Augraphy: A Data Augmentation Library for Document Images

Alexander Groleau[1], Kok Wei Chee[1], Stefan Larson[2], Samay Maini[1], and Jonathan Boarman[1]

[1] Sparkfish LLC (augraphy@sparkfish.com)
[2] Vanderbilt University

Abstract. This paper introduces *Augraphy*, a Python library for constructing data augmentation pipelines which produce distortions commonly seen in real-world document image datasets. *Augraphy* stands apart from other data augmentation tools by providing many different strategies to produce augmented versions of clean document images that appear as if they have been altered by standard office operations, such as printing, scanning, and faxing through old or dirty machines, degradation of ink over time, and handwritten markings. This paper discusses the *Augraphy* tool, and shows how it can be used both as a data augmentation tool for producing diverse training data for tasks such as document denoising, and also for generating challenging test data to evaluate model robustness on document image modeling tasks.

1 Introduction and Motivation

The *Augraphy* library is designed to introduce realistic degradations to images of documents, producing training datasets for supervised learning by pairing synthetic dirty documents with their clean ground truth document image sources. The need for such a data augmentation tool in document image analysis is twofold: existing document analytics software must be robust against various forms of noise, and ground truth images are unavailable for most of the documents recorded within human history. Automated recovery of knowledge stored in such documents is thwarted by the presence of distortion artifacts, which typically increase with time. As such, there is an ongoing *and growing* need to improve both document restoration techniques and software processes, unlocking contents for use in search or other tasks and providing challenging data for tooling development. Augraphy answers this demand by supporting document-modality tasks such as reconstruction (Figure 1), denoising (Figure 5), and robustness testing (Figure 6).

An ongoing digital transformation has reduced everyday reliance on paper. However, major industries such as healthcare [35], insurance [15], legal [46], financial services [10], government [1], and education [2] continue to produce physical documents that are later digitally archived.

Paper documents accumulate physical damage and alterations through printing, scanning, photocopying and regular use. Real-world processes introduce many types of distortions that result in color changes, shadows in scanned documents, and degraded

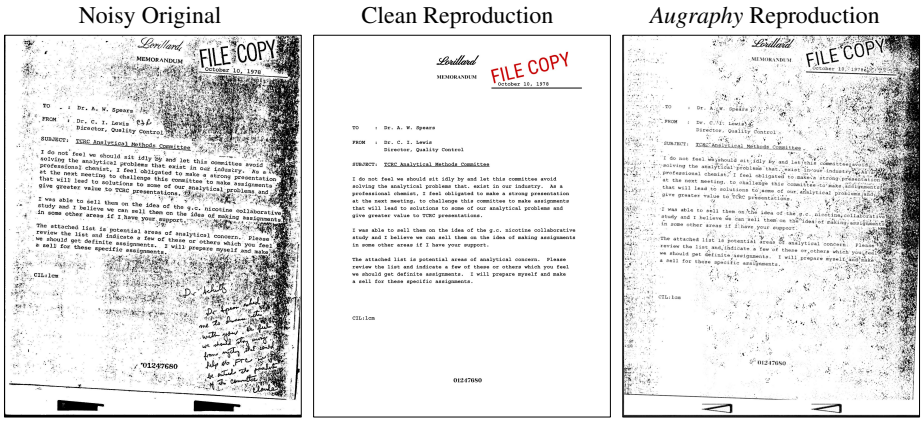

Fig. 1. *Augraphy* can be used to introduce noisy perturbations to document images like the original noise seen above, a real-life sample from RVL-CDIP. We demonstrate this by reproducing a clean image, then introducing noise with *Augraphy*.

text where low ink, annotations, highlighters, or other markings add noise that hinder semantic access by digital means.

Document image analysis tasks are impacted by the presence of such noise; high-level tasks like document classification and information extraction are frequently expected to perform well even on noisily-scanned document images. For instance, the RVL-CDIP document classification corpus [16] consists of scanned document images, many of which have substantial amounts of scanner-induced noise, as does the FUNSD form understanding benchmark [23]. Other intermediate-level tasks like optical character recognition (OCR) and page layout analysis may perform optimally if noise in a document image is minimized [9,38,42].

Attempts have been made to denoise document images in order to improve document tasks [7,14,29,36,37]. However, they have lacked the benefit of large datasets with ground truth images to train those denoisers.

A tool such as *Augraphy* that helps to produce copious amounts of training data with document noise artefacts is essential in this space to developing advanced document binarization and denoising processes.

For this reason we introduce *Augraphy*, an open-source Python-based data augmentation library for generating versions of document images that contain realistic noise artefacts commonly introduced via scanning, photocopying, and other office procedures. *Augraphy* differs from most image data augmentation tools by specifically targeting the types of alterations and degradations seen in document images.

Augraphy offers *30* individual augmentation methods out-of-the-box across three "phases" of augmentations, and these individual phase augmentations can be composed together along with a "paper factory" step where different paper backgrounds can be added to the augmented image. The resulting images are realistic, noisy versions of clean documents, as evidenced in Figure 1, where we apply *Augraphy* augmentations to a clean document image in order to mimic the types of noise seen in a real-world noisy document image from RVL-CDIP.

Several research efforts have used *Augraphy*: Larson et al. (2022) [30] used *Augraphy* to mimic scanner-like noise for evaluating document classifiers trained on RVL-CDIP; Jadhav et al (2022) [21] used *Augraphy* to generate noisy document images for training a document denoising GAN; and Kim et al. (2022) [27] used *Augraphy* as part of a document generation pipeline for document understanding tasks.

Table 1. Comparison of various image-based data augmentation libraries. Number of augmentations is a rough count, and many augmentations in other tools are what *Augraphy* calls Utilities. Further, many single augmentations in *Augraphy* — geometric transforms, for example — are represented by multiple classes in other libraries.

Library	Number of Augmentations	Document Centric	Pipeline Based	Python	License
Augmentor [4]	27	✗	✓	✓	MIT
Albumentations [5]	216	✗	✓	✓	MIT
imgaug [25]	168	✗	✓	✓	MIT
Augly [39]	34	✗	✓	✓	MIT
DocCreator [24]	12	✓	✗	✗	LGPL-3.0
Augraphy (ours)	*30*	✓	✓	✓	MIT

2 Related Work

This section reviews prior work on data augmentation and available tooling for robustness testing, particularly in the context of document understanding, processing, and analysis tasks.

