

Received May 4, 2022, accepted June 6, 2022, date of publication June 14, 2022, date of current version June 21, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3183068

FR-DETR: End-to-End Flowchart Recognition With Precision and Robustness

LIANSHAN SUN¹, HANCHAO DU¹, AND TAO HOU

Shaanxi Joint Laboratory of Artificial Intelligence, School of Electronic Information and Artificial Intelligence, Shaanxi University of Science and Technology, Xi'an 710021, China

Corresponding author: Lianshan Sun (sunlianshan@sust.edu.cn)

This work was supported by the Scientific Research Program through the Education Department of Shaanxi Provincial Government under Grant 17JK0087.

ABSTRACT Traditional flowchart recognition methods have difficulties in detecting newly added symbols and distinguishing targets from complex backgrounds like line texture. Existing deep-learning-based object detectors and line segment detectors are promising in recognizing and distinguishing targets from texture backgrounds. However, using two separate detectors will inevitably cause unnecessary training and inference costs. Moreover, the insufficient volume and diversity of currently available dataset limit the effectiveness of model training. To address these issues, this paper proposes an end-to-end multi-task network FR-DETR (Flowchart Recognition DETECTION TRANSFORMER) and a new dataset for precise and robust flowchart recognition. FR-DETR comprises a CNN backbone and a shared multi-scale Transformer structure to perform symbol detection and edge detection using shared feature maps and respective prediction heads in a coarse-to-fine refinement process. The coarse stage analyzes features with low resolution and suggests candidate regions that contain potential targets for the fine stage to produce accurate predictions using features with high resolution. Meanwhile, a new dataset is constructed to provide more symbol types and complex backgrounds for network training and evaluation. It contains more than 1000 machine-generated flowchart images, 25K+ symbol instances with nine categories, and 20K+ line segments. The experiments show that FR-DETR achieves an overall precision and recall of 94.0% and 93.1% on the proposed dataset, and 98.7% and 98.1% on the CLEF-IP dataset, respectively, which all outperform the prior methods.

INDEX TERMS Flowchart recognition, multi-task learning, multi-scale Transformer, object detection, line segment detection.

I. INTRODUCTION

Flowchart recognition is an essential sub-task in research on document analysis and recognition [1]. The critical problem of flowchart recognition is to recognize and refine the structural semantics of flowcharts. There are two study areas for flowchart recognition: handwritten flowchart recognition and machine-generated flowchart recognition. In the past few years, many researchers have mainly focused on analyzing handwritten flowcharts, while machine-generated ones have been rarely concerned. However, understanding the structural semantics of machine-generated flowchart images is crucial for many structural-semantic-based tasks, such as patent retrieval, automatic code generation, and task-oriented dialogue systems [2]–[6].

The associate editor coordinating the review of this manuscript and approving it for publication was Yonghong Peng¹.

Dataset available online: https://github.com/harolddu/frdetr_dataset

Existing methods [2], [3], [8]–[10], [23] for recognizing machine-generated flowchart mainly extract the entire structure by analyzing the connected components in images and then identify specific structures using manually chosen features, which results in their failures of detecting broken edges and dotted lines. Traditionally, these methods focus on flowcharts in binary images or images with simple backgrounds. However, as information technologies advance and data volumes increase, flowchart images are becoming more diverse and complex to meet various aesthetic requirements. In detail, flowchart structures now have more varied and colorful symbols and more complex backgrounds, leading to problems such as decrements in recognition accuracy as well as incompatibility and inflexibility of manually chosen features. Meanwhile, traditional detection methods normally aim to perform universal detection, which may produce redundant results and cause models' inability to separate targets from interference (e.g., line-shape texture backgrounds).

Different from traditional methods, deep-learning-based computer vision technologies are capable of focusing on desired targets, which can be summarized into symbols and edges in flowcharts. Deep-learning-based object detectors are currently widely used for detecting objects within images. Although these approaches perform well in symbol detection, their bounding box representation makes it difficult to recognize line segments that are short or nearly parallel to the axes. Some deep-learning-based line segment detectors that treat edges as endpoint pairs can be used to achieve good line segment detection performance. Consequently, the flowchart recognition task can be divided into recognizing symbols using object detection and detecting edges using line segment detection. As shown in Fig. 1, symbols and arrows are indicated by bounding boxes and edges are indicated by line segments.

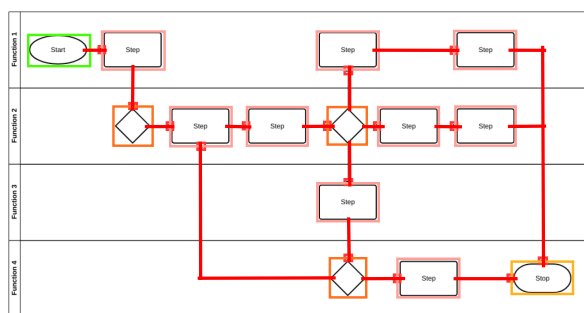


FIGURE 1. Representation of proposed flowchart recognition method: bounding boxes localize each symbol and arrow, line segments denote connecting edges, regardless of texture backgrounds.

Many methods manage these two tasks separately. For example, some works [11]–[14] handle object detection, while some others [15]–[19] deal with line segment detection. Whereas these methods can achieve promising performance, the sequential processing of these tasks results in high training and inference costs. Some multi-task models [20]–[22], [25], [26] achieve competitive performance by combining different tasks into a model with one shared encoder and separated decoders. In flowchart recognition, information processed by different tasks is inherently correlated, such as symbols are connected by edges and edges are connecting symbols. We believe a multi-task model is more suitable for flowchart recognition because (1) it enables information to be shared between two tasks of object detection and line segment detection, which can simplify the network structure; and (2) it reduces the training and analysis process by managing two tasks simultaneously.

Based on these works, this paper proposes an end-to-end multi-task network architecture named FR-DETR, the first deep learning system for machine-generated flowchart recognition to the best of our knowledge. By fusing DETR (DEtection TRansformer) [14] and LETR (LinE segment TRansformers) [19], FR-DETR has a CNN backbone and a multi-scale Transformer structure to perform fine-grained symbol detection and edge detection

respectively. In addition, to satisfy the requirements of data volume and complexity for model training, a new flowchart dataset is also constructed. Compared with the only publicly accessible machine-generated flowchart dataset CLEF-IP (Cross-Language Evaluation Forum Intellectual Property), which has around 60 images, the new dataset has more than 1000 machine-generated flowchart images, 25K+ symbol instances with nine categories, and 20K+ line segments.

