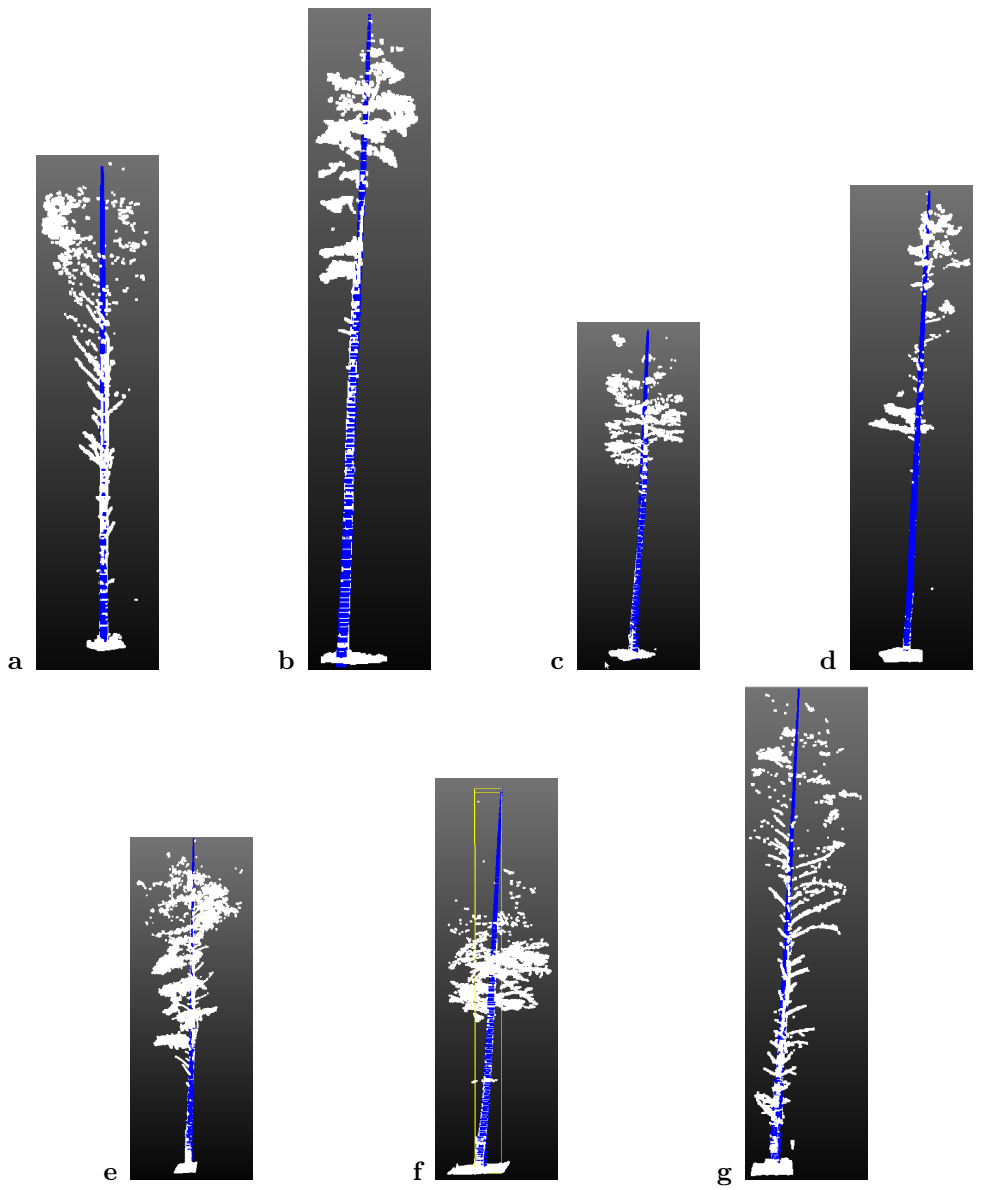


Fig. 6. Average estimation error graphs for the datasets: (**a**) "Hajnówka", (**b**) "Jedwabno", (**c**) "Nowe Ramuki", (**d**) "R_ilaw", (**e**) "R_lidz", (**f**) "Wejherowo", (**g**) "Wipsowo". See text for explanations.

8. Impact

ForestTaxator represents a novel software for the forest inventory. With its publication, *ForestTaxator* becomes freely available to the scientific community and business practitioners. The impact of the software on the community is significant. It addresses the gap in the existing software. For the benefit of users the Table 2 summarizes a comparison between *ForestTaxator* and other free software. In general, there are five main functionalities that are important in this field.

First is a digital elevation model (DEM) and digital terrain model (DTM). The digital elevation model is a 3D representation of the terrain elevations found on the earth's surface. DEMs are generated from variably-spaced Lidar ground points, or they can be created using a raster grid. The DTM is a DEM in which terrain data have been further enhanced by including vector features of the natural terrain, such as rivers and ridges, providing for greater accuracy as it contains additional information defining the terrain in areas where Lidar data alone are insufficient to do the job effectively.

The second functionality allows to detect the stem of the tree. Third functionality classifies the point cloud into woody and soft parts, i.e., leaves. Forth functionality provides tree parameters, such as: diameter at breast height, tree height, tree taper curve or tree volume.

Finally, our software reconstructs the quantitative structure models (QSMs) of trees from point clouds. A QSM consists of a hierarchical collection of cylinders which estimate topological, geometrical and volumetric details of the woody structure of the tree. The input point cloud, which is usually produced by a terrestrial laser scanner, must contain only one tree, but the point cloud may contain also some points from the ground and understorey.

As shown in the Table 2, *ForestTaxator* possesses a number of advantages as compared to other software. The attractiveness of this software is due to the fact that it provides all necessary functionalities used in the forest inventory field. *ForestTaxator* is freely available to the scientific community and for commercial purposes. We believe the impact of the software on the community will be significant.

9. Conclusions

Today's technical possibilities make it possible to collect data about the environment in the form of three-dimensional scans containing millions of points describing the structure of the scanned objects. There are various methods of performing a scan, including the LiDAR-based methods. Three-dimensional scans are used in many social and scientific fields as well as in civil engineering. In the case of large-area or high-resolution scans, the size of the data may necessitate the use of an appropriate approach to reduce memory

Tab. 2. Comparison between *ForestTaxator* and other free software used for the forest inventory. Standalone means standalone application; DTM – DTM extraction; Stem – tree stem detection; Wood/leaf – point cloud classification into wood/leaf components; Params – Basic tree parameters extraction; QSM – QSM extraction.

Name [reference]	Platform	DTM	Stem	Wood/leaf	Params	QSM
DendroCloud [1]	standalone	+	+	−	+	−
3DForest [35]	standalone	+	+	−	+	+
Computree [34]	standalone	+	+	−	+	+
Simple-Forest [33]	standalone	+	+	−	+	+
AdTree [36]	standalone	−	−	−	+	+
SSSC [37]	standalone	+	+	+	−	−
TreeQSM [11]	Matlab	−	−	−	+	+
TreeLS [38]	R	−	+	−	+	−
TLSeparation [25]	Python	−	−	+	−	−
ForestTaxator	C#	+	+	+	+	+

usage or speed up the search of the set. A scanner using LiDAR is characterized by very high precision, short scanning time and a long range of tens or even hundreds of meters.

Genetic algorithms are a type of heuristic that works well where no other methods are known. They use terminology borrowed from biology, and more specifically from the field of species evolution. The advantage of these algorithms is their great versatility. One basic scheme is always used, which can be extended with additional elements. Another advantage of genetic algorithms is their susceptibility to parallelization. The most illustrative case is the use of multiple populations that may exchange from time to time with a small number of individuals to increase genetic diversity.

We described in detail an attempt to use a 3D point cloud and genetic algorithms to detect and approximate the diameter of the cross-sections of trees. The work covers all stages of data analysis – from loading them into memory to creating the tree model. The work proved that the use of hybrid genetic algorithms for automatic inventory of trees in the forest makes sense – the obtained results were consistent with the reference data to a large extent, while the time of genetic calculations remained very short.

References

[1] J. Elseberg, D. Borrmann, and A. Nuchter. Efficient processing of large 3D point clouds. In *Proc. XXIII Int. Symp. Information, Communication and Automation Technologies ICAT 2011*, pages 1–7, Sarajevo, Bosnia Herzegovina, 27-29 Oct 2011. doi:10.1109/icat.2011.6102102.

[2] G. Krok, B. Kraszewski, and K. Stereńczak. Application of terrestrial laser scanning in forest inventory – an overview of selected issues. *Forest Research Papers*, 81(4):175–194, 2020. doi:10.2478/frp-2020-0021.

[3] A. Bienert, S. Scheller, E. Keane, et al. Tree detection and diameter estimations by analysis of forest terrestrial laserscanner point clouds. In *Proc. Workshop Laser Scanning and SilviLaser 2007 ISPRS 2007*, pages 50–55, Espoo, Finland, 12-14 Sep 2007. `https://www.isprs.org/proceedings/xxxvi/3-w52/final_papers/Bienert_2007.pdf`.

[4] A. Konieczny, and B. Neroj. Projekt działania programu do obliczania miąższości drzew na podstawie danych skanowania naziemnego (TLS). Presentation from "Narada Koordynatorów SIP", Zakopane, 23–25 Feb 2016. `https://www.geomatyka.lasy.gov.pl/documents/25999395/0/Konieczny-TLS.pdf/b13219cc-1608-4004-8692-2de4d0d44a5e`. [Online; accessed 12 Nov 2020].

[5] M. Małaszek. ForestTaxator library. `https://github.com/maciej-malaszek/forest-taxator`. [Online; accessed 10 Dec 2020].

[6] X. Liang, V. Kankare J. Hyyppä, et al. Terrestrial laser scanning in forest inventories. *ISPRS Journal of Photogrammetry and Remote Sensing*, 115:63–77, 2016. doi:10.1016/j.isprsjprs.2016.01.006.

[7] S. Bauwens, H. Bartholomeus, K. Calders, and P. Lejeune. Forest inventory with terrestrial LiDAR: A comparison of static and hand-held mobile laser scanning. *Forests*, 7(12):127, 2016. doi:10.3390/f7060127.

[8] P. Wezyk, K. Koziol, M. Glista, and M. Pierzchalski. Terrestrial laser scanning versus traditional forest inventory first results from the Polish forests. In *Proc. Workshop Laser Scanning and SilviLaser 2007 ISPRS 2007*, pages 12–14, Espoo, Finland, 12-14 Sep 2007. `http://foto.hut.fi/ls2007/posters/Wezyk_ls2007_poster.pdf`

[9] M. Zasada, K. Stereńczak, W. M. Dudek, and A. Rybski. Horizon visibility and accuracy of stocking determination on circular sample plots using automated remote measurement techniques. *Forest Ecology and Management*, 302:171–177, 2013. doi:10.1016/j.foreco.2013.03.041.

[10] R. Astrup, M. J. Ducey, A. Granhus, et al. Approaches for estimating stand-level volume using terrestrial laser scanning in a single-scan mode. *Canadian Journal of Forest Research*, 44(6):666–676, 2014. doi:10.1139/cjfr-2013-0535.

[11] P. Raumonen, M. Kaasalainen, M. Åkerblom, et al. Fast automatic precision tree models from terrestrial laser scanner data. *Remote Sensing*, 5(2):491–520, 2013. doi:10.3390/rs5020491.

[12] P. Wilkes, A. Lau, M. Disney, et al. Data acquisition considerations for Terrestrial Laser Scanning of forest plots. *Remote Sensing of Environment*, 196:153–520, 2017. doi:10.1016/j.rse.2017.04.030.

[13] A. M. Kim, R. C. Olsen, and M. Béland. Simulated full-waveform lidar compared to Riegl VZ-400 terrestrial laser scans. *Laser Radar Technology and Applications XXI*,9832:242–255, 2016. doi:10.1117/12.2223929.

[14] M. T. Vaaja, J. P. Virtanen, M. Kurkela, et al. The effect of wind on tree steam parameter estimation using terrestrial laser scanning. *International Society for Photogrammetry and Remote Sensing Workshop on Laser Scanning*, III-8:117–122, 2016. doi:10.5194/isprsannals-iii-8-117-2016.

[15] D. Seidel, S. Fleck, and C. Leuschner. Analyzing forest canopies with ground-based laser scanning: A comparison with hemispherical photography. *Agricultural and Forest Meteorology*, 154:1–8, 2012. doi:10.1016/j.agrformet.2011.10.006.

[16] M. Dassot, T. Constant, and M. Fournier. The use of terrestrial LiDAR technology in forest science: Application fields, benefits and challenges. *Annals of Forest Science*, 68(5):959–974, 2011. doi:10.1007/s13595-011-0102-2.

[17] L. J. Chmielewski, M. Bator, M. Zasada, et al. Fuzzy Hough transform-based methods for extraction and measurements of single trees in large-volume 3d terrestrial LIDAR data. *In Computer Vision and Graphics: Proc. ICCVG 2010*, Part I, volume 6374 of *Lecture Notes in Computer Science*, pages 265—274, Warsaw, Poland, 20-22 Sep 2010. Springer. doi:10.1007/978-3-642-15910-7_30.

[18] E. Lindberg, J. Holmgren, K. Olofsson, and H. Olsson. Estimation of stem attributes using a combination of terrestrial and airborne laser scanning. *European Journal of Forest Research*, 131(6):1917–1931, 2012. doi:10.1007/s10342-012-0642-5.

[19] W. Zhang, P. Wan, T. Wang, et al. A novel approach for the detection of standing tree stems from plot-level Terrestrial Laser Scanning data. *Remote Sensing*, 11(2):211, 2019. doi:10.3390/rs11020211.

[20] A. T. Albrecht, M. Fortin, U. Kohnle, and F. Ningre. Coupling a tree growth model with storm damage modeling–conceptual approach and results of scenario simulations. *Environmental Modelling & Software*, 69:63–76, 2015. doi:10.1016/j.envsoft.2015.03.004.

[21] D. E. Goldberg. *Genetic Algorithms*. Pearson Education India, 2013.

[22] T. Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, 1996.

[23] A. Burt, M. Disney, and K. Calders. Extracting individual trees from lidar point clouds using *treeseg*. *Methods in Ecology and Evolution*, 10(3):438–445, 2018. doi:10.1111/2041-210x.13121.

[24] K. Olofsson, and J. Holmgren. Single tree stem profile detection using Terrestrial Laser Scanner data, flatness saliency features and curvature properties. *Forests*, 7(12):207, 2016. doi:10.3390/f7090207.

[25] M. B. Vicari, M. Disney, P. Wilkes, et al. Leaf and wood classification framework for terrestrial LiDAR point clouds. *Methods in Ecology and Evolution*, 10(5):680–690, 2019. doi:10.1111/2041-210x.13144.

[26] H. Yoon, H. Song, and K. Park. A phase-shift laser scanner based on a time-counting method for high linearity performance. *Review of Scientific Instruments*, 82(7):075108, 2011. doi:10.1063/1.3600456.

[27] J. D. Schaffer, R. Caruana, L. J. Eshelman, and R. Das. A study of control parameters affecting online performance of genetic algorithms for function optimization. In Schaffer J. D. (ed.), *Proc. 3rd Int. Conf. Genetic Algorithms*, pages 51–60, Morgan Kaufmann Publishers Inc., San Francisco, CA, 1989.

[28] K. Stereńczak, S. Miścicki, M. Zasada et al. Project *Remote sensing based assessment of woody biomass and carbon storage in forests (REMBIOFOR)*. Funded by the Polish National Centre for Research and Development (NCBiR) within the program Natural Environment, Agriculture and Forestry BIO-STRATEG 2015, under the agreement no. BIOSTRATEG1/267755/4/NCBR/2015. `http://rembiofor.pl/en/305-2/`. [Online; accessed 10 Dec 2020].

[29] M. Małaszek, A. Zembrzuski, and K. Gajowniczek. Supplementary Materials to this paper – file `DetailedResults.xlsx`. Published together with this paper. doi:10.22630/MGV.2022.31.1.2.

[30] M. Małaszek. GeneticToolkit library. `https://github.com/maciej-malaszek/GeneticToolkit`. [Online; accessed 10 Dec 2020].

[31] CloudCompare. 3D point cloud and mesh processing software. Open Source Project. `https://www.danielgm.net/cc/`. [Online; accessed 10 Feb 2021].

[32] T. de Conto, Jean-Romain, C. Hamamura and A. Marcozzi. TreeLS. `https://github.com/tiagodc/TreeLS`. [Online; accessed 10 Feb 2021].

[33] J. Hackenberg, H. Spiecker, K. Calders, et al. SimpleTree—An efficient open source tool to build tree models from TLS clouds. *Forests*, 6(11):4245–4294, 2015. doi:10.3390/f6114245.

[34] J. Trochta, M. Krůček, T. Vrška, and K. Král. 3D Forest: An application for descriptions of three-dimensional forest structures using terrestrial LiDAR. *PLOS ONE*, 12(5):e0176871, 2017. doi:10.1371/journal.pone.0176871.

[35] A. Othmani, L. F. C. Lew Yan Voon, C. Stolz, and A. Piboule. Single tree species classification from Terrestrial Laser Scanning data for forest inventory. *Pattern Recognition Letters*, 34(16):2144–2150, 2017. doi:10.1016/j.patrec.2013.08.004.

[36] S. Du, R. Lindenbergh, H. Ledoux, et al. AdTree: Accurate, detailed, and automatic modelling of laser-scanned trees. *Remote Sensing*, 11(18):2074, 2019. doi:10.3390/rs11182074.

[37] D. Wang. Unsupervised semantic and instance segmentation of forest point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 165:86–97, 2020. doi:10.1016/j.isprsjprs.2020.04.0.

[38] T. De Conto, K. Olofsson, E. B. Görgens, et al. Performance of stem denoising and stem modelling algorithms on single tree point clouds from terrestrial laser scanning. *Computers and Electronics in Agriculture*, 143:165–176, 2017. doi:10.1016/j.compag.2017.10.019.

Attention-Based Deep Learning Model for Arabic Handwritten Text Recognition

Takwa Ben Aïcha Gader, Afef Kacem Echi
University of Tunis, ENSIT-LaTICE, Tunis, Tunisia
takwa.ben.aichaa@gmail.com, ff.kacem@gmail.com

Abstract. This work proposes a segmentation-free approach to Arabic Handwritten Text Recognition (AHTR): an attention-based Convolutional Neural Network – Recurrent Neural Network – Connectionist Temporal Classification (CNN-RNN-CTC) deep learning architecture. The model receives as input an image and provides, through a CNN, a sequence of essential features, which are transferred to an Attention-based Bidirectional Long Short-Term Memory Network (BLSTM). The BLSTM gives features sequence in order, and the attention mechanism allows the selection of relevant information from the features sequences. The selected information is then fed to the CTC, enabling the loss calculation and the transcription prediction. The contribution lies in extending the CNN by dropout layers, batch normalization, and dropout regularization parameters to prevent over-fitting. The output of the RNN block is passed through an attention mechanism to utilize the most relevant parts of the input sequence in a flexible manner. This solution enhances previous methods by improving the CNN speed and performance and controlling over model over-fitting. The proposed system achieves the best accuracy of 97.1% for the IFN-ENIT Arabic script database, which competes with the current state-of-the-art. It was also tested for the modern English handwriting of the IAM database, and the Character Error Rate of 2.9% is attained, which confirms the model's script independence.

Key words: Arabic handwriting recognition, attention mechanism, BLSTM, CNN, CTC, RNN.

1. Introduction

Handwriting recognition is one of the critical research areas of Optical Character Recognition (OCR)[1], which consists of converting text on images into machine-encoded text. This step is vital, considering the importance of digitization in nowadays life. Handwriting recognition is employed in various domains such as document processing [25], writer identification, office automation, signature verification [5] automatic cheque processing in banks [26], postal code recognition [9], etc.

In this work, we present a segmentation-free approach to address the problem of Arabic Handwritten Text Recognition (AHTR). It is crucial since the Arabic alphabet is the second most widely used alphabetic writing system globally, and 234 million people speak Arabic. Researchers obtained very encouraging results in machine-printed Arabic text recognition because of the text's homogeneity, such as within and across word spaces, spacing within successive text lines, and character sizes. Oppositely, handwritten Arabic recognition is classified as a complex task for many reasons: the multiple writing styles, the Arabic script's vast variability, the use of ligatures and diacritics, the cursive

[1] All the abbreviations are explained in Table 6 at the end of the paper, p. 67.

nature where alphabets are written in a joint flowing style which may cause touched and overlapped characters (see Figure 1, letters with the same shapes, such as ب, ت and ث, which can only be distinguished by dots figured under or above the base alphabet, the change of the same character's shape according to its position in the word (see Figure 2) and the absence of significant available Arabic databases.

Deep Learning-based models have been the basis of most Computer Vision tasks [7, 48], performing higher performances than other state-of-the-art approaches. More particularly, models based on neural networks have given exciting results in handwriting recognition. As it has proven its success in this area, many works based on the architecture Convolutional Neural Network – Recurrent Neural Network – Connectionist Temporal Classification (CNN-RNN-CTC) have been proposed to recognize cursive scripts such as Arabic, Parsi, and Latin to avoid segmenting words into characters or isolating merged characters.

In the mentioned architecture, the text image is sent through convolutional layers to get the features from the image. Later these features are fed to recurrent neural network architecture, which outputs softmax probabilities over the vocabulary. These outputs from different time steps are fed to the CTC decoder to get the raw text from images. Note that CTC is designed for tasks where we need alignment between sequences but where that alignment is difficult. We used it to align each character to its location in the text. By just mapping the image to text and not worrying about each character's alignment to the input image's location, one should calculate the loss and train the network.

