Revisiting the privacy of censored credentials

Viktor Garske[1] Andreas Noack[2]

Abstract: On the internet, you find numerous images like screenshots where secret parts are hidden with irreversible redaction techniques like pixelation or blurring. In this paper, we propose a system that recovers information from redacted text in raster graphics using a composition of a Convolutional Neural Network (CNN), a Recurrent Neural Network (RNN) using Long short-term memory (LSTM) and a Connectionist Temporal Classification (CTC) layer to output the most probable character sequence. We furthermore show that our model operates in an automated pipeline, performs on blurred images without modification and is even able to compensate JPEG quality loss. Finally, our test results indicate that a generic neural network can be trained successfully to assist the recovery of pixelized or blurred information on screenshots or high-quality photos.

Keywords: Neural networks, privacy, machine learning, computer vision, password, credentials, pixelized

1 Introduction

On different platforms on the internet, users utilize images like screenshots to explain complex issues or just to make things clear. In some cases, however, those screenshots contain credentials like usernames, addresses or even passwords that the uploaders do not want to share with everyone else. This is where redaction techniques like pixelation or blurring come into play, to keep the credentials secret.

Image manipulation tools like *GIMP* provide functions to redact information in photos or screenshots conveniently. From a security point of view, pixelation of data is oftenly less secure than uploaders think, as shown in previous work by Garske; Noack [GN22].

In this paper we analyze the redaction quality of Gaussian Blur and investigate whether it delivers similar weak results than pixelation or can keep the redacted data secret. Our working hypothesis is that Gaussian Blur performs even less secure than pixelation. Although pixelation and blurring are both lossy functions, blurring makes intense use of local redundancy which probably makes it easier to extract remaining information.

We contribute a generic anti-redaction system consisting of a preprocessor and a set of neural networks that is able to extract information from redacted image snippets. Our system

[1] University of Applied Sciences Stralsund, Department of Electrical Engineering and Computer Science, Zur Schwedenschanze 15, 18435 Stralsund, Germany, viktor.garske@hochschule-stralsund.de
[2] University of Applied Sciences Stralsund, Department of Electrical Engineering and Computer Science, Zur Schwedenschanze 15, 18435 Stralsund, Germany, andreas.noack@hochschule-stralsund.de

doi:10.18420/inf2022_04

provides the user with the most probable character sequence of the redacted data while using a Convolutional Neural Network (CNN), a Recurrent Neural Network (RNN) with Long short-term memory (LSTM) and a Connectionist Temporal Classification (CTC) layer in its backend. Through the use of a preprocessor, our system is resilient to JPEG quality loss. Additionally, this paper contributes a security comparison between pixelation and Gaussian Blur for different pixel block sizes and blurring radii.

The organization of this paper is as follows: Section 2 deals with related and previous work on the topic of recovering information from redacted images. We give an overview about the Gaussian Blur technique in Section 3. Section 4 introduces our approach to gain information from pixelized character sequences using a system of neural networks. In Section 5, we demonstrate how accurate our solution is. Finally, Section 7 concludes this paper and provides an outlook.

2 Related Work

In the literature, several methods for recovering redacted information have been discussed. The recovering algorithms can either be classified by their target, e.g. faces, license plates or text images or by their technique, e.g. brute-force or neural network approaches.

One big research field deals with deblurring of objects in images and videos. Blurring can occur inadvertently as a result of camera shaking, that is why the motivations for related work are multifaceted and not only privacy related. Xu et al. [Xu17], for example, use generative adversarial networks (GAN) with novel training losses for deblurring faces and text in images. In addition to it, they compare several alternative methods and state that they achieve better results for finer details. Menon et al. [Me20] contribute the PULSE algorithm enabling reliable reconstruction of pixelized faces in images using GANs and Latent Space Exploration. Their system enables users to output realistic images in high-resolution. Our approach differs from GAN approaches as we do not aim for an improved image but predict the underlying text sequence.

A forensic use case for the recovery of text in images arises from low-resolution images of CCTV. Kaiser et al. [Ka21] evaluated opportunities for text recovering from license plates, especially from low-quality JPEG images. They also incorporate neural networks in their work.

Machine Learning is not necessarily required for such tasks: Cavedon et al. [CFV11] studied possibilities for full reconstruction of pixelized parts in videos. This is possible because the redacted part under the pixelation mask moves in videos, hence disclosing additional information that can be used for the reconstruction with a Maximum a Posteriori approach. They tested out their solution by recovering license plates and faces, where they were able to reveal information believed to be hidden.