The main contributions of this paper are:

- 1) It demonstrates that the machine-generated flowchart recognition can be accomplished using deep-learning-based object detection and line segment detection to handle the increasing symbol diversity and background complexity of flowchart images.
- 2) It proposes an end-to-end multi-task learning network that combines object detection and line segment detection to reduce the costs caused by separate models. The model jointly detects symbols and edges in flowcharts and employs a multi-scale Transformer structure to improve the recognition accuracy of both tasks.
- 3) The proposed method achieves a better recognition accuracy that outperforms the prior machine-generated flowchart recognition methods.
- 4) A new machine-generated flowchart dataset is established to address data shortages for deep learning model training.

II. RELATED WORK

A. FLOWCHART RECOGNITION

Most prior studies of machine-generated flowchart recognition took place after the CLEF-IP in 2011, which was held by the Information Retrieval Facility (IFR) and released a public machine-generated flowchart dataset. Rusiñol *et al.* [7], [8] summarized flowcharts as structure layer and text layer, and then performed recognition after layer separation. The structure extracted by connected component analysis was used to distinguish symbols with geometric moments descriptor and blurred shape model. Mörzinger *et al.* [9] separated text and diagram before studying the visual features of a structure at the pixel level. Thean *et al.* [10] suggested a symbol recognition method based on text-graphic segmentation, shape-squeezing, and symmetric features. Zhang [23] proposed a corner-based structural model (CBSM) based on the analysis of different corners and symbol shapes. The CBSM recognizes symbols by defining corner classification and corner-based spatial constraints for each kind of graphic shapes. These methods mostly rely on the results of connected component analysis and different manually chosen features.

Over the last decade, research on handwritten flowchart recognition has received increasing attention. Carton *et al.* [27] fused structural and statistical information to compute grammatical descriptions for each type of symbol. Lemaitre *et al.* [28] analyzed flowchart structure based on the description and modification of segment (DMOS) and structural information. Bresler *et al.* [29] solved a max-sum model optimization task to obtain the best symbol

description. Schäfer et al. [30] proposed Arrow R-CNN to detect symbols and connecting edges by adding a keypoint head to Faster R-CNN. The keypoint head is designed to detect the arrow and tail belonging to a connecting edge as two keypoints. Arrow R-CNN was the first deep-learning-based flowchart recognition approach.

B. TRANSFORMER ARCHITECTURE

In recent years, Transformer [32] has become the standard backbone for many natural language processing (NLP) models and has achieved remarkable success. Its self-attention and cross-attention mechanisms can build strong relations among sequence-format inputs. Recently, Transformer has attracted more researchers and steps in the fields of computer vision [34]. Carion et al. [14] introduced a Transformer based end-to-end object detection network DETR that removed pre-designed anchor and non-maximum suppression (NMS). It detects objects using interactions between a fixed number of queries and encoded image features. Following the basic query concept of DETR, Xu et al. [19] proposed a Transformer based line segment detector LETR. Unlike the prior line segment detection approaches consisting of heuristics-guided steps, the LETR detector directly regressed the endpoints of a line segment and achieved state-of-the-art performance on relative line segment detection datasets.

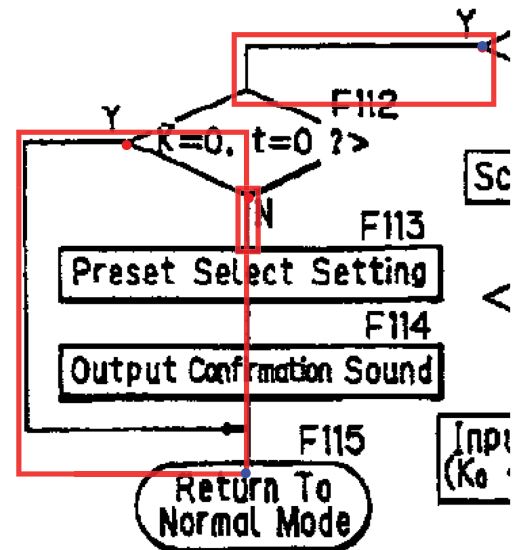
C. MULTI-TASK APPROACHES

Traditionally, task-oriented networks have been independently designed and trained for each task. However, as humans concurrently solve multiple tasks, many real-world problems are also multi-modal. This motivates researchers to study generalized methods that can accomplish all desired tasks with a single model. Deep-learning-based MTL (multi-task learning) models aim to improve network generalization and the capability of jointly learning shared information. Compared with single-task models, multi-task networks have advantages such as a reasonable reduction in model size and increment in inference speeds by sharing inherent parts of network structure [24]. Vandenhende et al. [25] considered the importance of task interactions while distilling information during multi-task learning. Hu et al. [26] jointly learned multiple tasks across different domains with a unified Transformer. Wu et al. [21] introduced a multi-task network that can jointly handle object detection, drivable area segmentation, and lane detection in autonomous driving.

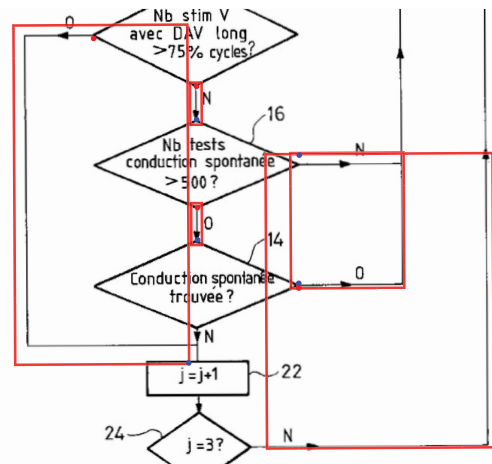
III. PROPOSED APPROACH

A. MOTIVATION

Although Arrow R-CNN [30] achieves remarkable results in handwritten flowcharts recognition, it is not fully applicable for recognizing machine-generated flowcharts. Every connecting edge detected by Arrow R-CNN must contain an arrow as a keypoint. However, several connecting edges within a machine-generated flowchart may share one arrow, which causes failures in detecting edges and keypoints when



(a) Errors when edges have no arrow



(b) Errors when edges have a nested layout

FIGURE 2. Due to the complex structure that machine-generated flowcharts have, errors occur when applying arrow R-CNN. bounding boxes indicate entire connecting edges, and each red point and blue point represent the start and tail of belonged edge. (a) shows the detection errors of box and keypoint when edges have no arrow. (b) shows the detection errors when edges have a nested layout.

applying Arrow R-CNN. As shown in Fig. 2, although the representation of connecting edges is adjusted to an entire connection between two symbols, errors in detecting edges and keypoints still occur. Based on the structural analysis of connecting edges, using line segment representation can better handle the dilemma in which Arrow R-CNN is stuck.