With this in mind, we tried to shed new light on previous works and propose an efficient AHTR system that uses a CNN, a RNN and a CTC block. Of particular interest, our system has three vital points. First, the CNN is extended by dropout layers. Second, to extract features from input images, the CNN employs batch normalization and dropout regularization parameters to prevent over-fitting and to improve system performance. Third, the output of the RNN block is passed through an Attention mechanism. This allows the essential information to be selected from the feature sequence resulting from the RNN and then transmitted to the CTC block. Thus, to recognize a text, the system does not segment it into words or characters in advance; rather, it recognizes the input text by extracting a feature map from the input image using a CNN and transfers it to the RNN layer, where an attention-based Bidirectional Long Short-Term Memory Network (BLSTM) with CTC is applied for sequence labeling. This solution enhances previous methods by improving the model speed and performance and controlling model over-fitting. From the functional point of view, the user simply performs the training of the network by just mapping the image to text, without worrying about each character's alignment to the location in the input, and the system calculates the loss and presents the results. We will discuss each of the steps in the further sections of the paper.

Fig. 1. Some Arabic script writing characteristics. Created by the Authors of [6]; reused under the CC BY 4.0 license.

Isolated form	Contextual forms		
	Final	Medial	Initial
ب	ـب	ـبـ	بـ
س	ـس	ـسـ	سـ
ض	ـض	ـضـ	ضـ
ه	ـه	ـهـ	هـ
ئ	ـئ	ـئـ	ئـ
غ	ـغ	ـغـ	غـ

Fig. 2. Contextual forms of some Arabic letters.

The paper is organized as follows. Section 2 briefly reviews related works, and section 3 details the proposed models and the system's architecture. In section 4, we present the training process and discuss the obtained results.

2. Related Works

The recent progress in deep learning technology influenced the Arabic text recognition problem, where several deep learning-based approaches were proposed. The first deep learning-based approach for AHTR in images was introduced in 2008 by Graves and Schmidhuber [33]. The authors used a Multi-Dimensional Long Short Term Memory (MDLSTM) network and the CTC. The proposed model was evaluated on the IFN/ENIT dataset [57] and reached an accuracy of 91.4%. The same model was used in 2013 by Rashid et al. [60], where the MDLSTM was used on the input images to extract features fed to the CTC layer (120 units), which allows data labeling. The proposed method reached a recognition rate of 99% on the Arabic Printed Text Image database (APTI) [63]. In the same year, Chherawala et al. in [19] used the MDLSTM model on

some handcrafted features and raw pixels extracted from IFN/ENIT, and they achieved an accuracy of 88.8%.

Continuing the study of the deep-learning-based approaches evolution, we mention the one proposed by Abandah et al. in 2014 [1]. It is based on the segmentation of cursive words into graphemes. A features vector is extracted and passed to a BLSTM, which transcript sequences by graphemes exploiting. In 2016, Ahmad et al. [2] used the three variants, LSTM, BLSTM, and MDLSTM, for Pashto (which uses the Arabic alphabet) handwritten text recognition. Their study offers the best performance using the MDLSTM model on the KPTI database with an error rate of 9.22%.

Meanwhile, Elleuch et al. [29] introduced a handwritten character recognition approach based on a CNN-SVM architecture. The CNN is used for feature extraction, and the SVM with Radial Basis Function (RBF) kernel for classification. The authors improved the results by adding the dropout technique, which temporarily removes some units from the network to prevent the system from over-fitting problems. The preformance of the system was evaluated on three datasets: HACDB [49] with 24 classes, HACDB with 66 classes, and IFN/ENIT with 56 classes, and error rates achieved were 2.09%, 5.83%, and 7.05%, respectively. During the same year, an RNL-based MDLSTM and dropout were introduced by Maalej et al. [52].

In 2017, Chen et al. [17] presented a segmentation-free approach of RNN with a four-layer bidirectional Gated Recurrent Unit (GRU) network with a CTC output layer and combined it with the dropout technique. The authors evaluated the system performance on the IFN/ENIT database with the "abcd-e" scenario. Accuracy of 86.4% was reached. El-Sawy et al. [27] proposed a CNN-based in-depth learning architecture for Arabic handwriting character recognition. The authors applied an optimization process that increased the model performance. However, this method's weak point was its incapacity to manage significant inputs and share their weights.

The same year, Ahmad et al. in [3] presented an MDLST-CTC architecture for recognizing Arabic handwritten text. They used data augmentation to improve the model performances and reached a CRR level of 80.02% on the KHATT dataset. Among the recent works, we cite the one proposed in 2018 by M. Amrouch et al. [8]. It is a CNN-based HMM model, where CNN is a prominent feature extractor, and the Hidden Markov Model (HMM) baseline system is a recognizer. The model was validated using the two IFN/ENIT database scenarios, "abc-d" and "abcd-e". It reached a recognition accuracy of 88.95% and 89.23%, respectively.

Continuing with the progress of Handwritten Arabic recognition, we mention two works presented in 2019. The first one [28] is a Convolutional Deep Belief Network (CDBN) framework proposed to recognize low/high-level dimensional data. The authors used data augmentation and a dropout regularization to increase the model's performance and avoid over-fitting. The model was first evaluated on the HACDB characters database and achieved an accuracy rate of 98.86%. Moreover, second, on the IFN/ENIT

words database, it reached an accuracy of 92.9%. The second one presented in [56] was designed to recognize an image of Arabic text/characters. The introduced model takes a single line of Arabic text and segments it into words and letters. The trained model recognizes these image fragments as characters. The model evaluation was done on a custom dataset, and reached a CRR of 83%. In 2020, authors in [4] proposed a deep learning-based approach for Arabic text extraction. They used preprocessing, including pruning of extra white spaces plus de-skewing the skewed text lines. Furthermore, they trained the proposed MDLSTM-CTC model on the KHATT database with data augmentation. They achieved a Character Recognition rate (CRR) of 80.02%. In the same year, authors in [61] introduced a CNN-RNN-based model for word recognition. They worked on Persian handwritten text, which has the same properties as Arabic. Their main improvement was not using the segmentation step. In [6], a supervised Deep CNN (DCNN) model was used to address the challenges of recognizing offline handwritten Arabic text, including isolated digits, characters, and words. The model reached an accuracy of 99.95% on the SUST-ALT database.

To clotureclose our related works section, we mention the work [10], where authors proposed an approach based on sequentially transferring the mid-level word image representations through two consecutive phases strategy based on transfer learning. They used the ResNet18 model that has been pre-trained on the ImageNet dataset. They achieved a recognition accuracy of 96.11% on the IFN/ENIT database.

The related works have been synthetically compared in Tab. 1.

3. Proposed System

This section will introduce the proposed deep neural network for AHTR inspired by the work in [62]. It comprises three principal end-to-end parts: a CNN, and an RNN, followed by a CTC. This combination is the best choice as it currently outperforms all other approaches. The CNN is used for sequence feature extraction from the input images. Furthermore, the RNN is used to propagate information within this sequence. For each sequence element, it outputs a matrix of character scores. The CTC operation is set up to calculate the loss value to train the proposed model and to perform the inference at this stage. The CTC decodes the RNN's output matrix to infer the text contained in the input image. These two associated networks with the CTC make word-level recognition possible without character-level segmentation. Figure 3 shows an overview of the AHTR system and Figure 4 details it. The proposed neural network (NN) can be described by a mathematical formula (1) that performs the mapping between an image I and a character sequence S:

$$\mathrm{NN} : I_{W \times H} \rightarrow (c_1, c_2, ..., c_n)_{0 \leq n \leq L} \ , \tag{1}$$

Tab. 1. Comparison of the state-of-the-art methods.

Year	Method	Advantages	Limits
2008	[33]	The general system applied for Arabic as English. The system works directly on raw pixel data and requires minimal changes for languages with different alphabets.	When applying the model to other languages, the dimensionality of the networks should be modified to match the data.
2013	[60]	The method performs very well on printed Arabic text recognition, even for very low resolution and small font size images.	It is not evaluated for handwriting text.
2013	[19]	The MDLSTM can automatically learn features from the input image (automatically learned features).	Despite their ability to learn features, the architecture of MDLSTM networks can limit their performance, especially the amount of horizontal sub-sampling.
2014	[1]	We describe a robust rule-based segmentation algorithm that uses particular feature points identified in the word skeleton to segment the cursive words into graphemes.	Improper segmentation of the graphemes leads to a lousy extraction of the features, therefore an inadequate recognition.
2016	[2]	It is a comparison, not an approach that showed that MDLSTM achieved a good performance on their KPTI (Pashto language) database.	-
2016	[29]	They used a CNN-based-SVM model, which automatically extracts features from the raw images and performs classification.	A non-generic system. The proposed architecture must be extended to deal with handwritten words in different languages and enhance the recognition rate.
2016	[52]	Dropout.	-
2017	[17]	The use of GRU units and dropout.	-
2017	[27]	The use of an optimization process.	The method is incapable of managing significant inputs and sharing their weights.
2018	[8]	The ability to extract automatically salient features directly from raw pixels.	Low Recognition Rate compared with recent models.
2019	[56]	Using a preprocessing step to improve the quality of the images and to segment text lines to words and characters.	The recognition rate is low for letters with loops and those including dots.
2020	[4]	MDLSTM has the advantage of scanning the Arabic text lines in all directions. A pre-processing step includes pruning of extra white spaces plus de-skewing the skewed text lines.	Limited dataset.
2020	[61]	Using the advantages of both CNN and RNN for the word recognition purpose. The method benefits from CTC for eliminating the segmentation procedure.	-
2021	[6]	Using the Transfer Learning (TL)-based feature extraction. The approach can effectively deal with high-dimensional data by automatically and contextually extracting the best features. PGeneral model.	A non-generic system. The database has an insufficient number of training samples.
2021	[10]	Using the transfer learning	The model misclassifies words with similarities in shape and number of characters.

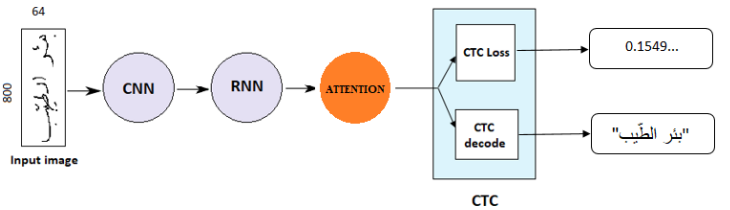


Fig. 3. The proposed AHTR pipeline.

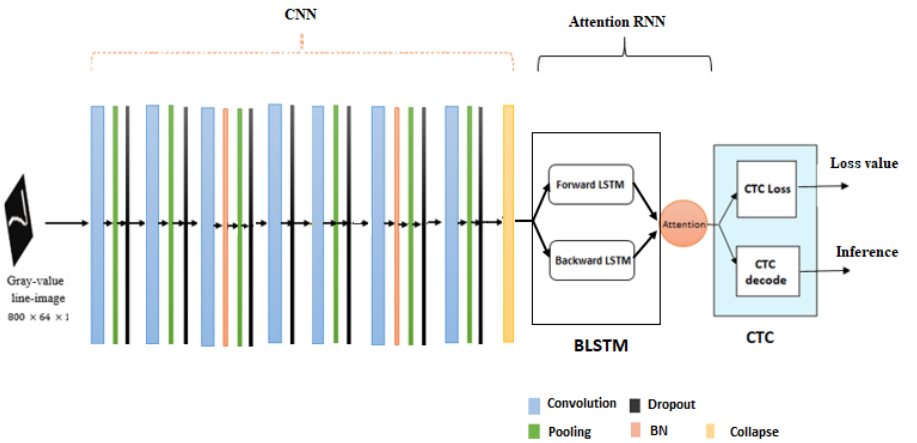


Fig. 4. The used Neural Network architecture for the text-line recognition.

where L is the max sequence length and $c_i, i \in \{1..L\}$ are the predicted characters. It transforms an image $I_{W \times H}$ to a sequence of characters $(c_1, c_2, ..., c_n)$ with a length L. As the recognition is done on a character level, the model can recognize text that does not belong to the training data. This is a strong point of the proposed model. We describe the model and the used dataset in the rest of this section.

3.1. CNN

CNNs are specific neural networks that apply convolution in place of general matrix multiplication in at least one of their layers. These networks have succeeded in several fields, such as automatic image classification, multi-object detection, object localization, handwritten digit recognition, and object classification. With this success, the application of CNNs in machine learning projects has increased drastically. Over time, several

approaches have been used to improve the performance of CNNs. We cite, for example, the development of computational systems, the design of regularization techniques such as the batch normalization and the dropout method, adding hidden layers, and the abundance of the training data. With all these stratagems, CNNs performance increased over time.

Recall that a CNN consists of an input layer, hidden layers, and an output layer sequentially connected. Each convolutional layer has input from the preceding layer convolved with trained filters. Hidden layers' inputs and outputs are masked by the activation function (generally the *Relu*) and the final layer's convolution. The convolution output can be followed by other layers, such as fully connected, pooling, and normalization layers. Every neuron in one layer is connected to another in a fully connected layer. The pooling operation reduces the risk of over-fitting. It minimizes the data size by combining the neuron clusters' outputs at one layer into a single neuron in the next layer. There are two well-used types of pooling: max pooling and average pooling. Taking the maximum value of each local neuron cluster in the feature map is max-pooling, and taking the average value is average pooling. While the Batch Normalization (BN) layer is added to a sequential model to standardize the input or the outputs, it provides each network layer to learn more independently. In normalization, the input layer is scaled by the activations. The normalization layer is usually set just after the convolution and pooling layers.

The first stage of our model is a CNN with seven layers (see Table 2). These layers are trained to select essential features from the input image. The CNN takes an image of size (800×64) and returns a features sequence of size (100×512). Each model's layer consists of a convolution (with 5×5 or 3×3 kernel) followed or not by one of the pooling (a 2×2 or 1×2 pooling) or BN operations. A dropout is added at the end of each layer to prevent over-fitting in the model. They are added to randomly and temporarily remove some percentage (at a given rate, see Table 2) of neurons and their connections.

3.2. Att-BLSTM

A Bidirectional Long Short-Term Memory Network (BLSTM) is a neural network model for training sequential data. It uses two isolated LSTMs, a forward LSTM reading the input sequence from left to right and a backward LSTM reading the sequence from right to left. The LSTM model was first proposed in [37] to defeat the gradient vanishing problem. The BLSTM model was introduced to get high-level features from the input features sequence. Further, the BLSTM networks extend the LSTMs by including a second layer, where the hidden-to-hidden connections flow in an opposite temporal order. The model is then able to manipulate past and future information.

In the proposed AHTR, we used an Attention-based BLSTM (see Figure 5) with 512 hidden cells for each LSTM. It employs a neural attention mechanism to extract relevant information from an input features sequence and explicitly includes the potential of

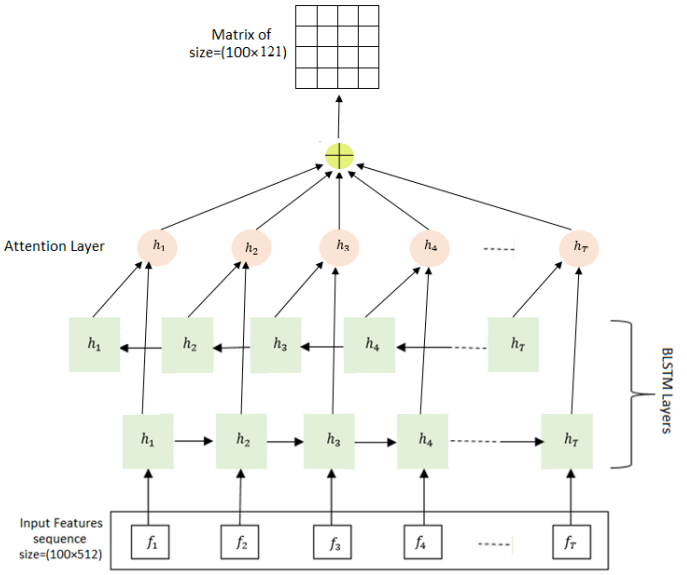


Fig. 5. An overview of the used Att-BLSTM architecture.

handling different writing styles. Attention-based networks have shown striking success in various deep learning domains. The attention mechanism is applied to an image to search in specific regions like a human when looking for a particular pattern, orientating himself to specific zones in the image.

The BLSTM includes two LSTMs, which are forward and backward, respectively. It takes as input a feature map with a size of (100×512), 512 features per time-step, and outputs a matrix consisting of combined LSTM output vectors of size (100×1024); 1024 for each time step. The attention block produces a weighted vector and multiplies features from each step by the latter. It gives an output sequence, which is mapped to a matrix of size (100×121), where 100 is the text line max length and 121 is the number of characters contained in the IFN/ENIT dataset considering the black character ('-'). The number 121 is also the number of classes C, so the length of the labeled axis in the CTC layer will be 121. Therefore there are 121 entries for each of the 100 time steps.

3.3. CTC

As mentioned before, our model comprises a CNN, an RNN, and a CTC. The CNN allows feature sequence extraction from input images, and the RNN propagates information through this sequence and produces a matrix of character scores for each sequence

element. The CTC is set up for sequence labeling without input segmentation. In other words, the CTC is a softmax layer, which produces probabilities corresponding to all the possible label alignments of the input sequence with length T, where T is the length of the probabilities sequence fed to the CTC function. for all steps. It interferes in two steps: 1) during training, it takes the Att-BLSTM output matrix and the ground truth text to calculate the loss value, and 2) while inference, where it takes just the output matrix and provides the predicted text with a max length of 100 characters. The CTC was first introduced in [32]. Let us detail how it works in the rest of this section. It is essential to know that while training, the CTC takes the output matrix and the ground truth text and does not try to learn each ground truth text's character position. Still, it tries all possible alignments of the ground truth text in the image and calculates the sum of the scores. The ground truth text's score is high if only the sum of the alignment scores is high.

There are three CTC fundamental functions to detail:

1. **Text encoding**: To defeat the significant problem of the database annotation when characters take more than one time-step in the image and duplicated characters are provided (an example is given in Figure 6). The CTC is set up to overcome this problem by representing redundant characters with a single one. It uses alphabet labels (composed of all characters that occurred in the training data). The CTC adds a blank label (indicated by '-') to specify no label at a particular time position in the output sequence. For example, in Figure 6b, the CNN result (annotation) for the input image is '-ننققة-', which will be decoded to 'نقة'.
2. **Loss calculation**: The CTC calculates the loss function and backpropagates it to the NN to restart the end-to-end learning process (training). As mentioned before, the Att-BLSTM produces a matrix of scores for each character at each time step. The loss is calculated by adding all scores of all possible alignments of the ground truth text. Figure 7 gives a simple example for two-time steps with a reduced matrix (where the alphabet is composed of three characters {أ,ي,–}. Supposing that the current ground truth is "ي," then all feasible paths are: "يي", "ي–" and "–ي." The probability of the ground truth occurring is calculated by: $P = 0.12 + 0.24 + 0.18 = 0.54$, where corresponding character scores are multiplied to get the score for one path. The CTC applies the negative logarithm to the obtained probability to calculate the loss value L: $L = -\log(P)$.
3. **Text decoding**: For CTC decoding (inference), we used the best path decoding method to decode the output probability matrix as follows:
   (a) For each time step, it determines the character with the maximum probability, and at the final time step, the best path is calculated.
   (b) To determine the resulting text, it removes blanks and duplicated characters.

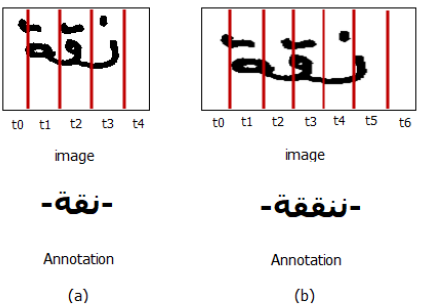


Fig. 6. (**a**) Image annotation where each character takes up one time-step. (**b**) Image annotation where a few characters take up more than one time-step.

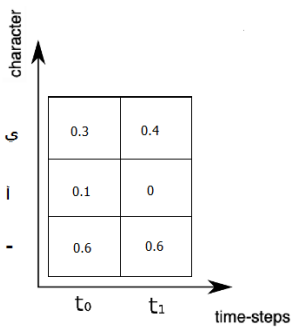


Fig. 7. The Att-BLSTM output matrix representing the character probability at each time step.

3.4. Configuration

We implemented the proposed offline Arabic handwritten text recognition model on a Hp Z-440 workstation with 16 GB RAM using Keras [34] (a Python deep learning library created by Google).

