Vision-Based Approach to Noisy Text Recognition

Han Yang (12531717)
Institute of Informatics, LMU Munich
ha.yang@campus.lmu.de

Yian Yu (12676653)
Institute of Informatics, LMU Munich
yian.yu@campus.lmu.de

Abstract

For many natural language processing (NLP) tasks, noise in the text of sequence can bring an out-of-vocabulary (OOV) problem, which will probably lead to poor performance in downstream tasks. Inspired by human vision robustness, the image-based method could be vital to text recognition, particularly when dealing with noisy text content. In this paper, our proposed CNN model-based approach can effectively bolster the robustness of text recognition by leveraging the principles of visual processing at the word level. We also conducted experimentation and rigorous evaluation to assess the model's robustness in terms of the three dimensions that affect text recognition: fonts, noise types, and the proportions of added noise.

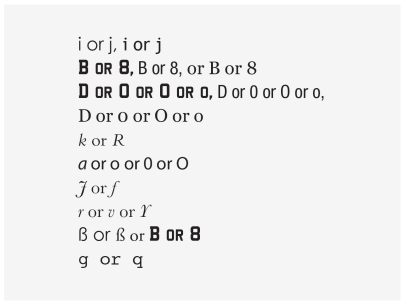
1. Introduction

Convolutional Neural Networks (CNNs) [5] are one of the most successful areas of application of deep learning algorithms, where 2D CNN are often applied to image-based text recognition.

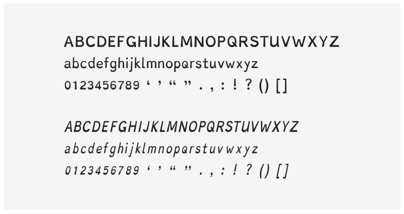
It's an interesting phenomenon that for words that are misspelled or blurry to see, people can often intuitively tell which word it is. For example, the pair of words "`apple`" and "α`pple`", where the letter a is replaced by the Greek letter alpha, or the pair of words "`Apple`" and "`4pple`", where the letter `A` is replaced by the number `4`, are very easy for human beings to notice, be distinguished and be identified as the same word. Such letter substitutions are very common on the web community like Reddit[1]. However, for text-based natural language processing tasks, e.g. machine translation or text representation techniques, the substitution brings additional difficulty for machines to identify such word pairs as the corresponding word recorded in the tokenizer, and it will even cause the out-of-vocabulary (OOV) problem because the substituted words like "`4pple`" do not exist in the dictionary at all.

In addition to letters that can be easily confused with

[1] https://www.reddit.com/



(a) Different Typefaces [1]


ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789 ' ' " " . , : ! ? ( ) [ ]
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789 ' ' " " . , : ! ? ( ) [ ]

(b) Christian Boer, typeface: Dyslexie Regular and Italic. [1]

Figure 1. Recognition of letters can be influenced by fonts

other characters, sometimes fonts can interfere with recognition too. Some variations of the letters are shown below in figure 1.

This leads to the superiority of the vision-based approach as the key algorithm in dealing with such problems. Aiming to address the OOV issue of the noised texts of sequence, we designed a pipeline incorporating the advantage of CNN-based image recognition. Our experiments, conducted with 3 fonts, 4 noises, and 5 noise levels, demonstrated a favorable enhancement of overall robustness. Moreover, the result of this project can also be used to refine the embedding generated by the vision-based pipeline, and finally improve the downstream tasks.