Table 1 compares *Augraphy* with other image augmentation libraries and tools, highlighting that most existing data augmentation libraries do not specifically cater to the corruptions typically encountered in document image analysis datasets. Instead of general image manipulation, *Augraphy* focuses on document-specific distortions and alterations that are better-suited to challenging and evaluating OCR, object-detection, and other models with images which still retain sufficient quality to remain human-readable. Crucially, *Augraphy* is also designed to be easily integrated into machine learning pipelines alongside other popular Python packages used in machine learning and document analysis.

2.1 Data Augmentation

A wide variety of data augmentation tools and pipelines exist for machine learning tasks, including natural language processing (e.g., [13,12,48]), audio and speech processing (e.g., [28,33,34]), and computer vision and image processing.

As it pertains to computer vision, popular tools such as *Augly* [39], *Augmentor* [4], *Albumentations* [5], and *imgaug* [25] provide image-centric augmentation strategies including general-purpose transformations like rotations, warps, masking, and color modifications. In contrast, as seen in Table 1, only *Augraphy* offers a data augmentation

library that supports imitating the corruptions commonly seen in document analysis corpora and is also easily integrated in modern machine learning training processes which rely on Python, including use of specialized paper, ink and post-processing pipelines not seen elsewhere. Best-effort attempts to use *Albumentations* to recreate the document in Figure 1 yielded results so poor and unlike the target that we could not in good faith include them in this paper.

DocCreator [24] is the next best option, providing an image-synthesizing tool for creating synthetic images that mimic common corruptions seen in document collections. However, *DocCreator* differs from *Augraphy* in several crucial ways. The first difference is that *DocCreator*'s augmentations are meant to imitate those seen in historical (e.g., ancient or medieval) documents, while *Augraphy* is intended to replicate a wider variety of both historical and modern scenarios. *DocCreator* was also written in the C++ programming language as a monolithic *what-you-see-is-what-you-get (WYSIWYG)* tool, and does not have a scripting or API interface to enable use in a broader machine learning pipeline. *Augraphy*, on the other hand, is written in Python and can be easily integrated into machine learning model development and evaluation pipelines, alongside other Python packages like PyTorch [40].

2.2 Robustness Testing

Testing a model's robustness with challenging datasets is a vital step to evaluate a model's resilience against real-world scenarios, helping developers identify and address weaknesses for enhanced effectiveness in document processing tasks.

Prior work for evaluating image classification and object detection models includes image blur; contrast, brightness and color alterations; masking; geometric transforms; pixel-level noise; and compression artefacts (e.g., [11,18,19,20,26,44,47]). More specific to the document understanding field, recent prior work has used basic noise-like corruptions to evaluate the robustness of document classifiers trained on RVL-CDIP [43]. Our paper also uses robustness testing of OCR and face-detection models, as a way to showcase *Augraphy's* efficacy in producing document-centric distortions which challenge these models while remaining human-readable.

3 Document Distortion, Theory & Technique

As referenced in the *Related Work* section of this paper, many general-purpose image distortion techniques exist for data augmentation and robustness testing. While these techniques are useful in general image analysis and understanding, they bear little resemblance to the features commonly found in real-world documents.

Augraphy's suite of augmentations was designed to faithfully reproduce the level of complexity in the document lifecycle as seen in the real world. Such real-world conditions and features have direct implementations either within the library or on the development roadmap.

Some techniques exist for introducing these features into images of documents, including but not limited to the following:

1. Text can be generated independently of the paper texture, and can be overlaid onto the "paper" by a number of blending functions, allowing a variety of paper textures to be used.
2. Similarly, any markup features may be generated and overlaid by the same methods.
3. Documents can be digitized with a commercial scanner, or converted to a continuous analog signal and back with a fax machine.
4. The finished document image can be used as a texture and attached to a 3D mesh, then projected back to 2 dimensions to simulate physical deformation. DocCreator's 3D mesh support here is excellent and has inspired *Augraphy* to add similar functionality in its roadmap. In the meantime, *Augraphy* currently offers support for paper folds and book bindings.

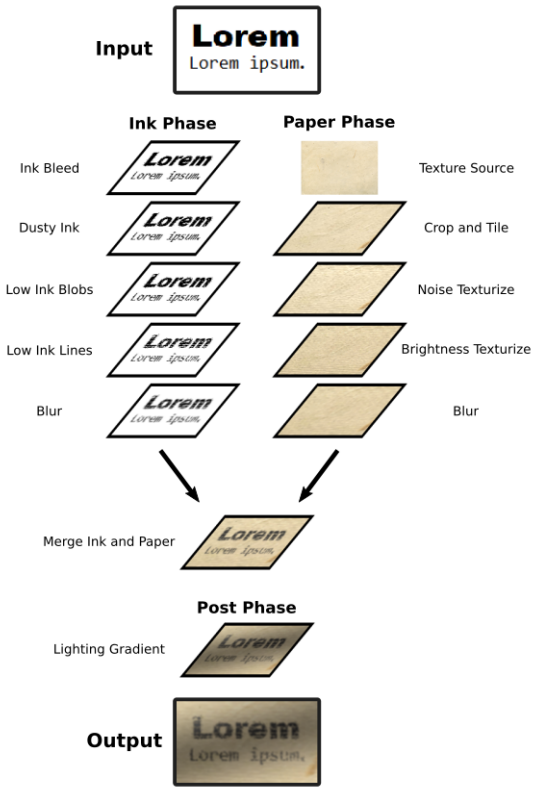

Fig. 2. Visualization of an *Augraphy* pipeline, showing the composition of several image augmentations together with a specific paper background

Augraphy attempts to decompose the lifetime of features accumulating in a document by separating the pipeline into *three phases*: **ink**, **paper**, and **post**. The ink phase exists to sequence effects which specifically alter the printed ink — like bleedthrough from too much ink on page, extraneous lines or regions from a mechanically-faulty printer, or fading over time — and transform them prior to "printing".

The paper phase applies transformations to the underlying paper on which the ink gets printed; here, a `PaperFactory` generator creates a random texture from a set of given texture images, as well as effects like random noise, shadowing, watermarking, and staining. After the ink and paper textures are separately computed, they are merged in the manner of Technique 1 from the previous section, simulating the printing of the document.