This section gives a detailed description of the proposed model architecture's configuration: the general hyperparameters, the CNN hyperparameters, the RNN hyperparameters, the attention blocks, and the CTC hyperparameters. The general hyperparameters provide those for the proposed model and the configuration network, which enclose the optimizer, learning rate, batch size, number of convolution layers, number of LSTM layers, the attention mechanism, and the CTC configuration. These hyperparameters are illustrated as follows:

- **Optimizer**: The adaptive learning rate optimization algorithm incrementally updates the CNN's weights after each epoch passes over the training dataset. The optimization algorithm used in our experiments is the Adam optimizer, presented by Diederik Kingma from OpenAI and Jimmy Ba from the University of Toronto in 2014 [44]. We choose it because it is a popular algorithm in the deep learning area and reaches good results fast. We set the Adam parameters as follows:
  - **Alpha:** is the learning rate or step size. The proportion that weights are updated was set to 10^{-4}.
  - **Beta1:** is the exponential decay rate for the first moment estimates is set to 0.9, the default value in the Keras deep learning library.
  - **Beta2:** is the exponential decay rate for the second-moment estimates is set to 0.999, the default value in the Keras library.
  - **Epsilon:** is a tiny number to prevent any division by zero in the implementation, and it was set to its default value, which is 10^{-7}.
- **Learning rate:** the learning rate controls how much to modify the model in reply to the calculated error each time the model weights are updated. Selecting a reasonable learning rate is a difficult task, where a too-small value can lead to a lengthy training process that might get stuck. At the same time, a too-large value can lead to learning a set of sub-optimal weights too fast or to an inconsistent training process. In our experiments, the learning rate is set to 10^{-4}.
- **Batch size:** it presents the total number of training samples presented in a single batch; in our training, we set it to 260.
- **The CNN configuration:**
  - **The number of convolution layers:** since handwriting recognition is a complex task, our CNN must be a deep architecture; with more than three layers to guarantee a good features extraction from the input images. We limited our model to 7 layers since an enormous number of layers can increase the number of weights and the complexity of the model.
  - **Regularization**: more precisely, the dropout; to handle the problem of overfitting. It is a simple method that randomly drops nodes out of the network, and it has a regularizing impact as the remaining nodes should adjust to pick up the slack of the removed nodes. So a dropout is added at the end of each layer (see Table 2 for the dropout rates setting).
  - **Convolution batch normalization**: is used to improve a neural network's performance and is designed to automatically standardize the inputs to a deep learning neural network layer. In our CNN architecture, we added batch normalization to the third and sixth layers between the convolution and max-pooling operations, and we used it in a binary range $(0, 1)$. The layer will transform inputs to be standardized, meaning they will have a mean of zero and a standard deviation of one. The momentum parameter is set to its default value, 0.99.

- **Convolution activation function**: In a network layer, the activation function specifies how the weighted sum of the input is transformed into an output of one single or multiple nodes. The activation function controls how sufficiently the network model learns the training dataset in the hidden layers. In the output layer, the activation function defines the type of predictions the model should make. In our CNN, we used the *ReLU* function for the hidden layers and the last layer since the last layer is not making predictions. It produces $100 \times 1 \times 512$ vectors that will be first passed to a collapse layer to remove dimension to 100×512; this feature matrix will be passed as input to the RNN block.
- **Convolution kernel size:** Each convolution kernels' number belongs to the set $\{1, 64, 128, 256, 512\}$.
- **Convolution kernels:** Each convolution kernel's size belongs to the set $\{1, 2, 4, 8, 16, 32, 64\}$.

- **The number of LSTM layers:** We used two LSTMs with 512 hidden cells for each one in the proposed architecture.
- **Attention mechanism:** Integrating attention mechanisms into deep-learning applications has proven to be a notable improvement in many applications, such as image recognition and machine translation. It is added to deep learning models to select which information to reserve to achieve the best use of the limited resources. More precisely, it is the power to dynamically select and use the essential parts of the available information as a human brain does. We added an Attention layer after the BLSTM using Keras in the proposed architecture. We set the `return_sequences` parameter to `True` when creating the BLSTM model to return the hidden units' output for all the previous time steps. The attention is calculated by a weighted sum of the value vectors resulting from the RNN.
- **CTC configuration:** The CTC translates a prediction into a label sequence. Its input is a sequence of observations, and the outputs are a sequence of labels, including blank outputs. In our architecture, the sequence max length is 100, the class number is 121, and the used decoder is the `bestpath` decoder.

The global structure of the used model is shown in Figure 4, and a detailed architecture is presented in Table 2.

4. Experimental Results

4.1. Training

The parameters used in the training process are listed in Tab. 3.

4.1.1. Databases

We used the IFN/ENIT [57] database for the model training. The Institute of Communications Technology in Germany has created IFN in association with the National

Tab. 2. Architecture for the used CNN. Abbreviations: Pooling (Pool), batch normalization (BN), convolutional layer (Conv).

Type	**Description**	**Output size**
Input	gray-value text line image	$800 \times 64 \times 1$
Conv + Pool + Dropout	kernel 5×5, pool 2×2, rate(0.1)	$400 \times 32 \times 64$
Conv + Pool + Dropout	kernel 5×5, pool 1×2, rate(0.2)	$400 \times 16 \times 128$
Conv + BN + Pool + Dropout	kernel 3×3, pool 2×2, rate(0.25)	$200 \times 8 \times 128$
Conv + Dropout	kernel 3×3, rate(0.3)	$200 \times 8 \times 256$
Conv + Pool + Dropout	kernel 3×3, pool 2×2, rate(0.35)	$100 \times 4 \times 256$
Conv + BN + Pool + Dropout	kernel 3×3, pool 1×2, rate(0.4)	$100 \times 2 \times 512$
Conv + Pool + Dropout	kernel 3×3, pool 1×2, rate(0.45)	$100 \times 1 \times 512$
Collapse	remove dimension	100×512
Att-BLSTM	bidir., 2 layers, 512 hidden cells	100×1024
Projection	Projection into C classes	$100 \times C$
CTC	Loss calculation and decoding	≤ 100

Engineers School of Tunis (ENIT) in Tunisia. It consists of five sets: a, b, c, d, e, containing 32492 images of handwritten names of 937 Tunisian towns written by more than 1000 different writers. Furthermore, two new sets, f and s, are added for word recognition evaluation. The set f was created in Tunisia by new writers who did not build the old five sets. The set s was collected in the United Arab Emirates (UAE) by students at Sharjah.

Furthermore, we trained the model on a Latin database called IAM [54]. It contains 1066 forms of handwritten English text produced by approximately 400 different writers. They were scanned at 300 dpi resolution and saved as PNG images with 256 gray levels. A total of 82227-word instances out of 10841 words occur in this database. The dataset can be used to train and test English handwritten text recognition. We kept the same parameters except for the number of classes, fixed at 54, which is the number of different characters in the IAM database.

Tab. 3. Parameter settings

Image pre-processing	Segmenting the IFN/ENIT database to three sets: set_1 composed of sets a, b, c, and s for training, set_2 is d for validation, and set e for testing. All images are resized to $(800 \times 64 \times 1)$. No other processing.
Training setting	`initial_epoch` = 0, `no_improvement` = 30, `learning_rate` = 10^{-4}, `initial_weights` = 0.01, `batch_size` = 260, Evaluation metrics: `Accuracy` and `Loss`,
CTC setting	`max_sequence_length` = 100, `Decoder_Type` = `bestpath`, and the number of classes $C = 121$

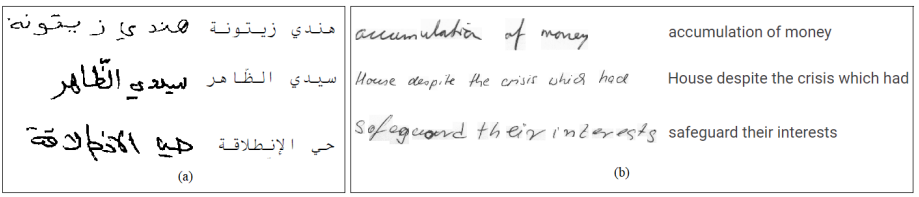


Fig. 8. (**a**) Samples from the IFN/ENIT database and their transcriptions. (**b**) Samples from the IAM database and their transcriptions.

Samples of the used databases and their transcriptions are given in Figure 8.

4.1.2. Training and Validation

We used precision and loss to evaluate the model's training on the IFN/ENIT and the IAM datasets. Table 3 displays the parameter settings when training the models.

- Training on the IFN/ENIT database:
  The training is stopped after 525 epochs, and the resulting curves show excellent training. As shown in Figure 9, the learning loss curve decreases to the point of stability, as does the validation loss plot, revealing a small gap with the learning loss plot. We conclude that the model loss is lower on the training dataset than on the validation dataset. In parallel, the accuracy plots of training and validation increase to the point of stability with a small gap. To conclude, the accuracy and loss curves show a good fit.
- Training on the IAM database:
  Figure 10 presents the loss and accuracy plots during training and validation, and they show a good fit. Training loss curves decrease slowly with slight noisy movements during the training and validation. Likewise, the training and validation accuracy curves increase gradually with subtle noisy movements to reach a height accuracy during training and validation.

4.2. Test

We reported the same evaluation metrics used in the benchmarks to facilitate comparison to other systems to access the performance of the proposed architecture on the test databases. On IAM, we show our results using Character Error Rate (CER) measures. Whereas on IFN/ENIT, we used the string accuracy metric. Recall that:

$$\mathrm{CER}(\%) = \frac{\sum_{i=1}^{n}(\mathrm{dist}_c(P_i, \mathrm{True}_i))}{\sum_{i=1}^{n}\mathrm{len}_c(\mathrm{True}_i)}\ , \tag{2}$$

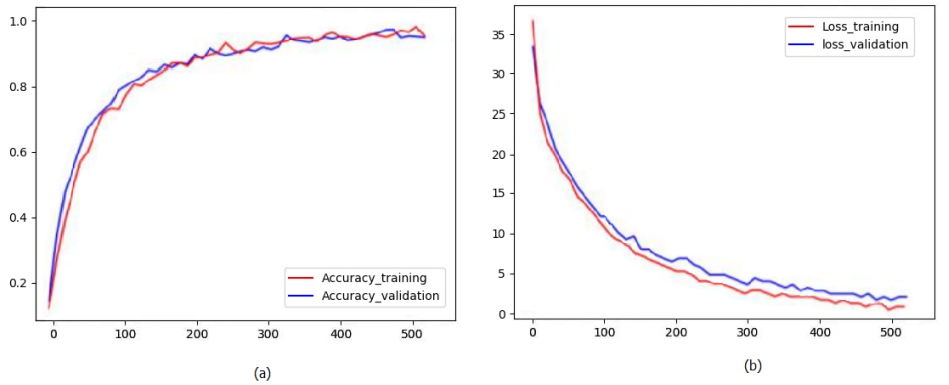


Fig. 9. Training curves: (**a**) Training and Validation Accuracy, and (**b**) Training and Validation Loss on IFN/ENIT database.

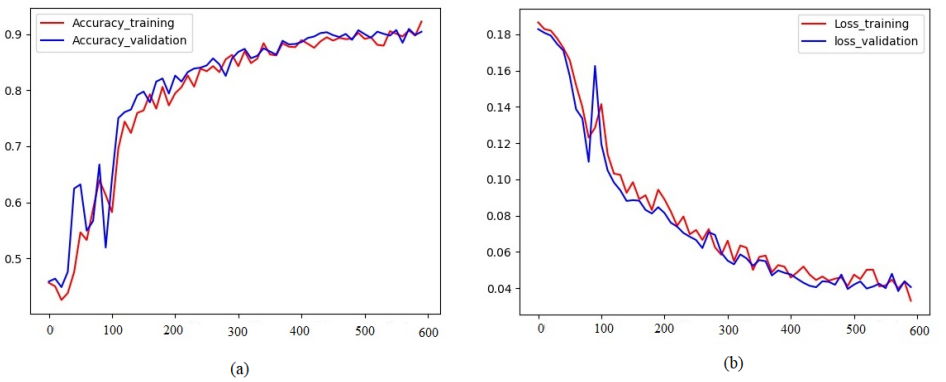


Fig. 10. Training curves: (**a**) Training and Validation Accuracy, and (**b**) Training and Validation Loss on IAM database.

where dist_c is the Levenshtein distance calculated at character level (including spaces), len_c is the number of characters in the input string, P_i is the string of characters to be recognized for the i^{th} input image and True_i is the true transcription of the i^{th} image.

$$\text{Accuracy} = \frac{\sum_{i=1}^{n}(P_i = \text{True}_i)}{n} \times 100\ [\%]\ , \tag{3}$$

where P_i is the string of characters that the model recognizes for the i^{th} input image, True_i is the true transcription of the i^{th} image, and n is the size of the test database.

Tab. 4. Performance comparisons with IFN/ENIT Database.

Methods	**Architecture**	**Accuracy**
[52]	MDLSTM-CTC	88.38%
[29]	CNN-SVM	92.95%
[39]	Dynamic Bayesian Network	82%
[8]	CNN-HMM	89.23%
[11]	HMM(128 Mixtures)	93%
[53]	CNN-BLSTM	92.21%
[36]	ANN	87.10%
[1]	RNN	94.45%
[30]	DBN + CDBN	96.23%
[52]	MDLSTM with dropout	94.65%
[27]	CNN	94.9%
[43]	Bayesian + CNN	95.2%
[61]	CNN + RNN	96.75%
Proposed Method	**CNN + Att-BLSTM + CTC + dropout**	**97.1%**

Table 4 gives recognition rates of other systems tested on the IFN/ENIT database. It is shown that our result is very prominent compared to other systems. Our trained model gives an impressive recognition rate of 97.1% for the testing set, whereas Table 5 shows CER rates of other systems tested on the IAM database. A CER of 2.9 % is attained, confirming the model's script independence. Interestingly, our system shows a clear advantage over the other systems tested on the IAM database. This result has further strengthened our conviction that the proposed model is script-independent.

Figure 11 presents examples of the model's inferences on English, French, and Italian handwritten text images.

5. Conclusion

Handwriting recognition is a dynamic field of research that regularly needs an accuracy increase. In this work, we proposed a deep learning approach based on a CNN-Att-BLSTM-CTC architecture with dropout and batch-normalization to recognize Arabic handwritten text accurately in images. Texts can be of various sizes and writing styles. We improved the used CNN with dropout, temporarily removing some units from the network to prevent the system from over-fitting problems. The CNN is used for sequence feature extraction and passes its output to the ATT-BLSTM to propagate information. For each sequence element, it outputs a matrix of character scores. Then the CTC operation is set up to calculate the loss value, train the proposed model, and to perform inference at this stage. The CTC decodes the Att-BLSTM's output matrix to infer the

Tab. 5. Performance comparisons with IAM Database.

Methods	**Architecture**	**CER**
[15]	CNN + BLSTM + CTC	3.2%
[65]	MDLSTM + CTC	3.5%
[16]	MDLSTM + MLP/HMM	3.6%
[12]	MDLSTM + CTC	4.4%
[59]	CNN + LSTM+CTC	4.4%
[13]	MDLSTM + Attention	4.4%
[40]	Transformer	4.6%
[22]	LSTM + HMM	4.7%
[66]	LSTM + HMM	4.8%
[55]	CNN + LSTM + Attention	4.8%
[67]	CNN + CTC	4.9%
[21]	CNN + LSTM + Attention	4.9%
[45]	LSTM + HMM	5.1%
[58]	MDLSTM + CTC	5.1%
[35]	CNN + BLSTM + Attention + CTC	5.1%
[24]	CNN + BLSTM	5.7%
[41]	CNN + BGRU + GRU + Attention	5.7%
[38]	CNN + CTC	6.1%
[14]	MDLSTM + CTC	6.6%
[42]	CNN + BGRU + GRU	6.8%
[20]	CNN + BLSTM + LSTM	8.1%
[46]	GMM/HMM	8.2%
[64]	CNN + LSTM + Attention	8.8%
[47]	CNN + LSTM + CTC	9.7%
[31]	MLP/HMM	9.8%
[18]	MDLSTM + CTC	11.1%
[23]	MLP/HMM	12.4%
[51]	MDLSTM + CTC	17.0%
[50]	BLSTM + CTC	18.2
Proposed method	**CNN + Attention-Blstm + CTC + dropout + BN**	**2.9%**

text contained in the input image. The proposed model is validated on the IFN/ENIT database. According to the experimental results, the accuracy reaches 97.1%. The feature extraction, training, and recognition components of the model are all designed to be script-independent. The model parameters are estimated automatically from the training data without the need for laborious handwritten rules. It requires no pre-segmentation of the data, either at the word level or at the character level. Thus, it can handle languages with cursive handwritten scripts straightforwardly. We tested it on the IAM database, and a CER of 2.9% was attained.

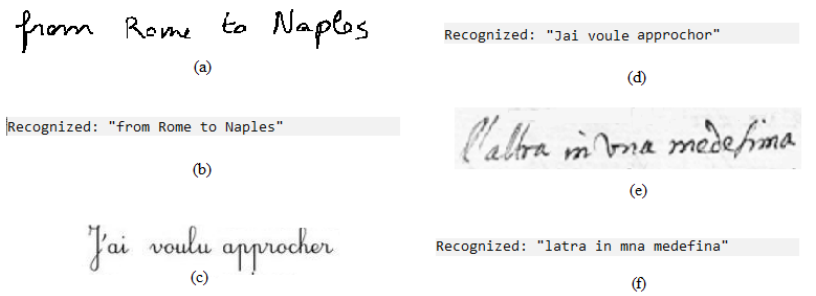


Fig. 11. Model's results on handwritten Latin text lines; where (**a**) and (**b**) are the handwritten English text image, and its recognition result, (**c**) and (**d**) are the handwritten French text image, and its recognition result, respectively, and (**e**) and (**f**) are handwritten Italian text image and its recognition result.

In future work, we intend to train the developed model on a complex database, such as HACDB, KHATT, etc., to improve our model's performance, employing deeper models. We also aim to extend the set of input images to recognize text lines with larger sizes and inclinations. We also plan to add a second Att-BLSTM before the CTC layer to improve the results.

Tab. 6. Table of Abbreviations.

AHTR	Arabic Handwritten Text Recognition
APTI	Arabic Printed Text Image database
Att	Attention
BLSTM	Bidirectional Long Short Term Memory
BN	Batch Normalization
CDBN	Convolutional Deep Belief Network
CER	Character Error Rate
CNN	Convolutional Neural Network
Conv	Convolutional layer
CRR	Character Recognition Rate
CTC	Connectionist Temporal Classification
DCNN	Deep CNN
GRU	Gated Recurrent Unit
HMM	Hidden Markov Model
LSTM	Long Short Term Memory
MDLSTM	Multi-Dimensional Long Short Term Memory
NN	Neural Network
OCR	Optical Character Recognition
Pool	Pooling
RNN	Recurrent Neural Network
RBF	Radial Basis Function

References

[1] G. A. Abandah, F. T. Jamour, and E. A. Qaralleh. Recognizing handwritten Arabic words using grapheme segmentation and recurrent neural networks. *International Journal on Document Analysis and Recognition*, 17(3):275–291, 2014. doi:10.1007/s10032-014-0218-7.

[2] R. Ahmad, M. Z. Afzal, S. F. Rashid, M. Liwicki, T. Breuel, and A. Dengel. KPTI: Katib's Pashto text imagebase and deep learning benchmark. In *Proc. 2016 15th Int. Conf. Frontiers in Handwriting Recognition (ICFHR)*, pages 453–458, Shenzhen, China, 23-26 Oct 2016. IEEE. doi:10.1109/ICFHR.2016.0090.

[3] R. Ahmad, S. Naz, M. Z. Afzal, et al. KHATT: A deep learning benchmark on Arabic script. In *Proc. 2017 14th IAPR Int. Conf. Document Analysis and Recognition (ICDAR)*, pages 10–14, Kyoto, Japan, 9-15 Nov 2017. IEEE. doi:10.1109/ICDAR.2017.358.

[4] R. Ahmad, S. Naz, M. Z. Afzal, et al. A deep learning based Arabic script recognition system: Benchmark on KHAT. *International Arab Journal of Information Technology*, 17(3):299–305, 2020. doi:10.34028/iajit/17/3/3.

[5] R. Ahmed, K. Dashtipour, M. Gogate, A. Raza, et al. Offline Arabic handwriting recognition using deep machine learning: A review of recent advances. In *Advances in Brain Inspired Cognitive Systems. Proc. Int. Conf. Brain Inspired Cognitive Systems (BICS) 2019*, pages 457–468, Guangzhou, China, 13-14 Jul 2019. 2020. Springer International Publishing. doi:10.1007/978-3-030-39431-8_44.

[6] R. Ahmed, M. Gogate, A. Tahir, et al. Novel deep convolutional neural network-based contextual recognition of Arabic handwritten scripts. *Entropy*, 23(3):340, 2021. doi:10.3390/e23030340.

[7] A. A. Al Rababah. Neural networks precision in technical vision systems. *International Journal of Computer Science and Network Security*, 20(3):29–36, 2020. `http://paper.ijcsns.org/07_book/202003/20200305.pdf`.

[8] M. Amrouch, M. Rabi, and Y. Es-Saady. Convolutional feature learning and CNN based HMM for Arabic handwriting recognition. In *Image and Signal Processing. Proc. Int. Conf. Image and Signal Processing (ICISP) 2018*, volume 9887 of *Lecture Notes in Computer Science*, pages 265–274, Cherbourg, France, 2-4 Jul 2018. Springer. doi:10.1007/978-3-319-94211-7_29.

[9] Z. Asebriy, S. Raghay, O. Bencharef, and Y. Chihab. Comparative systems of handwriting Arabic character recognition. In *Proc. 2014 2nd World Conf. Complex Systems (WCCS)*, pages 90–93, Agadir, Morocco, 10-12 Nov 2014. doi:10.1109/ICoCS.2014.7060923.

[10] M. Awni, M. I. Khalil, and H. M. Abbas. Offline Arabic handwritten word recognition: A transfer learning approach. *Journal of King Saud University – Computer and Information Sciences*, 34(10, Part B):9654–9661, 2022. doi:10.1016/j.jksuci.2021.11.018.

[11] S. A. Azeem and H. Ahmed. Effective technique for the recognition of offline Arabic handwritten words using hidden markov models. *International Journal on Document Analysis and Recognition*, 16(4):399–412, 2013. doi:10.1007/s10032-013-0201-8.

[12] T. Bluche. *Deep Neural Networks for Large Vocabulary Handwritten Text Recognition*. PhD thesis, Université Paris 11, 2015.

