ImageTra: Real-Time Translation for Texts in Image and Video

Hour Kaing[1] **Jiannan Mao**[2] **Haiyue Song**[1]
Chenchen Ding[1] **Hideki Tanaka**[1] **Masao Utiyama**[1]
[1]ASTREC, UCRI, NICT, Japan [2]Gifu University, Gifu, Japan
[1]{hour_kaing, haiyue.song, chenchen.ding,
hideki.tanaka, mutiyama}@nict.go.jp
[2]mao@mat.info.gifu-u.ac.jp

Abstract

There has been a growing research interest in in-image machine translation, which involves translating texts in images from one language to another. Recent studies continue to explore pipeline-based systems due to its straightforward construction and the consistent improvement of its underlined components. However, the existing implementation for such pipeline often lack extensibility, composability, and support for real-time translation. Therefore, this work introduces *ImageTra*—an open-source toolkit designed to facilitate the development of the pipeline-based system of in-image machine translation. The toolkit integrates state-of-the-art open-source models and tools, and is designed with a focus on modularity and efficiency, making it particularly well-suited for real-time translation. The toolkit is released at https://github.com/hour/imagetra.

1 Introduction

In-image machine translation (IIMT) refers to the task of translating texts in an image from one language to another, and generating a new image that embeds the translations (Mansimov et al., 2020; Tian et al., 2023, 2025). Ultimately, the background and text style of the translations inherit the characteristics of the original texts, as illustrated in Figure 1. Such systems have a significant value for research and a wide range of applications, including platform-independent automatic subtitle translation, manga translation, and real-time translation from camera input. Although commercial products like Google Translate offer real-time translation features, their underlying technical solutions are not transparent and difficult to customize for other purposes or downstream applications.

Recent studies continue to explore pipeline-based systems due to their straightforward construction (Qian et al., 2024; Vaidya et al., 2025; Kaing et al., 2025). These systems typically consist of three main components: optical character recognition (OCR), machine translation (MT), and scene text editing (STE). The pipeline-based systems remain the state-of-the-art compared to end-to-end solutions (Salesky et al., 2024), primarily because of the task difficulty and data bottlenecks where the models training rely heavily on synthetic data in end-to-end approaches (Li et al., 2025). In contrast, the data available for each component of the pipeline-based systems is richer and multilingual (Long et al., 2022; Bañón et al., 2020), making it easier to enhance individual components and, in turn, improve the overall performance of the pipeline-based systems.

Figure 1: A translation example using ImageTra.

Although open-source tools are available for each component thanks to active research communities, they were developed and improved independently, without consideration for compatibility with other components when building a pipeline-based system. While assembling these tools into an IIMT pipeline may seem straightforward, researchers and practitioners still need to invest a significant amount of effort in implementation. Although several studies have released code to reproduce their pipelines (Qian et al., 2024; Vaidya et al., 2025), existing implementations often lack extensibility and composability, and they do not support real-time translation.

This work aims to reduce that burden by introducing `ImageTra`—an open-source toolkit designed to facilitate the development of IIMT systems. For composibility, our toolkit enable researchers and practitioners to plug and play various state-of-the-art open-source models and tools to

Proceedings of The 14th International Joint Conference on Natural Language Processing and The 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics: System Demonstrations, pages 1–8

build IIMT systems capable of translating text in images, videos, or a live camera video (real-time translation)[1]. For extensibility, the toolkit is feasible to customize each component to support other tools or models, which is beneficial for both academic and industrial research. Both composibility and extensibility of our toolkit are demonstrated in Section 3.3. Additionally, for real-time translation, our toolkit has features to enhance efficiency such as translation caching, text tracking, and API hosting. For accessibility, our toolkit is implemented in Python and is pip-installable, and is released under the MIT license, permitting unrestricted use, modification, and redistribution.

For evaluation, we assess the efficiency and quality of various pipeline configurations, with particular focus on OCR, and analyze the impact of the translation caching and text tracking features on efficiency in Section 4. The experimental results demonstrate that both features significantly reduce the pipeline's inference time.