After "printing", the document enters the post phase, wherein it may undergo modifications that might alter an already-printed document out in the world. Augmentations are available here which simulate the printed page being folded along multiple axes, marked by handwriting or color highlighter, faxed, photocopied, scanned, photographed, burned, stained, and so on. Figure 2 shows the individual phases of an example pipeline combining to produce a noised document image.

4 Augraphy

Augraphy is a lightweight Python package for applying realistic perturbations to document images. It is registered on the Python Package Index (PyPI) and can be installed simply using `pip install augraphy`.

Augraphy requires only a few commonly-used Python scientific computing or image handling packages, such as NumPy [17], and has been tested on Windows, Linux, and Mac computing environments and supports recent major versions of Python 3. Below is a basic out-of-the-box *Augraphy* pipeline demonstrating its usage:

```
import augraphy; import cv2
pipeline = augraphy.default_augraphy_pipeline()
img = cv2.imread("image.png")
data = pipeline.augment(img)
augmented_img = data["output"]
```

Listing 1.1. Augmenting a document image with *Augraphy*.

Augraphy is designed to be immediately useful with little effort, especially as part of a preprocessing step for training machine learning models, so care was taken to establish good defaults. *Augraphy* provides *30* unique augmentations, which may be sequenced into pipeline objects which carry out the image manipulation. The default *Augraphy* pipeline (used in the code snippet above) makes use of all of these augmentations, with starting parameters selected after manual visual inspection of thousands of images. Users of the library can define directed acyclic graphs of image transformations via the `AugraphyPipeline` API, representing the passage of a document through real-world alterations.

4.1 Augraphy Augmentations

Augraphy provides *30* out-of-the-box augmentations, listed in Table 2. As mentioned before, Ink Phase augmentations include those that imitate noisy processes that occur in a document's life cycle when ink is printed on paper. These augmentations include

`BleedThrough`, which imitates what happens when ink bleeds through from the opposite side of the page. Another, `LowInkLines`, produces a streaking behavior common to printers running out of ink.

Augmentations provided by the Paper Phase include `BrightnessTexturize`, which introduces random noise in the brightness channel to emulate paper textures, and `Watermark`, which imitates watermarks in a piece of paper. Finally, the Post Phase includes augmentations that imitate noisy-processes that occur after a document has been created. Here we find `BadPhotoCopy`, which uses added noise to generate an effect of dirty copier, and `BookBinding`, which creates an effect with shadow and curved lines to imitate how a page from a book might appear after capture by a flatbed scanner.

Table 2. Individual *Augraphy* augmentations for each augmentation phase, in suggested position within a pipeline. Augmentations that work well in more than one phase are listed in the last column.

Ink Phase	Paper Phase	Post Phase	Multiple
BleedThrough	ColorPaper	BadPhotoCopy	BrightnessTexturize
LowInkPeriodicLines	Watermark	BindingsAndFasteners	DirtyDrum
LowInkRandomLines	Gamma	BookBinding	DirtyRollers
InkBleed	LightingGradient	Folding	Dithering
Letterpress	SubtleNoise	JPEG	Geometric
	DelaunayTessellation	Markup	NoiseTexturize
	VoronoiTesselation	Faxify	Scribbles
	PatternGenerator	PageBorder	LinesDegradation
			Brightness

In addition to document-specific transformations, *Augraphy* also includes transformations like blur, scaling, and rotation, reducing the need for additional general-purpose augmentation libraries. Descriptions of all *Augraphy* augmentations are available online, along with the motivation for their development and usage examples. [3]

4.2 The Library

There are four "main sequence" classes in the *Augraphy* codebase, which together provide the bulk of the library's functionality, and when composed generate realistic synthetically-augmented document images:

`Augmentation`. The `Augmentation` class is the most basic class in the project, and essentially exists as a thin wrapper over a probability value in the interval [0,1]. Every augmentation contained in a pipeline has its own chance of being applied during that pipeline's execution.

`AugmentationResult`. After an augmentation is applied, the output of its execution is stored in an `AugmentationResult` object and passed forward through the pipeline. These also record a full copy of the augmentation runtime data, as well as any metadata that might be relevant for debugging or other advanced use.

[3] `https://augraphy.readthedocs.io/en/latest/`

`AugmentationSequence`. A list of `Augmentations` — together with the intent to apply those Augmentations in sequence — determines an `AugmentationSequence`, which is itself both an `Augmentation` and callable (behaves like a function). In practice, these are model the pipeline phases discussed previously; they are essentially lists of `Augmentation` constructor calls which produce callable `Augmentation` objects of the various flavors explored in `AugmentationSequences` are applied to the image during each of the `AugmentationPipeline` phases, and in each case yield the image, transformed by some of the Augmentations in the sequence.

`AugmentationPipeline`. The bulk of the heavy lifting in *Augraphy* resides in the Augmentation pipeline, which is our abstraction over one or more events in a physical document's life.

Consider the following sequence:

1. Ink is adhered to paper material during the initial printing of a document.
2. The document is attached to a public bulletin board in a high-traffic area.
3. The pages are annotated, defaced, and eventually torn away from their securing staples, flying away in the wind.
4. Fifty years later, the tattered pages are discovered and turned over to library archivists.
5. These conservationists use delicate tools to carefully position and record images of the document, storing these in a digital repository.

An `AugmentationPipeline` represents such document lifetimes by composing augmentations modeling each of these individual events, while collecting runtime metadata about augmentations and their parameters and storing copies of intermediate images, allowing for inspection and fine-tuning to achieve outputs with desired features, facilitating (re)production of documents as in Figure 1.

Realistically reproducing effects in document images requires careful attention to how those effects are produced in the real world. Many issues, like the various forms of misprint, only affect text and images on the page. Others, like a coffee spill, change properties of the paper itself. Further still, there are transformations like physical deformations which alter the geometry and topology of both the page material and the graphical artifacts on it. Effectively capturing processes like these in reproducible augmentations means separating our model of a document pipeline into ink, paper, and post-processing layers, each containing some augmentations that modify the document image as it passes through. With *Augraphy*, producing realistically-noisy document images can be reduced to the definition and application of one or more *Augraphy* pipelines to some clean, born-digital document images.

