Enhancing Chart-to-Text Conversion

Tianhe Yu
Stanford
theoyu@stanford.edu

Abstract

Chart-to-text conversion is a challenging task that aims to generate natural language descriptions from graphical representations of data. While recent advances in vision-language pretraining have shown promising results, current approaches often struggle to capture the intricate details and relationships present in charts. In this paper, we explore techniques to enhance the chart-to-text conversion capabilities of the Matcha framework, which employs a Vision Transformer (ViT) encoder and a transformer decoder. We analyze the limitations of the pretrained Matcha model using saliency maps and propose two approaches to improve the encoder: self-supervised contrastive learning with SimCLR and replacing the ViT encoder with a Swin Transformer. Our experiments highlight the potential of these techniques to capture fine-grained details and improve chart understanding. However, we also encounter challenges related to computational efficiency and the need for further architecture search. The initial experiments with the Swin Transformer yield a high loss value, indicating the need for extensive fine-tuning and adaptation to the unique characteristics of chart data. Despite the challenges, our work demonstrates the importance of developing specialized architectures and training strategies for chart-to-text conversion tasks.

1. Introduction

Vision-language pretraining has led to remarkable progress on tasks like image captioning [8], visual question answering, and image summarization. However, chart-to-text conversion, while similar in nature, remains a challenging open problem. Current work in retrieval augmented generation (RAG) focuses primarily on text-only setups, neglecting the wealth of information available in charts and graphs that are common in business and scientific contexts. Enabling RAG systems to understand and convert charts into natural language would significantly enhance their applicability and utility.

Transformers [16] have become the dominant architecture for both the encoder and decoder in these models. They consist of self-attention layers that allow each token to attend to all other tokens in the sequence, enabling efficient parallel computation and capturing long-range dependencies. Large language models (LLMs), such as GPT-4 [1], are transformer-based models trained on vast amounts of text data, allowing them to generate human-like text and perform various language tasks. A Vision Transformer (ViT) is a type of transformer architecture specifically designed for processing visual data[3]. It divides the image into a grid of fixed-size patches (e.g., 16x16 pixels). Each patch is flattened into a 1D vector and treated as a "token" (analogous to words in a text sequence). Then embeds the path vector into a higher-dimensional space using a learned embedding matrix and add positional embeddings. These encoded path are then sent a stacked of transformer encoder layers and generates hidden vector representations.

The machine learning community has explored chart understanding from multiple angles. ChartOCR [12] and the approach in [6] leverage OCR tools to extract key points and numbers from chart images. More recently, there has been a shift towards end-to-end deep learning methods. Pix2struct [7] and Matcha [10] use a vision transformer (ViT) encoder to process the input chart image and a transformer decoder to generate the corresponding text or structured representation. They demonstrates the effectiveness of incorporating ViT structure in encoder-decoder network for vision-language models.

The goal of our model is to convert input chart images into a structured table format, with delimiters denoting the chart type and values on the x and y axes. We train and evaluate on datasets the Benetech Kaggle competition [5], which contains over labeled 60000 charts. Performance is measured using the cross-entropy loss between the model's predicted token scores and ground truth labels, to quantify the accuracy of the generated structured tables.

In this work, we first analyze the behavior of the pretrained Matcha model by generating saliency maps to visualize where the model attends in the input image. We then explore two approaches to improve the encoder: 1) self-supervised contrastive learning with augmentations suitable

for chart images, and 2) replacing the ViT encoder with a Swin Transformer to better capture fine-grained details. Our experiments demonstrate the potential of these techniques to enhance chart understanding while highlighting challenges related to computational efficiency and the need for architecture search.