2 Related Works

In-Image Machine Translation. This task is recently getting many attentions due to the advancement of machine learning techniques. Many works have explored the end-to-end approaches (Mansimov et al., 2020; Salesky et al., 2021; Ma et al., 2023; Lan et al., 2024; Tian et al., 2023, 2025), which are not generalized yet particularly on the scene text scenario. Meanwhile, the pipeline-based systems have been explored as well by integrating OCR, MT, and STE systems into a pipeline (Qian et al., 2024; Vaidya et al., 2025; Kaing et al., 2025). The implementation of these works are mostly opened mainly to reproduce their results, which are not easy to be adapted especially to support real-time translation.

Optical Character Recognition. This task is a long-standing task in computer vision, and modern OCR systems face increasingly complex challenges, such as handling document-level and scene text images. These tasks are typically divided into two subtasks: text detection (Zhou et al., 2017; Baek et al., 2019; Liao et al., 2020) and text recognition (Shi et al., 2016; Bautista and Atienza, 2022; Du et al., 2025). A wide range of open-source OCR tools are available, many of which continue to improve in terms of accuracy, efficiency, and language coverage. For example, but not limited to, DocTR supports multiple model architectures, allowing users to choose specific models for detection and recognition. Similarly, OpenOCR supports several detection and recognition models including its own state-of-the-art models (Du et al., 2025). On the other hand, tools like EasyOCR and PaddleOCR emphasize support for diverse languages, with PaddleOCR also being recognized for its computational efficiency (Cui et al., 2025).

Machine Translation. Text-based machine translation provides an efficient way to transform information from one language into another (Bahdanau et al., 2014; Vaswani et al., 2017). Many effective techniques have been introduced for improving translation quality of low-resource languages such as data augmentation (Sennrich et al., 2016; Kaing et al., 2024), multilingual training (Dabre et al., 2020; Costa-Jussà et al., 2022), and multimodal training (Elliott et al., 2016; Hirasawa et al., 2020; Gu et al., 2021), among others. Multilingual models offer other advantages especially their efficiency, as they can handle diverse languages within a single model. Therefore, our toolkit begins with support for the NLLB200 model family (Costa-Jussà et al., 2022), which is considered the state-of-the-art in multilingual machine translation.

Scene Text Editing. This task involves modifying texts within natural scene images while preserving the visual consistency and contextual integrity of surrounding elements. SRNet was the first model introduced for this purpose, explicitly separating foreground and background components (Wu et al., 2019). Subsequent works extended this approach to better handle irregular or curved text (Yang et al., 2020), enable character-level editing (Roy et al., 2020; Qu et al., 2023), and incorporate diffusion-based frameworks (Zeng et al., 2024; Fang et al., 2025). Subramanian et al. (2021) extended SRNet to edit videos and incorporated additional techniques to make the edited text in videos smoother. It is worth noting that these studies primarily focus on model architecture and are restricted to editing English text. Besides the fact that SRNet remains impressive and has consistently been used as a baseline, it has recently been adapted to perform cross-lingual editing, particularly within the IIMT pipeline, including English↔Hindi (Vaidya et al., 2025) and English→Japanese (Kaing et al., 2025). Hence, our toolkit begins with SRNet support in this version.

[1]A video is technically a set of images and the IIMT task can be generalized to a video as well as a live camera video.

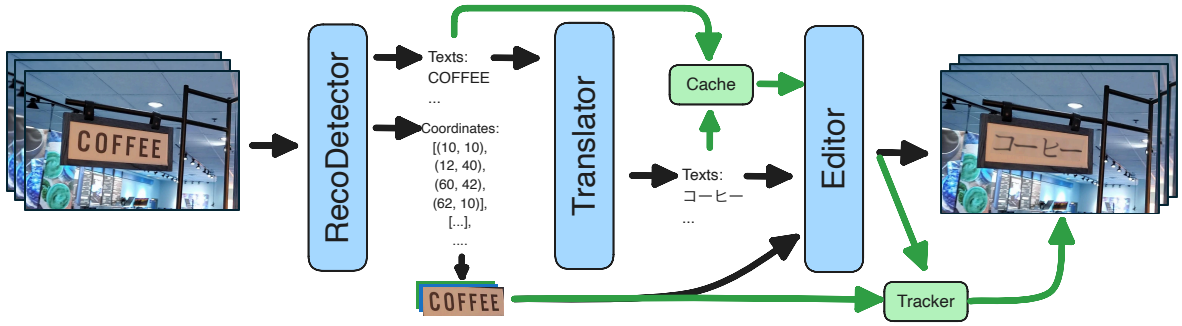

Figure 2: Overview of a pipline in ImageTra. Green indicate proposed features to enhance real-time translation.