Line segments are not appropriately described with bounding boxes because of their highly variable aspect ratio and limited choices of anchors. Many works [16]–[18] follow the procedure of first producing junction proposals and then converting them into line segments, which causes the performance of these methods to rely heavily on junction detection. However, in a flowchart, junctions appear on both connecting edges and symbols, which brings numerous redundant

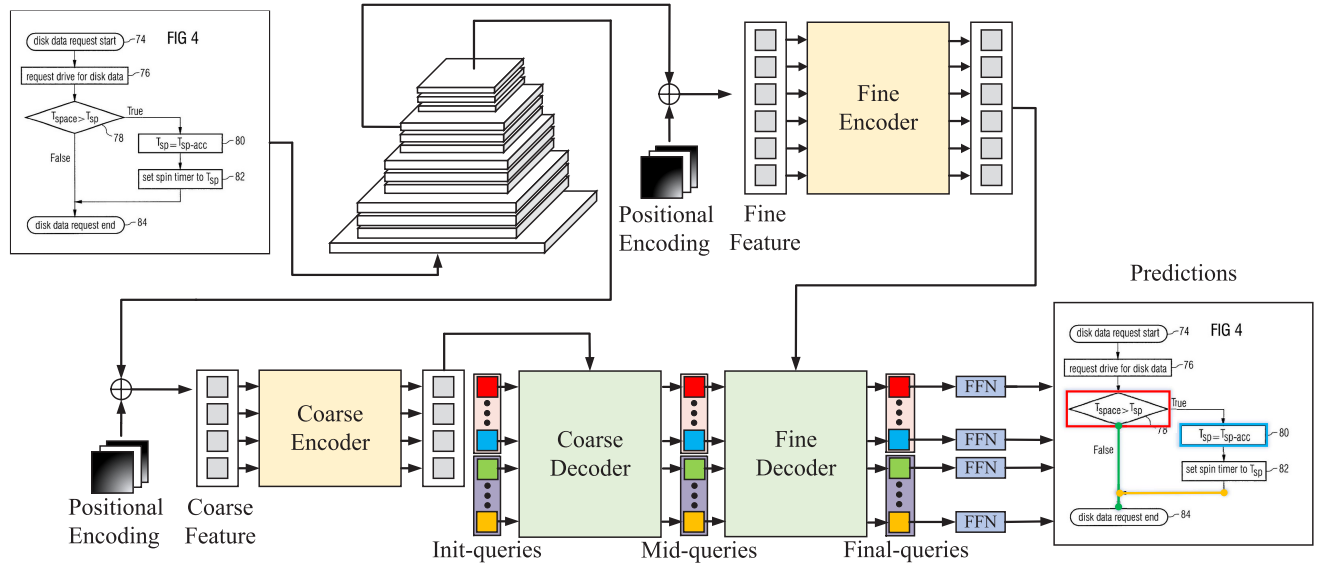


FIGURE 3. FR-DETR architecture. A convolution network is used as backbone to produce two feature maps with different resolutions from an input flowchart. The coarse encoder-decoder structure first encodes the smaller feature map and then generates the mid-queries with the interactions between encoded features and the init-queries. The fine encoder-decoder structure encodes the feature map with higher resolution and outputs final-queries based on the interaction between the corresponding encoded feature and mid-queries. In the end, the final-queries are fed into feed-forward networks to make the final predictions.

instances when using junction based approaches. In addition, due to convolution having a fixed respect field, CNN-based models have limitations in building long-range relations for targets like long lines. Recently, a Transformer-based line segment detector LETR [19] suggested a model that directly regresses the endpoints coordinates of each line segment. The attention mechanism introduced by Transformer perfectly meets the need to distinguish line segments between wanted and unwanted ones.

Inspired by the aforementioned methods, with the aim of improving the flowchart recognition accuracy and reducing the costs of using isolated models, this paper modifies LETR into a multi-task model to jointly accomplish the two detection tasks. The model selected to perform symbol detection is DETR. The reasons can be concluded as follows: (1) DETR has a similar Transformer-based structure to LETR, which maximizes structure sharing between the two models and the overall cost reduction. (2) Other CNN-based object detection models and LETR have few shareable parts, which unavoidably results in insufficient structure sharing and difficulties in jointly analyzing features.

B. THE FR-DETR MODEL

The overall architecture of FR-DETR is designed based on a multi-scale Transformer encoder-decoder structure as depicted in Fig. 3. The proposed flowchart recognition process can be divided into four sub-tasks: feature extraction, feature encoding, feature decoding and target prediction.

1) FEATURE EXTRACTION

FR-DETR uses a CNN backbone to extract a feature map $f_0 \in \mathbb{R}^{C \times H \times W}$ from an input image $x \in \mathbb{R}^{3 \times H_0 \times W_0}$.

The channel dimension of the feature map f_0 is then reduced from C to a smaller dimension d to obtain a new feature map $f \in \mathbb{R}^{d \times H \times W}$ by 1×1 convolution. To meet the encoder’s requirement, which expects input in the sequence format, the feature map f is flattened to create another feature map $z \in \mathbb{R}^{d \times HW}$.

2) FEATURE ENCODING

The Transformer encoder is stacked with multiple encoding layers. Each layer consists of a multi-head self-attention module and a feed-forward network (FFN). The encoding layers receive processed features from their predecessor layer and deliver the output features to the corresponding FFN after learning the pairwise relations between the input and output. In general, the flattened feature map $z \in \mathbb{R}^{d \times HW}$ is encoded into a new feature map $z' \in \mathbb{R}^{d \times HW}$. The positional encoding of f is added to guarantee the flattened feature map z not to lose the spatial relations.

3) FEATURE DECODING

Following the standard architecture of Transformer, the decoder transforms each N embeddings of symbol and line segment using the multi-head self-attention and cross-attention module. Like the positional encoding of the encoder, the input embeddings are learnable positional information that is added to the input of each layer and named as target queries. Each decoding layer receives $z' \in \mathbb{R}^{d \times HW}$ from the last encoding layer and two types of target queries $b \in \mathbb{R}^{d \times N}$ and $l \in \mathbb{R}^{d \times N}$, namely symbol queries and line queries, from its predecessor decoding layer. Both types of queries are first processed by the self-attention model, and then, each entity

in the queries is assigned to different regions of positional encoding by the cross-attention module. The output of the decoder is then used to predict the final results using an FFN.

4) TARGET PREDICTION

The final results of symbols and line segments are predicted by an FFN. Specifically, the coordinates of symbols and line segments are computed by a multi-layer perceptron (MLP) with three layers, and the confidence of the predicted targets is produced by a linear projection layer.

C. COARSE-TO-FINE STRATEGY

The detection of small objects, such as arrows in a flowchart, empirically needs high resolution feature maps to achieve better results. However, directly processing high resolution image features with Transformer incurs a high computation cost. In contrast to object detection, which mainly focuses on local and neighborhood regions, line segment detection needs to consider the fine-grained local features and the global information. According to previous works [11], [12], an efficient way to tackle the problem is a sequential two-stage structure, whose former component produces suggested regions for the other component to perform exact detection. Following this idea, FR-DETR performs both the desired tasks in a refinement process using a coarse-to-fine strategy. This strategy enables FR-DETR to learn from multi-scale image features and produce precise predictions. In general, the model first analyzes the global information to locate possible targets coarsely and then uses the location information to examine local features and perform fine-grained recognition.