2. Related Work

Optical character recognition (OCR) domain is one of the most successful applications of CNNs, and its effectiveness has been demonstrated by Jaderberg, Simonyan et al. in their thesis *Reading text in the wild with CNN* [4] back in 2016. As their algorithms are mainly divided into two stages: scene text detection based on computer vision and machine learning and text recognition based on deep learning. Their model is based on a word-based classification approach. To solve the problem of undefined words, they sampled 90k most common words, and the problem of insufficient samples was solved by synthesizing the data. The end-to-end training model CRNN [8] proposed by B. Shi et al. in the same year can train CNN and RNN jointly. The image features are extracted using standard CNN and represented as feature vectors using Map-to-Sequence, and the feature vectors are identified using bi-directional Long short-term memory (LSTM) [3] in the Recurrent Neural Network (RNN) [6] layer to get the probability distribution of each column of the features. Finally the optimal label sequences are solved in the Transcription layer by using the CTC and forward-backward algorithms. This model offers several noteworthy advantages, including its ability to accommodate inputs of arbitrary lengths, obviating the necessity for character calibration within the training set, and its versatility in utilizing samples both with and without dictionaries.

Visual text representations prove to be highly advantageous within the domain of natural language processing. Salesky et al. [7] proposed an approach using visual text representations to create continuous vocabularies by processing rendered text images instead of using limited text embedding. This approach demonstrates the potential for enhanced robustness and generalizability, especially in the context of machine translation, which is commonly characterized by challenges such as noise (e.g., misspellings, variant characters, encoding errors, etc.). They conducted experiments on multiple language pairs and datasets, showing the advantages and scalability of visual text representation compared to models based on subword segmentation.

3. Method

3.1. Conceptual Design

We have designed an automatic pipeline for the experimental process as shown in Figure 6. Generally speaking, each word, no matter a word presented in the corpora, or a noised word, will be first transferred into an image, which will be classified as a token of a "clean" word.

First we counted the frequency of each word in the original English dataset and filtered out the 4571 English words and created {`word : id`} and {`id : word`} dictionaries respectively, which map the words to token ID and reversely. The word in the form of text will be rendered as an image by utilizing PYGAME Python package[2], which will be used as the input of a CNN-based classifier later. The token ID will be then converted to a one-hot vector, which is used as the target in training the classifier or evaluating the performance.

Subsequent to the establishment of the dataset and dataloader, we proceed with the model training phase, concurrently fine-tuning the aforementioned variables along with the remaining hyperparameters to attain optimal outcomes.

Subsequently, a comprehensive evaluation is conducted, encompassing a comparison of accuracy outcomes under consistent font conditions or within identical noise categories. This analytical approach enables us to discern the influence of various variables on the model's performance in the realm of text recognition.

3.2. CNN-based Model

Our approach employs CNN to decipher the localized characteristics within images. This strategy offers the advantage of significantly diminishing the number of parameters, leading to notable reductions in learning time, as well as a decreased requirement for voluminous training data.

[2]https://www.pygame.org/news.

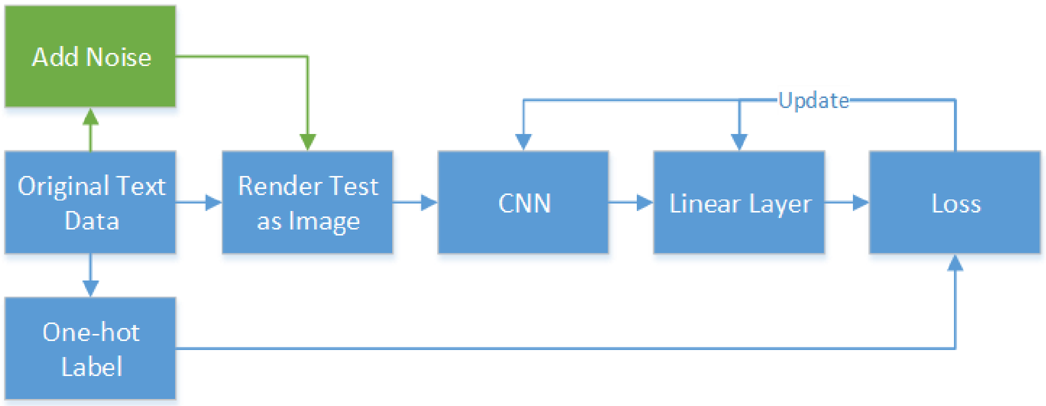


Figure 2. Pipeline Process

Set	Number of Sentences	Number of Tokens
Training set (/en-de//tok/ted_train_en-de.tok.clean.en)	151,627	$3,037,569$
Developing set (/en-de/tok/ted_dev_en-de.tok.en)	1,958	38,438
Testing Set (/en-de/tokted_test1_en-de.tok.en)	1,982	36,499

