Acquiring custom OCR system with minimal manual annotation

Jan Hula*, David Mojžíšek, David Adamczyk, Radek Čech
Institute for Research and Applications of Fuzzy Modeling
Ostrava, Czechia
*jan.hula@osu.cz

***Abstract*—We describe a development of a custom OCR system, which is designed specifically for a linguistic analysis of texts printed during the early modern period. This analysis requires precise detection of individual graphemes, and we, therefore, could not apply standard approaches that transcribe whole lines in an end-to-end fashion. We also describe our use of synthetically generated images, which allow us to avoid manual annotation of a large training set.**

***Index Terms*—OCR, Synthetic Data, Historical Texts, Neural Networks**

I. Overview

Optical Character Recognition (OCR) is nowadays considered a solved problem with very little space for innovation [1]. Still, some applications have very specific requirements for which custom solutions are needed. Here, we focus on a problem arising in a linguistic analysis of printed texts from the early modern period (1500-1750). The solution to this problem must include a precise detection of graphemes because the analysis will deal with relative sizes of printed graphemes and spaces between them as described in the next section. Because we have not found an open-source system which would fulfill our requirements, we decided to design the system by ourselves. It follows a two-stage pipeline. The first stage is based on a state-of-the-art model for object detection to detect individual graphemes and in the second stage, individual graphemes are recognized using bidirectional LSTM [2] which recognizes every grapheme in a context in which it appears. We solve these two stages separately. After training the grapheme detection model, we use it to detect thousands of instances of generic graphemes which we cut-out and cluster using a simple K-means algorithm. We then label these clusters and use them to pre-label training examples for the recognition model.

As annotating bounding boxes for individual graphemes can be a tedious task, we wanted to avoid as much of manual annotation as possible. For this purpose, we synthesize artificial examples of printed pages which help us to bootstrap a labeling process with a model pre-trained on these examples. Our methodology follows a simple principle where we train a weaker method requiring few training examples to pre-label training data for a stronger method. In this contribution, we describe the whole pipeline of our solution. We believe that it contains ideas which can generalize to similar problems.

II. Motivation and Problem Statement

The approach presented in this paper is motivated by needs that have emerged in linguistics focused on the development of the Czech orthographic system. Specifically, Voit [3] introduced new explanation of the usage of orthographic variations (such as ”uo” $\sim$ ”ů” / ”ú”, ”ie” $\sim$ ”ij” $\sim$ ”j”) in 16th century. He claims that the usage of either singlegraphic or digraphic grapheme was caused by pragmatic factors related to typesetting praxis. According to him, a typesetter was forced to fulfill the following requirements: 1) to align the right edge of the text, 2) to avoid splitting of words at the end of the line. In other words, an option to use either longer or shorter realizations of the grapheme was a tool for dilation or compression of text in the line.

This explanation offers setting up several empirically testable hypotheses. For instance, ”the higher the number of types in the line, the higher probability of occurrence of digraphic grapheme” or ”the higher the number of types in the line, the lower the number of spaces in the line”. Testing of hypotheses of this kind must be performed on a large sample of original texts. Further, proper testing needs careful operalization, which is not a trivial task in this case. For instance, a width of both the grapheme and space must be determined unambiguously. To our knowledge, up to now, only in [4], the problem had been analyzed empirically. However, they used the sample consisting of only four texts which were transcribed and annotated manually. The lack of adequately processed documents is the main obstacle for a thorough analysis of this phenomenon, and this could be solved by automatization of the annotation process.

In other words, we want to prepare an annotated dataset (together with the annotation tool) for further linguistic analysis. This analysis should confirm or disprove the hypothesis that changes in written Czech language were driven by technological limitations and needs in typesetting practice.

III. Description of the data

In this section, we briefly describe our data and their specifics. We are working with scanned printed documents mainly from the second half of the 17th century. They were printed in different Czech cities (Praha, Litomyšl, Olomouc, etc.). Not all of the documents are clearly readable - few pages are torn, or the text is faded out. Also, the typeset is very specific and different from contemporary documents. Even Czech

native speakers are not able to fluently read such texts if they are not trained. In figures 1,2,3 we present a few examples that point out some problematic areas in our dataset. Our dataset contains hundreds of scanned documents without annotations. We also dispose of dozens of transcriptions containing similar language, which we leverage when training the classification model, as described in section VIII.