In the coarse stage, FR-DETR studies a low resolution feature map to identify potential regions containing symbols and line segments. The low resolution feature map sent into the coarse encoder is the output of the ResNet [33] C_5 layer, and its size is $\frac{1}{32}$ of the original image resolution. After the encoding process, the encoded features and init-target queries are then passed into each decoding layer's cross-attention module and self-attention module. The predictions produced by the coarse stage are considered as potential target regions and received by fine decoding layers as mid-target queries. The coarse stage is important for improving the accuracy of the fine stage and can reduce the computation cost compared with directly processing high resolution features.

In the fine stage, based on the suggested potential regions, FR-DETR makes detailed predictions using a feature map with $\frac{1}{16}$ of the original image resolution, which is the output of the ResNet C_4 layer and is twice the size of the feature map used in the coarse stage. In general, fine decoding is similar to that in the coarse stage. The main difference is that it processes information with more details and focuses on the suggested regions to conclude predictions, making the fine stage crucial for accomplishing precise fine-grained detection.

While performing prediction, each detection task has one shared prediction head for the coarse and fine stage to make predictions precisely and progressively.

D. TARGET PREDICTION LOSS

In the encoding-decoding system, the multi-scale encoder-decoder gradually rectifies each N initial queries of symbol and line segment to achieve final queries in the same amount, respectively. In the prediction process, each type of final queries is fed into its corresponding FFN that consists of a classifier and a regressor to predict the category confidence p and the coordinates of every target. If a final query belongs to symbol detection, the coordinates prediction is in the format of bounding box $b = (cx, cy, w, h)$, which denotes the center point, width and height of the box. Otherwise, a line segment prediction l is represented by two endpoints (p_1, p_2) , where $p_1 = (x_1, y_1), p_2 = (x_2, y_2)$.

The set of predictions \hat{t} has N targets, and the set of ground truth t has M elements, normally $N > M$. To assign a bipartite matching between the predictions and ground truth, t is assumed as a set that padded with no object (\emptyset) to meet the size of \hat{t} . In this case, an optimization for the bipartite matching is used to find the permutation with the lowest cost:

$$\mathcal{L}_{match}(t, \hat{t}) = \sum_{i=1}^N \mathbb{I}_{\{c_i \neq \emptyset\}} [\lambda_1 \mathcal{L}_{target}(t_i, \hat{t}_{\hat{\sigma}_i}) - \lambda_2 p_{\hat{\sigma}_i}] \quad (1)$$

$$\hat{\sigma} = \arg \min_{\sigma} \mathcal{L}_{match}(t, \hat{t}) \quad (2)$$

where $\hat{\sigma}$ is the indices of the matched predictions in set \hat{t} . \mathcal{L}_{match} computes the total pair-wise cost of matching the ground truth t_i and prediction $\hat{t}_{\hat{\sigma}_i}$. \mathbb{I} is an indicator function used to drop \emptyset when computing \mathcal{L}_{match} , c_i represents the class of t_i . \mathcal{L}_{target} denotes the matching cost functions for symbol detection \mathcal{L}_{symbol} and line segment detection \mathcal{L}_{line} .

$$\mathcal{L}_{symbol}(b_i, \hat{b}_{\hat{\sigma}_i}) = \lambda_{L_1} \|b_i - \hat{b}_{\hat{\sigma}_i}\|_1 + \lambda_{IoU} \mathcal{L}_{IoU} \quad (3)$$

$$\mathcal{L}_{line}(l_i, \hat{l}_{\hat{\sigma}_i}) = \lambda_{L_1} \|l_i - \hat{l}_{\hat{\sigma}_i}\|_1 \quad (4)$$

where \mathcal{L}_{symbol} consists of L_1 loss and IoU loss. The L_1 loss computes the distance between the ground truth and symbol prediction, and \mathcal{L}_{IoU} uses the generalized IoU loss to measure the similarity between bounding boxes. Likewise, \mathcal{L}_{line} represents L_1 the distance of corresponding end point pair between two line segments of the ground truth and prediction.

For the two detection tasks, each task loss must also evaluate the results of classification. By adding a cross-entropy loss term, each task loss can be represented as:

$$\hat{\mathcal{L}}_{symbol}(b, \hat{b}) = \sum_{i=1}^N \lambda_{cls} \mathcal{L}_{cls}(b_i, \hat{b}_{\hat{\sigma}_i}) + \mathcal{L}_{symbol}(b_i, \hat{b}_{\hat{\sigma}_i}) \quad (5)$$

$$\hat{\mathcal{L}}_{line}(l, \hat{l}) = \sum_{i=1}^N \lambda_{cls} \mathcal{L}_{cls}(l_i, \hat{l}_{\hat{\sigma}_i}) + \mathcal{L}_{line}(l_i, \hat{l}_{\hat{\sigma}_i}) \quad (6)$$

where $\hat{\mathcal{L}}_{symbol}$ and $\hat{\mathcal{L}}_{line}$ are each task's loss term, \mathcal{L}_{cls} denotes the cross-entropy loss term for classification. The total loss of FR-DETR is formulated as:











$$\mathcal{L}_{total} = \hat{\mathcal{L}}_{symbol} + \hat{\mathcal{L}}_{line} \quad (7)$$

IV. EXPERIMENT AND RESULTS

A. DATASET

(1) CLEF-IP dataset. The original CLEF-IP dataset released in 2011 contains machine-generated flowchart images and other structural diagrams. After removing non-flowcharts, approximately 60 images remain, most of which are provided by the European Patent Office (EPO) for the patent retrieval study. These flowcharts have a simple white background, and their structures are drawn in black. The dataset has three main symbols: rectangle for processing action, diamond for decision, and oval for terminator.

TABLE 1. Statistic of the new dataset.

Symbol	Class	Counts
	arrow	13183
	rec	7309
	diamond	2603
	oval	1344
	ellipse	244
	circle	239
	parallel	501
	document	186
	hex	157
	line	20433

(2) The proposed dataset. To enrich the symbol category and structural complexity, public flowchart images are collected through the Internet by using image search engines, such as Google Image, Bing Image and Baidu Image. After filtering low quality images and removing duplicates, a dataset containing more than 1,000 images is constructed. The new dataset includes 25K+ symbol instances and 20K+ line segments. Statistical details are shown in Table 1. Moreover, the backgrounds of the images are not all purely white. For example, some samples have colorful backgrounds with line textures.

The FR-DETR model is trained on the new dataset, which is randomly divided into 800 training images and 200 testing images. Data augmentation methods including random resize, random flip, and random crop, are taken through all experiments. Usually, applying augmentations [31], such as copy-pasting, is an effective and efficient way to improve the detection results of small objects. Due to the arrow class, the only category of small objects in the dataset, already has the largest amount, specific augmentation for such class is unnecessary. The input images are resized to 600 pixels on the longer side during the training process and at least 600 pixels on the shorter side during the evaluation process.