3 The Toolkit: ImageTra

ImageTra is a playground where researchers and practitioners can quickly build an IIMT pipeline using the existing tools and models. The pipeline in ImageTra is composed of three main components: `RecoDetector`, `Translator`, and `Editor`. The composed pipeline can then be applied to an image, a video, or a live stream video. Additional features include translation caching, text tracking, and API hosting for server-client use cases, as illustrated in Figure 2.

3.1 Main Components

RecoDetector. This component takes an image as input, detects text regions, recognizes the texts inside the regions, and returns the coordinates of the regions and the recognized texts. Currently, our toolkit wraps four popular open-source OCR tools under this component: DocTR[2], OpenOCR[3], EasyOCR[4], and PaddleOCR[5].

Translator. This component takes recognized texts from `RecoDetector` and translates them into a specific language. We wrap the family of NLLB models (Costa-Jussà et al., 2022) and the TexTra API service[6] under this component.

Editor. This component aims to convert the translated texts from `Translator` into pieces of images and embed them into the whole scene image. The component takes several inputs, including the coordinates of detected texts and the translations. Two types of editors can be created in this component. The first one is a renderer that outputs a rendered translation. To fit the translation into the region of its original text, the font size is estimated based on the character length of the translations and the width and height of the region. It's worth noting that the lengths of the translation and its original text are often different, which may not fit the translation nicely. The second type of editor returns a fused translation that shares the same background and text style as its original text. For the second type of editor, we wrap the conventional SRNet model (Wu et al., 2019), which also leverages the rendered translation as part of its generation.

3.2 Real-Time Video Translation

To improve the efficiency of real-time translation, we introduce three additional features: translation caching, text tracking, and an API-based service.

Translation Caching. When translating texts in video, texts repetition across video frames is inevitable and using a translator to translate them all the time is not efficient especially when the translator model is large and the inference speed is slower. This could make the pipeline slow and less practical for real-time translation. We address this by introducing a simple solution by caching the already translated sentence using a dictionary that maps a source text with a target text.

Text Tracking. Translation caching eliminates redundant translator calls by reusing previously processed texts. To further improve efficiency, we apply text tracking to match detected regions across consecutive frames, allowing translated text from the prior frame to be directly reused in the current frame. This reduces computation for both the translator and the editor, thereby accelerating the pipeline. Specifically, the tracker identifies text regions matched between the previous and current frames. If a matched region in the previous frame has a score greater than or equal to that of its counterpart in the current frame, the region in the current frame is directly replaced with the translated text

[2] https://github.com/mindee/doctr
[3] https://github.com/Topdu/OpenOCR
[4] https://github.com/JaidedAI/EasyOCR
[5] https://github.com/PaddlePaddle/PaddleOCR
[6] https://mt-auto-minhon-mlt.ucri.jgn-x.jp

image from the previous frame. For tracking, we leverage the existing multi-object tracking library—boxmot[7], which supports a variety of algorithms, including botsort (Aharon et al., 2022), strongsort (Du et al., 2023), deepocsort (Maggiolino et al., 2023), bytetrack (Zhang et al., 2022), and ocsort (Cao et al., 2023).

API. The pipeline can also be served as an API, which allows us to run the pipeline on a server and call the pipeline from a local machine. This will enable broader application especially for real-time translation scenario where a local machine has a camera but no GPU.

3.3 Usage Examples

Here we present a basic example for building an IIMT pipeline and how to use it to translate texts in an image and a video with a few lines of code. Specifically, we first create the three components and then integrate them into the pipeline named `Image2Image`. Since this pipeline can take multiple images as input, we can use it to translate a video frame-by-frame. Lastly, the translated image and video are saved simply using a `save` function.

```python
from imagetra.detector.paddleocr import PaddleOCRRecoDetector
from imagetra.translator.nllb import NLLBTranslator
from imagetra.editor.render import RenderEditor
from imagetra.pipeline.img2img import Image2Image

recodetector = PaddleOCRRecoDetector(lang='en')
translator = NLLBTranslator(
    'facebook/nllb-200-distilled-600M',
    trg_lang='jpn_Jpan',
    cache=True, # translation cache
)
editor = RenderEditor('<font_path>')

pipeline = Image2Image(
    recodetector=recodetector,
    editor=editor,
    translator=translator,
)

from imagetra.common.media import Image
img = Image.load('image.jpeg')
result = pipeline([img])[0]
result.img.save('result.jpeg')

from imagetra.common.media import Video
video = Video.load('video.mp4')
results = pipeline(video.frames)
for i, result in enumerate(results):
    video.replace(result.img, i)
video.save('result.mp4')
```