Table 1. Table 1: Statistical information of the English part in en-de set in the MTTT dataset

The input images are uniformly sized at 25×130 pixels. In the context of the CNN architecture, we incorporate sixteen 3×3 filters, which traverse the image in a block-by-block manner with a stride of 1, effectively covering the entire image span during the scanning process. Following the CNN, a linear layer with softmax performs as a classifier, predicting the probability of the input image being classified as each token.

We trained 10 epochs, and during each epoch, the neural network processes batches of training data, makes predictions, calculates the loss between these predictions and the actual target values, and then performs backpropagation to update the network's weights and biases in order to minimize this loss.

3.3. Loss Function

The loss function in a CNN serves as a measure of the disparity between the predicted outputs of the network and the actual ground truth values. One commonly used loss function in image classification tasks is the Cross-Entropy Loss (also known as Logarithmic Loss or Softmax Loss). It computes the discrepancy between the predicted class probabilities and the true class labels. Mathematically, for a single example, the Cross-Entropy Loss is given by:

$$L(y, \hat{y}) = -\sum_i y_i \log\left(\hat{y}_i\right)$$

where:
- L represents the loss,
- y_i is the true probability of class i,
- $\hat{y}_i$ is the predicted probability of class i.

After completing an epoch, the average loss for that epoch is calculated by summing up all the individual losses and then dividing by the total number of batches or samples processed in that epoch.

4. Experiments

4.1. Dataset

We employ the Multitarget TED Talks Task (MTTT) dataset [2], as originally introduced in the referenced paper [7]. This dataset encompasses a collection of 30 TED Talks, currently serving as a congenial platform for a machine translation competition. The entirety of the data is initially presented in spoken form, transcribed into English, and subsequently translated by the TED translators. The dataset comprises a diverse array of 20 languages, yielding a total of 20 parallel translation pairs. Each language within this dataset is accompanied by distinct subsets, including training, development, testing, and metadata sets.

For our specific text recognition task, we focus on the English portion of the en-de (English-German) translation set, designated as the text-based data. Although the primary purpose of this dataset is for machine translation, the English texts within it are notably clean and plentiful, making them suitable for our task. In table 1, we present the relevant statistical characteristics of this chosen subset.

Utilizing this dataset, our methodology involves the intentional introduction of noise, particularly at the word-level granularity, into the textual content. The resulting noisy text is then transformed into image representations, serving as the input for a CNN. The original text (token) before being polluted by the noise will be used as the golden label.

4.2. Variables and Experimental Condition

For the independent variables, we first classified the type of noise into three categories: Greek letters[3], Cyrillic letters[4], and leetspeak[5], which is a system of modified spellings used primarily on the Internet. The correspondence between the original clean letter and the noised letter

[3]https://web.mit.edu/jmorzins/www/greek-alphabet.html
[4]https://www.russlandjournal.de/en/learn-russian/russian-alphabet/
[5]https://github.com/floft/leetspeak/blob/master/LeetSpeak.py

Example words	10%	20%	30%	40%	50%
United	Un!ted	Un!t3ol	Un!t3ol	Un!t3ol	Un!t3ol
illusion	illu5!on	i1lu5!on	i1lu5!on	i1lu5!on	i1lu5!0n
Friday	Frida¥	Frida¥	Fr!d@¥	l=r!d@¥	l=r!ol@¥

Table 2. Example of replacing each character with different probabilities.

is shown in figure 4. The letter will be mapped to the most similar character during the noising process. The noises in the Greek and Cyrillic alphabet are one Greek or Cyrillic character visually similar to the original letter, while the leetspeak noise might be a longer sequence, e.g. letter "v" will be replaced by "\/". Since several letters are not able to correspond with visually similar noise, they remain the same during the noise process.