[13] T. Bluche. Joint line segmentation and transcription for end-to-end handwritten paragraph recognition. In *Advances in Neural Information Processing Systems 29 – Proc. 30th Conf. NIPS 2016*, volume 29, pages 838–846, Barcelona, Spain, 5-10 Dec 2019. Curran Associates, Inc. `https://proceedings.neurips.cc/paper/2016/file/2bb232c0b13c774965ef8558f0fbd615-Paper.pdf`.

[14] T. Bluche, J. Louradour, and R. Messina. Scan, Attend and Read: End-to-end handwritten

paragraph recognition with MDLSTM attention. In *Proc. 2017 14th IAPR Int. Conf. Document Analysis and Recognition (ICDAR)*, pages 1050–1055, Kyoto, Japan, 9-15 Nov 2017. IEEE. doi:10.1109/ICDAR.2017.174.

[15] T. Bluche and R. Messina. Gated convolutional recurrent neural networks for multilingual handwriting recognition. In *Proc. 2017 14th IAPR Int. Conf. Document Analysis and Recognition (ICDAR)*, pages 646–651, Kyoto, Japan, 9-15 Nov 2017. IEEE. doi:10.1109/ICDAR.2017.111.

[16] D. Castro, B. L. D. Bezerra, and M. Valença. Boosting the deep multidimensional long-short-term memory network for handwritten recognition systems. In *Proc. 2018 16th Int. Conf. Frontiers in Handwriting Recognition (ICFHR)*, pages 127–132, Niagara Falls, NY, USA, 5-8 Aug 2018. IEEE. doi:10.1109/ICFHR-2018.2018.00031.

[17] L. Chen, R. Yan, L. Peng, A. Furuhata, and X. Ding. Multi-layer recurrent neural network based offline Arabic handwriting recognition. In *Proc. 2017 1st Int. Workshop on Arabic Script Analysis and Recognition (ASAR)*, pages 6–10, Nancy, France, 3-5 Apr 2017. IEEE. doi:10.1109/ASAR.2017.8067749.

[18] Z. Chen, Y. Wu, F. Yin, and C.-L. Liu. Simultaneous script identification and handwriting recognition via multi-task learning of recurrent neural networks. In *Proc. 2017 14th IAPR Int. Conf. Document Analysis and Recognition (ICDAR)*, pages 525–530, Kyoto, Japan, 09-15 Nov 2017. IEEE. doi:10.1109/ICDAR.2017.92.

[19] Y. Chherawala, P. P. Roy, and M. Cheriet. Feature design for offline Arabic handwriting recognition: Handcrafted vs automated? In *Proc. 2013 12th IAPR Int. Conf. Document Analysis and Recognition (ICDAR)*, pages 290–294, Washington, DC, USA, 25-28 Aug 2013. IEEE. doi:10.1109/ICDAR.2013.65.

[20] A. Chowdhury and L. Vig. An efficient end-to-end neural model for handwritten text recognition. *arXiv*, 2018. arXiv:1807.07965v2. doi:10.48550/arXiv.1807.07965.

[21] D. Coquenet, C. Chatelain, and T. Paquet. End-to-end handwritten paragraph text recognition using a vertical attention network. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):508–524, 2022. doi:10.1109/TPAMI.2022.3144899.

[22] P. Doetsch, M. Kozielski, and H. Ney. Fast and robust training of recurrent neural networks for offline handwriting recognition. In *Proc. 2014 14th Int. Conf. Frontiers in Handwriting Recognition (ICFHR)*, pages 279–284, Hersonissos, Greece, 01-04 Sep 2014. IEEE. doi:10.1109/ICFHR.2014.54.

[23] P. Dreuw, P. Doetsch, C. Plahl, and H. Ney. Hierarchical hybrid MLP/HMM or rather MLP features for a discriminatively trained gaussian HMM: A comparison for offline handwriting recognition. In *2011 18th IEEE Int. Conf. Image Processing (ICIP)*, pages 3541–3544, Brussels, Belgium, 11-14 Sep 2011. IEEE. doi:10.1109/ICIP.2011.6116480.

[24] K. Dutta, P. Krishnan, M. Mathew, and C.V. Jawahar. Improving CNN-RNN hybrid networks for handwriting recognition. In *Proc. 2018 16th Int. Conf. Frontiers in Handwriting Recognition (ICFHR)*, pages 80–85, Niagara Falls, NY, USA, 5-8 Aug 2018. IEEE. doi:10.1109/ICFHR-2018.2018.00023.

[25] B. El Qacimy, A. Hammouch, and M. A. Kerroum. A review of feature extraction techniques for handwritten Arabic text recognition. In *Proc. 2015 Int. Conf. Electrical and Information Technologies (ICEIT)*, pages 241–245, Marrakech, Morocco, 25-27 Mar 2015. IEEE. doi:10.1109/EITech.2015.7162979.

[26] B. El Qacimy, M. A. Kerroum, and A. Hammouch. Word-based Arabic handwritten recognition using SVM classifier with a reject option. In *Proc. 2015 15th Int. Conf. Intelligent Systems Design and Applications (ISDA)*, pages 64–68, Marrakech, Morocco, 14-16 Dec 2015. IEEE. doi:10.1109/ISDA.2015.7489190.

[27] A. El-Sawy, M. Loey, and H. El-Bakry. Arabic handwritten characters recognition using convolutional neural network. *WSEAS Transactions on Computer Research*, 5:11–19, 2017. https://www.wseas.com/journals/articles.php?id=3300.

[28] M. Elleuch and M. Kherallah. Convolutional deep learning network for handwritten arabic script recognition. In *Proc. Int. Conf. Hybrid Intelligent Systems (HIS 2019)*, volume 1179 of *Advances in Intelligent Systems and Computing*, pages 103–112, Sehore, India, 10-12 Dec 2019. Springer. doi:10.1007/978-3-030-49336-3_11.

[29] M. Elleuch, R. Maalej, and M. Kherallah. A new design based-SVM of the CNN classifier architecture with dropout for offline Arabic handwritten recognition. *Procedia Computer Science*, 80:1712–1723, 2016. doi:10.1016/j.procs.2016.05.512.

[30] M. Elleuch, N. Tagougui, and M. Kherallah. Deep learning for feature extraction of Arabic handwritten script. In *Computer Analysis of Images and Patterns. Proc. Int. Conf. Computer Analysis of Images and Patterns (CAIP) 2015*, volume 9257 of *Lecture Notes in Computer Science*, pages 371–382, Valletta, Malta, 2-4 Sep 2015. Springer. doi:10.1007/978-3-319-23117-4_32.

[31] S. España-Boquera, M. J. Castro-Bleda, J. Gorbe-Moya, and F. Zamora-Martinez. Improving offline handwritten text recognition with hybrid HMM/ANN models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(4):767–779, 2010. doi:10.1109/TPAMI.2010.141.

[32] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *ICML '06: Proc. 23rd Int. Conf. Machine Learning*, pages 369–376, Pittsburgh, PA, USA, 25-29 Jun 2006. doi:10.1145/1143844.1143891.

[33] A. Graves and J. Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in Neural Information Processing Systems 21 – Proc. 22nd Conf. NeurIPS 2008*, volume 21, pages 545–552. Curran Associates, Inc., 2008. https://proceedings.neurips.cc/paper/2008/file/66368270ffd51418ec58bd793f2d9b1b-Paper.pdf.

[34] Keras Special Interest Group. Keras. simple. flexible. powerful. https://keras.io.

[35] L. Gui, X. Liang, X. Chang, and A. G. Hauptmann. Adaptive context-aware reinforced agent for handwritten text recognition. In *Proc. 29th British Machine Vision Conference (BMVC) 2018*, volume 207, Newcastle, United Kingdom, 3-6 Sep 2018. British Machine Vision Association and Society for Pattern Recognition. http://bmvc2018.org/contents/papers/0628.pdf.

[36] S. Haboubi, S. Maddouri, N. Ellouze, and H. El-Abed. Invariant primitives for handwritten Arabic script: A contrastive study of four feature sets. In *Proc. 2009 10th IAPR Int. Conf. Document Analysis and Recognition (ICDAR)*, pages 691–697, Barcelona, Spain, 26-29 Jul 2009. IEEE. doi:10.1109/ICDAR.2009.281.

[37] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi:10.1162/neco.1997.9.8.1735.

[38] X. Huang, L. Qiao, W. Yu, et al. End-to-end sequence labeling via convolutional recurrent neural network with a connectionist temporal classification layer. *International Journal of Computational Intelligence Systems*, 13(1):341–351, 2020. doi:10.2991/ijcis.d.200316.001.

[39] K. Jayech, M. A. Mahjoub, and N. E. B. Amara. Arabic handwritten word recognition based on dynamic Bayesian network. *International Arab Journal of Information Technology*, 13(6B):1024–1031, 2016. doi:10.34028/iajit/16/13/6B. https://iajit.org/PDF/Vol.13,No.3/7681.pdf.

[40] L. Kang, P. Riba, M. Rusiñol, et al. Pay attention to what you read: Non-recurrent handwritten text-line recognition. *Pattern Recognition*, 129:108766, 2022. doi:10.1016/j.patcog.2022.108766.

[41] L. Kang, P. Riba, M. Villegas, et al. Candidate fusion: Integrating language modelling into a

sequence-to-sequence handwritten word recognition architecture. *Pattern Recognition*, 112:107790, 2021. doi:10.1016/j.patcog.2020.107790.

[42] L. Kang, J. I. Toledo, P. Riba, et al. Convolve, attend and spell: An attention-based sequence-to-sequence model for handwritten word recognition. In *Proc. 40th German Conf. Pattern Recognition (GCPR) 2018*, volume 11269 of *Lecture Notes in Computer Science*, pages 459–472, Stuttgart, Germany, 9-12 Oct 2018. Springer. doi:10.1007/978-3-030-12939-2_32.

[43] A. Khémiri, A. K. Echi, and M. Elloumi. Bayesian versus convolutional networks for Arabic handwriting recognition. *Arabian Journal for Science and Engineering*, 44(11):9301–9319, 2019. doi:10.1007/s13369-019-03939-y.

[44] 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.

[45] M. Kozielski, P. Doetsch, and H. Ney. Improvements in RWTH's system for off-line handwriting recognition. In *Proc. 2013 IAPR 12th Int. Conf. Document Analysis and Recognition (ICDAR)*, pages 935–939, Washington, DC, USA, 25-28 Aug 2013. IEEE. doi:10.1109/ICDAR.2013.190.

[46] M. Kozielski, D. Rybach, S. Hahn, et al. Open vocabulary handwriting recognition using combined word-level and character-level language models. In *Proc. 2013 IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, pages 8257–8261, Vancouver, Canada, 26-31 May 2013. IEEE. doi:10.1109/ICASSP.2013.6639275.

[47] P. Krishnan, K. Dutta, and C. V. Jawahar. Word spotting and recognition using deep embedding. In *Proc. 2018 13th IAPR Int. Workshop on Document Analysis Systems (DAS)*, pages 1–6, Vienna, Austria, 24-27 Apr 2018. IEEE. doi:10.1109/DAS.2018.70.

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

[49] A. Lawgali, M. Angelova, and A. Bouridane. HACDB: Handwritten Arabic characters database for automatic character recognition. In *Proc. European Workshop on Visual Information Processing (EUVIP)*, pages 255–259, Paris, France, 10-12 Jun 2013. `https://ieeexplore.ieee.org/abstract/document/6623974`.

[50] M. Liwicki, A. Graves, and H. Bunke. Neural networks for handwriting recognition. In *Computational Intelligence Paradigms in Advanced Pattern Classification*, volume 386 of *Studies in Computational Intelligence*, pages 5–24. Springer, 2012. doi:10.1007/978-3-642-24049-2_2.

[51] J. Louradour and C. Kermorvant. Curriculum learning for handwritten text line recognition. In *Proc. 2014 11th IAPR Int. Workshop on Document Analysis Systems (DAS)*, pages 56–60, Tours, France, 07-10 Apr 2014. IEEE. doi:10.1109/DAS.2014.38.

[52] R. Maalej and M. Kherallah. Improving MDLSTM for offline Arabic handwriting recognition using dropout at different positions. In *Artificial Neural Networks and Machine Learning. Proc. Int. Conf. on Artificial Neural Networks (ICANN) 2016*, volume 9887 of *Lecture Notes in Computer Science*, pages 431–438, Barcelona, Spain, 6-9 Sep 2016. Springer. doi:10.1007/978-3-319-44781-0_51.

[53] R. Maalej and M. Kherallah. Convolutional neural network and BLSTM for offline Arabic handwriting recognition. In *Proc. 2018 Int. Arab Conf. Information Technology (ACIT)*, pages 1–6, Werdanye, Lebanon, 28-30 Nov 2018. IEEE. doi:10.1109/ACIT.2018.8672667.

[54] U.-V. Marti and H. Bunke. The IAM-database: an English sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*, 5:39–46, 2002. doi:10.1007/s100320200071.

[55] J. Michael, R. Labahn, T. Grüning, and J. Zöllner. Evaluating sequence-to-sequence models for handwritten text recognition. In *Proc. 2019 IAPR Int. Conf. Document Analysis and Recognition (ICDAR)*, pages 1286–1293, Sydney, NSW, Australia, 20-25 Sep 2019. IEEE. doi:10.1109/ICDAR.2019.00208.

[56] A. Mohsin and M. Sadoon. Developing an Arabic handwritten recognition system by means of artificial neural network. *Journal of Engineering and Applied Sciences*, 15(1):1–3, 2019. doi:10.36478/jeasci.2020.1.3.

[57] V. Märgner and H. El Abed. IFN/ENIT-database. Database of handwritten Arabic words, 2002. `http://ifnenit.com`.

[58] V. Pham, T. Bluche, C. Kermorvant, and J. Louradour. Dropout improves recurrent neural networks for handwriting recognition. In *Proc. 2014 14th Int. Conf. Frontiers in Handwriting Recognition (ICFHR)*, pages 285–290, Hersonissos, Greece, 01-04 Sep 2014. IEEE. doi:10.1109/ICFHR.2014.55.

[59] J. Puigcerver. Are multidimensional recurrent layers really necessary for handwritten text recognition? In *Proc. 2017 14th IAPR Int. Conf. Document Analysis and Recognition (ICDAR)*, pages 67–72, Kyoto, Japan, 9-15 Nov 2017. IEEE. doi:10.1109/ICDAR.2017.20.

[60] S. F. Rashid, M.-P. Schambach, J. Rottland, and S. von der Nüll. Low resolution Arabic recognition with multidimensional recurrent neural networks. In *Proc. 4th Int. Workshop on Multilingual OCR (MOCR '13)*, pages 1–5, Washington, DC, USA, 24 Aug 2013. doi:10.1145/2505377.2505385.

[61] V. M. Safarzadeh and P. Jafarzadeh. Offline Persian handwriting recognition with CNN and RNN-CTC. In *Proc. 2020 25th Int. Computer Conf., Computer Society of Iran (CSICC)*, pages 1–10, Tehran, Iran, 1-2 Jan 2020. IEEE. doi:10.1109/CSICC49403.2020.9050073.

[62] H. Scheidl. Build a handwritten text recognition system using TensorFlow. In B. Huberman et al., editors, *Towards Data Science*, 2015. [Accessed 4 May 2022]. `https://towardsdatascience.com/build-a-handwritten-text-recognition-system-using-tensorflow-2326a3487cd5`.

[63] F. Slimane, R. Ingold, S. Kanoun, et al. A new Arabic printed text image database and evaluation protocols. In *Proc. 2009 10th IAPR Int. Conf. Document Analysis and Recognition (ICDAR)*, pages 946–950, Barcelona, Spain, 26-29 Jul 2009. IEEE. doi:10.1109/ICDAR.2009.155.

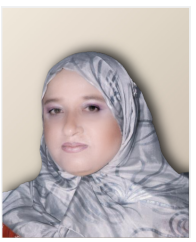
[64] J. Sueiras, V. Ruiz, A. Sanchez, and J. F. Velez. Offline continuous handwriting recognition using sequence to sequence neural networks. *Neurocomputing*, 289:119–128, 2018. doi:10.1016/j.neucom.2018.02.008.

[65] P. Voigtlaender, P. Doetsch, and H. Ney. Handwriting recognition with large multidimensional long short-term memory recurrent neural networks. In *Proc. 2016 15th Int. Conf. Frontiers in Handwriting Recognition (ICFHR)*, pages 228–233, Shenzhen, China, 23-26 Oct 2016. IEEE. doi:10.1109/ICFHR.2016.0052.

[66] P. Voigtlaender, P. Doetsch, S. Wiesler, et al. Sequence-discriminative training of recurrent neural networks. In *Proc. 2015 IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, pages 2100–2104, South Brisbane, QLD, Australia, 19-24 Apr 2015. IEEE. doi:10.1109/ICASSP.2015.7178341.

[67] M. Yousef, K. F. Hussain, and U. S. Mohammed. Accurate, data-efficient, unconstrained text recognition with convolutional neural networks. *Pattern Recognition*, 108:107482, 2020. doi:10.1016/j.patcog.2020.107482.

Takwa Ben Aïcha Gader is a Ph.D. student and a member of the Laboratory for Technologies of Information and Communication – LaTICE, at the National High School of Engineers of Tunis. Received her engineering teleinformatics degree from the Higher Institute of Computer Science and Communication Technologies of Hammam Sousse, Sousse, Tunisia, in 2014. (ORCID: 0000-0002-3786-3649)



Afef Kacem Echi Dr. Afef Kacem Echi received M.Sc. and Ph.D. degrees in Computer Sciences from the National School of Computer Sciences of Tunis in 1997 and 2001, respectively. Since 2000, she has been an assistant in the computer science department at the Faculty of Sciences of Monastir and was appointed Assistant Professor there in 2002. Dr. Kacem is responsible for the research area of Analysis and Recognition of Handwriting and document at the Laboratory for Technologies of Information and Communication – LaTICE, at the National High School of Engineers of Tunis. She has authored over 50 papers in various national and international journals and conference proceedings. (ORCID: 0000-0001-9219-5228)

Lexicon and Attention Based Handwritten Text Recognition System

Lalita Kumari[1], Sukhdeep Singh[2], Vaibhav Varish Singh Rathore[3] and Anuj Sharma[1]
[1]*Department of Computer Science and Applications, Panjab University, Chandigarh, India*
https://anuj-sharma.in
[2]*D.M. College (Affil. to Panjab University), Moga, Punjab, India*
https://sites.google.com/site/sransingh13/
[3]*Computer Networking and Information Technology, PRL, Ahmedabad, Gujarat, India*
lalita@pu.ac.in, sukha13@ymail.com, vaibhav@prl.res.in, anujs@pu.ac.in

Abstract. The handwritten text recognition problem is widely studied by the researchers of computer vision community due to its scope of improvement and applicability to daily lives. It is a sub-domain of pattern recognition. Due to advancement of computational power of computers since last few decades neural networks based systems heavily contributed towards providing the state-of-the-art handwritten text recognizers. In the same direction, we have taken two state-of-the art neural networks systems and merged the attention mechanism with it. The attention technique has been widely used in the domain of neural machine translations and automatic speech recognition and now is being implemented in text recognition domain. In this study, we are able to achieve 4.15% character error rate and 9.72% word error rate on IAM dataset, 7.07% character error rate and 16.14% word error rate on GW dataset after merging the attention and word beam search decoder with existing Flor et al. architecture. To analyse further, we have also used system similar to Shi et al. neural network system with greedy decoder and observed 23.27% improvement in character error rate from the base model.

Key words: handwriting recognition, deep learning, word beam search, attention, neural network, lexicon.

1. Introduction

One of the major modes of communication is through handwritten text. From the 19^{th} century onwards, there is a rapid growth in the various computer technologies. One such domain is handwriting recognition [53]. It is a sub-domain of pattern recognition. The act of transcribing handwritten text into a digitized form is named as handwritten text recognition (HTR). This problem is widely studied in the research community due to its omnipresent need where people communicate and interact. The communication mode can be verbal, by sign language or with handwritten text. At present, handwritten text can be represented in two ways: online and offline. The online handwriting recognition is performed while the text to be recognized is written (e.g. by a pressure setting device), therefore temporal and geometric information is available. It is a way of processing and recognizing the text while writing or entering. The offline handwriting recognition is performed by collecting handwritten data and feeding these data into a computing device for recognition. The HTR task faces various challenges, like cursiveness of handwritten

text, different size and shape of each character and large vocabularies. There are many challenging aspects of any unconstrained handwriting recognition task. For example, in unconstrained handwriting recognition there is no control over author, writing styles and instrument used for writing. Moreover, different vertical and horizontal space present among different lines or different words in the same line causes uncertainty in a total number of lines and words, respectively, in a page and line of a handwritten document for an automatic text recognizer [33].

One of the key application areas of HTR is converting the ancient historical handwritten text into the digital form as a part of modern digital library. It helps in bridging the gap among computers and humans in various domains. Apart from this, invoices, notes, accidental claims, feedback forms are usually in the handwritten format and these can be used as digital footprints in the industrial domain. Despite from having developed many state of the art techniques, HTR domain is still far away from having a generic system that is able to read any handwritten text. Initial state of the art approaches of the HTR domain were based upon Hidden Markov Models (HMMs) [4, 9, 22, 24]. Recently, Neural Network (NN) based techniques have been on the rise among the state of the art methods, specially Convolutional Neural Network (CNN) techniques, in various challenging tasks.