You may have noticed that the translation caching is activated when the `Translator` component is created with `cache=True`. We can further speed up the pipeline on the video translation using a tracker. To do that, we just need to replace `Image2Image` with `Video2Vdieo` and specify the tracker type during inference, e.g, `bytetrack`. The rest of the codes are the same.

[7] https://github.com/mikel-brostrom/boxmot

```python
pipeline = Video2Video(...)
results = pipeline(video.frames, tracker_type='bytetrack')
```

As there are plenty of other tools and models that are not directly supported yet, we can also integrate them into the pipeline by simply inheriting the base class and overriding the main function of each component as follows.

```python
from imagetra.detector.base import BaseRecoDetector
from imagetra.translator.base import BaseTranslator
from imagetra.editor.base import BaseEditor

class CustomRecoDetector(BaseRecoDetector):
    def recodetect(self, imgs):
        # code here
        return bboxs, det_scores, texts, reco_scores

class CustomTranslator(BaseTranslator):
    def translate(self, texts, src_lang, trg_lang):
        # code here
        return translations

class CustomEditor(BaseEditor):
    def edit(self, texts, imgs):
        # code here
        return edited_imgs
```

The above example are explained in python code to show how a pipeline can be constructed and customized. Beside this, we can quickly run the pipeline using a command line interface, of which details can be found in our project homepage.

4 Evaluation

We assume that the overall quality of the pipeline primarily depends on the intrinsic quality of its underlying components, which we expect will continue to be actively improved and domain-generalized by their respective communities. Therefore, our evaluation focuses on the pipeline's efficiency and its trade-off with quality, particularly in the context of real-time translation. We analyze the impact of both translation caching and text tracking features, while also examining optimal configurations, with particular attention to the OCR component. To simulate a realistic real-time translation scenario, our evaluation is conducted on the DSText video dataset from ICDAR2023 (Wu et al., 2023), which provides ground-truth coordinates of English text and their transcriptions. Furthermore, the pipeline is constrained to translating only one frame at a time.

4.1 Impact of Translation Caching

Figure 3 compares the latency of the NLLB model family with 600M, 1.3B, and 3.3B parameters. The models translate the DSText dataset transcriptions into German, one frame at a time. Latency is measured as the average time per frame (in seconds)

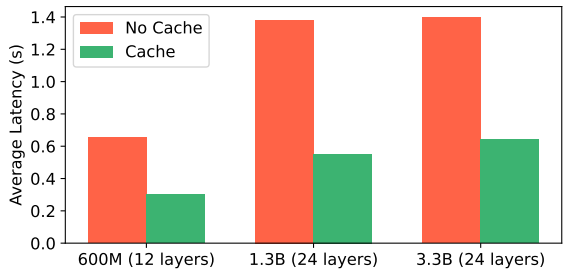

Figure 3: Translation latency of NLLB models across different sizes, with and without translation cache. The y-axis is the latency per frame in seconds.

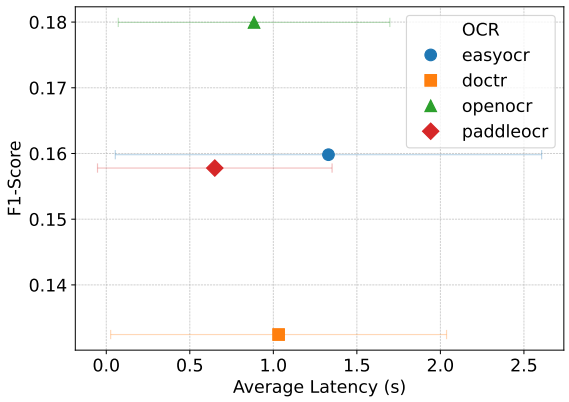

Figure 4: F1-score and average latency of the pipeline with different OCRs.

for each MT model. The results show that larger models tend to have slower inference speeds, as expected, and that model depth significantly contributes to higher latency. Our translation caching substantially improves inference speed, achieving a twofold increase in this experiment.