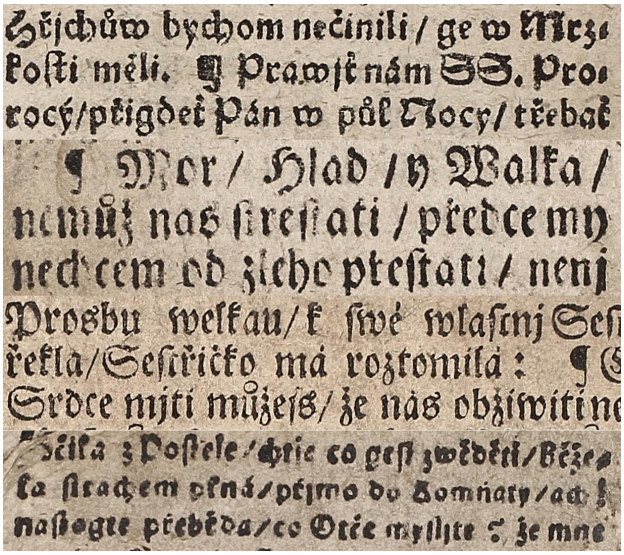

Fig. 1: Altough these text are from approximately the same period, they contain different typesets.

Fig. 2: Character difficulties. a) Ligatures, i.e. two or more characters printed as one. b) Some glyphs are hard to read and distinguish when the context is unknown.

IV. Outline of Our Pipeline

As most of OCR systems in use [5]–[8], we developed a pipelined system in which we solve individual steps independently of others. On a high level, we rotate each document so that lines are horizontally oriented, then we detect all graphemes in a given document, and finally, we classify each grapheme in the context of the text it appears in. The schematic representation of this process is depicted in figure 4. Most of modern OCR systems do not work by detecting individual graphemes but transcribe whole lines in an end-to-end fashion. This approach has the benefit that it does not require annotated

Fig. 3: Difficult pages. a) Some documents are damaged (for example folded or torn) or the qaulity of scan is low. b) Page containing an image and faded graphemes of different size.

bounding boxes. In our case, obtaining these bounding boxes is necessary, so we decided to split the pipeline to grapheme detection stage and grapheme classification stage. We could have also tried to classify and detect the graphemes in parallel, as it is usually done in object detection. By doing so, we would miss the opportunity to incorporate the structure of the text into the classification. The detection network would classify the grapheme as a patch in the image, instead of a grapheme in the sequence of graphemes. The second minor benefit of separating these two stages is the fact that we do not need to annotate the class of every bounding box, as will be described in section VIII. To obtain a final version of our system, we rely heavily on synthetic data and other tricks which allow us to avoid as much manual annotation as possible. Here follows a description of all the steps we execute during the construction of our system:

Alignment part

1) Train a neural network *NN-rot* (VGG-11) to regress angles of rotated images.
2) Use *NN-rot* to straighten all images in the dataset.

Detection part

3) Annotate bounding boxes around graphemes in 3 random pages.
4) Use the annotated graphemes to generate a large training set of synthetic images.
5) Pre-train a neural network *NN-det* to detect graphemes in synthetic images.
6) Fine-tune *NN-det* on the 3 annotated real images.
7) Pre-label 6 new pages with *NN-det* and correct wrong detections to enlarge the annotated set.
8) Repeat steps 4-7 two more times to increase the variability of the dataset and to obtain an accurate detection model.

Classification part

9) Detect few thousands of graphemes in still unlabeled images using *NN-det* trained in previous steps.
10) Cut out the detected graphemes and train an autoencoder to obtain low-dimensional representations for every grapheme.

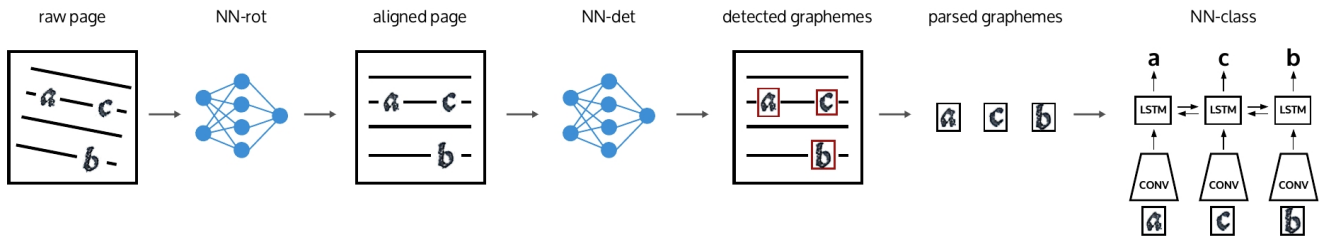

Fig. 4: A schematic representation of the pipeline used for every new image. The system first alignes the page according to a predicted angle from *NN-rot*; next it detects all graphems on the page with *NN-det*; these graphemes are parsed into a sequence which is finally classified by *NN-class*