McPherson et al. [MSS16] investigate the influence of deep learning techniques against image obfuscation. They experiment with specific datasets like the MNIST digits and demonstrate information recovery. Other than in our work, they do not focus on sequences of characters.

Hill et al. [Hi16] take such character sequences into account whereby their model is based on Hidden Markov Models (HMM) instead of neural networks. While their model is designed for pixelized images, they propose an additional pixelation step for blurred images in order to process them, too. This process differs from our work, because our model learns directly from blurred images without another pixelation step.

3 Redaction using Blurring

Blurring is one of several methods to obfuscate text in images. As with pixelation, the goal is to reduce information up to a point where no human can recognize the redacted text without raising too much visual attention.

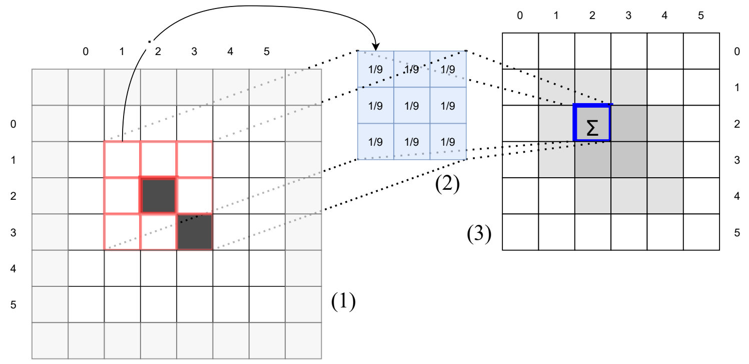


Fig. 1: Blurring

Fig. 1 illustrates the process for creating the blurred image. In order to compute the color value for a pixel on the position (x, y) in the redacted image, the algorithm averages the color values around the pixel (x, y) in the preimage.

$$\mathbf{S}_i \rightarrow \mathbf{S}_o : s_o(x,y)_m = \sum_{u=0}^{m-1} \sum_{v=0}^{m-1} s_e\left(x + \frac{m-1}{2} - u, y + \frac{m-1}{2} - v\right) \cdot h(u,v) \qquad (1)$$

The process can be summarized with e.g. Equation (1) (based on [Ni19, pp. 132 ff.]). Here, the pixel values of an input image S_i shall be mapped to an output image S_o. By selecting a proper m and kernel $\mathbf{H}$ with function $h(\cdot,\cdot)$, the output image will appear blurred. $\mathbf{H}$ and m depend on the selected technique. Fig. 1 shows a simple *box blur* where all eight pixels around a preimage pixel are averaged evenly, so that edges appear softer and contents will be harder to read. Another very common technique is the *Gaussian Blur* with $m = 3$ and the following kernel **H**.

$$\mathbf{H} = \frac{1}{16} \cdot \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} \tag{2}$$

The color values of some pixels are taken more into account than others. To make an image look blurrier, the radius and kernel size are increased accordingly. As the Gaussian Blur is based on the two dimensional Gaussian distribution, the kernel matrix **H** can be calculated with Equation (3).

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3}$$

After blurring, edges in the image appear smoother and contents are harder to read. This property is often desired to reduce image noise and details. In practice, blurring is provided by many implementations: image manipulation software like GIMP, ImageMagick or PIL offer different filters for image blurring.

4 Recovery of Redacted Information

The basic assumption for high-quality images is always the same: all redaction algorithms behave deterministically, i.e. two identical inputs always result in the same output. If we know the parameters for font family, font size, font weight or text color, we are able to compile a database to determine which input could have been created a given output. A naïve approach could incorporate a brute-force algorithm trying out all possibilities for a particular output sequence. This, however, requires huge resources. Machine Learning models can assist this process by predicting correct sequences for given blurred text images. In order to do so, the required underlying model has to be fitted to a particular training set which needs to match to the task.

Garske and Noack [GN22] create and demonstrate such an approach for pixelized images using a generic pipeline which is not tied to a specific redaction technique. This paper is based on that pipeline and neural network architecture (see Fig. 4 in Appendix). We investigate whether this approach also works for images that are not redacted by pixelation but by blurring.