Algorithm 1 Task-by-Task Training Strategy

Input:

Target network \mathcal{N} with parameters:
 $\Theta = \{\theta_{line}^c, \theta_{symbol}^c, \theta_{line}^f, \theta_{symbol}^f, \theta_{backbone}\}$;
 $// c = \text{coarse stage}, f = \text{fine stage}$

Training data set: \mathcal{S} ;

Preset train epochs:

$\Phi = \{\phi_{warmup}, \phi_{end}, \phi_c, \phi_f\}$

Output: Well-trained network: \mathcal{N}

```

1: for  $i$  in  $\{c, f\}$  do
2:    $\Theta \leftarrow \{\theta_{line}^i, \theta_{symbol}^i\}$ ; // Initialization
3:   if  $i = c$  then
4:      $\Theta \leftarrow \Theta \cup \{\theta_{backbone}\}$ ; // Train backbone in the coarse
       stage
5:   end if
6:   Train( $\mathcal{N}, \mathcal{S}$ ) for  $\phi_{warmup}$  epochs;
7:    $\Theta \leftarrow \Theta \setminus \{\theta_{symbol}^i\}$ ; // Freeze  $\theta_{symbol}^i$ 
8:   Train( $\mathcal{N}, \mathcal{S}$ ) for  $\phi_i$  epochs;
9:    $\Theta \leftarrow \Theta \cup \{\theta_{symbol}^i\} \setminus \{\theta_{line}^i\}$ ; // Freeze  $\theta_{line}^i$  and activate
        $\theta_{symbol}^i$ 
10:  Train( $\mathcal{N}, \mathcal{S}$ ) for  $\phi_i$  epochs;
11:   $\Theta \leftarrow \Theta \cup \{\theta_{line}^i\}$ ; // Activate all parameters
12:  Train( $\mathcal{N}, \mathcal{S}$ ) for  $\phi_{end}$  epochs; // Stage  $i$  finished
13: end for
14: return Trained Network  $\mathcal{N}$ ;

```

B. IMPLEMENTATION

1) NETWORK ARCHITECTURE

ResNet-50 and ResNet-101 are used as backbone to generate feature maps from the input image $x \in \mathbb{R}^{3 \times H_0 \times W_0}$. The coarse encoder and fine encoder take the output from the C_5 layer and C_4 layer, respectively. The output from C_5 layer is a low resolution feature map $f_0 \in \mathbb{R}^{C \times H \times W}$, where $C = 2048, H = \frac{H_0}{32}, W = \frac{W_0}{32}$. The output from C_4 layer is a feature map with higher resolution and $C = 1024, H = \frac{H_0}{16}, W = \frac{W_0}{16}$. Before being fed into the Transformer encoder, the feature map is compressed from C channels to 256 by a 1×1 convolution. Then, with a fixed sine/cosine positional encoding, the new feature map is sent into the Transformer encoder. To process multi-scale features, the network sets up two independent encoder-decoder structures. Similar to the standard Transformer structure, each encoder and decoder has six encoding layers and six decoding layers with eight attention heads. The two tasks share one encoder-decoder structure at each stage, and use respective prediction heads to produce the final prediction. The sharing structure can efficiently reduce the number of network parameters and the cost of learning and inferring.

2) TRAINING STRATEGY

A single GeForce RTX 3090 GPU is used to train and test the model through all experiments. The backbone weights are initialized using the pre-trained model provided by DETR. To accomplish coarse-to-fine detection, the network first

TABLE 2. Detection results of symbol and line segment/edge. (Edge is for arrow R-CNN only)

Network	Stage		Multi-Decoder	Symbol			Line Segment/Edge			Overall		
	One	Two		P%	R%	F1	P%	R%	F1	P%	R%	F1
DETR-R101 [14]	✓	-	-	97.9	97.3	0.976	-	-	-	-	-	-
LETR-R101 [19]	-	✓	-	-	-	-	98.9	99.1	0.990	-	-	-
Arrow R-CNN-R101 [30]	-	✓	-	90.9	83.7	0.872	81.2	77.3	0.792	86.2	80.9	0.836
FR-DETR-R50	✓	-	-	80.3	82.7	0.815	77.2	82.6	0.798	78.9	82.7	0.807
	-	✓	✓	87.6	89.3	0.884	92.3	90.5	0.914	89.6	89.8	0.897
FR-DETR-R101	-	✓	✓	89.1	91.2	0.901	93.6	91.7	0.926	91.0	91.4	0.912
	✓	✓	✓	91.8	91.7	0.917	95.6	93.3	0.944	93.5	92.4	0.929
	-	✓	✓	92.4	92.5	0.924	96.1	93.7	0.949	94.0	93.1	0.935

TABLE 3. Comparison of parameter volume and inference time among different models using ResNet50 as backbone.

Network	Task		Stage		#params	seconds per image
	Symbol	Line segment	One	Two		
DETR [14]	✓	-	✓	-	41.3M	5.23
LETR [19]	-	✓	✓	-	41.3M	4.99
	-	-	-	✓	59.1M	5.06
FR-DETR	✓	✓	✓	-	41.5M	5.26
			-	✓	59.5M	5.32