4.2 Performance of OCRs

Figure 4 compares four OCR tools in terms of latency and accuracy. For latency, we measure the inference time of the pipeline using different OCRs, while keeping the MT model (NLLB-600M) and the editor (RenderEditor) fixed. For accuracy, we compute F1-scores on the OCR outputs following the evaluation protocol of the end-to-end text detection and recognition task in ICDAR2019 (Nayef et al., 2019). Specifically, we first observed that the DSText dataset contains labels marked as "##DONT#CARE##", which are typically used when text in the image is unreadable by annotators due to low resolution or other distortions. Following the ICDAR2019 protocol, both ground-truth regions labeled as "##DONT#CARE##" and predicted regions overlapping with them are excluded

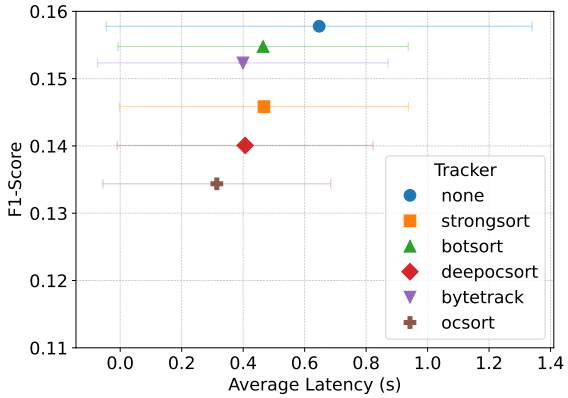

Figure 5: F1-score and average latency of the pipeline using different trackers.

from evaluation (Nayef et al., 2019). After this filtering, true positives are defined as the number of matched texts between the ground truth and predictions, denoted $|M|$. Two texts are considered matched if (i) the intersection-over-union (IoU) of their regions exceeds 0.5, and (ii) their surface forms are exactly identical. Precision P and recall R are then computed as $\frac{|M|}{|T|}$ and $\frac{|M|}{|G|}$, respectively, where $|T|$ is the number of predicted texts and $|G|$ is the number of ground-truth texts. The F1-score is calculated as $\frac{2 \cdot P \cdot R}{P+R}$.

The results show that OpenOCR achieve the best accuracy while PaddleOCR has the best inference speed. This findings is equivalent what is claimed by these tools, for instance, in each of their home page, the OpenOCR teams claims that their tool outperform PaddleOCR and the PaddleOCR teams present their tool as a lightweight OCR system. On another hand, EasyOCR maintains similar performance with PaddleOCR but has the worst inference time. The performance of DocTR is the worst but maintains similar inference speed with OpenOCR. To this end, this result suggest OpenOCR for accuracy and PaddleOCR for inference speed.

4.3 Impact of Text Tracking

Since PaddleOCR is the most efficient system among the four OCRs, we use it to establish a baseline performance and evaluate the impact of trackers on pipeline translation, as shown in Figure 5. The results demonstrate the efficiency gains from using trackers, which reduce the average latency per frame by up to 0.34 seconds. However, a trade-off between accuracy and efficiency is observed; for example, the fastest tracker, `ocsort`, achieves a 0.02 lower F1-score compared with the