The presented approach consists of two main components that aim to extract as much information as possible about the underlying text of blurred images:

1. A **preprocessor** that (a) converts pixelized images to grayscale and (b) aligns the image to left and center, and crops it to a uniform height. This is an important step to increase the accuracy of the following neural network.
2. A **neural network** including a CNN and RNN with the CTC loss function builds the core of our approach.

The proposed system generates key candidates that restrict the search space in order to optimize key recovery attacks.

4.1 Technical Remarks

The experiment is designed to approximate a real-world scenario as good as possible. With our training pipeline, text can be synthesized, blurred and saved closely to a real world sample like a screenshot that occurs with similar parameters and quality characteristics.

Furthermore, we focus on common passwords as training and test data to ensure that our neural network performs well on redacted credentials. We use a standard wordlist from the *SecLists* project by Miessler et al. [MHg21]. The selected wordlist is named `10-million-password-list-top-100000.txt` and contains the first 100,000 entries of a collection compiling the 10 million most used passwords, whereby we decided to only use the top 40,000 entries for performance reasons. The entries of the wordlist involve combinations of letters, numbers and special characters in one word.

In our redaction pipeline, text image synthesis is performed by the Python *Pillow* package [LCC21]. We use *GIMP* and the *PNG* file format with compression level 6, whereby level 1 has the best speed and level 9 the best compression rate. *PNG* is commonly used for lossless storage of high-quality images like screenshots.

4.2 Methodology

Our goal is to further analyze the previous presented pipeline and model. We want to investigate how a different redaction technique than pixelation and specific parameters like a different file format influence the performance of the prediction. In our study, blurring acts as the redaction technique and JPEG with its different quality levels as the file format.

The pipeline is able to automate the whole training process with just a set of parameters like font size or a particular wordlist. Other than some brute-force approaches, our model does not build on pixel-level analysis but works on a more abstract level due to its neural network backend. However, we want to determine whether the aforementioned neural network is able to predict the correct letters and digits with confidence when dealing with blurred images. For our experiments, we use GIMP 2.8.22 and the Gaussian Blur filter. All 40,000 training samples, each with dimensions of 256×32 pixels, are combined together to a collage with 51200×6400 pixels. See Fig. 4 in Appendix for the exact model configuration.

First, we analyze how the *blur radius* affects our results. Therefore we create a training set for each blur radius, fit a model using our neural network architecture and evaluate it using a test set. By setting the same seed for the random function, we ensure that the collages only differ in their blur radius.

Secondly, we will conduct this experiment three times in order to examine the role of the preprocessor. In the first pass, our preprocessor works as usual. The preprocessor aligns the image using the previously mentioned algorithm and applies an additional Gaussian Blur with radius 1 to remove noise, performed by the Python PIL library. In the second pass and third pass, we disable components of the preprocessor, i.e. the images will not be aligned and/or blurred. In each case the preprocessor measures the width of the text because this is necessary for the CTC layer of our neural network model.

5 Results

Like described in subsection 4.2, we analyze every radius three times: with full preprocessing (FPP), preprocessing without aligning and blurring (NAB = no aligning and blurring) and preprocessing without blurring but with aligning (NB = no blurring). Table 1 shows the Label Error Rate (LER) and the Sequence Error Rate (SER) for different blurring radii and each preprocessor option. A blur radius m means a radius with m pixels horizontal and m pixels vertically, or $m \times m$ pixels. As additional experiment, table 2 shows the LER and SER for different JPEG quality levels when a blur radius of 10×10 pixels is used.

In order to evaluate the effectiveness, two established metrics are used: the Label Error Rate (LER) as already proposed by Graves et al. [Gr06] and the Sequence Error Rate (SER) by Soullard et al. [SRP19]. Both metrics are also used in [GN22]. The Label Error Rate (LER) metric measures the **mean normalized** edit distances between classifications and preimage labels and is defined as follows:

$$LER(h, S') = \frac{1}{|S'|} \sum_{(\mathbf{x},\mathbf{z} \in S')} \frac{ED(h(\mathbf{x}), \mathbf{z})}{|\mathbf{z}|} \tag{4}$$

whereby the LER is computed for a temporal classifier h (i.e. the model) and a given test set S' that is disjoint from the training set S. x stands for the input sequences and z for corresponding labels. Both are compared with the edit distance function $ED(\cdot, \cdot)$ which is the number of characters that must be changed to make the input sequence equal to the corresponding label. The Sequence Error Rate (SER) is the percentage of predictions in the test set that are **not completely correct**.