11) Cluster the graphemes using K-means based on the representation from the autoencoder.
12) Create a labeled dataset of graphemes by manually labeling the clusters and deleting incorrectly assigned graphemes.
13) Use available transcriptions of texts from the same period containing the same language to generate training sequences of cut-out graphemes.
14) Train a convolutional bi-LSTM *NN-class* to classify each grapheme in the context of other graphemes using training examples from step 13.

By following these steps, we avoid manual labeling of hundreds of pages. Their details will be described in the subsequent section. At the end, we use *NN-rot*, *NN-det*, and *NN-class* for every new image.

V. Page alignment

The scanned documents in our dataset were not always aligned, and therefore, individual lines were not aligned horizontally. Page alignment is a part of all OCR systems because when the letters are aligned, the subsequent recognition model does not need to learn rotation invariance. We decided to train a neural network to predict the angles from cropped patches of the image. Fortunately, obtaining a labeled dataset for this task does not require a lot of manual effort. We manually aligned 30 pages, and these aligned images are then rotated by a random angle from an interval -15 to +15 degrees, quantized to increments of 0.5. Subsequently, crops are taken from these rotated images, and the rotation angle is saved as their target label. We train a convolutional network with ResNet-18 backbone [9] to predict these angles, and for every new image, we average predictions from crops taken from it. We achieve nearly perfect accuracy ($\sim$98%) with this approach.

VI. Synthetic Data and Domain Knowledge

One of the biggest drawbacks of data-driven approaches for problem-solving is that they often need a lot of labeled examples to be trained on. To avoid this problem and save a lot of manual annotation, we decided to use synthetic data, which we generate using our understanding of the problem domain. Synthetic data has been recently used in all kinds of domains of Computer Vision and Machine Learning in general [10]–[13], and their use in OCR for modern print marks one of their first successful use-case [14]. The main pitfall of using synthetic datasets to train systems for real data is that the training distribution may be very different from the testing distribution because, in some domains, it may not be trivial to synthesize realistic examples (e.g., synthesizing realistic images of human faces). OCR for modern print was a successful use-case mainly because it is possible to create highly realistic synthetic examples using known fonts and simple image distortions and noise. Creating realistic examples of old prints is more involved because the variability of the appearance of graphemes is larger due to all kinds of problems arising during the printing process (e.g., leakage of ink) and degradation of documents after long periods. Examples of such problems can be seen in figures 2 and 3.

After a visual inspection of many real documents, we model the realism and variability of the page as closely as possible. For this, we use cut-out examples of various graphemes [1] with background removed and documents containing blank pages. We generate each page by sampling random words from which we create whole lines and place them to an empty page. On each grapheme, we apply a random set of augmentations simulating fading of the ink, elastic distortion, and various kinds of noises and scratches. To gain the variability in the background, we also apply similar augmentations on the few blank pages we had at our disposal. Also, many documents contained random drops of ink, which could be possibly mistaken for a grapheme, and therefore we add such drops to the background at random positions. An example of a such generated page can be seen in figure 5. For our experiments, we created 300 synthetic pages together with ground truth bounding boxes for every grapheme.

VII. Glyph Detection

As described in section IV, we first detect all graphemes as one generic class, then we parse the segmented graphemes to a sequence of these graphemes, and finally, we classify each grapheme using a sequence-based model. For the detection of generic graphemes, we use a state-of-the-art model for object

[1] Synthetic data described in this section are used only to detect generic graphemes, i.e., to detect a grapheme without classifying it into a class.