Then, we set five levels of noise, i.e., we replaced each character with a probability of 10%, 20%, 30%, 40%, and 50%, respectively. The noise was applied to 4 fonts: Noto Sans, Mandatory, Turok and Typographer. The example of the fonts is shown in figure 3. We present an example of words in various noise levels in table 2. In the example, the word `United` is noised as `Un!ted` with the leetspeak noise under the noise level 10%, where the letter `i` is replaced by "`!`".

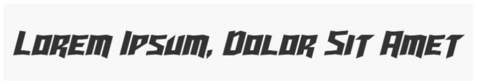
Therefore, for each font, we performed 5×3, i.e., 15 experiments with noises, and 1 experiment session without any noise. Thus there are a total of $(5 \times 3 + 1) \times 4$, i.e., 64 operations.

LOREM IPSUM, DOLOR SIT AMET

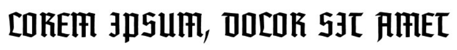
(a) Font: Noto Sans[6]

LOREM IPSUM, DOLOR SIT AMET

(b) Font: Mandatory[7]



(c) Font: Turok[8]



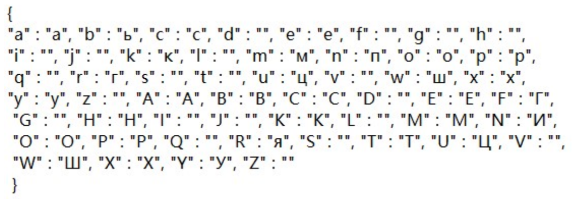
(d) Font: Typographer[9]

Figure 3. Four Fonts

5. Result

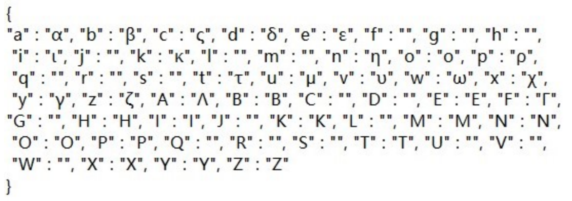
In this section, we will report our results.

[6]https://www.1001fonts.com/noto-sans-font.html
[7]https://www.fontrepo.com/fonts/mandatory/
[8]https://www.fontrepo.com/fonts/turok/
[9]https://www.1001fonts.com/typographertextur-font.html



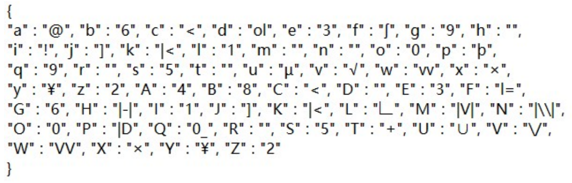
```
{
"a" : "a", "b" : "ь", "c" : "c", "d" : "", "e" : "e", "f" : "", "g" : "", "h" : "",
"i" : "", "j" : "", "k" : "к", "l" : "", "m" : "м", "n" : "п", "o" : "o", "p" : "p",
"q" : "", "r" : "г", "s" : "", "t" : "", "u" : "ц", "v" : "", "w" : "ш", "x" : "x",
"y" : "y", "z" : "", "A" : "A", "B" : "B", "C" : "C", "D" : "", "E" : "E", "F" : "Г",
 "G" : "", "H" : "H", "I" : "", "J" : "", "K" : "K", "L" : "", "M" : "M", "N" : "И",
 "O" : "O", "P" : "P", "Q" : "", "R" : "я", "S" : "", "T" : "T", "U" : "Ц", "V" : "",
 "W" : "Ш", "X" : "X", "Y" : "У", "Z" : ""
 }
```