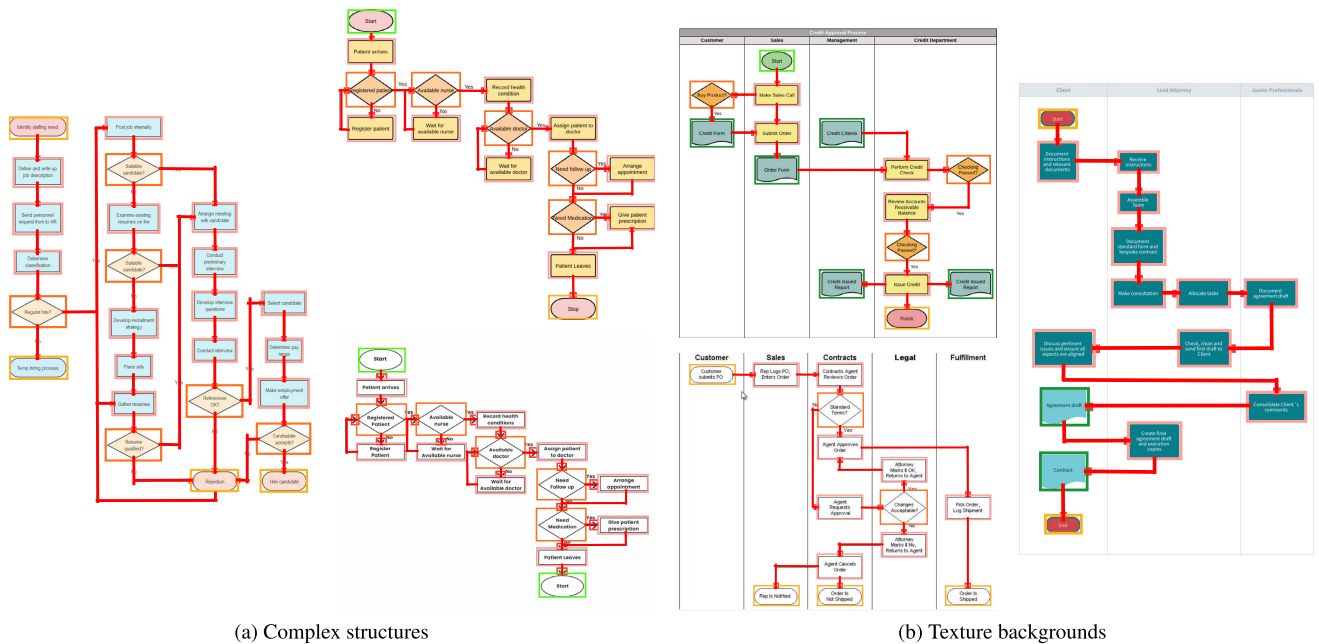


FIGURE 4. Detection results of FR-DETR on the new dataset. (a) shows the results of recognizing flowcharts with complex structures, dense targets and broken edges. (b) shows the recognition results of flowcharts with texture backgrounds.

trains only the coarse stage for 500 epochs. Subsequently, all parameters of the coarse encoder-decoder are frozen, and the fine stage is trained for 300 epochs after being initialized by coarse weights. Different training paradigms, such as end-to-end strategy and task-by-task strategy, are applied to train

the model. The end-to-end training simply trains the entire network at once, and the symbol detection and line segment detection are therefore trained jointly. The task-by-task training allows the model to focus on one task each time. In detail, the entire model is first trained for 50 warm-up epochs to

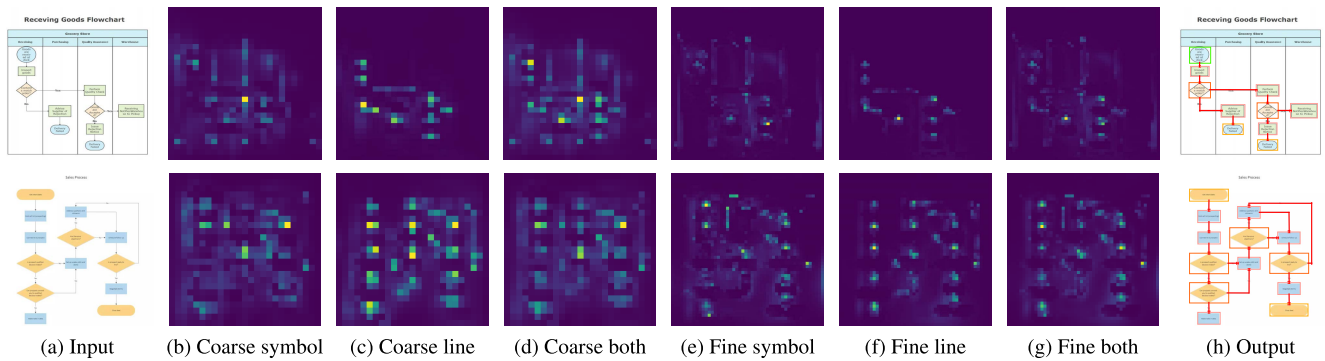


FIGURE 5. Visualization of FR-DETR's coarse-to-fine decoding process. The first column shows the input images. The (b) to (d) columns show the decoding results for symbols, line segments and both at the end of the coarse stage. Likewise, the (e) to (g) columns show the decoding results at the end of the fine stage. The last column shows detection results.

be roughly convergent. Then, while all the parameters of the other branch are frozen, each task is trained for 300 epochs in the coarse stage and 200 epochs in the fine stage. Finally, all the parameters are unfrozen and trained together for another 100 epochs. The specific training steps are presented in Algorithm 1. In the end-to-end training strategy, the learning rate is initially set to 0.0001 and multiplied by a factor of 0.1 every 200 epochs in the coarse stage and 120 epochs in the fine stage. In the task-by-task training strategy, learning rates are shared between tasks only during the warm-up and jointly training stage. AdamW is used as the optimizer and the weight decay is set to 10^{-4} .

C. EVALUATION METRIC

As pioneering flowchart recognition works, the evaluation metric selected in this paper is the F1-Score, which is a standard metric in flowchart recognition. The F1-Score is the weighted average of Precision and Recall, where Precision measures the percentage of correct prediction and Recall shows the percentage of correctly detected targets.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (8)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (9)$$

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (10)$$

where TP (true positive) denotes correct positive predictions of positive targets, FP (false positive) denotes incorrect positive predictions of negative targets, FN (false negative) denotes incorrect negative predictions of positive targets.

D. RESULTS AND COMPARISONS

Regarding the evaluation metric, Table 2 reveals the detection results of symbols and line segments on the new dataset. It is clear that the overall performance of FR-DETR is better than Arrow R-CNN. Table 2 also shows that all two-stage models achieve better results than the single-stage model, which means that the two-stage structure can effectively improve detection accuracy. The effectiveness of multiple

TABLE 4. Comparison of different training strategies.

Strategy	Symbol			Line Segment		
	P%	R%	F1	P%	R%	F1
End-to-end	87.6	89.3	0.884	92.3	90.5	0.914
Task-by-task	87.4	90.5	0.889	92.8	90.3	0.915

decoders is also tested. In this case, the network structure still has a shared backbone and encoders, but the decoding results sent into each prediction head are produced by their respective decoders. This structure allows each decoding module to focus on its specific task. The results show that the multi-decoder does improve the model performance. Compared with DETR and LETR, there is a slight gap between the results of FR-DETR and the single-task models, while the multi-task model makes it difficult to find an optimal global solution for all tasks. As shown in Table 3, using ResNet50 as the backbone, FR-DETR has an inference time similar to that of DETR and LETR. However, FR-DETR can productively reduce the total inference time with a slight increment in the number of parameters. Generally, FR-DETR can significantly reduce the time consumption with a small decrease in accuracy.

The comparison of different training strategies' performance is shown in Table 4. It shows that the two strategies achieve almost the same results, while the task-by-task training is slightly better than the end-to-end strategy.

FR-DETR is robust and can handle complex flowchart images. Unlike the CLEF-IP dataset, the new dataset provides flowcharts with more complex structures and backgrounds, enabling FR-DETR to address challenging problems such as broken edges and misleading line textures. As shown in Fig. 4, the images on the left have a dense layout of symbols and broken connecting edges, the images on the right are flowcharts with texture backgrounds. The detection results show that FR-DETR can accurately localize each symbol and line segment regardless of structure complexity, backgrounds and broken edges.