You can see example images of a random character sequences after the different preprocessing steps in Table 3. For every blur radius, a training set is created based on the different

Radius	LER			SER		
	FPP	NAB	NB	NAB	FPP	NB
2	17.73 %	29.99 %	34.20 %	64.06 %	53.91 %	84.38 %
3	19.98 %	88.66 %	25.90 %	100.00 %	62.11 %	71.48 %
4	12.22 %	55.52 %	28.54 %	94.92 %	46.09 %	64.84 %
5	18.24 %	20.04 %	36.36 %	63.28 %	53.13 %	72.27 %
6	33.60 %	49.20 %	32.54 %	86.72 %	73.44 %	75.39 %
7	51.95 %	34.27 %	28.67 %	76.17 %	92.58 %	66.02 %
8	28.97 %	79.24 %	61.22 %	100.00 %	68.75 %	95.70 %
9	60.50 %	31.35 %	46.92 %	76.17 %	96.48 %	90.23 %
10	50.97 %	74.88 %	37.17 %	100.00 %	94.53 %	83.98 %
11	47.88 %	37.76 %	43.09 %	82.42 %	94.14 %	89.06 %
12	56.12 %	52.06 %	61.29 %	96.88 %	97.66 %	97.66 %
13	55.20 %	61.38 %	53.34 %	100.00 %	94.92 %	92.97 %
14	54.27 %	42.06 %	59.01 %	87.89 %	95.70 %	98.05 %
15	58.33 %	34.63 %	53.93 %	81.64 %	98.05 %	96.48 %
16	40.41 %	56.12 %	47.62 %	95.31 %	83.98 %	90.23 %
20	46.74 %	55.65 %	54.21 %	96.48 %	88.28 %	97.66 %
24	55.72 %	57.69 %	50.67 %	98.44 %	98.83 %	95.70 %
32	57.12 %	62.16 %	63.04 %	98.44 %	98.83 %	100.00 %

Tab. 1: Error rate of the proposed depixelation system, see Fig. 2 and Fig. 3 for chart illustration and Tab. 3 for training set examples.

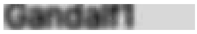
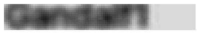
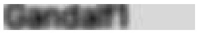
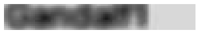
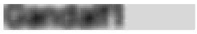
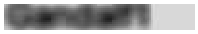
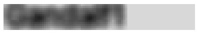
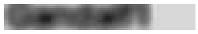
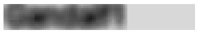
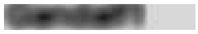
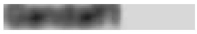
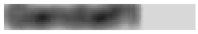
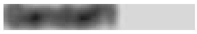
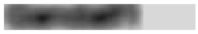
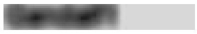
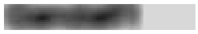
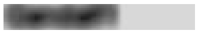
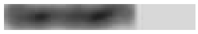
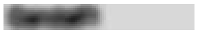
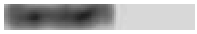
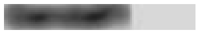
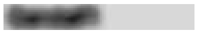
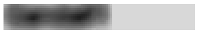
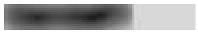
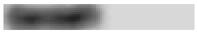
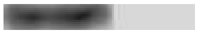
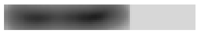
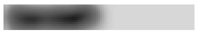
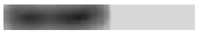
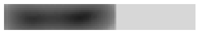
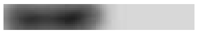
JPEG Quality Level	**LER**	**SER**
100	32.84%	75.00%
50	29.73%	69.14%
10	38.38%	81.64%

Tab. 2: JPEG Quality Loss Experiment with a blurring radius of 10×10 pixels

configurations described in subsection 4.2. Note that the scale of the input image does not change. This becomes clear in the NAB column where no alignment is performed. When increasing the blur radius, the blurred area takes more space on the canvas, especially towards the borders. The visualization component (preview) of our preprocessors with activated alignment scales the images up and shows a centered view so that it looks like the images would have different lengths which is not the fact.

6 Discussion

In our experiment, we analyze whether a Machine Learning model and corresponding pipeline for retrieving data from redacted images can be used for blurred images as well. The paper from Garske and Noack [GN22] demonstrates this on pixelized images only.