(a) Alphabet to Cyrillic Dictionary Cross Reference



```
{
"a" : "α", "b" : "β", "c" : "ς", "d" : "δ", "e" : "ε", "f" : "", "g" : "", "h" : "",
 "i" : "ι", "j" : "", "k" : "κ", "l" : "", "m" : "", "n" : "η", "o" : "o", "p" : "ρ",
 "q" : "", "r" : "", "s" : "", "t" : "τ", "u" : "μ", "v" : "υ", "w" : "ω", "x" : "χ",
 "y" : "γ", "z" : "ζ", "A" : "Λ", "B" : "B", "C" : "", "D" : "", "E" : "E", "F" : "Γ",
"G" : "", "H" : "H", "I" : "I", "J" : "", "K" : "K", "L" : "", "M" : "M", "N" : "N",
 "O" : "O", "P" : "P", "Q" : "", "R" : "", "S" : "", "T" : "T", "U" : "", "V" : "",
 "W" : "", "X" : "X", "Y" : "Y", "Z" : "Z"
}
```

(b) Alphabet to Greek Dictionary Cross Reference



```
{
"a" : "@", "b" : "6", "c" : "<", "d" : "ol", "e" : "3", "f" : "ʃ", "g" : "9", "h" : "",
"i" : "!", "j" : "]", "k" : "|<", "l" : "1", "m" : "", "n" : "", "o" : "0", "p" : "þ",
"q" : "9", "r" : "", "s" : "5", "t" : "", "u" : "μ", "v" : "√", "w" : "vv", "x" : "×",
"y" : "¥", "z" : "2", "A" : "4", "B" : "8", "C" : "<", "D" : "", "E" : "3", "F" : "l=",
"G" : "6", "H" : "|-|", "I" : "1", "J" : "]", "K" : "|<", "L" : "|_", "M" : "|V|", "N" : "|\\|",
"O" : "0", "P" : "|D", "Q" : "0_", "R" : "", "S" : "5", "T" : "+", "U" : "∪", "V" : "\/",
"W" : "VV", "X" : "×", "Y" : "¥", "Z" : "2"
}
```

(c) Alphabet to Leetspeak Dictionary Cross Reference

Figure 4. Three Noise Types

5.1. Evaluation

Initially, our objective was to undertake a macroscopic analysis aimed at visualizing the impact of fonts and noise ratios on the model's robustness within the context of text recognition.

As can be seen on the graph, fonts do make discernible variations to the accuracy with which the model performs text recognition. Furthermore, even when the model is trained using a higher percentage of noise added, the recognition success rate is lower. This observation aligns with our intuition that words become illegible even for human observers when each character has a 50% probability of being replaced.

In order to more intuitively see the effect of adding different levels of noise on the accuracy of the CNN model for text recognition, we presented on each font separately with fixed noise type. The model's recognition accuracy demonstrates a near-perfect performance of approximately 100% in the absence of noise. However, an evident trend emerges

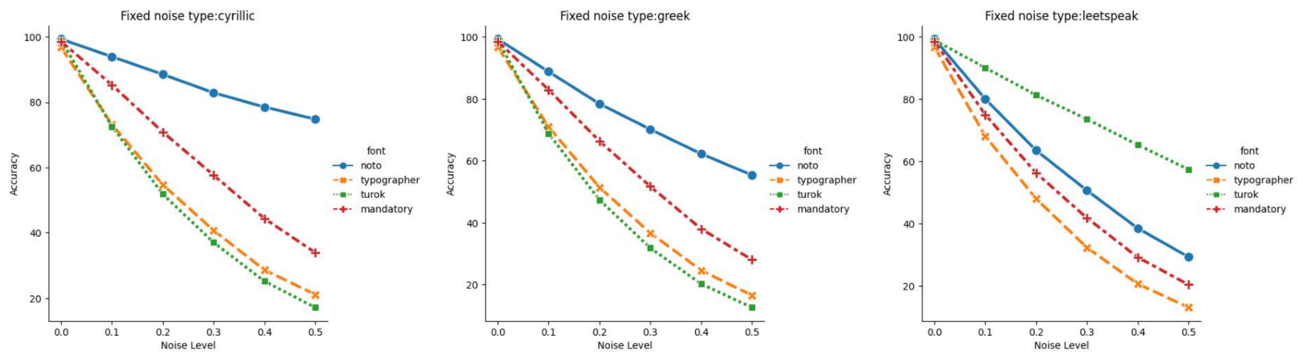


Figure 5. Text Recognition Accuracy when Noise Type is Fixed.