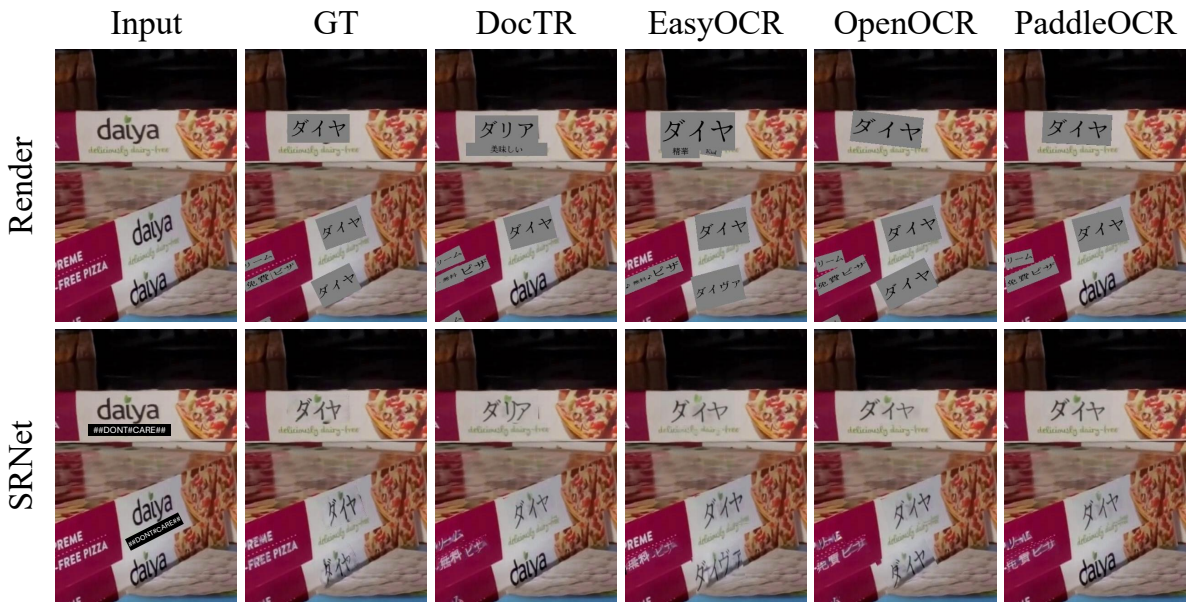

Figure 6: Comparison of outputs generated by pipelines using various OCRs (columns) and editors (rows). GT refers to ground truth where both coordinates and texts are provided in the dataset. The image in the second row under "Input" is used to highlight the "##DONT#CARE##" regions since it is identical with the image above it.

baseline. To this end, the results suggest employing `bytetrack` or `botsort` in the pipeline, as they achieve performance comparable to the baseline while reducing latency.

4.4 Qualitative Analysis

Figure 6 presents an output example translated from English to Japanese using pipelines that integrate different OCR systems and editors. For comparison, we also include the output generated from ground-truth detection and recognition (GT), excluding the "##DONT#CARE##" regions. The editors compared are Render and SRNet; for the latter, we retrained an English-to-Japanese SRNet model on synthetic data following the same settings as Kaing et al. (2025). The example shown is a cropped frame from a video in the DSText dataset. We selected a slow-motion video and chose a relatively sharp frame, further cropping it to enhance readability in the illustration.

Overall, the pipeline produces reasonable results across different OCRs. The outputs are generally consistent, with only minor variations in handling off-angle or blurry text and in the detected coordinates. Among them, OpenOCR achieves results most closely aligned with the ground truth (GT), consistent with the findings in Figure 4. When SRNet is used, the translations of the word "daiya" are clearly readable, and the original text is cleanly erased—except for the small green dot above the letter i, which is preserved. These results are notable given that the model was trained solely on synthetic data. Nevertheless, the model continues to face challenges with more complex cases, such as text over colorful backgrounds or characters with higher visual complexity (e.g., Kanji). We believe that integrating state-of-the-art scene text editors could substantially enhance the quality of the pipeline's outputs.

5 Conclusion

This work introduces ImageTra—an open-source toolkit for building pipeline-based IIMT systems that leverage state-of-the-art models and tools. The toolkit supports real-time translation and integrates two key features to enhance efficiency: translation caching and text tracking. Our evaluation demonstrates that these features significantly reduce inference latency without compromising accuracy.

While the current implementation yields promising results, there remains considerable room for improvement in both efficiency and accuracy. For instance, real-time translation could be enhanced by adopting lightweight, unified models that operate in a more end-to-end manner. Furthermore, translation quality could be improved by leveraging contextual information rather than translating words or phrases in isolation. Pursuing these directions is a core part of our development roadmap as we work toward practical, real-world applications.

References

Nir Aharon, Roy Orfaig, and Ben-Zion Bobrovsky. 2022. Bot-sort: Robust associations multi-pedestrian tracking. *arXiv preprint arXiv:2206.14651*.

Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun, and Hwalsuk Lee. 2019. Character region awareness for text detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9365–9374.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate.