TABLE 5. Comparison of FR-DETR and other methods on the CLEF-IP dataset.

Method	P%	R%	F1
FR-DETR-R50	98.7	98.1	0.984
CVC-UAB [7]	72.5	90.3	0.804
JOANNEUM [9]	79.8	85.6	0.826
INRIA [10]	89.1	87.9	0.885
Arrow R-CNN [30]	93.8	88.3	0.910

The demonstration of FR-DETR's coarse-to-fine decoding process is shown in Fig. 5. The results of the coarse stage are produced by the coarse decoder decoding features from ResNet's C_5 layer. Although the target information is will-captured, the features with a low resolution limit the model from making precise predictions. The outputs of the fine stage are generated by the fine decoder decoding high resolution features from ResNet's C_4 layer with target queries from the coarse stage. The overlay of attention heatmaps shows more detailed relations in the image space, which is the key to the detector performance.

The overall recognition results on the CLEF-IP dataset, as shown in Table 5, are improved to 98.7% precision and 98.1% recall by FR-DETR, which indicates that the proposed method outperforms the prior approaches.

V. CONCLUSION

This paper presents a new method for machine-generated flowchart recognition, which accomplishes the task by detecting symbols and connecting edges using deep-learning-based object detectors and line segment detectors. An end-to-end multi-task model named FR-DETR that contains a multi-scale Transformer structure is introduced to reduce the high cost caused by using separate models. Its well-performed joint detection of symbols and line segments significantly simplifies the flowchart recognition task. A new machine-generated flowchart dataset is also constructed for practical model training and evaluation.

Although FR-DETR outperforms other flowchart recognition methods, it is not lightweight enough to meet the requirements of real-time processing and mobile application development. In addition, recognizing flowcharts and understanding the corresponding structural semantics with an end-to-end model is still a challenge.

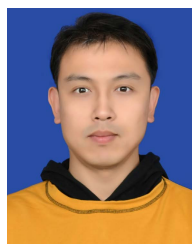
REFERENCES

- [1] J. Lladós, "Two decades of GREC workshop series. Conclusions of GREC2017," in *Graphics Recognition. Current Trends and Evolutions* (Lecture Notes in Computer Science), vol. 11009, A. Fornés and B. Lamiroy, Eds. Cham, Switzerland: Springer, Nov. 2018, pp. 163–168, doi: [10.1007/978-3-030-02284-6_14](https://doi.org/10.1007/978-3-030-02284-6_14).
- [2] N. Bhatti and A. Hanbury, "Image search in patents: A review," *Int. J. Document Anal. Recognit.*, vol. 16, no. 4, pp. 309–329, Dec. 2013, doi: [10.1007/S10032-012-0197-5](https://doi.org/10.1007/S10032-012-0197-5).
- [3] S. Adams, "Electronic non-text material in patent applications—Some questions for patent offices, applicants and searchers," *World Patent Inf.*, vol. 27, no. 2, pp. 99–103, Jun. 2005, doi: [10.1016/J.WPLI.2004.12.005](https://doi.org/10.1016/J.WPLI.2004.12.005).
- [4] F. Piroi, M. Lupu, A. Hanbury, A. P. Sexton, W. Magdy, and I. V. Filippov, "CLEF-IP 2012: Retrieval experiments in the intellectual property domain," in *Proc. CLEF Eval. Labs Workshop*, Rome, Italy, Sep. 2012, pp. 1–13. [Online]. Available: <http://ceur-ws.org/Vol-1178/CLEF2012wn-CLEFIP-PiroiEt2012.pdf>
- [5] C. Supaartagorn, "Web application for automatic code generator using a structured flowchart," in *Proc. 8th IEEE Int. Conf. Softw. Eng. Service Sci. (ICSESS)*, Beijing, China, Nov. 2017, pp. 114–117, doi: [10.1109/ICSESS.2017.8342876](https://doi.org/10.1109/ICSESS.2017.8342876).
- [6] J. Andreas et al., "Task-oriented dialogue as dataflow synthesis," *Trans. Assoc. Comput. Linguistics*, vol. 8, pp. 556–571, Sep. 2020, doi: [10.1162/TACL_A_00333](https://doi.org/10.1162/TACL_A_00333).
- [7] M. Rusiñol, L.-P. de Las Heras, J. Mas, O. R. Terrades, D. Karatzas, A. Dutta, G. Sánchez, and J. Lladós, "CVC-UAB's participation in the flowchart recognition task of CLEF-IP," in *Proc. CLEF Eval. Labs Workshop*, Rome, Italy, Sep. 2012, pp. 1–11. [Online]. Available: <http://ceur-ws.org/Vol-1178/CLEF2012wn-CLEFIP-RusinolEt2012.pdf>
- [8] M. Rusiñol, L.-P. de las Heras, and O. R. Terrades, "Flowchart recognition for non-textual information retrieval in patent search," *Inf. Retr.*, vol. 17, nos. 5–6, pp. 545–562, Oct. 2014, doi: [10.1007/S10791-013-9234-3](https://doi.org/10.1007/S10791-013-9234-3).
- [9] R. Mörzinger, R. Schuster, and A. Horti, "Visual structure analysis of flow charts in patent images," in *Proc. CLEF Eval. Labs Workshop*, Rome, Italy, Sep. 2012, pp. 1–8. [Online]. Available: <http://ceur-ws.org/Vol-1178/CLEF2012wn-CLEFIP-MorzingerEt2012.pdf>
- [10] A. Thean, J.-M. Deltorn, P. Lopez, and L. Romary, "Textual summarisation of flowcharts in patent drawings for CLEF-IP 2012," *CLEF 2012 Eval. Labs Workshop*, Rome, Italy, Sep. 2012, pp. 1–18. [Online]. Available: <http://ceur-ws.org/Vol-1178/CLEF2012wn-CLEFIP-TheanEt2012.pdf>
- [11] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Santiago, Chile, Dec. 2015, pp. 1440–1448, doi: [10.1109/ICCV.2015.169](https://doi.org/10.1109/ICCV.2015.169).
- [12] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017, doi: [10.1109/TPAMI.2016.2577031](https://doi.org/10.1109/TPAMI.2016.2577031).
- [13] G. Jocher et al., "Ultralytics/YOLOv5: V6.1—TensorRT, TensorFlow edge TPU and OpenVINO export and inference," Los Angeles, CA, USA, Tech. Rep., Feb. 2022. [Online]. Available: <https://zenodo.org/record/6222936> and <https://ultralytics.com>, doi: [10.5281/zenodo.6222936](https://doi.org/10.5281/zenodo.6222936).
- [14] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Proc. 16th Eur. Conf. Comput. Vis. (ECCV)*, Glasgow, U.K., Aug. 2020, pp. 213–229, doi: [10.1007/978-3-030-58452-8_13](https://doi.org/10.1007/978-3-030-58452-8_13).
- [15] G. Gu, B. Ko, S. Go, S.-H. Lee, J. Lee, and M. Shin, "Towards light-weight and real-time line segment detection," 2021, *arXiv:2106.00186*.
- [16] Z. Zhang, Z. Li, N. Bi, J. Zheng, J. Wang, K. Huang, W. Luo, Y. Xu, and S. Gao, "PPGNet: Learning point-pair graph for line segment detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Long Beach, CA, USA, Jun. 2019, pp. 7098–7107, doi: [10.1109/CVPR.2019.00727](https://doi.org/10.1109/CVPR.2019.00727).
- [17] Y. Zhou, H. Qi, and Y. Ma, "End-to-end wireframe parsing," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Seoul, South Korea, Oct. 2019, pp. 962–971, doi: [10.1109/ICCV.2019.00105](https://doi.org/10.1109/ICCV.2019.00105).
- [18] K. Huang, Y. Wang, Z. Zhou, T. Ding, S. Gao, and Y. Ma, "Learning to parse wireframes in images of man-made environments," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 625–626, doi: [10.1109/CVPR.2018.00072](https://doi.org/10.1109/CVPR.2018.00072).
- [19] Y. Xu, W. Xu, D. Cheung, and Z. Tu, "Line segment detection using transformers without edges," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 4257–4266, doi: [10.1109/CVPR46437.2021.00424](https://doi.org/10.1109/CVPR46437.2021.00424).
- [20] Y. Qian, J. M. Dolan, and M. Yang, "DLT-Net: Joint detection of drivable areas, lane lines, and traffic objects," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 11, pp. 4670–4679, Nov. 2019.
- [21] D. Wu, M. Liao, W. Zhang, X. Wang, X. Bai, W. Cheng, and W. Liu, "YOLOP: You only look once for panoptic driving perception," 2021, *arXiv:2108.11250*.
- [22] D. Vu, B. Ngo, and H. Phan, "HybridNets: End-to-end perception network," 2022, *arXiv:2203.09035*.