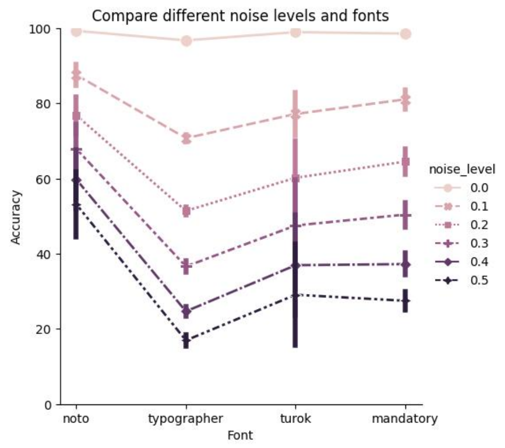


Figure 6. Overview of the Effect of Fonts and Noise Levels on Accuracy.

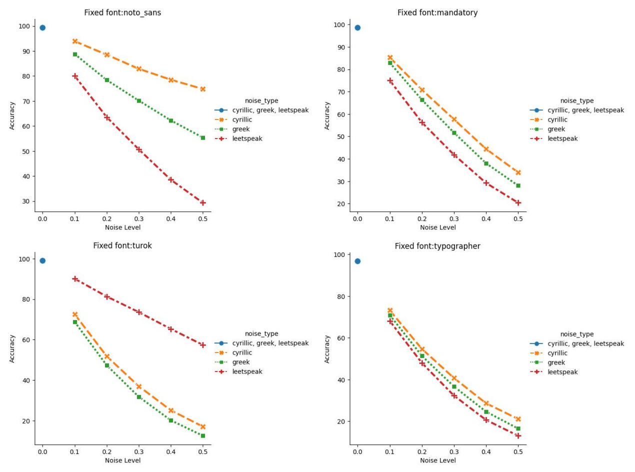


Figure 7. Text Recognition Accuracy when Fonts are Fixed.

as the proportion of noise within the word escalates, leading to a consistent decline in accuracy regardless of font.

We fixed the noise type in our subsequent experiment, which investigates how the fonts and noise ratio affect the model's robustness. As demonstrated by Cyrillic and Greek, the typeface Noto Sans produces the best model classification accuracy for general forms of noise. We also find it interesting that the Turok typeface seems to perform better when Leetspeak noise is present, providing the model with the maximum accuracy at all noise levels. This seems to suggest that fonts may be compatible with particular types of noise.

An intriguing observation emerges regarding the influence of noise types on the model's recognition capability. Specifically, the replacement of letters with distinct character sets such as Greek letters, Cyrillic letters, or leetspeak engenders divergent outcomes. Notably, cyrillic letters exhibit a relatively minimal impact on accurate recognition when subjected to misspelling, followed by Greek letters, whereas leetspeak poses the greatest challenge to successful recognition.

5.2. Statistical Analysis

We used ANOVA to analyze whether there were significant differences between all the data. The left side of the table below shows three sets of independent variables, C (font), C (noise type) and C (noise level). The right column shows the corresponding p-values. If the p-value is less than 0.05 (95%), the general hypothesis is rejected, which means that the variable has an effect on the results.

Consequently, it becomes evident that the three distinct variables, namely "Font," "Noise Type," and "Noise Level," underscore the statistical significance, as substantiated by p-values falling below the threshold of 0.05, which is also in line with the line graphs we obtained in the evaluation.

	df	sum_sq	mean_sq	F	PR(F)
C(font)	3.0	6159.253119	2053.084373	13.216646	1.472186e − 06
C(noise_type)	3.0	8230.472624	2743.490875	17.661109	4.485511e − 08
C(noise_level)	5.0	17914.068404	3582.813681	23.064215	3.208838e − 12
Residual	53.0	8233.062735	155.340806	NaN	NaN

Table 3. Analysis of Variance (ANOVA) for Variables.