In any typical HTR system an input image consists of a sequence of objects (characters) to be recognized, that contextually depend upon each other. The recognized text length also varies greatly; for example, English language word "To" is of length 2, "Tomorrow" is of length 8 and "The quick brown fox jumps over a lazy dog" is of length 33. The recognition of such sequences of objects is done either at word or line level. Although word level recognition systems are quite popular, for a generic system this technique is suboptimal. Firstly, for handwritten text, it is not always feasible to do word segmentation due to their proximity or partially overlapping locations. In addition to this, for densely written handwritten text, it is complex to detect large number of words and at last there may be scenarios where a word is not separated by a space which is a word separator considered by most of the linguistic systems [36]. Thus this study focuses on line level recognition architecture to make the system as generic as possible.

The CNN in connection with the RNN have been consistently performing good and are able to give state of the art results for the HTR problem [35, 38, 51]. The CNN typically has a convolutional layer that uses convolution operation for extracting the features of the given image by applying various filters. The RNN is used to capture sequences and contextual information hence is able to correlate the relationship among characters rather then treating them independently. The Bidirectional Long Short-Term Memory (BLSTM) [23, 27] RNN is used to capture long term dependencies and to consider more context in comparison to typical RNN architecture. Deep BLSTM RNN is widely used in speech recognition tasks [26, 46]. The output from BLSTM RNN is the character probability versus time matrix, where probabilities of specific characters are calculated

for each time step. In 2006, a technique was introduced to train and score NN architectures in which input sequence and output labels are given in the form of the input and output pair. The Connectionist Temporal Classification (CTC) function introduced in this technique does not depend upon the underlying network architecture [25]. Hence, in architecture trained by the CTC, output of the RNN is characters probabilities, including the blank character. A blank character is a special character introduced in CTC to handle duplicate characters. Obtaining the actual text from the output of the RNN is called decoding.

A typical HTR system includes various preprocessing steps to minimize the variation of text as much as possible. There are no generic preprocessing steps but these usually relie on the input of the HTR system [19].

In this study, we have usued the line based gated convolution text recognition system [15]. The attention was first introduced in Neural Machine Translation (NMT) task [2, 39]. In a typical NMT task attention is used to provide the importance of each word at a given time step while translating. Other than NMT, attention is also used in speech recognition tasks [14]. Some recent studies have also shown attention to be a promising method used with a HTR task. In this study, we have blended the attention mechanism with CTC based NN system to learn and propagate the image features and utilize the benefits obtained from both the techniques The key contributions of the present study are as follows.

- The text recognition system is explained in an end-to-end manner and in a simplified way.
- Popular attention mechanism is merged with existing state-of-the-art HTR techniques.
- Two separate NN systems are taken and merged with attention module to make our observation generalized.
- Attention module is merged with Flor et al. and Shi et al. architecture to learn and propagate the image features efficiently while training of the NN system.
- Additionally, Word Beam Search (WBS) decoder [49] has been added as a post processing step to improve the accuracy in Flor et al. system [15].
- Greedy decoding method is used with Shi et al. architecture [51] to show the percentage of improvement we observed by adding the attention module to the existing NN system.

The rest of the paper is organized as follows. Section 2 covers the key contributions in the domain of the HTR. In section 3 the system design is discussed. Section 4 briefs the experimental setup. In Section 5 the results of this study and the comparison with other works are presented. Section 6 contains the discussion. At last, conclusions are presented in Section 7.

2. Related Work

In this section, we have discussed the key contributions in the domain of the HTR. We especially focused on the methods and techniques involving line level HTR. Early

works in this domain have used dynamic programming based methods to segment and identify the words at character level using optimum path finding algorithm [3, 11]. Further improvement was observed by using the HMM based techniques in the HTR [55]. Standard HMM based methods lack handling the long sequences of characters as per the Markov assumption. To improve the accuracy further, these models are combined with other basic NN systems. In one such method, an HMM/ANN based architecture is presented in which trigram Language Model (LM) is used for recognition purposes [18]. Later, more advanced NN layers systems are studied in connection with the HMMs. In a similar study, the HMM based HTR architecture is used that improves the accuracy of recognition on IAM and RIMES dataset by applying preprocessing steps, discriminative HMM training and discriminative feature extraction. In [34], an LSTM network is used for feature extraction, and word and character level LMs are used to improve the accuracy even more. Further, in one such study, the activation function of the gated units of the LSTM is modified to make the overall system robust. A combination of HMM and LSTM is used as a recognizer [17].

Current state-of-the-art are NN based systems. In one such system, the Convolutional Recurrent Neural Network (CRNN) architecture is introduced that is a combination of the CNN and the RNN and is able to produce state-of-the-art results in recognizing sequential objects in scene-text images [51] (available also in [50]). Similar to CRNN, one variant of RNNs, that is Multidimensional Long Short-Term Memory (MDLSTM) network, has been widely studied by the research community [28]. The MDLSTM architectures provide the recurrence in both directions (horizontal and vertical) along the given image [37]. Thus, two dimensional data of unconstrained handwritten text can be processed. Utilizing this nature of the MDLSTM, many page level recognition systems of the HTR are also proposed. In one such study, the proposed architecture is able to recognize paragraph level texts. External segmentation of the paragraph into lines is prone to errors and these errors propagate from segmentation to recognition. By doing an implicit recognition, this study resolves the error generated due to poor segmentation. In this study, the MDLSTM-RNN is used with attention mechanism. In [6] (published earlier as [5]), a trigram LM is used that was trained on LOB, Brown and Wellington corpora [30]. In a similar work [8] (available also in [7]), authors used an attention based model for end-to-end HTR. In this approach, an MDLSTM, CTC and attention based model is gradually trained from consecutive images of words to a complete text line. As the size of the sentence increases accuracy increases gradually. To handle the paragraph data, augmentation techniques are used to generate enough training data and the model is modified with curriculum learning to adapt to recognition at paragraph level. These networks are complex and require the use of a large number of computational resources. The NN architecture based on convolutional and 1D-LSTM layers is able to learn similar features with a significantly smaller computational cost [45]. Some notable

state-of-the-art systems are only made up of CNN layers or attention techniques without any recurrent layer [13, 42, 43, 44, 52, 57, 58].

3. System Design

In this section, we presented an overview of each module proposed in the text recognition system in detail. In this study, we have used modified Bahdanau attention [2], which was successfully used in two state-of-the-art NN systems [15] and [51]. Both of these systems take the input image and extract its features. The propagated features are processed by the attention and the RNN layer to produce the occurrence probability of each character at each time step. Figure 1 presents the system architecture in detail. Later in the discussion section, more is written on the position of attention layer in the NN model.

3.1. Feature Extraction

In both above mentioned systems, a greyscale image is taken as an input and its feature map is produced as an output. A set of convolution layers is used to extract these input image features. The heart of a convolution layer is a convolution operation. A kernel slides over a given input image to produce its feature map. The convolution operation is followed by Rectified Linear Unit (ReLU) activation function, which produces the non-linearity in the NN system. Quick convergence was observed in ReLU comparison to the other activation function of the same class. As shown in Fig. 1 the input image is first pre-processed then this processed image is passed through a set of convolution and fully gated convolution layers in Flor et al. [15] to extract the most relevant features. While in Shi et al. [51], a series of convolution layers with varying kernel sizes are used to extract features of images.

3.2. Attention Module

The CNN output is a four dimensional vector which is reshaped to three dimensions (batch size, time-steps, features at each time step). These feature maps $(f_1, f_2..., f_n)$ are fed to the attention module as input. The motivation for applying attention is towards getting a more powerful representation using a weighted context vector (C). At a given timestep, it is computed using Eq. (1),

$$C = \sum_{i=1}^{T} s_i f_i^{\mathrm{CNN}}, \tag{1}$$

where $f_{\mathrm{i}}^{\mathrm{CNN}}$ is feature information at i_{th} time step, T is the total number of time steps and s_i is attention weight corresponding to $f_{\mathrm{i}}^{\mathrm{CNN}}$ which is calculated using Eq. (2),

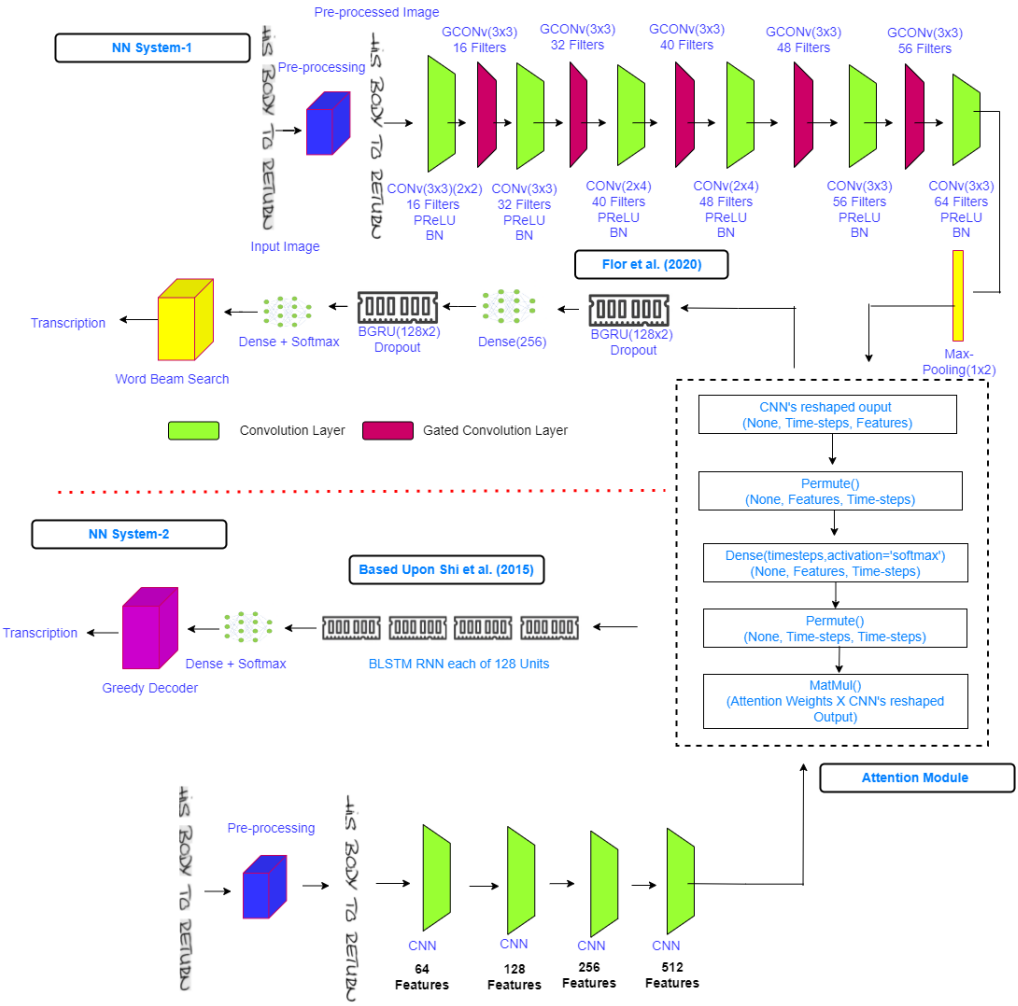


Fig. 1. System Architecture

$$s_i = \frac{\exp(a_i)}{\sum_{k=1}^{T} \exp(a_k)}, \tag{2}$$

where a_i is alignment score of feature f_i at a given time step. It is learned using Feed Forward Neural Network (FNN) while training of the NN model. The inputs of different

times are aggregated using attention based weighting. Thus, attention mechanism is applied in the time step dimension such that at each step more relevant features will be send to further RNN layers. Before applying the attention mechanism, we have to first permute the reshaped output of CNN. The current output without permutation will be of form (batch size, time steps, features per time step) and by applying dense layer without permuting it would be understood that there was no feature exchange due to attention mechanism, which is a false assumption. So, after the permutation operation, FFN layer is used to get attention probabilities. This operation will be performed for each time step. Thus we obtained a matrix that contains the weighted attention probabilities at each time step, which further multiplied to feature vectors to obtain the context vector. This context vector will be given to the next layer for further processing.

3.3. Recurrent Layers

Due to its property of having internal memory, the recurrent layers are widely used in sequence learning tasks. HTR is a sequence learning problem when we identify the probability of character occurrence given as an input image at each time step. As shown in Fig. 1, the output of the attention layer is given to Bidirectional Gated Recurrent Unit (BGRU) in Flor et al. system and similarly, the attention output is given to stacked BLSTM layers in Shi et al. system. Both systems used bidirectional stacked layers for processing input in forward and backward directions.

3.4. WBS Decoder

In a NN system, when trained with CTC loss function, the recurrent layer produces the character probabilities at each time-step. These probabilities are mapped to final character sequences using various decoding algorithms such as greedy, token passing and WBS decoder. This was proposed in [49]. The prefix tree made from the available corpus is internally used by this technique to decide which path to take while decoding at each time step. A *beam* is simply one possible character sequence. The *number* of beams that takes part in the next time step is equal to beam width except at $t = 0$. At the final time step, the beam that has the highest probability is selected as the final sequence and given as the output of the recognizer. We have used `Beam Width` $= 50$, and `Processing Mode` as `'NGram'` while applying this decoder in the present study [49].

4. Experimental Setup

In this section, we discuss the experimental work done in this study. This includes discussion on the datasets, preprocessing, data augmentation, evaluation metrics and training details. We have used the basic model of Flor et al. [15,16] and WBS according to [48,49]. The NN systems were implemented using the Keras package [29] in Python.

Tab. 1. Train, validation and test splits of IAM and GW datsets

SNo.	Dataset	Train set	Validation set	Test set
1	IAM Dataset (No. of characters=79)	6,161	900	1,861
2	GW Dataset (No. of characters=82)	325	168	163

4.1. Dataset

In this study, we evaluated our model on IAM [40] and GW [21] benchmarked datasets. These datasets contain pages of handwritten texts. These datasets contain the text images and their transcription at word, line and paragraph levels. Each of both datasets is discussed below.

4.1.1. IAM

The IAM dataset [40] contains the handwritten English text forms used by text recognizers for training and testing purposes. Data are present at word, line and paragraph levels. In this work, we have used the data present at line level with standard split as defined in Table 1. The IAM dataset contains 657 writers, 1539 pages of scanned text, 13353 text lines and 115320 words. All the data is in labelled format. LOB corpus is used to build the IAM dataset.

4.2. George Washington dataset (GW)

This dataset [21] contains the English letters written by George Washington to their associates in 1755. It has a total of 20 pages whose data is annotated at the word level, making 5000 words in total. We have used the train, test and validation split as specified in Table 1.

4.3. Preprocessing

The preprocessing techniques are used to improve the quality of degraded handwritten text documents. In this study, we have used illumination compensation [10], binarization [47] and deslanting [56] as preprocessing techniques. Figure 2 shows the image of the IAM dataset after each preprocessing technique.

4.4. Evaluation Metric

The evaluation metric is used to identify how well the proposed system is performing in comparison to earlier studies. We have used the standard evaluation metric Character Error Rate (CER) and Word Error Rate (WER). It is based on the Levenshtein

distance (LD). It is formulated as follows,

$$\text{WER} = \frac{S_{\text{word}} + D_{\text{word}} + I_{\text{word}}}{N_{\text{word}}} \tag{3}$$

where S_{word} is the number of substitutions required, D_{word} is the number of deletions required, I_{word} is the number of insertions required at word level, and N_{word} is the total number of characters in the ground truth sentence.

CER is the same as WER; the only difference is that in CER we work on the character level instead of the word level like in WER.

4.5. Data Augmentation

Data augmentation techniques are used to provide different variations of the samples available for training. For a NN system to be able to learn properly, the right amount of training data is required. The model can either be overfit or underfit based on the availability of the data. We have used random morphological and displacement transformations such as resizing, rotation, image displacement, erosion and dilation.

4.6. Training Details

In this section, we discuss the training and testing algorithm used in the present system. Algorithm 4.1 presents the training strategy and explains the details of the attention module. Algorithm 4.2 explains the decoding using the WBS decoder. In the Shi et al. system, the training architecture is the same except for the number and type of layers, but for the decoding purpose, we have used best path decoding to produce results in that setup.

Explanation We will discuss the training process as in algorithm 4.1 in line by line manner, as follows.

Fig. 2. Results of pre-processing techniques

Algorithm 4.1 Training Details

Input line images $I_1, I_2, ...I_n$ and ground truth $y_1, y_2, ...y_n$
Result Trained model weights on minimizing the validation loss

1: epochs=1000, batch=16, lr=0.001, stop_tolerance=20, reduce_tolerance=15; //initialize the training parameters
2: **procedure** ATTENTION(RNN_{out})
3: RNN_{out}=Permute(2,1)RNN_{out}; //permute the time and feature axis
4: α_p=Dense(timestep,softmax); //Calculate the attention weights
5: α_p=Permute(2,1)α_p;
6: $\text{Context}_{\text{vec}}$=Multiply($\text{RNN}_{\text{out}}$,$\alpha_p$); //Calculate the context vector
7: **return** $\text{Context}_{\text{vec}}$;
8: **end procedure**
9: **procedure** MAIN()
10: init model(); //Initialize the model framework
11: **for** i=1 to batch **do**
12: augmentImage(I_i); //Augment text line images
13: CNN_i=(I_i); //Extract features of the Image
14: Reshape_i=Reshape(CNN_i); //Reshape output of CNN for further proc.
15: $\text{RNN}_{\text{inp}-i}$=Attention($Reshape_i$); //Attention module
16: $\hat{y_i}$=RNN($\text{RNN}_{\text{inp}-i}$); //Processing RNN Layers
17: $\hat{y_i}$=Dense(timestep,$Num_{\text{char}} + 1$(for CTC blank)); //Finding the character occurrence at each time step
18: δ_{ctc}+=L_{ctc}(y_i,$\hat{y_i}$); //Computing CTC loss
19: **end for**
20: Backward(δ_{ctc}); //Updating model weights using backpropagation
21: **end procedure**

Line 1: Define training model parameters such as batch size learning rate, early stopping criteria and the total number of epochs.

Line 3: Permute the dimension of the output to enable feature exchange.

Line 4: Apply the dense() layer along the time step.

Line 5: Permute back the attention vector

Line 6: Obtaining the context vector by multiplying the features with the attention weights for each step.

Line 7: Returns the context vector to the main function to be further processed by RNN layers.

Line 10: Load the NN model.

Line 12: For a given batch, augment the preprocessed image.

Line 13: Extract the features of the image using a series of convolutions and gated convolutions operations.
Line 14: Converting 4D feature maps to 3D vectors to be further processed by attention and RNN layers.
Line 15: Applied the attention as defined above and obtained the context vector.
Line 16: Find the predicted character occurrence from the output of RNN
Line 17: Map the output of RNN to the number of characters of dataset + 1 for the sequence prediction.
Line 18: Compute the CTC loss from predicted and actual character sequence.
Line 20: Based upon the Loss value, train the NN system using backpropagation.

Algorithm 4.2 Prediction process

Input Text line image I, $E_{\text{test}_{\text{corpus}}}$, E_{chars}, $E_{\text{wordchars}}$
Result Prediction of image text with CER and WER

1: BW=50, mode='NGrams', smooth=0.01; //Initializing the WBS decoding params.
2: initNNModel(); //Loading of the trained model with all the layers
3: output=Modelpredict(I); //Predicting the text in the Image
4: $\hat{y}$= Decoder(BW,mode,smooth = 0.01, $D_{\text{test}_{\text{corpus}}}$, D_{chars}, $D_{\text{wordchars}}$); //Applying WBS decoding algorithm
5: CER,WER = accuracy(y,$\hat{y}$); //compute CER and WER

Explanation We will discuss prediction process as in algorithm 4.2 in line by line manner, as follows,
Line 1: Input text line image I. First, initialize the parameters of WBS decoding.
Line 2: Build the model.
Line 3: Process the image as per the trained model.
Line 4: RNN's output dimensions are swapped as per predefined input accepted by the WBS decoder.
Line 5: Computation of character occurrence using WBS decoding algorithm.
Line 6: Estimate the accuracy of the model on test images.

5. Results and Comparison

In this section, the results obtained in the present study have been discussed and compared with the other state-of-the-art methods. This HTR system recognizes the handwritten text on the line level, so the results are compared with other state-of-the-art line level systems. We are able to achieve 4.12% CER and 9.72% WER on the IAM dataset, and 7.07% CER and 16.14% WER on the GW dataset having Flor et al. as our based model and WBS as a decoding algorithm. We have also implemented Shi

et al. architecture and merged the attention module with it. Similar improvements were observed in that architecture reported in Table 4. The greedy decoder was used in this system. The given attention module is providing a 23.27% improvement with respect to the basic model.