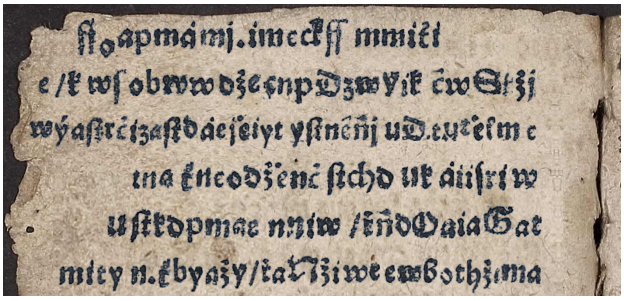

Fig. 5: An example of a crop from artificially generated page. We used different grapheme and background augmentations to add variability.

detection called RetinaNet [15], which we slightly modify to suit our task. Object detection models are usually categorized into one-stage and two-stage methods. As the names suggest, one-stage methods classify and detect objects in one stage, whereas two-stage methods first propose a candidate bounding boxes, which are then refined and classified in the second stage. One-stage methods are usually faster but less precise due to the imbalance of positive and negative bounding box proposals. RetinaNet solves this short-coming of one-stage methods by using special loss function called Focal Loss [15], which takes this imbalance into account. Therefore RetinaNet is a fast and accurate architecture for object detection.

Most of the time, models for object detection use pre-trained backbones (feature extractors) trained on big classification datasets such as ImageNet [16]. These backbones extract already useful features that can be leveraged by subsequent layers in the network. The pre-trained backbone is most useful when the pre-training domain is similar to the target domain. In our case, the images of scanned printed documents are very different from photographs capturing random objects, and therefore we do not use a pre-trained backbone but train it from scratch on our synthetic dataset.

As most of the one-stage detectors, RetinaNet uses anchors when detecting individual objects. During detection, the image is divided into a grid of rectangular cells, and inside each cell, multiple anchors of predefined sizes and proportions are used to detect possible objects. The final bounding box is being regressed from each such anchor and classified as a particular class or as a background. Setting up the sizes and ratios correctly is an important step. It, for example, does not make sense to have anchors large in size if we know that there won't be large objects within any image. We, therefore, take special care to set up these sizes and ratios so that they cover the sizes of graphemes in our dataset. For a more complete overview of current methods in object detection see [17].

Also, we wanted to retain the resolution of images as high as possible, and so we cut the whole page into overlapping patches of size 256x256 px and process each patch separately. After the system processes all patches from an image, we merge all boundary boxes in a post-processing stage. We train the detection model on 300 synthetically generated pages, and

TABLE I: Comparison of average precision (AP) and Focal loss between the same model (RetinaNet) trained with and without syntetic images.

Training Data	AP	Focal Loss
With synthetic data	**0.3110**	**1.143**
Without synthetic data	0.03767	2.181

then we fine-tune it on three real and manually labeled pages. This model already produces quite precise bounding boxes, so we use it to pre-label six more pages in which we manually correct wrong predictions. We then enlarge the training set of synthetic and real images using these newly labeled pages. We repeat this process two times and thus acquire a model of satisfying accuracy. In table I, we compare average precision and Focal loss between the same model trained with and without synthetic images created from three annotated pages. As these metrics are not easily interpretable, we show a visual example in figure 6.

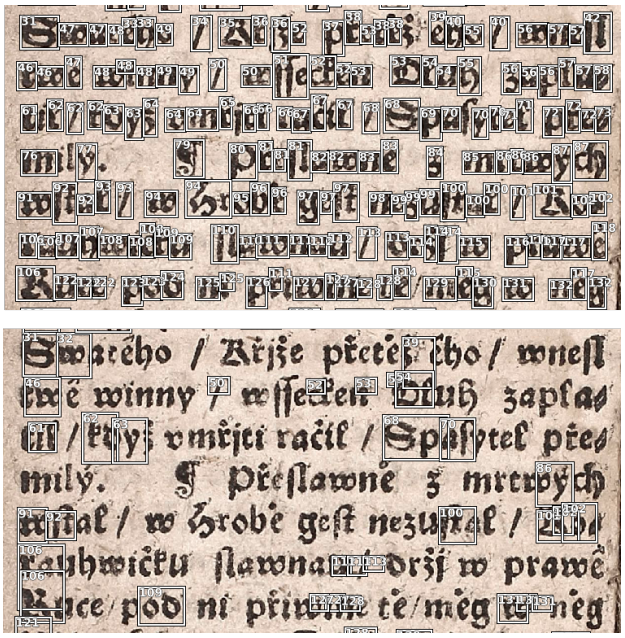

Fig. 6: A visual example of a quality of detection after the first round using only 3 annotated images. The top image shows results from a model trained on synthetic images and fine-tuned on the 3 annotated images. The bottom image shows results from a model trained only on the 3 annotated images (with standard augmentations, i.e. resize, lightness, etc.).