Example word	Prediction word	Evaluation	Added noise	Prediction with noise	Evaluation
`attract`	`attract`	true	`attra<t`	`attract`	true
`abstract`	`abstract`	true	`@b5tract`	`celebrate`	false
`previous`	`previous`	true	`prev!ou5`	`previous`	true
`obvious`	`obvious`	true	`o6v!0usly`	`carpeting`	false
`College`	`College`	true	`<0lle93`	`ended`	false
`colleagues`	`colleagues`	true	`<olle@gu3s`	`imaginative`	false

Table 4. Negative Examples for Case Study.

In essence, each of these variables bears a determinative influence on the model's results for text recognition.

5.3. Case Study

In the task of text recognition based on vision methods, we focus on whether words with similar characters can be successfully distinguished. To elucidate this principle, we have deliberately selected some negative examples for certain words for the case study.

Consequently, our observations indicate that the visually-based model achieves a remarkably high level of recognition accuracy, consistently hovering around 100% for the unaltered original words. This noteworthy performance implies the model's proficiency in distinguishing even minutely similar words, including cases of differing grammatical forms (e.g., singular-plural variations). However, as the analysis extends to words featuring substituted characters, the visual model experiences a pronounced decline in its ability to accurately differentiate such instances.

6. Limitations

Our proposed method and experiments showed an improvement in the robustness of identifying an input word and the influence of the factors like fonts. In the experiments, we evaluated the intrinsic performance, i.e. accuracy of the classification. Since the NLP scenario suffers often the OOV problem, it is also exciting to compare the extrinsic performance of the downstream tasks, e.g. machine translation or other natural language generation tasks.

7. Conclusion and Future Work

In this project, we aimed at the OOV problem in the NLP area and proposed a pipeline to improve the robustness of identifying a text of sequence by utilizing the image-based CNN method at the word level. Our experiments showed the robustness of our models against different kinds and levels of noise. We also conducted a case study showing how font influences the robustness of our pipeline. As a suggestion for future work, applying this method to various granularity and downstream tasks is also worth expecting.

Contribution

We hereby declare that all work on this project was done independently by us, and all help documents are listed in the reference or footnotes. Concretely, **Yian Yu** is responsible for background research on letter or word misrecognition, preparing font banks for experiments, the implementation of image rendering, and analyzing and evaluating the results, while the implementation of the pipeline, model, dataset & dataloader, and the training are done by **Han Yang**.

Our code and text-data are available: https://github.com/ANNAISDEVIL/cvdl_ss23/tree/release. We uploaded our models and slide here: https://drive.google.com/drive/folders/1xBEqIwnwxeQifTPyx4qfNNKoSe4PJdJ9?usp=sharing. Our video could be watched at: https://drive.google.com/file/d/1-JUROBlSOLcN5M21Ci_TEvUzzm_RNmI0/view?usp=drive_link

Additionally, we are deeply **grateful** to the Computer Vision & Learning Group[10] for providing this lecture that focuses on both theory and practice!

[10] https://ommer-lab.com/

References

[1] Thomas Bohm. Letter and symbol misrecognition in highly legible typefaces for general, children, dyslexic, visually impaired and ageing readers. *Information Design Journal*, 21(1):34–50, 2014. 1

[2] Kevin Duh. The multitarget ted talks task. http://www.cs.jhu.edu/~kevinduh/a/multitarget-tedtalks/, 2018. 3

[3] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 2

[4] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Reading text in the wild with convolutional neural networks. volume 116, pages 1–20. Springer, 2016. 1

[5] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015. 1

[6] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986. 2

[7] Elizabeth Salesky, David Etter, and Matt Post. Robust open-vocabulary translation from visual text representations. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7235–7252, Online and Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics. 2, 3

[8] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(11):2298–2304, 2016. 2