Tab. 2. Comparison of present work with other state-of-the-art line level works on IAM Dataset

S No.	Reference	Method	CER	WER
1	Puigcerver et al. [45]	CNN + LSTM + CTC	4.4	12.2
2	Chowdhury et al. [12]	CNN + BLSTM + LSTM	8.1	16.7
3	Michael et al. [41]	CNN + LSTM + Attention	4.87	-
4	Kang et al. [31, 32]	Transformer	4.67	15.45
5	Yousef et al. [58]	CNN + CTC	4.9	-
6	Flor et al. [15]	CNN + BGRU + CTC	3.72	11.18
7	Present Work	CNN + Attention + BGRU + CTC	**4.15**	**9.72**

Tab. 3. Comparison of present work with other state-of-the-art line level works on GW Dataset

S No.	Reference	Method	CER	WER
1	Toledo et al. [54]	CNN + BLSTM + CTC	7.32	-
2	Almazan et al. [1]	Word Embedding	17.40	-
3	Fischer et al. [20]	HMM + RNN	20	-
4	Present Work	CNN + Attention + BGRU + CTC	**7.07**	**16.14**

Tab. 4. Similar NN System to Shi et al. recreated and attention module applied CNN as in Flor et al. except using greedy decoder instead of WBS

Architecture	CER(in %)
Model Based Upon Shi et. al	11.00
Model Based Upon Shi et al + Attention	8.44

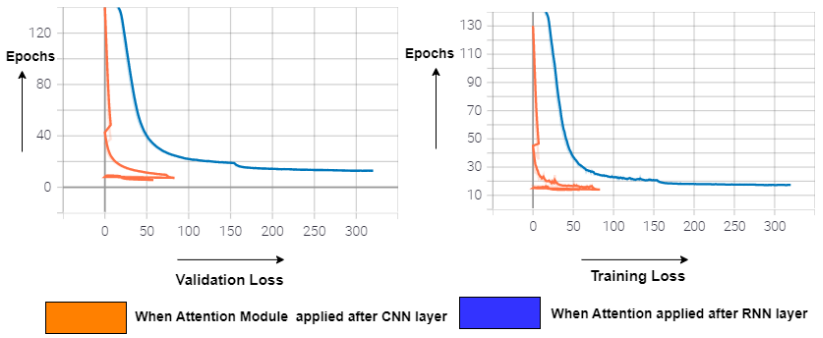


Fig. 3. Comparison of Training and Validation loss at different places of attention module

6. Discussion

In the proposed study, the attention mechanism helped in identifying relevant features in the given input image. We have experimentally found two possible positions where this attention block can be plugged in. Through extensive experiments and as per the graphs shown in Fig. 3, it is evident that the use of the attention mechanism before the recurrent layers and after the CNN layer helps the model to converge quicker with better accuracy. As shown in Fig. 3, with the same hyper-parameters for training, the model learns better and converges quickly when the attention module is applied after CNN layers.

7. Conclusion

In the present study, we have merged attention with two state-of-the-art NN systems that are Flor et al. and a small version of Shi et al. We were able to achieve 4.15% CER and 9.72% WER on the IAM dataset, and 7.07% CER and 16.14% WER on the GW dataset. We have also observed a 23.17% improvement in CER from the base model by applying attention module in the NN system similar to Shi et al. system. The accuracies obtained after applying the attention module favour our hypothesis that attention helps in learning the image features better. The position of applying the attention module is a critical step to consider, which we addressed in the discussion section. In future, we will work on extending this work to page or paragraph level.

Acknowledgements
This research is funded by the Government of India, University Grant Commission, under the Senior Research Fellowship scheme. The authors acknowledge PRL's supercomputing resource Vikram-100 (https://www.prl.res.in/prl-eng/hpc/vikram_hpc) made available for conducting the research reported in this paper.

References

[1] J. Almazan, A. Gordo, A. Fornes, and E. Valveny. Word spotting and recognition with embedded attributes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(12):2552–2566, 2014. doi:10.1109/TPAMI.2014.2339814.

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *Proc. 3rd Int. Conf. Learning Representations, ICLR 2015*, San Diego, CA, 7-9 May 2015. Accessible in arXiv. doi:10.48550/arXiv.1409.0473.

[3] R. E. Bellman and S. E. Dreyfus. *Applied Dynamic Programming*, volume 2050 of *Princeton Legacy Library*. Princeton University Press, 2015. doi:10.1515/9781400874651.

[4] A.-L. Bianne-Bernard, F. Menasri, Al-Hajj M. R., et al. Dynamic and contextual information in HMM modeling for handwritten word recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(10):2066–2080, 2011. doi:10.1109/TPAMI.2011.22.

[5] T. Bluche. Joint line segmentation and transcription for end-to-end handwritten paragraph recognition. *arXiv*, 2016. arXiv:1604.08352. doi:10.48550/arXiv.1604.08352.

[6] T. Bluche. Joint line segmentation and transcription for end-to-end handwritten paragraph recognition. In *Advances in Neural Information Processing Systems 29 – Proc. 30th Conf. NIPS 2016*, volume 29, pages 838–846, Barcelona, Spain, 5-10 Dec 2019. Curran Associates, Inc. `https://proceedings.neurips.cc/paper/2016/file/2bb232c0b13c774965ef8558f0fbd615-Paper.pdf`.

[7] T. Bluche, J. Louradour, and R. Messina. Scan, Attend and Read: End-to-end handwritten paragraph recognition with MDLSTM attention. *arXiv*, 2016. arXiv:1604.03286. doi:10.48550/arXiv.1604.03286.

[8] T. Bluche, J. Louradour, and R. Messina. Scan, Attend and Read: End-to-end handwritten paragraph recognition with MDLSTM attention. In *Proc. 2017 14th IAPR Int. Conf. Document Analysis and Recognition (ICDAR)*, pages 1050–1055, Kyoto, Japan, 9-15 Nov 2017. IEEE. doi:10.1109/ICDAR.2017.174.

[9] T. Bluche, H. Ney, and C. Kermorvant. Tandem HMM with convolutional neural network for handwritten word recognition. In *Proc. 2013 IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, pages 2390–2394, Vancouver, Canada, 26-31 May 2013. IEEE. doi:10.1109/ICASSP.2013.6638083.

[10] K.-N. Chen, C.-H. Chen, and C.-C. Chang. Efficient illumination compensation techniques for text images. *Digital Signal Processing*, 22(5):726–733, 2012. doi:10.1016/j.dsp.2012.04.010.

[11] W.-T. Chen, P. Gader, and H. Shi. Lexicon-driven handwritten word recognition using optimal linear combinations of order statistics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(1):77–82, 1999. doi:10.1109/34.745738.

[12] A. Chowdhury and L. Vig. An efficient end-to-end neural model for handwritten text recognition, 2018. arXiv:1807.07965v2. doi:10.48550/arXiv.1807.07965.

[13] D. Coquenet, Y. Soullard, C. Chatelain, and T. Paquet. Have convolutions already made recurrence obsolete for unconstrained handwritten text recognition? In *Proc. 2019 Int. Conf. Document Analysis and Recognition Workshops (ICDARW)*, volume 5, pages 65–70, Sydney, NSW, Australia, 20-25 Sep 2019. doi:10.1109/ICDARW.2019.40083.

[14] A. Das, J. Li, G. Ye, et al. Advancing acoustic-to-word CTC model with attention and mixed-units. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(12):1880–1892, 2019. doi:10.1109/TASLP.2019.2933325.

[15] A. F. de Sousa Neto, B. L. D. Bezerra, A. H. Toselli, and E. B. Lima. HTR-Flor: A deep learning system for offline handwritten text recognition. In *Proc. 2020 33rd SIBGRAPI Conference on*

Graphics, Patterns and Images (SIBGRAPI), pages 54–61, Porto de Galinhas, Brazil, 07-10 Nov 2020. doi:10.1109/SIBGRAPI51738.2020.00016.

[16] A. Flor de Sousa Neto. handwritten-text-recognition. GitHub repository, 2020. `https://github.com/arthurflor23/handwritten-text-recognition`.

[17] P. Doetsch, M. Kozielski, and H. Ney. Fast and robust training of recurrent neural networks for offline handwriting recognition. In *Proc. 2014 14th Int. Conf. Frontiers in Handwriting Recognition (ICFHR)*, pages 279–284, Hersonissos, Greece, 01-04 Sep 2014. IEEE. doi:10.1109/ICFHR.2014.54.

[18] P. Dreuw, P. Doetsch, C. Plahl, and H. Ney. Hierarchical hybrid MLP/HMM or rather MLP features for a discriminatively trained gaussian HMM: A comparison for offline handwriting recognition. In *2011 18th IEEE Int. Conf. Image Processing (ICIP)*, pages 3541–3544, Brussels, Belgium, 11-14 Sep 2011. IEEE. doi:10.1109/ICIP.2011.6116480.

[19] S. España-Boquera, M. J. Castro-Bleda, J. Gorbe-Moya, and F. Zamora-Martinez. Improving offline handwritten text recognition with hybrid HMM/ANN models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(4):767–779, 2010. doi:10.1109/TPAMI.2010.141.

[20] A. Fischer. *Handwriting Recognition in Historical Documents*. PhD thesis, Universität Bern, Switzerland, 13 Mar 2012. `https://www.researchgate.net/publication/259346163`.

[21] A. Fischer, A. Keller, V. Frinken, and H. Bunke. Lexicon-free handwritten word spotting using character HMMs. *Pattern Recognition Letters*, 33(7):934–942, 2012. Special Issue on Awards from ICPR 2010. doi:10.1016/j.patrec.2011.09.009.

[22] A. Fischer, K. Riesen, and H. Bunke. Graph similarity features for HMM-based handwriting recognition in historical documents. In *Proc. 2010 12th Int. Conf. Frontiers in Handwriting Recognition (ICFHR)*, pages 253–258, Kolkata, India, 16-18 Nov 2010. IEEE. doi:10.1109/ICFHR.2010.47.

[23] V. Frinken and S. Uchida. Deep BLSTM neural networks for unconstrained continuous handwritten text recognition. In *Proc. 2015 13th Int. Conf. Document Analysis and Recognition (ICDAR)*, pages 911–915, Tunis, Tunisia, 23-26 Aug 2015. IEEE. doi:10.1109/ICDAR.2015.7333894.

[24] A. Giménez, I. Khoury, J. Andrés-Ferrer, and A. Juan. Handwriting word recognition using windowed Bernoulli HMMs. *Pattern Recognition Letters*, 35:149–156, 01 2014. doi:10.1016/j.patrec.2012.09.002.

[25] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *ICML '06: Proc. 23rd Int. Conf. Machine Learning*, pages 369–376, Pittsburgh, PA, USA, 25-29 Jun 2006. doi:10.1145/1143844.1143891.

[26] A. Graves and N. Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *Proc. 31st Int. Conf. Machine Learning (ICML'14)*, volume 32 of *ACM Proceedings*, pages II–1764–II–1772, Beijing, China, 21-26 Jun 2014. JMLR.org. `https://dl.acm.org/doi/abs/10.5555/3044805.3045089`.

[27] A. Graves, M. Liwicki, S. Fernández, et al. A novel connectionist system for unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):855–868, 2009. doi:10.1109/TPAMI.2008.137.

[28] A. Graves and J. Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in Neural Information Processing Systems 21 – Proc. 22nd Conf. NeurIPS 2008*, volume 21, pages 545–552. Curran Associates, Inc., 2008. `https://proceedings.neurips.cc/paper/2008/file/66368270ffd51418ec58bd793f2d9b1b-Paper.pdf`.

[29] Keras Special Interest Group. Keras. simple. flexible. powerful. `https://keras.io`.

[30] S. Johansson, G. N. Leech, and H. Goodluck. *Manual of Information to accompany the Lancaster-Oslo/Bergen Corpus of British English, for use with digital Computers*. Department of English, University of Oslo, Oslo, Norway, 1978.

[31] L. Kang, P. Riba, M. Rusiñol, et al. Pay attention to what you read: Non-recurrent handwritten text-line recognition. *arXiv*, 2020. arXiv:2005.13044. doi:10.48550/arXiv.2005.13044.

[32] L. Kang, P. Riba, M. Rusiñol, et al. Pay attention to what you read: Non-recurrent handwritten text-line recognition. *Pattern Recognition*, 129:108766, 2022. doi:10.1016/j.patcog.2022.108766.

[33] G. Kim, V. Govindaraju, and S. N. Srihari. An architecture for handwritten text recognition systems. *International Journal on Document Analysis and Recognition*, 2(1):37–44, 1999. doi:10.1007/s100320050035.

[34] M. Kozielski, P. Doetsch, and H. Ney. Improvements in RWTH's system for off-line handwriting recognition. In *Proc. 2013 IAPR 12th Int. Conf. Document Analysis and Recognition (ICDAR)*, pages 935–939, Washington, DC, USA, 25-28 Aug 2013. IEEE. doi:10.1109/ICDAR.2013.190.

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

[36] L. Kumari and A. Sharma. A review of deep learning techniques in document image word spotting. *Archives of Computational Methods in Engineering*, 29(2):1085–1106. doi:10.1007/s11831-021-09605-7.

[37] L. Kumari, S. Singh, and A. Sharma. Page level input for handwritten text recognition in document images. In J. H. Kim et al., editors, *Proc. 7th Int. Conf. Harmony Search, Soft Computing and Applications (ICHSA)*, volume 140 of *Lecture Notes on Data Engineering and Communications Technologies*, pages 171–183, Seoul, South Korea, 23-24 Feb 2022. Springer Nature Singapore. doi:10.1007/978-981-19-2948-9_17.

[38] Y. Le Cun, B. Boser, J. S. Denker, et al. Handwritten digit recognition with a back-propagation network. In *Advances in Neural Information Processing Systems 2 – Proc. Conf. NeurIPS 2008*, volume 2, page 396–404, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc. `https://proceedings.neurips.cc/paper/1989/file/53c3bce66e43be4f209556518c2fcb54-Paper.pdf`.

[39] M.-T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. In *Proc. EMNLP 2015*, Lisbon, Portugal, 17-21 Sep 2015. Accessible in arXiv. doi:10.48550/ARXIV.1508.04025.

[40] U.-V. Marti and H. Bunke. The IAM-database: an English sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*, 5(1):39–46, 2002. doi:10.1007/s100320200071.

[41] J. Michael, R. Labahn, T. Grüning, and J. Zöllner. Evaluating sequence-to-sequence models for handwritten text recognition. In *Proc. 2019 IAPR Int. Conf. Document Analysis and Recognition (ICDAR)*, pages 1286–1293, Sydney, NSW, Australia, 20-25 Sep 2019. IEEE. doi:10.1109/ICDAR.2019.00208.

[42] J. Poulos and R. Valle. Character-based handwritten text transcription with attention networks. *Neural Computing and Applications*, 33(16):10563–10573, 2021. doi:10.1007/s00521-021-05813-1.

[43] A. Poznanski and L. Wolf. CNN-N-Gram for handwriting word recognition. In *Proc. 2016 IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 2305–2314, Las Vegas, NV, USA, 27-30 Jun 2016. doi:10.1109/CVPR.2016.253.

[44] R. Ptucha, F. Petroski Such, S. Pillai, et al. Intelligent character recognition using fully convolutional neural networks. *Pattern Recognition*, 88:604–613, 2019. doi:10.1016/j.patcog.2018.12.017.

[45] J. Puigcerver. Are multidimensional recurrent layers really necessary for handwritten text recognition? In *Proc. 2017 14th IAPR Int. Conf. Document Analysis and Recognition (ICDAR)*, pages 67–72, Kyoto, Japan, 9-15 Nov 2017. IEEE. doi:10.1109/ICDAR.2017.20.

[46] H. Sak, A. Senior, and F. Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Proc. Annual Conf. of the International Speech Communication Association (Interspeech)*, pages 338–342, Singapore, 14-18 Sep 2014. doi:10.21437/Interspeech.2014-80.

[47] J. Sauvola and M. Pietikäinen. Adaptive document image binarization. *Pattern Recognition*, 33(2):225–236, 2000. doi:10.1016/S0031-3203(99)00055-2.

[48] H. Scheidl. CTCWordBeamSearch. GitHub repository, 2019. `https://github.com/githubharald/CTCWordBeamSearch`.

[49] H. Scheidl, S. Fiel, and R. Sablatnig. Word Beam Search: A connectionist temporal classification decoding algorithm. In *Proc. 2018 16th Int. Conf. Frontiers in Handwriting Recognition (ICFHR)*, pages 253–258, Niagara Falls, NY, USA, 5-8 Aug 2018. IEEE. doi:10.1109/ICFHR-2018.2018.00052.

[50] B. Shi, X. Bai, and C. Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *arXiv*, 2015. arXiv:1507.05717. doi:10.48550/arXiv.1507.05717.

[51] B. Shi, X. Bai, and C. Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(11):2298–2304, 2017. doi:10.1109/TPAMI.2016.2646371.

[52] F. Such Petroski, D. Peri, F. Brockler, et al. Fully convolutional networks for handwriting recognition. In *Proc. 2018 16th Int. Conf. Frontiers in Handwriting Recognition (ICFHR)*, pages 86–91, Niagara Falls, NY, USA, 5-8 Aug 2018. IEEE. doi:10.1109/ICFHR-2018.2018.00024.

[53] D. Suryani, P. Doetsch, and H. Ney. On the benefits of convolutional neural network combinations in offline handwriting recognition. In *Proc. 2016 15th Int. Conf. Frontiers in Handwriting Recognition (ICFHR)*, pages 193–198, Shenzhen, China, 23-26 Oct 2016. IEEE. doi:10.1109/ICFHR.2016.0046.

[54] J. I. Toledo, S. Dey, A. Fornes, and J. Llados. Handwriting recognition by attribute embedding and recurrent neural networks. In *Proc. 2017 14th IAPR Int. Conf. Document Analysis and Recognition (ICDAR)*, volume 01, pages 1038–1043, Kyoto, Japan, 9-15 Nov 2017. IEEE. doi:10.1109/ICDAR.2017.172.

[55] A. Vinciarelli. A survey on off-line cursive word recognition. *Pattern Recognition*, 35(7):1433–1446, 2002. doi:10.1016/S0031-3203(01)00129-7.

[56] A. Vinciarelli and J. Luettin. A new normalization technique for cursive handwritten words. *Pattern Recognition Letters*, 22(9):1043–1050, 2001. doi:10.1016/S0167-8655(01)00042-3.

[57] M. Yousef and T. Bishop. OrigamiNet: Weakly-supervised, segmentation-free, one-step, full page text recognition by learning to unfold. In *Proc. 2020 IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 14698–14707, Seattle, WA, USA, 13-19 Jun 2020. IEEE. doi:10.1109/CVPR42600.2020.01472.

[58] M. Yousef, K. F. Hussain, and U. S. Mohammed. Accurate, data-efficient, unconstrained text recognition with convolutional neural networks. *Pattern Recognition*, 108:107482, 2020. doi:10.1016/j.patcog.2020.107482.

Lalita Kumari is pursuing her PhD in Computer Science with the Department of Computer Science and Applications at Panjab University Chandigarh under UGC Senior Research Fellowship Scheme. Her research interest includes Machine Learning and Pattern Recognition in the area of cursive handwriting recognition. She is also interested in other areas of Pattern Recognition, Machine Learning and Image Processing.
ORCID: 0000-0002-4406-3324 .

Sukhdeep Singh is a doctor in Computer Science and works at DM college Moga. He has done his postdoctoral research at Boise State University, USA and received PhD in Computer Science from DCSA Panjab University Chandigarh. He has also done his MCA from DCSA Panjab University. His research interest includes Pattern Recognition and Machine Learning, especially in handwritten text recognition.
Homepage: https://sites.google.com/site/sransingh13/ .

Vaibhav Varish Singh Rathore is working with PRL Ahmedabad. He graduated from the National Institute of Technology Allahabad, India. His research interests include Pattern Recognition, Machine Learning, High-Performance Computing, Networking and Cyber Security.
ORCID: 0000-0003-2045-5339 .
Homepage: https://www.prl.res.in/~vaibhav .

Anuj Sharma is working with DCSA Panjab University Chandigarh. He has done his Post-Doc at RWTH Aachen, Germany and received PhD in Computer Science from Thapar Institute of Engineering and Technology, India. His research interests include Pattern Recognition and Machine Learning in the areas of Document Analysis, Handwriting and Image Recognition.
Homepage: https://anuj-sharma.in .

Person Re-Identification Accuracy Improvement by Training a CNN with the New Large Joint Dataset and Re-Rank

Rykhard Bohush[1], Sviatlana Ihnatsyeva[1], Sergey Ablameyko[2,3]
[1] *Polotsk State University, Novopolotsk, Belarus*
[2] *Belarusian State University, Minsk, Belarus*
[3] *United Institute for Informatics Problems of NAS of Belarus, Minsk, Belarus*
rbohush@gmail.com

Abstract. The paper is aimed to improve person re-identification accuracy in distributed video surveillance systems based on constructing a large joint image dataset of people for training convolutional neural networks (CNN). For this aim, an analysis of existing datasets is provided. Then, a new large joint dataset for person re-identification task is constructed that includes the existing public datasets CUHK02, CUHK03, Market, Duke, MSMT17 and PolReID. Testing for re-identification is performed for such frequently cited CNNs as ResNet-50, DenseNet121 and PCB. Re-identification accuracy is evaluated by using the main metrics Rank, mAP and mINP. The use of the new large joint dataset makes it possible to improve Rank1 mAP, mINP on all test sets. Re-ranking is used to further increase the re-identification accuracy. Presented results confirm the effectiveness of the proposed approach.

Key words: convolution neural network, PolReID, re-identification, large-scale dataset, re-rank.

1. Introduction

Due to the widespread use of intelligent video surveillance systems, the task of re-identifying a person in a distributed camera system is an urgent task. In general, re-identification is the task of identifying the person you are looking for in a different place or time using distributed video surveillance systems in a city [34]. Such a system extracts features of the analyzed person and compares them with features of other persons in the existing dataset. Finding and highlighting effective distinguishing features manually is a long and time-consuming process, and for re-identification task, due to the ambiguity of appearance from several angles, lighting variations, different camera resolutions, occlusions, it would take an irrationally large amount of time [7]. That is why the re-identification task remained unresolved for a long time. The recent achievements in the field of deep learning, in particular, the development of convolutional neural networks (CNN) allowed to perform the re-identification process with sufficiently high accuracy and in a reasonable time.

Despite the successful application of CNN for this task, a large number of problems still have to be faced when developing a re-identification system. Feature extraction is a difficult process in ReID system, due to the ambiguity of people appearance from various angles, lighting variations, different camera resolutions, occlusions.