There are also two classes that provide additional critical functionality in order to round out the *Augraphy* base library:

`OneOf`. To model the possibility that a document image has undergone one and only one of a collection of augmentations, we use `OneOf`, which simply selects one of those augmentations from the given list, and uses this to modify the image.

`PaperFactory`. We often print on multiple sizes and kinds of paper, and out in the world we certainly *encounter* such diverse documents. We introduce this variation into the `AugmentationPipeline` by including `PaperFactory` in the `paper` phase of the pipeline. This augmentation checks a local directory for images of paper to crop, scale, and use as a background for the document image. The pipeline contains edge

detection logic for lifting only text and other foreground objects from a clean image, greatly simplifying the "printing" onto another "sheet", and capturing in a reproducible way the construction method used to generate the NoisyOffice database [49]. Taken together, `PaperFactory` makes it trivial to re-print a document onto other surfaces, like hemp paper, cardboard, or wood.

Interoperability and flexibility are core requirements of any data augmentation library, so *Augraphy* includes several utility classes designed to improve developer experience:

`ComposePipelines`. This class provides a means of composing two pipelines into one, allowing for the construction of complex multi-pipeline operations.

`Foreign`. This class can be used to wrap augmentations from external projects like *Albumentations* and *imgaug*.

`OverlayBuilder`. This class uses various blending algorithms to fuse foreground and background images together, which is useful for "printing" or applying other artifacts like staples or stamps.

`NoiseGenerator`. This class uses make blobs algorithm to generate mask of noises in different shape and location.

5 Deep Learning with *Augraphy*

Augraphy aims to facilitate rapid dataset creation, advancing the state of the art for document image analysis tasks. This section describes a brief experiment in which *Augraphy* was used to augment a new collection of clean documents, producing a new corpus which fills a feature gap in existing public datasets. This collection was used to train a denoising convolutional neural network capable of high-accuracy predictions, demonstrating *Augraphy*'s utility for robust data augmentation.

All code used in these experiments is available in the `augraphy-paper` GitHub repository [4].

5.1 Model Architecture

To evaluate *Augraphy*, we used an off-the-shelf Nonlinear Activation-Free Network (NAFNet), [8], making only minor alterations to the model's training hyperparameters, changing the batch size and learning epoch count to fit our training data.

5.2 Data Generation

Despite recent techniques ([6], [32]) for reducing the volume of input data required to train models, large datasets remain king; feeding a model more data during training can help ensure better latent representations of more features, improving robustness of the model to noise and increasing its capacity for generalization to new data.

[4] `https://github.com/sparkfish/augraphy-paper`

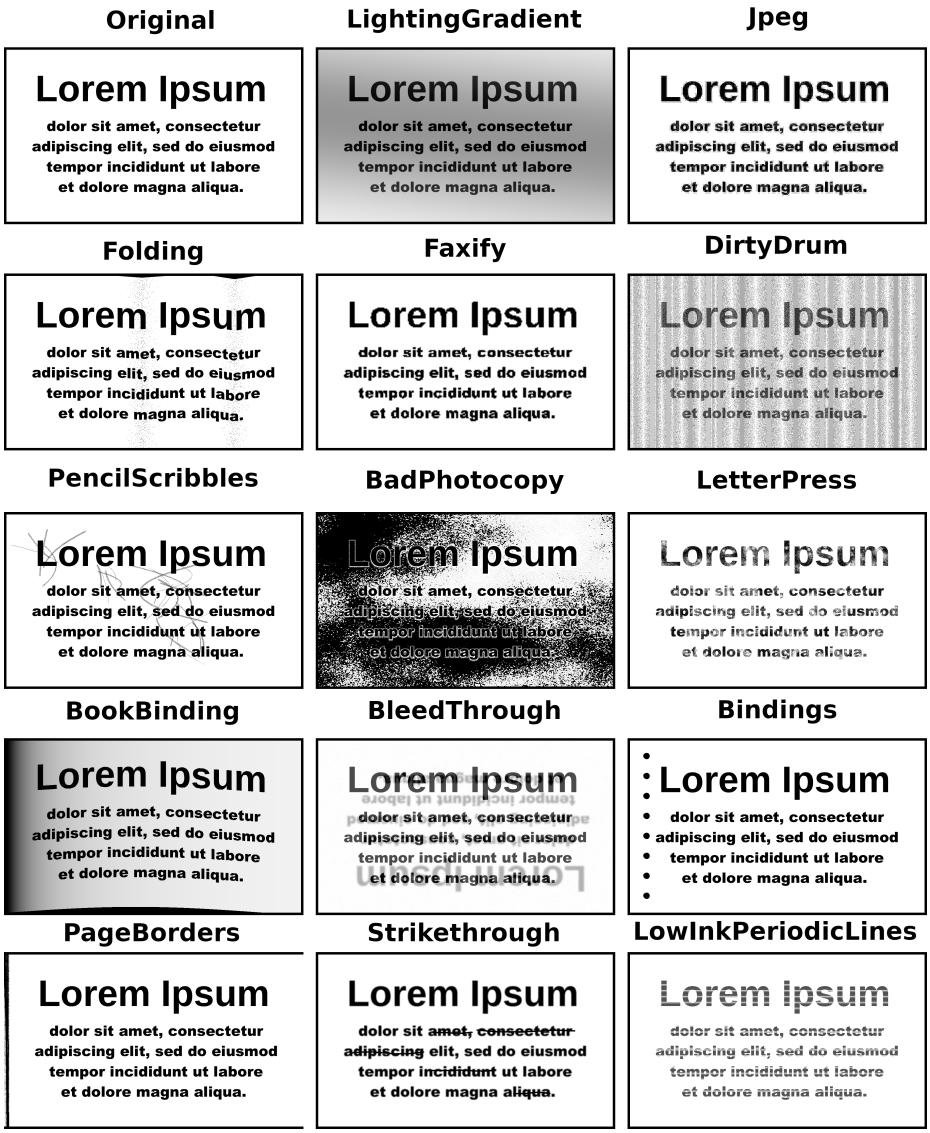

Fig. 3. Examples of some *Augraphy* augmentations.