VIII. Glyph Classification

In the last section, we described the model for grapheme detection. We use this model to detect generic graphemes, which will be parsed into a sequence where the order of graphemes is the same as the order in which we would read

the text. These sequences are then classified with a sequence-based model, which we describe in this section. We use a sequence-based model because, in some instances, the identity of a grapheme may not be recognizable without a context. The sequence-based model can leverage statistical regularities within sequences of letters found within the use of a language.

First, we need to create a dataset for classification. Again, we want to avoid manual annotation as much as possible. We came up with two ways how to achieve it. Our classification model is convolutional bidirectional LSTM which takes a sequence of cut-out graphemes and produces a sequence of labels, one label for each grapheme. Therefore we need to obtain labeled training examples of such sequences. Our idea was to use transcriptions of texts which contain a language being used in our documents. Given these transcripts, we can create many different training sequences by sampling graphemes according to the letters in the text. This has an advantage that using one sentence, we can generate many different training examples by sampling different examples for the same letter each time. In order to do this, we need to have many examples of each letter. To avoid manual annotation of separate graphemes, we use the detection model to cut-out thousands of generic graphemes, which we then cluster and label only the clusters. This lowers the amount of work we need to do by order of magnitude.

In order to cluster the graphemes, we first train an autoencoder with ResNet-18 in the encoder to obtain a low-dimensional representation of every grapheme. Umap [18] visualization of this low-dimensional space can be seen in 7. We then cluster all graphemes using a simple K-means algorithm. We set K to 2*C where C is the number of classes because when K is close to C, the algorithm mixes too many examples of different classes together in the same cluster. We manually check all clusters and remove incorrectly assigned examples. Thus, we acquire thousands of labeled examples with very little manual effort. From these, we construct a dataset of labeled sequences.

Our classification model first extracts low-dimensional (512) representation of each grapheme by processing it with a backbone from ResNet-18, and the sequence of these low-dimensional representations then goes as an input into bidirectional LSTM (with 2 layers, both of which have the output dimension equal to 512). Finally, the sequence of representations in the hidden layer of the LSTM is then processed by a linear layer with a softmax to predict the class of the grapheme at every position of the sequence.

In table II, we show a comparison between our model, which takes the context of each grapheme into account and a model with the same backbone that classifies each grapheme separately.

Fig. 7: Umap visualization of low-dimensional representation of cut-out graphemes obtained from a trained autoencoder.

TABLE II: Comparison between a model which classifies each grapheme separately and a model which takes the ohter graphemes in the same context into account. Both models share the same backbone (ResNet-18).

Method	Validation Accuracy
With bi-LSTM	**0.9337**
Without bi-LSTM	0.498

IX. Related Work

The OCR problem has been studied for many years [1]. Especially for modern prints, there exist software solutions with nearly perfect accuracy. Most of these systems use a pipelined approach such that in one step, individual lines are segmented out, and subsequently, they are transcribed into sequences of letters by a neural network that processes the whole line as a sequence of vertical strips of pixels. The number of such strips in a line does not correspond to the number of letters (labels) in that line, and therefore special loss function called CTC loss [19] is used to account for the alignment of these strips and labels. The same loss function is frequently being used in speech recognition, where the same problem arises. Examples of systems based on this approach include Tesseract [7], Calamari [6] build on top of TensorFlow and OCRopus [5] build on top of PyTorch. We started our development with OCRopus but soon realized that we need a custom solution. We have also tried a system called OCR4ALL [8], which was developed specifically for historical OCR. It is targeted mostly on people with no coding skills, so the emphasis is being given mainly on user-friendliness and not much on customizability. In most of the use-cases, it is not needed to segment out individual graphemes, and so there is no need to innovate over these solutions. As mentioned in the motivation, our use case was quite specific, and therefore we needed to develop our own solution.