CNN-based ReID systems are highly dependent on the size and quality of the datasets used for training. It is obvious that the dataset must contain a large amount of various data, and the larger the dataset, the more accurate the re-identification result will be. It is also desirable to use a dataset that will have the maximum similarity to the data that the re-identification algorithm will have to work with.

Also, the data for training and testing should be independent and identically distributed. However, experiments show that such models are well suited for a training set and will perform poorly in an invisible domain [2].

This leads to the main problem: how to increase the training set without using additional data? One of the ways to do it is to add augmentation tools such as reflection vertically or horizontally, rotation, changes in brightness and contrast, color fluctuations and others to the existing dataset. The use of data augmentation enables to diversify the training set. With a small original dataset size, these transformations may also not be enough to obtain satisfactory re-identification results. Increasing training sample size and diversity can also be achieved by combining existing datasets.

So, the success of solving re-identification problem depends a lot from datasets that will be used for network training. That's why we detail consider this task here.

In this paper, we analyze the existing datasets and discuss a problem of increasing a data volume for person re-identification task. Approaches for increasing images number in datasets are described. Based on this analysis, we propose a new large joint dataset for person re-identification task. Our dataset includes the existing public datasets CUHK02, CUHK03, Market, Duke, MSMT17 and our collected PolReID. Testing for re-identification was performed for such famous CNNs as ResNet-50, DenseNet121 and PCB. Re-identification accuracy was evaluated by using the main metrics Rank, mAP and mINP. The use of the new large joint dataset allowed us to improve Rank1 mAP, mINP on all test sets.

2. Overview of datasets

It is known that if the training and test samples were obtained under the same video surveillance conditions, then the re-identification accuracy is higher than if these conditions are different. Such a problem is called domain shift [39], a partial solution of which can be the several datasets combination [12, 14, 22]. This is increasing the variety of training examples and allows to extract features that are more resistant to changes in the domain.

Various re-identification systems types use different datasets. For research systems, datasets are used that consist of individual person images – bounding boxes. The largest scale of them are CUHK01 [19], CUHK02 [18], CUHK03 [20], Market-1501 [41], DukeMTMC-ReID [25], MSMT17 [32]. For re-identification systems that use temporal information, sets include of people images obtained from several consecutive frames,

Tab. 1. Comparative table of datasets for re-identification.

Dataset	Number of cameras	Number of persons	Bounding boxes
PRW	6	932	34 304
CUHK-SYSU	6	8 432	96 143
MARS	6	1 261	1 191 003
LPW	3,4,4	2 731	592 438
Market1501	6	1 501	32 217
CUHN01	2	971	3 884
CUHN02	10	1 816	7 264
CUHN03	6	1 360	13 164
MSMT17	15	4 101	126 441
VIPeR	2	632	1 264
PRID	2	934	24 541
QMUL iLIDS	2	119	476
Airport	6	9 651	39 902
CUHK-PEDES	–	13 003	80 412
ICFG-PEDES	–	4 102	52 522
LR-PRID	2	100	200
LR-VIPeR	2	632	1 264
SYSU-MM01	6	491	38 271
RegDB	2	412	8 240
PKU-Sketch	2	200	400

tracklets. Tracklets are contained in such datasets as LPW [27], MARS [40]. More complex re-identification systems assume that person must first be detected in the image and only then identified. For such systems datasets are used, consisting of frames from video surveillance cameras, for example PRW [42], CUHK-SYSU [35]. Comparison of datasets is shown in Table 1.

Datasets CUHK02 and CUHK03 were generated on the University campus of Hong Kong. Each image for each person is obtained from two cameras. CUHK02 had two cameras at five different locations on campus, while CUHK03 had cameras installed in pairs at three locations. CUHK02 includes 7264 images for 1816 persons, CUHK03 has 13 164 bounding boxes for 1360 IDs. Images are not divided into test and training sets, and it is proposed to perform testing for images of 100 randomly selected persons, and use the rest for training. A small number of video surveillance scenes, the images number for each person, the random nature of the test images choice in each experiment can be attributed to the shortcomings of these datasets, which were partially taken into account when forming the Market-1501 dataset. Frames from six video surveillance cameras were used to form it located near the supermarket at Tsinhua University. The

dataset includes 32 668 images for 1501 people. In the test sample for 750 persons, there are 19 732 bounding boxes, including 2739 distractors, and the remaining images are assigned to the training sample. Distractors in Market-1501 include examples where the detector mistook some objects for believable person images. Additionally, upon request, 500 000 distractors can be obtained under the same conditions as the labeled images.

DukeMTMC-ReID is a subset of the Duke Multi-Target Multi-Camera (MTMC) dataset generated from video 85 minutes at 60 frames per second. Eight CCTV cameras were used, located on the Duke University campus. Duke MTMC is designed for multi-tracking and based on it is marked up for re-identification tasks. The algorithm is trained on 16 522 images of 702 persons and 17 661 of 702 other identities for testing.

Market-1501 and DukeMTMC-ReID are based on the frame received from outdoor security cameras, which limits the variety of lighting options to natural light only. This was taken into account when developing the MSMT17 dataset. For its formation, frames from twelve outdoor surveillance cameras and three cameras installed indoors were used. Video recording was carried out at different day times for four days. Person number in MSMT17 is 4101, for which there are 126 441 images. The training sample includes 32 621 images for 1041 people. The test sample includes 93 820 bounding boxes for 3060 persons.

For training and testing of heterogeneous re-identification systems special datasets are used, where text (CUHK-PEDES [17], ICFG-PEDES [5]), a low-resolution image (LR-PRID [21], LR-VIPeR [15]), an image from an infrared camera (SYSU-MM01 [33], RegDB [23] or drawing (PKU-Sketch [24]).

The CUHK-PEDES dataset [17] combines five existing datasets, such as CUHK03 [20], Market-1501 [41], dataset from [36], VIPeR [8] and CUHK01 [19], and each image is annotated with two text descriptions in English received from crowdsourcing employees, which can be used as a query. The text contains information about the appearance of a person, his actions, and poses. Each text description contains an average of 23.5 words. Another dataset for heterogeneous re-identification systems by text description is the ICFG-PEDES dataset [5], which contains an average of 37.2 words with a more detailed description of appearance than CUHK-PEDES, and is formed based on the MSMT17 dataset [32].

The datasets LR-PRID [21], LR-VIPeR [15] are formed on the basis of images from the datasets PRID [10] and VIPeR [8], respectively, and for each person there is a pair of images, one of which has a low resolution and the other high. These datasets are used for heterogeneous re-identification systems with images of different resolutions.

SYSU-MM01 [33] was obtained from two infrared and four RGB cameras. It contains 15 712 images from an infrared camera and 22 559 color images for 491 people. RegDB dataset [23] includes 10 color images taken during the day and 10 thermal images from a night IR camera for 412 people. Both sets are used in heterogeneous re-identification systems with images from infrared and RGB cameras.

In [24], a dataset is proposed, including two images from different cameras for two hundred people, and one sketch for a person. To create sketches, volunteers were involved, who described the people appearance to five artists, to train an open-world cross-modality re-identification system. Sketch drawn according to the description is used to search for a person whose photo is not in this system. Another dataset for open-world re-identification systems is the MPR Drone [16], which differs from traditional sets in that a flying drone camera takes people images. Since only one camera is used, the whole set consists of two parts. The first part is marked up for 113 610 detected bounding rectangles, and the second contains raw frames for the first part.

Paper [6] presents a large unlabeled LUPerson dataset for unsupervised training of re-identification systems, which was created using more than seventy thousand street videos from various cities, which includes more than four million images for two hundred thousand people.

3. Re-identification for different algorithms and datasets

As we know the accuracy of a person re-identification in a distributed video surveillance system is largely determined by the number and variety of the image database that is used in CNN training. An images set obtained under the same conditions in the same video surveillance system is called a domain. Such a set can be divided into two parts, the first one is used for training (source domain), and the second one (target) is used for testing. The target domain is invisible if it differs from the source. Each image in the dataset is affected by a combination of factors, including camera resolution, the same background, lighting conditions, and the person appearance.

Training on a single domain allows providing sufficiently high accuracy only for this system. However, when using such a trained CNN for another domain that is unknown (invisible) to the system, the re-identification accuracy will be significantly lower. Thus, if datasets obtained under different conditions were used for training and testing, then there is a problem of domain transfer (domain change) for real ReID systems. An increase in accuracy can be provided by the search for new methods and algorithms, as well as combining several datasets.

Domain adaptation approaches were considered in various papers. In [31] and [14] training results are shown for different datasets composition. Analysis of results shows that an increase in the training set has a positive effect on the re-identification accuracy. Thus, in [31], adding MSMT17 to the synthetic dataset used as a training sample allows increasing the re-identification accuracy by 12.7% in mAP for DukeMTMC-ReID. Data inclusion from target domain into training sample allows increasing Rank1 for MSMT17 by more than 45% in [31]. Similarly, in [14], the use of target domain images for training allows increasing mAP for Market-1501 from 33.9% to 82.3% and for DukeMTMC-ReID from 33.6% to 73.2%. Maximum accuracy in mAP among the considered algorithms

was achieved using the IDM algorithm [3,4] for the Market-1501 and MSMT17 datasets, which became possible using the features of intermediate domains generated during the learning process.

The JVTC approach [38] and the UnrealPerson synthetic dataset as training proved to be the most effective when using DukeMTMC-ReID as the target domain. Table 2 shows an efficiency of cross-domain person re-identification for different datasets.

Tab. 2. Efficiency of person re-identification for different algorithms and datasets.

Algorithm	Year	Training dataset	Metrics	Testing Dataset		
			(%)	Market	Duke	MSMT17
Open-ReID [31]	2020	RandPerson [31]	mAP	28.8	27.1	6.3
			Rank1	55.6	47.6	20.1
		RandPerson [31] +MSMT17	mAP	35.8	39.8	36.8
			Rank1	62.3	61.0	65.0
SNR [14]	2020	Market	mAp	84.7	33.6	-
			Rank1	94.4	55.1	-
		Duke	mAP	33.9	72.9	-
			Rank1	66.7	84.4	-
		Market+ Duke+CUHK+ MTMC17	mAP	82.3	73.2	-
			Rank1	93.4	85.5	-
NRMT [39]	2020	Market	mAP	–	62.3	19.8
			Rank1	–	78.1	43.7
		Duke	mAP	72.2	–	20.6
			Rank1	88.0	–	45.2
CBN [38]	2021	UnrealPerson [38]	mAP	54.3	49.4	15.3
			Rank1	79.0	69.7	38.5
JVTC [38]	2021	UnrealPerson [38]	mAP	80.2	75.2	34.8
			Rank1	93.0	88.3	68.2
QAConv [30]	2022	ClonedPerson [30]	mAP	21.8	–	18.5
			Rank1	22.6	–	49.1
IDM [3]	2022	Market	mAp	–	73.2	40.2
			Rank1	–	85.5	69.9
		Duke	mAP	85.3	–	40.5
			Rank1	94.2	–	69.5
		MSMT17	mAP	85.2	73.6	-
			Rank1	94.1	84.6	-

4. New large joint ReID dataset

At the first stage, we significantly increased the PolReID dataset [12] from 5609 images for 54 people to 52 035 bounding boxes for 657 people [13]. The training set includes 397 identities (32 516 bounding boxes), test sample – 259 IDs (19519 images). Examples of images are shown in Fig. 1. Statistical information for PolReID composition is given in Tab. 3.

To form PolReID, video sequences were obtained, the total duration of which was 8 hours 1 minute 53 seconds. Cameras in different locations from several angles took each person images. Various camera number from two to nine carried out video surveillance. In total, 839 ways of placing cameras with different characteristics (resolution, frames number per second) were used to create the set. Video surveillance was carried out under dissimilar weather conditions for four seasons. The cameras were installed indoors with natural and artificial lighting, outdoors for all day. The presence of various attributes in people in the form of bags, packages, backpacks, briefcases, grocery baskets, scarves, hats,

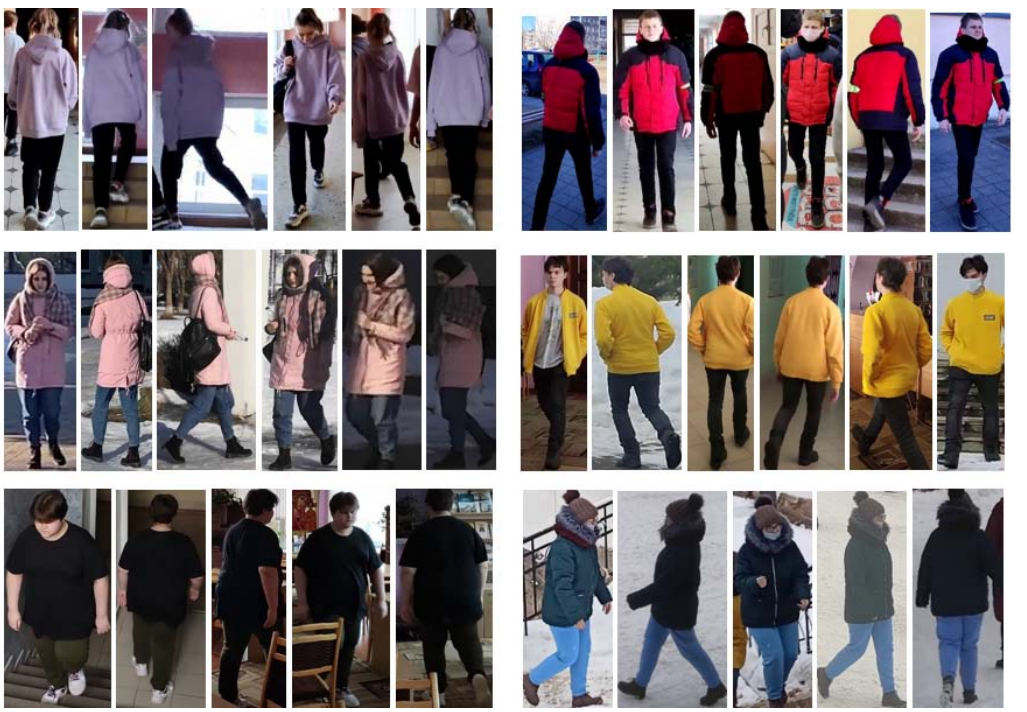


Fig. 1. Some images from PolReID dataset.

Tab. 3. PolReID composition.

Characteristic		Number of persons
Gender	Male	440
	Female	217
Age	18–30 years	524
	Over 30	133
Season	Summer	95
	Autumn and Spring	274
	Winter	288
Shooting conditions	Indoors	340
	Outdoors	214
	Outdoors and	103
	indoors	103
Mask availability	Masked	177
	Without mask	447
	Masked and unmasked	33

glasses, folders for papers, notebooks, phones, headphones, food and drinks changes over time, which makes it possible to study and take into account minor appearance changes. The person detection on frames and the formation of bounding boxes were performed using the YOLOv4 CNN [1, 29].

At the second stage, to increase training sample size and diversity, several common datasets were combined, such as CUHK02, CUHK03, Market-1501, DukeMTMC-ReID, MSMT17 and PolReID.

When merging several datasets, one has to face such difficulties as various file names in sets, sundry names and location of directories. In this regard, when adding each new set to the training sample, a separate software implementation of the renaming process all files and placing them in the appropriate folders was developed. The training set includes images from such directories as "train" and "train_all" (Fig. 2). Directory "train_all" includes also instance validation images to determine the accuracy in the training process. Each file has the same name format, and a character "_" separates each value. For example, in Fig. 2 two images of a person are shown. According to the first file name 00007_c01s06_028546_04.jpg, we can say that this is a person whose ID is 00007, obtained from frame 28546 of the sixth video sequence from the first surveillance camera. In total, four different people were present in this frame. For the second file, whose name is 00007_c02s03_071052_01.jpg, we can say that this is an image of the same person, but it was taken from frame number 71 052 in the third sequence from the second surveillance camera. The total images number in the training set of the joint dataset was 115 956 bounding boxes for 6174 people.

Fig. 2. The structure of the joint training dataset.

5. Metrics for re-identification

Choice of metrics is critical task for evaluating re-identification results. The most common group of metrics is RankN, which includes Rank1, Rank5, Rank10, and mAP.

RankN group characterizes ranking quality and shows the percentage of queries number for which the correct result was among the first N results. Accordingly, Rank1 metric shows the queries percentage for which the first candidate image ID matches the query ID. If $N = 5$, then Rank5 shows the queries percentage for which among the first five given candidate images there was a correct solution. For the first ten candidate images are considered. To calculate RankN, the number sum ratio queries for which the correct solution was found among the first returned results to the total queries number Q is determined:

$$\text{RankN} = \frac{\sum K_{i,N}}{Q}\,, \tag{1}$$

where i – query number, $K_{i,N}$ – i-th query for which the correct solution was found among the first N returned results.

Metric mAP estimates the mean value of the average precision for all queries and is calculated with the formula:

$$\text{mAP} = \frac{1}{Q}\sum_{i=1}^{Q}\text{AP}_i\,, \tag{2}$$

where Q – the total number of queries, AP – average precision defined as the area under the precision-recall curve, where $\text{pr} = \frac{\text{TP}}{\text{TP+FP}}$ – precision, TP – number of true positive

identifications or simply true positive, FP – false positive, rc = $\frac{\mathrm{TP}}{\mathrm{TP+FN}}$ – recall, FN – false negative.

In re-identification systems, it is a priority that the correct predictions are at the beginning of ranked list and have as few false predictions as possible. It should be noted that RankN and mAP metrics do not reflect the difficulty of finding correctly identified person images for a request. With the same Rank metrics for different requests, the AP accuracy for them may differ. To take into account the search for the hardest match correct predictions, mINP (mean Inverse Negative Penalty) metric is proposed in [37]. Analysis of this metric makes it possible to eliminate the dominance of light matches that affect the Rank and mAP metrics. Additional metrics are introduced for calculation mINP: Negative Penalty (NP) assigned for incorrect predictions for the i-th query and reducing correct re-identification probability if the most difficult match is incorrectly found and Inverse Negative Penalty (INP) – the reciprocal for NP. Growth NP indicates an improvement in system performance. mINP characterizes the average INP value for all requests and is calculated as:

$$\mathrm{mINP} = \frac{1}{Q}\sum_i (1 - \mathrm{NP}_i) = \frac{1}{Q}\sum_i \left(1 - \frac{R_i^{\mathrm{hard}} - |G_i|}{R_i^{\mathrm{hard}}}\right) = \frac{1}{Q}\sum_i \frac{|G_i|}{R_i^{\mathrm{hard}}}, \tag{3}$$

where Q – total number of queries, $\mathrm{NP}_i = \frac{R_i^{\mathrm{hard}} - |G_i|}{R_i^{\mathrm{hard}}}$ – Negative Penalty, R_i^{hard} – position of hardest correct prediction, $|G_i|$ – total correct predictions number for the query.

INP enables to evaluate all correct matches finding complexity. The larger this value, the better the system is at finding all people with the same ID. Accordingly, one should strive to reduce NP and reduce the distance from ranking list beginning to position of the most difficult image to search for, which may be incorrectly identified.

6. Training and testing

To test re-identification task in our experiments, we used the algorithm from [43, 44]. Training on datasets was carried out for 60 epochs at learning rate of 0.05 and batch size of 32. During the learning process from 30 to 50 epochs fluctuations have been observed around in the Top1 error minimum. Top1 error rate indicates how many times the CNN has predicted the correct label with the highest probability. Therefore, to get as close as possible to the minimum of the function after epoch 40, the learning rate should be decreased by a factor of 0.1. Figure 3 shows Top1 error graphs during training the re-identification model.

The work [26] considered only the effect of reducing the learning rate on the value of the loss function. Experimental results for the re-identification accuracy for several datasets and CNN are presented in Table 4. Three CNN DenseNet-121 [11], ResNet-50 [9], PCB [28] have been used for feature extraction.

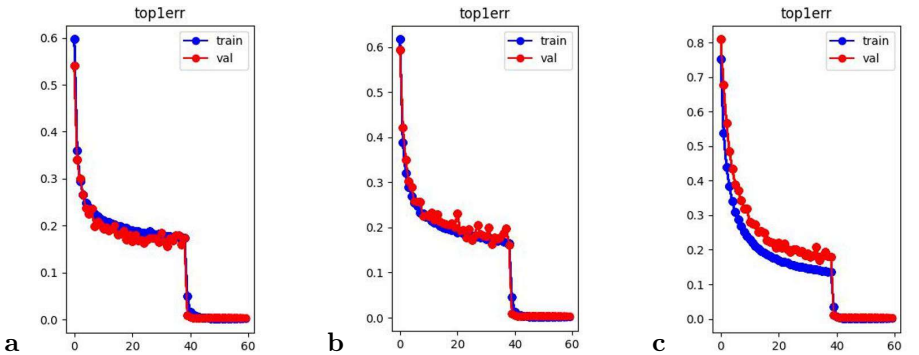


Fig. 3. Top1 error graphs of training and validation. (**a**) For DenseNet-121; (**b**) For ResNet-50; (**c**) For PCB.

Tab. 4. Experimental results for different datasets and CNN.