To train our model, we developed the *ShabbyPages* dataset [41], a real-world document image corpus with *Augraphy*-generated synthetic noise. Gathering the 600 ground-truth documents was the most challenging part, requiring the efforts of several crowd-workers across multiple days. Adding the noise to these images was trivial however; *Augraphy* makes it easy to produce large training sets. For additional realistic variation, we also gathered 300 paper textures and used the PaperFactory augmentation to "print" our source documents onto these.

The initial 600 PDF documents were split into their component pages, totaling 6202 clean document images exported at 150 pixels per inch. We iteratively developed an *Augraphy* pipeline which produced satisfactory output with few overly-noised images, then ran each of the clean images through this to produce the base training set. Patches were taken from each of these images and used for the final training collection.

Another new dataset — *ShabbyReal* — was produced as part of the larger *ShabbyPages* corpus [41]; this data was manually produced by applying sequences of physical operations to real paper. *ShabbyReal* is fully out-of-distribution for this experiment, and also has a very high degree of diversity, providing good conditions for evaluating *Augraphy*'s effect.

5.3 Training Regime

The NAFNet was given a large sample to learn from: for each image in the ShabbyPages set, we sampled ten patches of 400x400 pixels, using these to train the instance across just 16 epochs.

The model was trained with mean squared error as the loss function, using the Adam optimizer and cosine annealing learning rate scheduler, then evaluated with the mean average error metric.

5.4 Results

Sample predictions from the model on the validation task are presented in Figure 4. The SSIM score indicates that these models generalize very well, though they do over-compensate for shadowing features around text, by increasing the line thickness in the prediction, resulting in a bold font.

To compare the model's performance on the validation task, we considered the root mean square error (RMSE), structural similarity index (SSIM), and peak signal-to-noise ratio (PSNR) metrics.

Table 3. Denoising performance on *ShabbyReal* validation task

Metric	Score
SSIM	0.71
PSNR	32.52
RMSE	6.52

As can be seen in Figure 4, the model was able to remove a significant amount of blur, line noise, shadowing from tire tracks, and watercoloring.

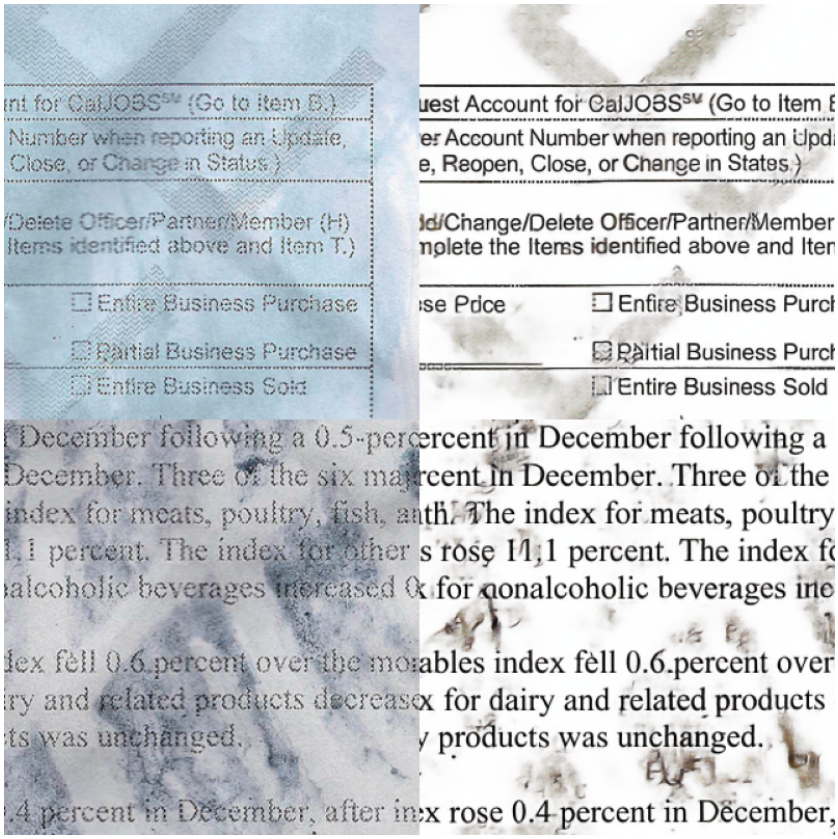

Fig. 4. Sample *ShabbyReal* noisy images (left) and their denoised counterparts (right).

6 Robustness Testing

In this section, we examine the use of *Augraphy* to add noise which challenges existing models.

We first use *Augraphy* to add noise to an image of text with known groundtruth. The Tesseract [45] pre-trained OCR model's performance on the clean image is compared to the OCR result on the *Augraphy*-noised image using the Levenshtein distance.

Then, we add *Augraphy*-generated noise to document images which contain pictures of people, and perform face-detection on the output, comparing the detection accuracy to that of the clean images.

6.1 Character Recognition

We first compiled 15 ground-truth, noise-free document images from a new corpus of born-digital documents, whose ground-truth strings are known. We then used Tesseract to generate OCR predictions on these noise-free documents, as a baseline for comparison. We considered these OCR predictions as the ground-truth labels for each docu-