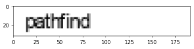
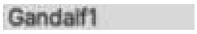
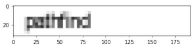
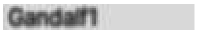
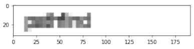
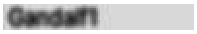
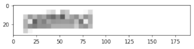
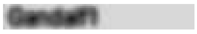
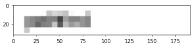
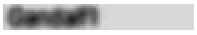
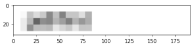
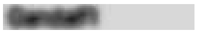
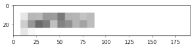
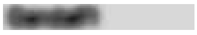
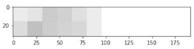
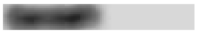
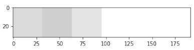
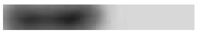
Radius	FPP	NAB	NB	Best LER
2				17.73 % (FPP)
3				19.98 % (FPP)
4				12.22 % (FPP)
5				18.24 % (FPP)
6				32.54 % (NB)
7				28.67 % (NB)
8				28.97 % (FPP)
9				31.35 % (NAB)
10				37.17 % (NB)
11				37.76 % (NAB)
12				52.06 % (NAB)
13				53.34 % (NB)
14				42.06 % (NAB)
15				34.63 % (NAB)
16				40.41 % (FPP)
20				46.74 % (FPP)
24				50.67 % (NB)
32				57.12 % (FPP)

Tab. 3: Examples for each blur radius in the different test passes

Our test results indicate that it is actually possible to train such a network on blurred images. Naturally, when the blur radius rises, the LER and SER will rise, too. However, the network can still recover information even if a human has trouble to decipher the image. We assume a blur radius of 10 pixels as the limit up to where a human can still recognize all characters without problems.

When considering pixelation and blurring, the question arises how they relate to each other. Tab. 4 provides a visual aid for comparing some block sizes of pixelized texts and blur radii of blurred texts. It becomes clear that users need a higher blur radius in order to achieve a comparable visual redaction level as with pixelation.

When comparing the LERs for both anti-redaction techniques (e.g. 32x32 px), we can see that our initial working hypothesis is confirmed. Retrieving data from a blurred text works at least as accurate as for pixelized text. That means blurring has in the best case the same security as pixelation, rather a little bit less. Note that [GN22] use a slightly different dataset than it is used in this experiment: for performance reasons, only 40,000 instead of 100,000 unique passwords are used as a wordlist for the training. Additionally, we anticipate that each model is trained for those parameters (font size, etc.) that are also used for the test images.

Block Size	Example Image	LER	Blur Radius	Example Image	LER
2 × 2		2.99 %	2 × 2		17.73 %
3 × 3		16.00 %	5 × 5		18.24 %
4 × 4		20.47 %	7 × 7		28.67 %
5 × 5		36.17 %	9 × 9		31.35 %
6 × 6		28.92 %	10 × 10		37.17 %
7 × 7		41.17 %	12 × 12		52.06 %
8 × 8		49.93 %	14 × 14		42.06 %
16 × 16		61.09 %	20 × 20		46.74 %
32 × 32		69.25 %	32 × 32		57.12 %

Tab. 4: Visual aid for the comparison of pixelized and blurred images. Block sizes and blur radii are optically fitted. The examples for pixelized images are taken from [GN22].

7 Conclusion and Future Work

In this paper, we analyze the privacy of blurring whereby our research especially focuses on the obfuscation of credentials like passwords. We show that a neural networks can not only be used for recovering information from pixelized images but also from blurred images.

We point out that higher blur radii are required to enable comparable results regarding pixelation. If investigators are able to guess parameters like font family or font size, they can run a dictionary-like attack to create a model that can reconstruct redacted parts in e.g. publicly available images. Besides, it turned out that our pipeline and model are able to handle JPEG quality loss with only slightly increased error rates.

There is still room for improvement: the current architecture and preprocessor can be further examined to improve the LER and SER metrics and thus the results. In addition to it, the influence of other parameters like different font families can be investigated. Moreover, aspects of transfer learning remain open: the model can be extended in a way to transfer the knowledge from specific parameters to applications with more generic parameters. By doing so, one model could support different font sizes or even a variety of redaction techniques without the need for retraining every time.

References

[CFV11] Cavedon, L.; Foschini, L.; Vigna, G.: Getting the face behind the squares: Reconstructing pixelized video streams. In: 5th USENIX Workshop on Offensive Technologies (WOOT 11). 2011.