Dataset for train		Dataset for test								
		Market-1501			DukeMTMC-ReID			PolReID		
	Metrics	DenseNet	ResNet	PCB	DenseNet	ResNet	PCB	DenseNet	ResNet	PCB
Market-1501	Rank1(%)	88.9	83.3	92.7	37.2	30.6	40.4	63.7	57.6	62.6
	mAP(%)	73.0	71.2	77.7	20.2	15.9	22.2	34.6	29.4	35.3
	mINP	0.41	0.39	0.4	0.03	0.02	0.03	0.06	0.04	0.05
Duke MTMC -ReID	Rank1(%)	49.2	43.9	55.1	81.5	79.0	84.9	74.2	67.9	72.2
	mAP(%)	21.7	18.9	25.9	64.8	62.4	70.3	43.4	37.2	40.8
	mINP	0.03	0.02	0.03	0.27	0.25	0.30	0.08	0.06	0.07
MSMT 17	Rank1(%)	54.2	48.5	55.5	55.6	50.8	54.4	83.6	79.7	86.4
	mAP(%)	26.4	22.8	25.7	34.5	30.8	33.3	58.1	52.9	60.6
	mINP	0.05	0.04	0.04	0.06	0.06	0.06	0.13	0.11	0.14
PolReID	Rank1(%)	49.7	39.7	47.4	54.8	47.6	51.1	93.9	91.2	94.2
	mAP(%)	24.7	18.2	21.1	33.2	27.2	29.2	77.4	74.2	83.3
	mINP	0.05	0.03	0.03	0.06	0.04	0.03	0.27	0.25	0.34
Presen-ted Dataset	Rank1(%)	**94.1**	92.1	93.1	**86.5**	84.2	86.4	95.3	94.1	**95.4**
	mAP(%)	**83.3**	80.6	81.6	**74.0**	71.2	73.9	83.8	80.9	**84.7**
	mINP	**0.57**	0.51	0.49	**0.39**	0.35	0.37	**0.35**	0.32	0.35

Experiments had two stages. At the first stage, unlike [26] CNN training is carried out not only on Market-1501, DukeMTMC-ReID, MSMT17, but also on PolReID dataset, and accuracy is assessed using three metrics as mAP, Rank1, and mINP. The tests were performed for different domains. The obtained results show that re-identification accuracy decreases significantly for invisible domains. The maximal values of Rank1, mAP and mINP metrics for cross-domain re-identification have been obtained by training on dataset MSMT17 and testing on PolReID, Rank1 = 86.38%, mAP = 60.62%, mINP = 0.142.

This can be explained by the fact that MSMT17 includes the largest number of people images, which were obtained under different conditions (Tab. 1). PolReID and MSMT17 include data obtained by indoor and outdoor surveillance cameras, by using various lighting and weather conditions, etc.

At the second stage, a large joint set that included CUHK02, CUHK03, Market-1501, DukeMTMC-ReID, PolReID, MSMT17 was used for training. The training and test samples do not overlap. This approach allowed increasing re-identification accuracy for all three metrics. Maximum values were obtained for PolReID: Rank1 = 95.41%, mAP = 84.74%, mINP = 0.345. In [12], the joined training set was also used, and despite the fact that its size was larger (8 690 IDs/537 109 images), the results in the current experiment turned out to be better for similar test sets. This is primarily due to the fact that in [12] the LPW dataset [27] consisting of tracklets was included in the training set. A large images number with similar features led to an uneven distribution of the training examples variety.

PCB network is the most effective when the source and target domains match, as well as for PolReID and Market1501 for cross-domain re-identification. DenseNet-121 is the most effective for DukeMTMC-ReID, Market-1501 and DukeMTMC-ReID when trained on a jointed dataset.

Another key difference from [26] is that re-ranking is performed to further improve the accuracy of re-identification [45]. The k-inverse feature vector is computed for the image. To calculate it, k-inverse nearest neighbors are used by the Jaccard distance:

$$d(p, g_i) = 1 - \frac{|R^*(p,k) \cap R^*(g_i,k)|}{|R^*(p,k) \cup R^*(g_i,k)|}, \tag{4}$$

where $R^*(p,k)$ and $R^*(g_i,k)$ – k-reciprocal nearest neighbors, p – query, g – gallery image.

Therefore, it is of interest to study this approach for the used CNN and datasets. However, PCB conducts uniform partition on the conv-layer for learning part-level features. It does not explicitly partition the images.

After passing through the network, the feature vector enters the classification layer. During testing, pieces are concatenated to form the final descriptor of the input image.

Tab. 5. Experimental results for person re-identification with re-rank.

Dataset for train		Dataset for test					
		Market-1501		DukeMTMC-ReID		PolReID	
	Metrics	DenseNet	ResNet	DenseNet	ResNet	DenseNet	ResNet
Market-1501	Rank1(%)	91.75	90.83	44.30	36.49	68.65	60.27
	mAP(%)	86.52	84.92	33.57	25.80	46.09	39.53
	mINP	0.709	0.675	0.094	0.064	0.125	0.092
DukeMTMC-ReID	Rank1(%)	53.15	47.68	85.55	83.03	77.92	72.20
	mAP(%)	32.95	27.46	80.82	78.11	56.41	48.65
	mINP	0.098	0.068	0.567	0.526	0.155	0.116
MSMT17	Rank1(%)	58.58	52.02	61.89	57.99	87.27	83.24
	mAP(%)	39.23	33.18	51.06	46.33	71.31	65.96
	mINP	0.146	0.112	0.188	0.159	0.250	0.211
PolReID	Rank1(%)	53.95	43.71	61.98	54.26	94.44	91.86
	mAP(%)	35.25	25.80	49.44	41.83	85.23	82.42
	mINP	0.115	0.066	0.162	0.121	0.403	0.390
Presented Dataset	Rank1(%)	**94.80**	94.12	**88.73**	87.34	**96.45**	95.57
	mAP(%)	**92.01**	90.41	**86.03**	84.04	**90.85**	88.77
	mINP	**0.833**	0.788	**0.663**	0.613	**0.495**	0.457

This makes the CNN sensitive to the content of each part and leads to significant errors when searching for the k-reciprocal nearest neighbors to the query image.

The increase in mINP indicates that image number at rank list top has increased. The growth of this metric indicates the positive impact of this approach on re-identification (Tabs. 4 and 5).

Table 5 shows re-rank tests results only for the DenseNet-121 and ResNet-50 re-rank. The most efficient CNN using re-rank based on test results is DenseNet-121. Training on the joint dataset increased the accuracy scores for PolReID Rank1 = 96.45%, mAP = 90.85%, mINP = 0.495.

In [14] for the jointed set including Market-1501, DukeMTMC-ReID, CUHK and MSMT17 research was performed: for Market-1501 Rank1 = 93.4%, mAP = 82.3%, for DukeMTMC-ReID metrics Rank1 = 85.5, mAP = 73.2.

Our approach for Market-1501 dataset increases to Rank1 = 94.80%, mAP = 92.01% and for DukeMTMC-ReID to Rank1 = 88.73%, mAP = 86.03% (Tab. 5). Thus, the proposed approach makes it possible to obtain improved accuracy for people re-identification.

7. Conclusion

Convolutional neural networks are now widely used for video person re-identification task. However, to reach a good result, the networks must be appropriately trained. The success of solving re-identification task depends a lot on datasets that are used for training. That is why this task was analyzed in detail in our present paper.

We analyzed various existed datasets and their size and composition. A number of experiments with different size of datasets have been performed. Then, we considered a problem of forming a large dataset and propose a large joint dataset for person re-identification. This dataset includes the existed datasets CUHK02, CUHK03, Market, Duke, MSMT17 and our collected PolReID. The obtained unified dataset includes 6174 identifiers and 115 956 images.

The proposed new large dataset allows to improve re-identification metrics on all test sets. For further increasing the re-identification accuracy, we used re-rankings. The most efficient CNN using re-rank after training by presented dataset based on test results is DenseNet-121. We achieve for Market-1501 dataset 94.80% on Rank1, 92.01% on mAP, 0.833 on mINP and for DukeMTMC-ReID 88.73% on Rank1, 86.03% on mAP, 0.833 on mINP = 0.663 after re-rank. Training on the joint dataset and re-rank increased the accuracy scores for PolReID to Rank1 = 96.45%, mAP = 90.85%, mINP = 0.495.

References

[1] A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao. YOLOv4: Optimal speed and accuracy of object detection. arXiv, 2020. arXiv:2004.10934. doi:10.48550/arXiv.2004.10934.

[2] S. Bąk and P. Carr. One-shot metric learning for person re-identification. In *Proc. 2017 IEEE Conf. Computer Vision and Pattern Recognition (CVPR 2017)*, pages 1571–1580, Honolulu, HI, USA, 21-26 Jul 2017. doi:10.1109/CVPR.2017.171.

[3] Y. Dai, J. Liu, Y. Sun, et al. IDM: An intermediate domain module for domain adaptive person re-ID. In *Proc. 2021 IEEE/CVF Conf. Computer Vision (ICCV 2021)*, pages 11844–11854, Montreal, QC, Canada, 10-17 Oct. doi:10.1109/ICCV48922.2021.01165.

[4] Y. Dai, Y. Sun, J. Liu, et al. Bridging the source-to-target gap for cross-domain person re-identification with intermediate domains. ArXiv, 2022. arXiv:2203.01682v1. doi:10.48550/arXiv.2203.01682.

[5] Z. Ding, C. Ding, Z. Shao, and D. Tao. Semantically self-aligned network for text-to-image part-aware person re-identification. arXiv, 2021. arXiv:2107.12666v2. doi:10.48550/arXiv.2107.12666.

[6] D. Fu, D. Chen, J. Bao, et al. Unsupervised pre-training for person re-identification. In *Proc. 2021 IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR 2021)*, pages 14745–14754, Nashville, TN, USA, 20-25 Jun 2021. doi:10.1109/CVPR46437.2021.01451.

[7] S. Gong and T. Xiang. Person re-identification. In *Visual Analysis of Behaviour: From Pixels to Semantics*, pages 301–313, London, 2011. Springer. doi:10.1007/978-0-85729-670-2_14.

[8] D. Gray, S. Brennan, and H. Tao. Evaluating appearance models for recognition, reacquisition, and tracking. In *Proc. 10th IEEE Int. Workshop on Performance Evaluation of Tracking and*

Surveillance (PETS 2007), Sep 2007. https://www.researchgate.net/publication/228345677_Evaluating_appearance_models_for_recognition_reacquisition_and_tracking.

[9] K. He, X. Zhang, Sh. Ren, and J. Sun. Deep residual learning for image recognition. *2016 IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015. doi:10.1109/cvpr.2016.90.

[10] M. Hirzer, C. Beleznai, P. M. Roth, and H. Bischof. Person re-identification by descriptive and discriminative classification. In *Proc. Scandinavian Conf. Image Analysis (SCIA 2011)*, volume 6688 of *Lecture Notes in Computer Science*, pages 91–102, Ystad, Sweden, 23-25 May 2011. doi:10.1007/978-3-642-21227-7_9.

[11] G. Huang, Zh. Liu, and K. Q. Weinberger. Densely connected convolutional networks. *2017 IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2017. doi:10.1109/CVPR.2017.243.

[12] S. Ihnatsyeva, R. Bohush, and Ablameyko. Joint dataset for CNN-based person re-identification. In *Proc. 15th Int. Conf. Pattern Recognition and Information Processing (PRIP 2021)*, pages 33–37, Minsk, Belarus, 21-24 Sep 2021. United Institute of Informatics Problems, NAS Belarus, Minsk. https://elib.psu.by/handle/123456789/28586.

[13] SvetlanaIgn (S. Ihnatsyeva). PolReID. GitHub, Sep 2022. https://github.com/SvetlanaIgn/PolReID. [Accessed 1 Dec 2022].

[14] X. Jin, C. Lan, W. Zeng, et al. Style normalization and restitution for generalizable person re-identification. In *Proc. 2020 IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR 2020)*, pages 3140–3149, Seattle, WA, USA, 13-19 Jun 2020. doi:10.1109/cvpr42600.2020.00321.

[15] X. Jing, X. Zhu, F. Wu, et al. Super-resolution person re-identification with semi-coupled low-rank discriminant dictionary learning. *IEEE Transactions on Image Processing*, 26(3):1363–1378, 2015. doi:10.1109/TIP.2017.2651364.

[16] R. Layne, T. M. Hospedales, and S. Gong. Investigating open-world person re-identification using a drone. In Agapito L. et al., editors, *Computer Vision – Proc. European Conf. Computer Vision Workshops (ECCVW 2014)*, volume 8927, Part III of *Lecture Notes in Computer Science*, pages 225–240, Zurich, Switzerland, 6-7 Sep 2014. doi:10.1007/978-3-319-16199-0_16.

[17] S. Li, T. Xiao, H. Li, et al. Person search with natural language description. In *Proc. 2017 IEEE Conf. Computer Vision and Pattern Recognition (CVPR 2017)*, pages 5187–5196, Honolulu, HI, USA, 21-26 Jul 2017. doi:10.1109/CVPR.2017.551.

[18] W. Li and X. Wang. Locally aligned feature transforms across views. In *Proc. 2013 IEEE Conf. Computer Vision and Pattern Recognition (CVPR 2013)*, pages 3594–3601, Portland, OR, USA, 23-28 Jun 2013. doi:10.1109/CVPR.2013.461.

[19] W. Li, R. Zhao, and X. Wang. Human reidentification with transferred metric learning. In K. M. Lee et al., editors, *Computer Vision – Proc. 11th Asian Conf. Computer Vision (ACCV 2012)*, volume 7724 of *Lecture Notes in Computer Science*, pages 31–44, Daejeon, Republic of Korea, 5-9 Nov 2012. doi:10.1007/978-3-030-58555-6_14.

[20] W. Li, R. Zhao, T. Xiao, and X. Wang. DeepReID: Deep filter pairing neural network for person re-identification. In *Proc. 2014 IEEE Conf. Computer Vision and Pattern Recognition (CVPR 2014)*, pages 152–159, Columbus, OH, USA, 23-28 Jun 2014. doi:10.1109/CVPR.2014.27.

[21] X. Li, W. Zheng, X. Wang, et al. Multi-scale learning for low-resolution person re-identification. In *Proc. 2015 IEEE Int. Conf. Computer Vision (ICCV 2015)*, pages 3765–3773, Santiago, Chile, 7-13 Dec 2015. doi:10.1109/ICCV.2015.429.

[22] C. Luo, C. Song, and Z. Zhang. Generalizing person re-identification by camera-aware invariance learning and cross-domain mixup. In A. Vedaldi et al., editors, *Computer Vision – Proc. European*

Conf. Computer Vision (ECCV 2020), volume 12360 of *Lecture Notes in Computer Science*, pages 224–241, Glasgow, United Kingdom, 23-28 Aug 2020. doi:10.1007/978-3-030-58555-6_14.

[23] T. D. Nguyen, H. G. Hong, K. W. Kim, and K. R. Park. Person recognition system based on a combination of body images from visible light and thermal cameras. *Sensors*, 17(3):605, 2017. doi:10.3390/s17030605.

[24] L. Pang, Y. Wang, Y. Song, et al. Cross-domain adversarial feature learning for sketch re-identification. In *Proc. 26th ACM Int. Conf. Multimedia (MM '18)*, pages 609–617, Seoul, Republic of Korea, 22-26 Oct 2018. doi:10.1145/3240508.3240606.

[25] E. Ristani, F. Solera, R. S. Zou, et al. Performance measures and a data set for multi-target, multi-camera tracking. In G. Hua et al., editors, *Computer Vision – Proc. European Conf. Computer Vision Workshops (ECCVW 2020)*, volume 9914 of *Lecture Notes in Computer Science*, pages 17–35, Amsterdam, The Netherlands, 8-16 Oct 2016. doi:10.1007/978-3-319-48881-3_2.

[26] Y. Shiping, S. Ihnatsyeva, R. Bohush, C. Chen, and S. Ablameyko. Estimation CNN-based person re-identification accuracy in video using different datasets. In C.-H. Chen et al., editors, *Applied Mathematics, Modeling and Computer Simulation*, volume 30 of *Advances in Transdisciplinary Engineering*, pages 978–985. IOS Press, 2022. doi:10.3233/ATDE221122.

[27] G. Song, B. Leng, Y. Liu, et al. Region-based quality estimation network for large-scale person re-identification. *Proc. AAAI Conf. Artificial Intelligence*, 32(1):7347–7354, 2018. doi:10.1609/aaai.v32i1.12305.

[28] Y. Sun, L. Zheng, Y. Yang, et al. Beyond part models: Person retrieval with refined part pooling. In V. Ferrari et al., editors, *Computer Vision – Proc. European Conf. Computer Vision (ECCV 2017)*, volume 11208, Part IV of *Lecture Notes in Computer Science*, pages 501–518, Munich, Germany, 8-14 Sep 2018. doi:10.1007/978-3-030-01225-0_30.

[29] `Tianxiaomo`. Pytorch-YOLOv4. GitHub, 2020. `https://github.com/Tianxiaomo/pytorch-YOLOv4`. [Accessed 1 Dec 2022].

[30] Y. Wang, X. Liang, and S. Liao. Cloning outfits from real-world images to 3D characters for generalizable person re-identification. In *Proc. 2022 IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR 2022)*, pages 4890–4899, New Orleans, LA, USA, 18-24 Jun 2022. doi:10.1109/CVPR52688.2022.00485.

[31] Y. Wang, S. Liao, and L. Shao. Surpassing real-world source training data: Random 3d characters for generalizable person re-identification. In *Proc. 28th ACM Int. Conf. Multimedia (MM '20)*, Seattle, WA, USA, 12-16 Oct 2020. doi:10.1145/3394171.3413815.

[32] L. Wei, S. Zhang, W. Gao, and Tian Q. Person transfer GAN to bridge domain gap for person re-identification. In *Proc. 2018 IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR 2018)*, pages 79–88, Salt Lake City, UT, USA, 18-23 Jun 2018. doi:10.1109/CVPR.2018.00016.

[33] A. Wu, W. Zheng, H. Yu, et al. RGB-infrared cross-modality person re-identification. In *Proc. 2017 IEEE Int. Conf. Computer Vision (ICCV 2017)*, pages 5390–5399, Venice, Italy, 22-29 Oct 2017. doi:10.1109/ICCV.2017.575.

[34] D. Wu, S.-J. Zheng, X.-P. Zhang, et al. Deep learning-based methods for person re-identification: A comprehensive review. *Neurocomputing*, 337:354–371, 2019. doi:10.1016/j.neucom.2019.01.079.

[35] T. Xiao, S. Li, B. Wang, et al. Joint detection and identification feature learning for person search. In *Proc. 2017 IEEE Conf. Computer Vision and Pattern Recognition (CVPR 2017)*, pages 3376–3385, Honolulu, HI, USA, 21-26 Jul 2017. doi:10.1109/CVPR.2017.360.

[36] T. Xiao, S. Li, B. Wang, L. Lin, and X. Wang. End-to-end deep learning for person search. Xiaogang Wang: personal web page, 2016. `http://www.ee.cuhk.edu.hk/~xgwang/PS/paper.pdf`.

[37] M. Ye, J. Shen, G. Lin, et al. Deep learning for person re-identification: A survey and outlook. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(6):2872–2893, 2020. doi:10.1109/TPAMI.2021.3054775.

[38] T. Zhang, L. Xie, L. Wei, et al. UnrealPerson: An adaptive pipeline towards costless person re-identification. In *Proc. 2021 IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR 2021)*, pages 11501–11510, Nashville, TN, USA, 20-25 Jun 2021. doi:10.1109/CVPR46437.2021.01134.

[39] F. Zhao, S. Liao, G. Xie, et al. Unsupervised domain adaptation with noise resistible mutual-training for person re-identification. In A. Vedaldi et al., editors, *Computer Vision – Proc. European Conf. Computer Vision (ECCV 2020)*, volume 12356 of *Lecture Notes in Computer Science*, pages 526–544, Glasgow, United Kingdom, 23-28 Aug 2020. doi:10.1007/978-3-030-58621-8_31.

[40] L. Zheng, Z. Bie, Y. Sun, et al. MARS: A video benchmark for large-scale person re-identification. In B. Leibe et al., editors, *Computer Vision – Proc. European Conf. Computer Vision (ECCV 2016)*, volume 9910 of *Lecture Notes in Computer Science*, pages 868–884, Amsterdam, The Netherlands, 11-14 Oct 2016. doi:10.1007/978-3-319-46466-4_52.

[41] L. Zheng, L. Shen, L. Tian, et al. Scalable person re-identification: A benchmark. In *Proc. 2015 IEEE Int. Conf. Computer Vision (ICCV 2015)*, pages 1116–1124, Santiago, Chile, 7-13 Dec 2015. doi:10.1109/ICCV.2015.133.

[42] L. Zheng, H. Zhang, S. Sun, et al. Person re-identification in the wild. In *Proc. 2017 IEEE Conf. Computer Vision and Pattern Recognition (CVPR 2017)*, pages 3346–3355, Honolulu, HI, USA, 21-26 Jul 2017. doi:10.1109/CVPR.2017.357.

[43] Z. Zheng, X. Yang, Z. Yu, et al. Joint discriminative and generative learning for person re-identification. In *Proc. 2019 IEEE Conf. Computer Vision and Pattern Recognition (CVPR 2019)*, pages 2133–2142, Long Beach, CA, USA, 15-20 Jun 2019. doi:10.1109/CVPR.2019.00224.

[44] `layumi` (Z. Zheng). Person_reID_baseline_pytorch. GitHub. `https://github.com/layumi/Person_reID_baseline_pytorch`. [Accessed 1 Dec 2022].

[45] Z. Zhong, L. Zheng, D. Cao, and S. Li. Re-ranking person re-identification with k-reciprocal encoding. In *Proc. 2017 IEEE Conf. Computer Vision and Pattern Recognition (CVPR 2017)*, pages 3652–3661, Honolulu, HI, USA, 21-26 Jul 2017. doi:10.1109/CVPR.2017.389.