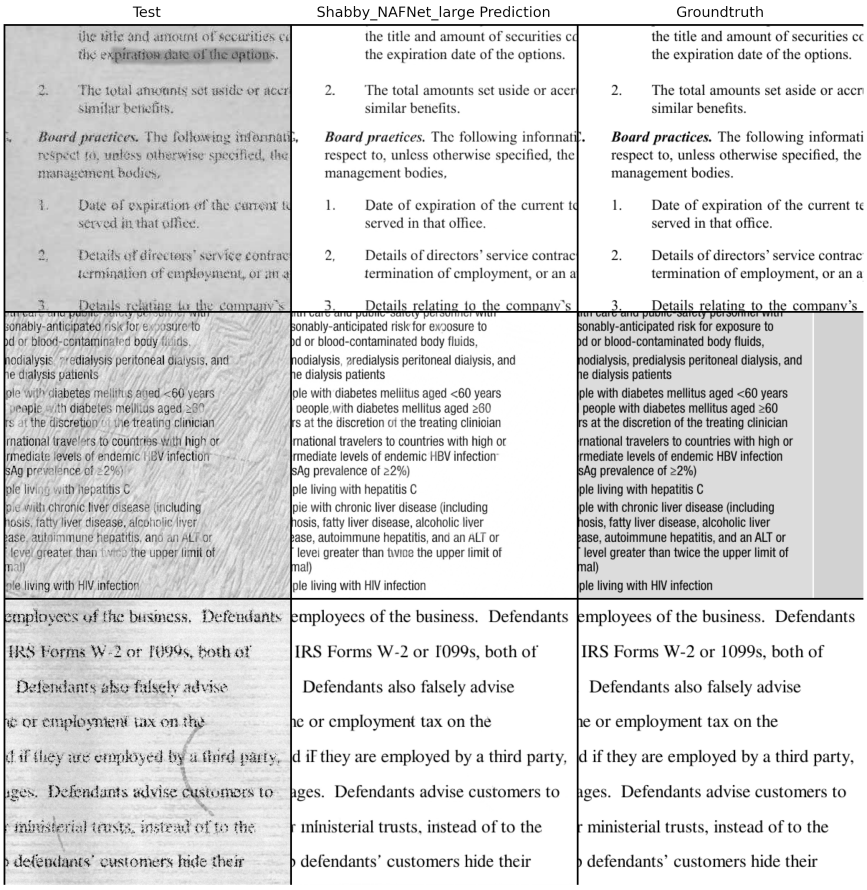

Fig. 5. Sample *ShabbyPages* noisy images (left), denoised by Shabby_NAFNet (center), and groundtruth (right).

ment. Next, we generated noisy versions of the 15 documents by running them through an *Augraphy* pipeline, and again used Tesseract to generate OCR predictions on these noisy documents. We compared the word accuracy rate on the noisy OCR results versus the ground-truth noise-free OCR results, and found that the noisy OCR results were on average 52% less accurate, with a range of up to 84%. This example use-case demonstrates the effectiveness of using *Augraphy* to create challenging test data that remains human-readable, suitable for evaluating OCR systems.

6.2 Face Detection Robustness Testing

In this section, we move beyond text-related tasks to robustness analysis of image classifiers and object detectors. In particular, we examine the robustness

Table 4. Face detection performance.

Model	Accuracy
Azure	57.1%
Google	60.1%
Amazon	49.1%
UltraFace	4.5%

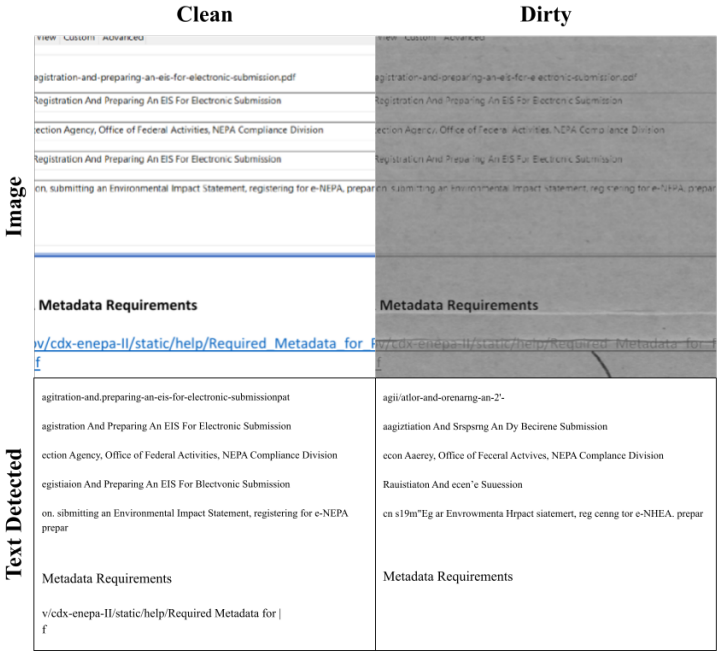

Fig. 6. Sample *ShabbyReal* clean/noisy image pair, and the OCR output for each. The computed Levenshtein distance between the OCR result strings is 203.

of face detection models on *Augraphy*-altered images containing faces, such as newspapers [31] and legal identification documents [3], which are often captured with noisy scanners.

We begin by sampling 75 images from the FDDB face detection benchmark [22], using *Augraphy* to generate 10 altered versions of each image, manually removing any augmented image where the face(s) is not reasonably visible. We then test four face detection models on the noisy and noise-free images. These models are: proprietary face detection models from Google,[5] Amazon,[6] and Microsoft[7], and lastly the freely-available UltraFace[8] model.

[5] https://cloud.google.com/vision/docs/detecting-faces
[6] https://aws.amazon.com/rekognition/
[7] https://azure.microsoft.com/en-us/services/cognitive-services/face
[8] https://github.com/onnx/models/tree/main/vision/body_analysis/ultraface

Table 4 shows face detection accuracy of both models on the noisy test set, and example images where no faces were detected by the Microsoft model are shown in Figure 7. We see that all four models struggle on the *Augraphy*-augmented data, with the proprietary models seeing detection performances drop to between roughly 50-60%, and UltraFace detecting only 4.5% of faces that it found in the noise-free data.

Fig. 7. Example augmented images that yielded false-positive predictions with the Azure face detector.

7 Conclusion and Future Work

We presented *Augraphy*, a framework for generating realistic synthetically-augmented datasets of document images. Other available image augmentation tools were examined and found to lack features needed for our purposes, motivating the creation of this library specifically targeting the types of alterations and degradations seen in document images.

We described the process for using *Augraphy* to create a new document image dataset containing synthetic real-world noise, then compared results obtained by training a convolutional NAFNet instance on this corpus.

Finally, we examined *Augraphy*'s efficacy in generating confounding data for testing the robustness of document vision models.

Future work on the *Augraphy* library will focus on adding new types of augmentations, increasing performance to enable faster creation of larger datasets on commodity hardware, providing more scale-invariant support so that augmentations perform well at all document image resolutions, and responding to community-initiated feature requests.

Augraphy is licensed under the MIT open source license, and readers are invited to share feedback and participate in its development on GitHub.