[GN22] Garske, V.; Noack, A.: Recovering information from pixelized credentials. In: *Lecture Notes in Informatics, Proceedings*, Sicherheit 2022, Apr. 8, 2022. Karlsruhe, pp. 129–141, Apr. 2022, ISBN: 978-3-88579-717-3.

[Gr06] Graves, A.; Fernández, S.; Gomez, F.; Schmidhuber, J.: Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. In: Proceedings of the 23rd International Conference on Machine Learning. ICML '06, Association for Computing Machinery, Pittsburgh, Pennsylvania, USA, pp. 369–376, 2006, ISBN: 1595933832, URL: https://doi.org/10.1145/1143844.1143891.

[Hi16] Hill, S.; Zhou, Z.; Saul, L.; Shacham, H.: On the (in) effectiveness of mosaicing and blurring as tools for document redaction. Proceedings on Privacy Enhancing Technologies 2016/4, pp. 403–417, 2016.

[Ka21] Kaiser, P.; Schirrmacher, F.; Lorch, B.; Rieß, C.: Learning to Decipher License Plates in Severely Degraded Images. In: MultiMedia FORensics in the WILD - accepted for publication. Jan. 11–11, 2021, URL: https://faui1-files.cs.fau.de/public/publications/mmsec/2021-Schirrmacher-LTD.pdf.

[LCC21] Lundh, F.; Clark, A.; Contributors: Pillow: The friendly PIL fork (Python Imaging Library) repository, Version 8.3.1, Apr. 1, 2021, URL: https://github.com/python-pillow/Pillow/tree/8.3.1.

[Me20] Menon, S.; Damian, A.; Hu, S.; Ravi, N.; Rudin, C.: Pulse: Self-supervised photo upsampling via latent space exploration of generative models. In: Proceedings of the ieee/cvf conference on computer vision and pattern recognition. Pp. 2437–2445, 2020.

[MHg21] Miessler, D.; Haddix, J.; g0tmi1k: SecLists, Mar. 1, 2021, URL: https://github.com/danielmiessler/SecLists.

[MSS16] McPherson, R.; Shokri, R.; Shmatikov, V.: Defeating Image Obfuscation with Deep Learning, 2016, arXiv: 1609.00408 [cs.CR].

[Ni19] Nischwitz, A.; Fischer, M.; Haberäcker, P.; Socher, G.: Bildverarbeitung: Band II des Standardwerks Computergrafik und Bildverarbeitung. Springer-Verlag, 2019, ISBN: 978-3-658-28704-7.

[SRP19] Soullard, Y.; Ruffino, C.; Paquet, T.: CTCModel: a Keras Model for Connectionist Temporal Classification, 2019, arXiv: 1901.07957 [cs.LG].

[Xu17] Xu, X.; Sun, D.; Pan, J.; Zhang, Y.; Pfister, H.; Yang, M.-H.: Learning to super-resolve blurry face and text images. In: Proceedings of the IEEE international conference on computer vision. Pp. 251–260, 2017.