Marta Bañón, Pinzhen Chen, Barry Haddow, Kenneth Heafield, Hieu Hoang, Miquel Esplà-Gomis, Mikel L. Forcada, Amir Kamran, Faheem Kirefu, Philipp Koehn, Sergio Ortiz Rojas, Leopoldo Pla Sempere, Gema Ramírez-Sánchez, Elsa Sarrías, Marek Strelec, Brian Thompson, William Waites, Dion Wiggins, and Jaume Zaragoza. 2020. ParaCrawl: Web-scale acquisition of parallel corpora. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4555–4567.

Darwin Bautista and Rowel Atienza. 2022. Scene text recognition with permuted autoregressive sequence models. In *European conference on computer vision*, pages 178–196.

Jinkun Cao, Jiangmiao Pang, Xinshuo Weng, Rawal Khirodkar, and Kris Kitani. 2023. Observation-centric sort: Rethinking sort for robust multi-object tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9686–9696.

Marta R Costa-Jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, and 1 others. 2022. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*.

Cheng Cui, Ting Sun, Manhui Lin, Tingquan Gao, Yubo Zhang, Jiaxuan Liu, Xueqing Wang, Zelun Zhang, Changda Zhou, Hongen Liu, Yue Zhang, Wenyu Lv, Kui Huang, Yichao Zhang, Jing Zhang, Jun Zhang, Yi Liu, Dianhai Yu, and Yanjun Ma. 2025. Paddleocr 3.0 technical report. *Preprint*, arXiv:2507.05595.

Raj Dabre, Chenhui Chu, and Anoop Kunchukuttan. 2020. A comprehensive survey of multilingual neural machine translation. *CoRR*, abs/2001.01115.

Yongkun Du, Zhineng Chen, Hongtao Xie, Caiyan Jia, and Yu-Gang Jiang. 2025. Svtrv2: Ctc beats encoder-decoder models in scene text recognition. In *ICCV*.

Yunhao Du, Zhicheng Zhao, Yang Song, Yanyun Zhao, Fei Su, Tao Gong, and Hongying Meng. 2023. Strongsort: Make deepsort great again. *IEEE Transactions on Multimedia*, 25:8725–8737.

Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia. 2016. Multi30K: Multilingual English-German image descriptions. In *Proceedings of the 5th Workshop on Vision and Language*, pages 70–74.

Zhengyao Fang, Pengyuan Lyu, Jingjing Wu, Chengquan Zhang, Jun Yu, Guangming Lu, and Wenjie Pei. 2025. Recognition-synergistic scene text editing. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 13104–13113.

Weiqi Gu, Haiyue Song, Chenhui Chu, and Sadao Kurohashi. 2021. Video-guided machine translation with spatial hierarchical attention network. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: Student Research Workshop*, pages 87–92.

Tosho Hirasawa, Zhishen Yang, Mamoru Komachi, and Naoaki Okazaki. 2020. Keyframe segmentation and positional encoding for video-guided machine translation challenge 2020.

Hour Kaing, Chenchen Ding, Hideki Tanaka, and Masao Utiyama. 2024. Robust neural machine translation for abugidas by glyph perturbation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 311–318.

Hour Kaing, Haiyue Song, Chenchen Ding, Jiannan Mao, Hideki Tanaka, and Masao Utiyama. 2025. Towards scene text translation for complex writing systems. In *NLP2025*, pages 234–238.

Zhibin Lan, Liqiang Niu, Fandong Meng, Jie Zhou, Min Zhang, and Jinsong Su. 2024. Translatotron-V(ison): An end-to-end model for in-image machine translation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 5472–5485.

Bo Li, Shaolin Zhu, and Lijie Wen. 2025. MIT-10M: A large scale parallel corpus of multilingual image translation. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 5154–5167.

Minghui Liao, Zhaoyi Wan, Cong Yao, Kai Chen, and Xiang Bai. 2020. Real-time scene text detection with differentiable binarization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 11474–11481.

Shangbang Long, Siyang Qin, Dmitry Panteleev, Alessandro Bissacco, Yasuhisa Fujii, and Michalis Raptis. 2022. Towards end-to-end unified scene text detection and layout analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1049–1059.

Cong Ma, Yaping Zhang, Mei Tu, Yang Zhao, Yu Zhou, and Chengqing Zong. 2023. E2timt: Efficient and effective modal adapter for text image machine translation. In *International Conference on Document Analysis and Recognition*, pages 70–88.