References

1. Albano, G.L., Sparro, M.: The role of digitalization in public service provision: Evidence and implications for the efficiency of local government services. Local Government Studies **44**(5), 613–636 (2018)
2. Alkhezzi, F., Alsabawy, A.Y.: Factors influencing the implementation of learning management systems in higher education: A case study. Education and Information Technologies **25**(4), 2827–2845 (2020)
3. Arlazarov, V., Bulatov, K., Chernov, T., Arlazarov, V.: Midv-500: A dataset for indentity document analysis and recognition on mobile devices in video stream. Computer Optics **43**(5) (2019), https://arxiv.org/ftp/arxiv/papers/1807/1807.05786.pdf
4. Bloice, M.D., Roth, P.M., Holzinger, A.: Biomedical image augmentation using Augmentor. Bioinformatics **35**(21), 4522–4524 (04 2019). https://doi.org/10.1093/bioinformatics/btz259, https://doi.org/10.1093/bioinformatics/btz259
5. Buslaev, A., Iglovikov, V.I., Khvedchenya, E., Parinov, A., Druzhinin, M., Kalinin, A.A.: Albumentations: Fast and flexible image augmentations. Information **11**(2) (2020). https://doi.org/10.3390/info11020125, https://www.mdpi.com/2078-2489/11/2/125
6. Cao, Y., Yu, H., Wu, J.: Training vision transformers with only 2040 images. CoRR **abs/2201.10728** (2022), https://arxiv.org/abs/2201.10728
7. Castro-Bleda, M.J., España-Boquera, S., Pastor-Pellicer, J., Zamora-Martínez, F.: The Noisy-Office Database: A Corpus to Train Supervised Machine Learning Filters for Image Processing. The Computer Journal **63**(11), 1658–1667 (11 2019). https://doi.org/10.1093/comjnl/bxz098, https://doi.org/10.1093/comjnl/bxz098
8. Chen, L., Chu, X., Zhang, X., Sun, J.: Simple baselines for image restoration. arXiv preprint arXiv:2204.04676 (2022)
9. Cheriet, M., Kharma, N., Liu, C.L., Suen, C.Y.: Character Recognition Systems: A Guide for Students and Practitioners. Wiley (2007)
10. Chuen, D.L.K., Deng, R.H.: Handbook of Blockchain, Digital Finance, and Inclusion: Cryptocurrency, FinTech, InsurTech, Regulation, ChinaTech, Mobile Security, and Distributed Ledger. Academic Press (2017)
11. Dodge, S., Karam, L.: Understanding how image quality affects deep neural networks. In: 2016 Eighth International Conference on Quality of Multimedia Experience (QoMEX). pp. 1–6 (2016). https://doi.org/10.1109/QoMEX.2016.7498955
12. Fadaee, M., Bisazza, A., Monz, C.: Data augmentation for low-resource neural machine translation. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). pp. 567–573. Association for Computational Linguistics, Vancouver, Canada (Jul 2017). https://doi.org/10.18653/v1/P17-2090, https://aclanthology.org/P17-2090
13. Feng, S.Y., Gangal, V., Wei, J., Chandar, S., Vosoughi, S., Mitamura, T., Hovy, E.: A survey of data augmentation approaches for NLP. In: Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021. pp. 968–988. Association for Computational Linguistics, Online (Aug 2021). https://doi.org/10.18653/v1/2021.findings-acl.84, https://aclanthology.org/2021.findings-acl.84
14. Gangeh, M.J., Plata, M., Motahari Nezhad, H.R., Duffy, N.P.: End-to-end unsupervised document image blind denoising. In: 2021 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 7868–7877 (2021). https://doi.org/10.1109/ICCV48922.2021.00779

15. Gatzert, N., Schmeiser, H.: The impact of various digitalization aspects on insurance value: An enterprise risk management approach. The Geneva Papers on Risk and Insurance-Issues and Practice **41**(3), 385–405 (2016)
16. Harley, A.W., Ufkes, A., Derpanis, K.G.: Evaluation of deep convolutional nets for document image classification and retrieval. In: International Conference on Document Analysis and Recognition (ICDAR) (2015)
17. Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M.H., Brett, M., Haldane, A., del Río, J.F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., Oliphant, T.E.: Array programming with NumPy. Nature **585**(7825), 357–362 (Sep 2020). https://doi.org/10.1038/s41586-020-2649-2, https://doi.org/10.1038/s41586-020-2649-2
18. Hendrycks, D., Dietterich, T.: Benchmarking neural network robustness to common corruptions and perturbations. In: International Conference on Learning Representations (ICLR) (2019)
19. Homeyer, A., Geißler, C., Schwen, L.O., Zakrzewski, F., Evans, T., Strohmenger, K., Westphal, M., Bülow, D., Kargl, M., Karjauv, A., Munné-Bertran, I., Retzlaff, C.O., Romero-López, A., Sołtysiński, T., Plass, M., Carvalho, R., Steinbach, P., Lan, Y.C., Bouteldja, N., Haber, D., Rojas-Carulla, M., Sadr, A.V., Kraft, M., Krüger, D., Tick, R., Lang, T., Boor, P., Müller, H., Hufnagl, P., Zerbe, N.: Recommendations on test datasets for evaluating ai solutions in pathology. arXiv preprint arXiv:2204.14226 (2022), https://arxiv.org/pdf/2204.14226.pdf
20. Hosseini, H., Xiao, B., Poovendran, R.: Google's cloud vision API is not robust to noise. arXiv preprint arXiv:1704:05051 (2017)
21. Jadhav, P., Sawal, M., Zagade, A., Kamble, P., Deshpande, P.: Pix2pix generative adversarial network with resnet for document image denoising. In: 2022 4th International Conference on Inventive Research in Computing Applications (ICIRCA) (2022)
22. Jain, V., Learned-Miller, E.: FDDB: A benchmark for face detection in unconstrained settings. Tech. Rep. UM-CS-2010-009, University of Massachusetts, Amherst (2010)
23. Jaume, G., Ekenel, H.K., Thiran, J.P.: Funsd: A dataset for form understanding in noisy scanned documents. In: Accepted to ICDAR-OST (2019)
24. Journet, N., Visani, M., Mansencal, B., Van-Cuong, K., Billy, A.: Doccreator: A new software for creating synthetic ground-truthed document images. Journal of Imaging **3**(4) (2017). https://doi.org/10.3390/jimaging3040062, https://www.mdpi.com/2313-433X/3/4/62
25. Jung, A.B., Wada, K., Crall, J., Tanaka, S., Graving, J., Reinders, C., Yadav, S., Banerjee, J., Vecsei, G., Kraft, A., Rui, Z., Borovec, J., Vallentin, C., Zhydenko, S., Pfeiffer, K., Cook, B., Fernández, I., De Rainville, F.M., Weng, C.H., Ayala-Acevedo, A., Meudec, R., Laporte, M., et al.: imgaug. https://github.com/aleju/imgaug (2020), online; accessed 01-Feb-2020
26. Karahan, S., Kilinc Yildirum, M., Kirtac, K., Rende, F.S., Butun, G., Ekenel, H.K.: How image degradations affect deep cnn-based face recognition? In: 2016 International Conference of the Biometrics Special Interest Group (BIOSIG). pp. 1–5 (2016). https://doi.org/10.1109/BIOSIG.2016.7736924
27. Kim, D., Hong, T., Yim, M., Kim, Y., Kim, G.: Technical report on web-based visual corpus construction for visual document understanding. arXiv preprint arXiv:2211.03256 (2022), https://arxiv.org/pdf/2211.03256.pdf
28. Ko, T., Peddinti, V., Povey, D., Khudanpur, S.: Audio augmentation for speech recognition. In: Proc. Interspeech 2015. pp. 3586–3589 (2015). https://doi.org/10.21437/Interspeech.2015-711