A Additional Information

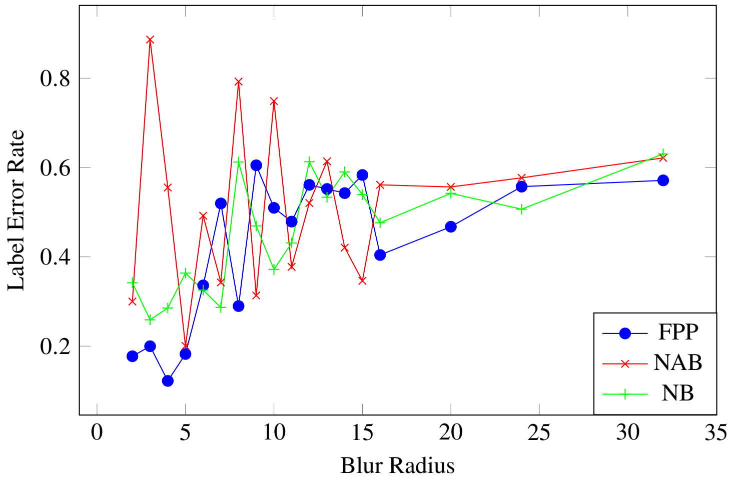


Fig. 2: Label error rates (LER) in relation to the blurring radius

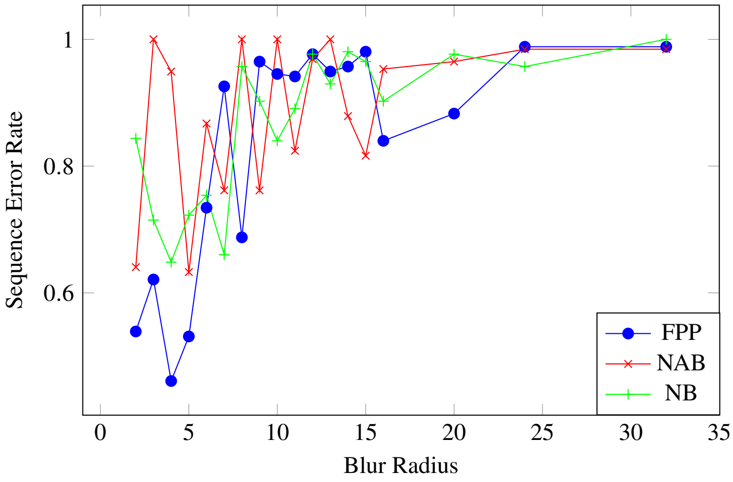


Fig. 3: Sequence error rates (SER) in relation to the blurring radius

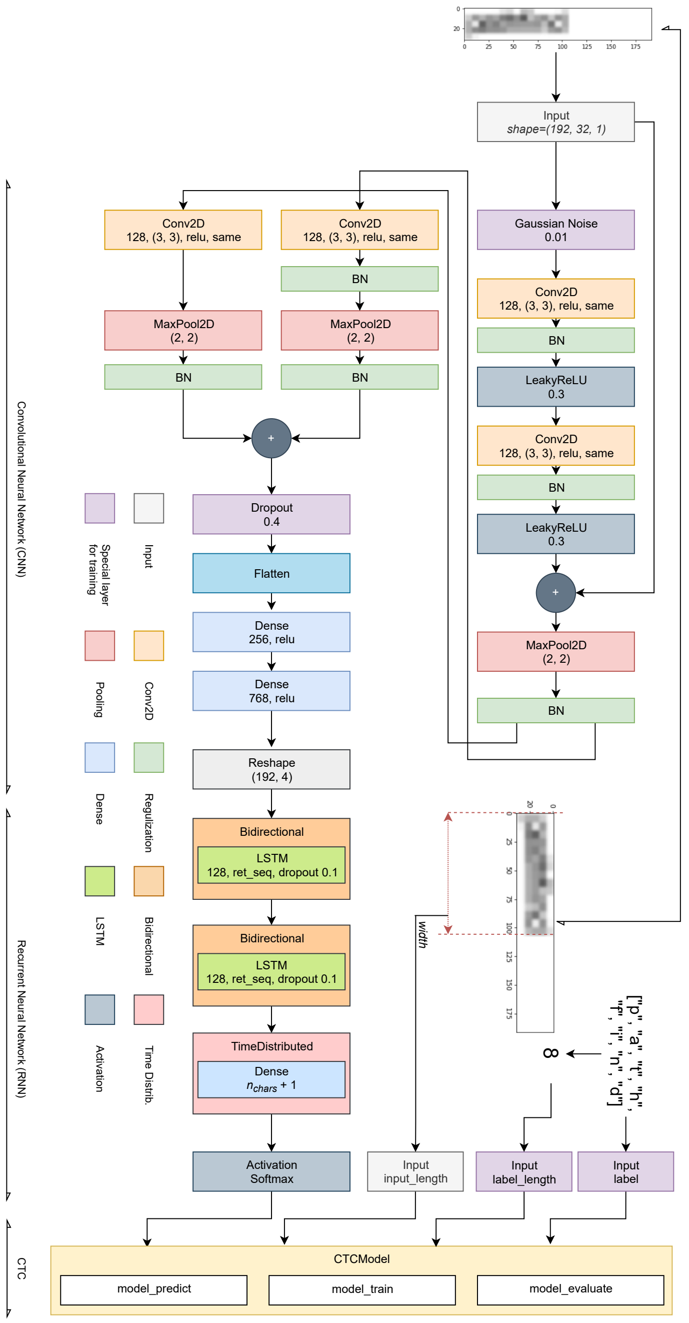


Fig. 4: Architecture of the neural network [GN22]