Gerard Maggiolino, Adnan Ahmad, Jinkun Cao, and Kris Kitani. 2023. Deep oc-sort: Multi-pedestrian tracking by adaptive re-identification. In *2023 IEEE International conference on image processing (ICIP)*, pages 3025–3029. IEEE.

Elman Mansimov, Mitchell Stern, Mia Chen, Orhan Firat, Jakob Uszkoreit, and Puneet Jain. 2020. Towards end-to-end in-image neural machine translation. In *Proceedings of the First International Workshop on Natural Language Processing Beyond Text*, pages 70–74.

Nibal Nayef, Yash Patel, Michal Busta, Pinaki Nath Chowdhury, Dimosthenis Karatzas, Wafa Khlif, Jiri Matas, Umapada Pal, Jean-Christophe Burie, Cheng-lin Liu, and 1 others. 2019. Icdar2019 robust reading challenge on multi-lingual scene text detection and recognition—rrc-mlt-2019. In *2019 International conference on document analysis and recognition (ICDAR)*, pages 1582–1587. IEEE.

Zhipeng Qian, Pei Zhang, Baosong Yang, Kai Fan, Yiwei Ma, Derek F. Wong, Xiaoshuai Sun, and Rongrong Ji. 2024. AnyTrans: Translate AnyText in the image with large scale models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 2432–2444.

Yadong Qu, Qingfeng Tan, Hongtao Xie, Jianjun Xu, Yuxin Wang, and Yongdong Zhang. 2023. Exploring stroke-level modifications for scene text editing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 2119–2127.

Prasun Roy, Saumik Bhattacharya, Subhankar Ghosh, and Umapada Pal. 2020. Stefann: scene text editor using font adaptive neural network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13228–13237.

Elizabeth Salesky, David Etter, and Matt Post. 2021. Robust open-vocabulary translation from visual text representations. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7235–7252.

Elizabeth Salesky, Philipp Koehn, and Matt Post. 2024. Benchmarking visually-situated translation of text in natural images. In *Proceedings of the Ninth Conference on Machine Translation*, pages 1167–1182.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96.

Baoguang Shi, Xiang Bai, and Cong Yao. 2016. 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.

Jeyasri Subramanian, Varnith Chordia, Eugene Bart, Shaobo Fang, Kelly Guan, Raja Bala, and 1 others. 2021. Strive: Scene text replacement in videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14549–14558.

Yanzhi Tian, Xiang Li, Zeming Liu, Yuhang Guo, and Bin Wang. 2023. In-image neural machine translation with segmented pixel sequence-to-sequence model. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 15046–15057.

Yanzhi Tian, Zeming Liu, Zhengyang Liu, and Yuhang Guo. 2025. Exploring in-image machine translation with real-world background. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 124–137.

Shreyas Vaidya, Arvind Kumar Sharma, Prajwal Gatti, and Anand Mishra. 2025. Show me the world in my language: Establishing the first baseline for scene-text to scene-text translation. In *International Conference on Pattern Recognition*, pages 312–328.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Liang Wu, Chengquan Zhang, Jiaming Liu, Junyu Han, Jingtuo Liu, Errui Ding, and Xiang Bai. 2019. Editing text in the wild. In *Proceedings of the 27th ACM international conference on multimedia*, pages 1500–1508.

Weijia Wu, Yuzhong Zhao, Zhuang Li, Jiahong Li, Mike Zheng Shou, Umapada Pal, Dimosthenis Karatzas, and Xiang Bai. 2023. Icdar 2023 competition on video text reading for dense and small text. In *Document Analysis and Recognition - ICDAR 2023: 17th International Conference, San José, CA, USA, August 21–26, 2023, Proceedings, Part II*, page 405–419.

Qiangpeng Yang, Jun Huang, and Wei Lin. 2020. Swaptext: Image based texts transfer in scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14700–14709.

Weichao Zeng, Yan Shu, Zhenhang Li, Dongbao Yang, and Yu Zhou. 2024. Textctrl: Diffusion-based scene text editing with prior guidance control. *Advances in Neural Information Processing Systems*, 37:138569–138594.

Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. 2022. Bytetrack: Multi-object tracking by associating every detection box. In *European conference on computer vision*, pages 1–21.

Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang. 2017. East: an efficient and accurate scene text detector. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 5551–5560.