29. Kulkarni, M., Kakad, S., Mehra, R., Mehta, B.: Denoising documents using image processing for digital restoration. In: Swain, D., Pattnaik, P.K., Gupta, P.K. (eds.) Machine Learning and Information Processing. pp. 287–295. Springer Singapore, Singapore (2020)
30. Larson, S., Lim, G., Ai, Y., Kuang, D., Leach, K.: Evaluating out-of-distribution performance on document image classifiers. In: Proceedings of the Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (2022), https://openreview.net/pdf?id=uDlkiCI5N7Y
31. Lee, B.C.G., Mears, J., Jakeway, E., Ferriter, M., Adams, C., Yarasavage, N., Thomas, D., Zwaard, K., Weld, D.S.: The newspaper navigator dataset: Extracting headlines and visual content from 16 million historic newspaper pages in chronicling america. In: Proceedings of the 29th ACM International Conference on Information Knowledge Management (CIKM) (2020), https://doi.org/10.1145/3340531.3412767
32. Lee, S.H., Lee, S., Song, B.C.: Vision transformer for small-size datasets (2021). https://doi.org/10.48550/ARXIV.2112.13492, https://arxiv.org/abs/2112.13492
33. Maguolo, G., Paci, M., Nanni, L., Bonan, L.: Audiogmenter: a matlab toolbox for audio data augmentation. Applied Computing and Informatics (2021)
34. McFee, B., Humphrey, E., Bello, J.: A software framework for musical data augmentation. In: Muller, M., Wiering, F. (eds.) Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015. pp. 248–254. Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015, International Society for Music Information Retrieval (2015)
35. Menachemi, N., Collum, T.H.: Benefits and drawbacks of electronic health record systems. Risk Management and Healthcare Policy **4**, 47–55 (2011)
36. Mohamed, S.S.A., Rashwan, M.A.A., Abdou, S.M., Al-Barhamtoshy, H.M.: Patch-based document denoising. In: 2018 International Japan-Africa Conference on Electronics, Communications and Computations (JAC-ECC) (2018)
37. Mustafa, W.A., Kader, M.M.M.A.: Binarization of document image using optimum threshold modification. Journal of Physics: Conference Series **1019**, 012022 (jun 2018). https://doi.org/10.1088/1742-6596/1019/1/012022, https://doi.org/10.1088/1742-6596/1019/1/012022
38. O'Gorman, L., Kasturi, R.: Document Image Analysis. IEEE Computer Society (1997)
39. Papakipos, Z., Bitton, J.: AugLy: Data augmentations for robustness. arXiv preprint arXiv:2201:06494 (2022)
40. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019), http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
41. Project, T.A.: Shabbypages dataset (2022), https://github.com/sparkfish/shabby-pages
42. Rotman, D., Azulai, O., Shapira, I., Burshtein, Y., Barzelay, U.: Detection masking for improved OCR on noisy documents. arXiv preprint arXiv:2205.08257 (2022)
43. Saifullah, Siddiqui, S.A., Agne, S., Dengel, A., Ahmed, S.: Are deep models robust against real distortions? a case study on document image classification. In: Proceedings of the 26th International Conference on Pattern Recognition (ICPR) (2022), https://www.computer.org/csdl/proceedings-article/icpr/2022/09956167/1IHoLM9J3gI

44. Schömig-Markiefka, B., Pryalukhin, A., Hulla, W., Bychkov, A., Fukuoka, J., Madabhushi, A., Achter, V., Nieroda, L., Büttner, R., Quaas, A., Tolkach, Y.: Quality control stress test for deep learning-based diagnostic model in digital pathology. Modern pathology : an official journal of the United States and Canadian Academy of Pathology, Inc **34**(12), 2098L = https://europepmc.org/articles/PMC8592835 (December 2021). https://doi.org/10.1038/s41379-021-00859-x
45. Smith, R.: An overview of the tesseract ocr engine. In: Ninth international conference on document analysis and recognition (ICDAR 2007). vol. 2, pp. 629–633. IEEE (2007)
46. Staudt, R.W., Medeiros, A.P.: Access to justice and technology clinics: A 4% solution. Chicago-Kent Law Review **88**(3), 695–728 (2015)
47. Vasiljevic, I., Chakrabarti, A., Shakhnarovich, G.: Examining the impact of blur on recognition by convolutional networks. arXiv preprint arXiv:1611.05760 (2016)
48. Wei, J., Zou, K.: EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). pp. 6382–6388. Association for Computational Linguistics, Hong Kong, China (Nov 2019). https://doi.org/10.18653/v1/D19-1670, https://aclanthology.org/D19-1670
49. Zamora-Martínez, F., España Boquera, S., Castro-Bleda, M.: Behaviour-based clustering of neural networks applied to document enhancement. In: Computational and Ambient Intelligence (2007), https://link.springer.com/chapter/10.1007/978-3-540-73007-1_18