The usefulness of synthetic datasets for training machine learning models was realized by many [10]–[12]. Synthetic

datasets are especially useful in robotics [13] and other domains where training on real data would be too expensive or infeasible. The main problem arising with the use of such datasets is that a machine learning model may overfit to specifics of synthetic examples and may not transfer well to real examples. As a possible solution to this problem, a technique called Domain Randomization became popular in recent years [20]–[22]. The idea behind Domain Randomization is that if we do not want to overfit to particular properties of data, such as for example color of objects when training object classifier, we should randomize that property as much as possible so that the model can not learn a statistical correlation between this property and some other variables of interest. We took inspiration in this idea when we were creating our synthetic dataset and randomized some properties of the generated images such as the size of graphemes, their sharpness, and other distortions.

X. Conclusion

In this contribution, we described creation of a custom OCR system explicitly designed to help linguistic analysis of printed texts from the early modern period. In contrast to mainstream OCR systems, we do not transcribe whole lines in an end-to-end fashion, but we first segment out individual graphemes, which are then classified using a sequence-based model. We also showed the usefulness of synthetically generated images and a bootstrapping process for annotation, which reduced the amount of manual work we needed to do by order of magnitude. In the future, we aim to design an intuitive user interface for our system which will be released together with the final version of the source code. We believe that our work, driven by the practical needs of linguists, is a valuable contribution to the interdisciplinary research between computer science and humanities.

Acknowledgment

The work was supported from ERDF/ESF "Centre for the development of Artificial Intelligence Methods for the Automotive Industry of the region" (No. CZ.02.1.01/0.0/0.0/17_049/0008414).

References

[1] D. Doermann, K. Tombre *et al.*, *Handbook of document image processing and recognition*. Springer, 2014.

[2] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[3] P. Voitem and J. Linka, "Udržet pravỳ okraj stránkové sazby (od literární historie k samostudiu)," *Česká literatura*, vol. 59, no. 2, pp. 242–260, 2011.

[4] R. Čech and J. Mačutek, "Orthography system of broadside ballads from 17. and 18. century. quantitative approach." Transformations of Czech 'kramářské písně' (Broadside Ballads) – media, traditions, contexts. Masaryk University in Brno, 2019.

[5] T. M. Breuel, "The ocropus open source ocr system," in *Document Recognition and Retrieval XV*, vol. 6815. International Society for Optics and Photonics, 2008, p. 68150F.

[6] C. Wick, C. Reul, and F. Puppe, "Calamari-a high-performance tensorflow-based deep learning package for optical character recognition," *arXiv preprint arXiv:1807.02004*, 2018.

[7] R. Smith, "An overview of the tesseract ocr engine," in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, vol. 2. IEEE, 2007, pp. 629–633.

[8] C. Reul, D. Christ, A. Hartelt, N. Balbach, M. Wehner, U. Springmann, C. Wick, C. Grundig, A. Büttner, and F. Puppe, "Ocr4all—an open-source tool providing a (semi-) automatic ocr workflow for historical printings," *Applied Sciences*, vol. 9, no. 22, p. 4853, 2019.

[9] S. Wu, S. Zhong, and Y. Liu, "Deep residual learning for image steganalysis," *Multimedia tools and applications*, vol. 77, no. 9, pp. 10 437–10 453, 2018.

[10] J. Hula, I. Perfilieva, and A. A. M. Muzaheed, "Towards visual training set generation framework," in *International Work-Conference on Artificial Neural Networks*. Springer, 2017, pp. 747–758.

[11] S. Sankaranarayanan, Y. Balaji, A. Jain, S. Nam Lim, and R. Chellappa, "Learning from synthetic data: Addressing domain shift for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3752–3761.

[12] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *European conference on computer vision*. Springer, 2016, pp. 102–118.

[13] P. Martinez-Gonzalez, S. Oprea, A. Garcia-Garcia, A. Jover-Alvarez, S. Orts-Escolano, and J. Garcia-Rodriguez, "Unrealrox: an extremely photorealistic virtual reality environment for robotics simulations and synthetic data generation," *Virtual Reality*, pp. 1–18, 2019.

[14] T. K. Ho and H. S. Baird, "Evaluation of ocr accuracy using synthetic data," in *Proceedings of the 4th Annual Symposium on Document Analysis and Information Retrieval*. Citeseer, 1995.

[15] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[17] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng, and R. Qu, "A survey of deep learning-based object detection," *IEEE Access*, vol. 7, pp. 128 837–128 868, 2019.

[18] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[19] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 369–376.

[20] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.

[21] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 1–8.

[22] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige *et al.*, "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 4243–4250.