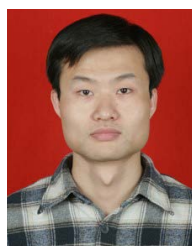
- [23] S. Zhang, "Research on flowchart recognition by fusing structural model and corner feature," M.S. thesis, College Electr. Inf. Eng., Shaanxi Univ. Sci. Technol., Xi'an, China, 2018. [Online]. Available: <https://oversea.cnki.net/KCMS/detail/detail.aspx?dbcode=CMFD&dbname=CMFD201802&filename=1018204634.nh>
- [24] S. Vandenhende, S. Georgoulis, W. Van Gansbeke, M. Proesmans, D. Dai, and L. Van Gool, "Multi-task learning for dense prediction tasks: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Jan. 26, 2021, doi: [10.1109/TPAMI.2021.3054719](https://doi.org/10.1109/TPAMI.2021.3054719).
- [25] S. Vandenhende, S. Georgoulis, and L. van Gool, "MTI-Net: Multi-scale task interaction networks for multi-task learning," in *Proc. 16th Eur. Conf. Comput. Vis. (ECCV)*, Glasgow, U.K., Aug. 2020, pp. 527–543, doi: [10.1007/978-3-030-58548-8_31](https://doi.org/10.1007/978-3-030-58548-8_31).
- [26] R. Hu and A. Singh, "UniT: Multimodal multitask learning with a unified transformer," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Montreal, QC, Canada, Oct. 2021, pp. 1419–1429, doi: [10.1109/ICCV48922.2021.00147](https://doi.org/10.1109/ICCV48922.2021.00147).
- [27] C. Carton, A. Lemaitre, and B. Coüasnon, "Fusion of statistical and structural information for flowchart recognition," in *Proc. 12th Int. Conf. Document Anal. Recognit.*, Washington, DC, USA, Aug. 2013, pp. 1210–1214, doi: [10.1109/ICDAR.2013.245](https://doi.org/10.1109/ICDAR.2013.245).
- [28] A. Lemaitre, H. Mouchère, J. Camillerapp, and B. Coüasnon, "Interest of syntactic knowledge for on-line flowchart recognition," in *Graphics Recognition. New Trends and Challenges (Lecture Notes in Computer Science)*, vol. 7423, Y.-B. Kwon and J.-M. Ogier, Eds. Berlin, Germany: Springer, Feb. 2013, pp. 89–98, doi: [10.1007/978-3-642-36824-0_9](https://doi.org/10.1007/978-3-642-36824-0_9).
- [29] M. Bresler, D. Prua, and V. Hlavac, "Modeling flowchart structure recognition as a max-sum problem," in *Proc. 12th Int. Conf. Document Anal. Recognit.*, Washington, DC, USA, Aug. 2013, pp. 1215–1219, doi: [10.1109/ICDAR.2013.246](https://doi.org/10.1109/ICDAR.2013.246).
- [30] B. Schäfer and H. Stuckenschmidt, "Arrow R-CNN for flowchart recognition," in *Proc. Int. Conf. Document Anal. Recognit. Workshops (ICDARW)*, Sydney, NSW, Australia, Sep. 2019, pp. 7–13, doi: [10.1109/ICDARW.2019.00007](https://doi.org/10.1109/ICDARW.2019.00007).
- [31] M. Kisantal, Z. Wojna, J. Murawski, J. Naruniec, and K. Cho, "Augmentation for small object detection," 2019, *arXiv:1902.07296*.
- [32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. 31th Int. Conf. Neural Inf. Process. Syst.*, Long Beach, CA, USA, Dec. 2017, pp. 6000–6010, doi: [10.48550/arXiv.1706.03762](https://doi.org/10.48550/arXiv.1706.03762).
- [33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778, doi: [10.1109/CVPR.2016.90](https://doi.org/10.1109/CVPR.2016.90).
- [34] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu, Z. Yang, Y. Zhang, and D. Tao, "A survey on vision transformer," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Feb. 18, 2022, doi: [10.1109/TPAMI.2022.3152247](https://doi.org/10.1109/TPAMI.2022.3152247).



LIANSHAN SUN received the Ph.D. degree in software engineering from Peking University, in 2009. He is an Associate Professor with the School of Electronic Information and Artificial Intelligence, Shaanxi University of Science and Technology. His main research interests include blockchain, information security, and artificial intelligence.



HANCHAO DU received the graduate degree in computer and its application from Xidian University, in 2015. He is currently pursuing the master's degree with the School of Electronic Information and Artificial Intelligence, Shaanxi University of Science and Technology. His main research interests include image recognition and artificial intelligence.



TAO HOU is a Lecturer with the School of Electronic Information and Artificial Intelligence, Shaanxi University of Science and Technology. His main research interests include software engineering and artificial intelligence.

...