2. Related Work

Chart understanding has been approached from multiple angles in prior work. ChartOCR [12] and [6] use OCR-based techniques to extract key information from chart images, while more recent methods like Pix2struct [7], Matcha [10], GIT [17] and Deplot [9] employ end-to-end deep learning with transformer architectures. Pix2struct [7] is a pretraining approach for learning visual representations from web page screenshots, with the goal of improving performance on downstream vision-language tasks. The model is trained to predict the DOM (Document Object Model) structure of a web page from its screenshot, using a transformer encoder-decoder architecture. The encoder processes the screenshot image, while the decoder generates a linearized representation of the DOM tree, capturing the hierarchical layout and content of the web page elements. Starting from Matcha, The model architecture consist of a vision transformer (ViT) encoder that processes the input chart image and a transformer decoder that generates the corresponding text or structured representation. To improve performance, Matcha enriches its visual reasoning capabilities through tasks including Language modeling, which predict the next word in a text sequence, given the preceding words and the chart image; Math reasoning, which solve mathematical problems that are grounded in the chart, such as finding the minimum, maximum, or average value of a data series; And chart derendering, which generate a structured representation of the chart, such as a table of data points or a JSON object capturing the chart type and attributes. The Matcha framework demonstrates the effectiveness of incorporating domain-specific reasoning tasks into the pretraining process for vision-language models. Continuing on this trend, GIT trained on a massive dataset of 30 million image-text pairs to improve accuracy. In Deplot work, people trained with both supervised learning, which is to minimize the cross-entropy loss between the generated table and the ground truth table on a dataset of real chart images that have been manually annotated with their corresponding tables; and self-supervised learning, where they reconstructed the input image from the generated table and predicting masked tokens in the table.

Our work builds most directly on Matcha [10]. We experimented to improve the model's encoder through self-supervised learning and architecture changes. This is inspired by the success of contrastive learning in computer vision [2] and the Swin Transformer's [11] ability to capture both local and global information .

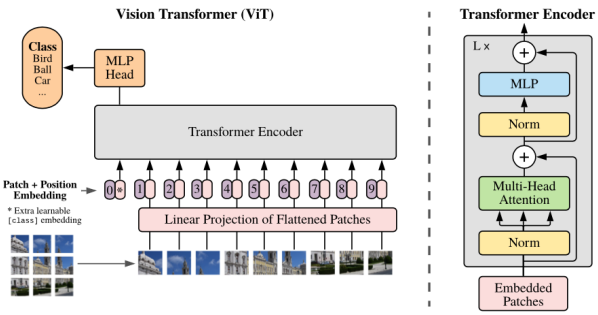


Figure 1: Vision Transformer Encoder Block .[3]

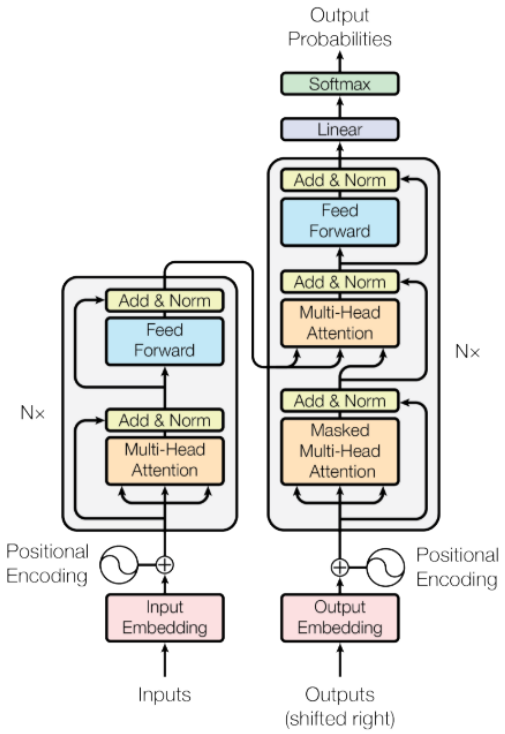


Figure 2: Encoder Block with Transformer Decoder Block [16]

Compared to prior work, our approach focuses on enhancing the feature extraction capabilities of the encoder through self-supervision and architecture search rather than scaling up pretraining data/model size or introducing additional task-specific modules. We evaluate on standard chart image datasets and metrics to facilitate comparison with existing methods.

3. Data

We utilize the kaggle Benetech Charts Dataset [5], which contains approximately 65,000 annotated scientific figures across four chart types: vertical and horizontal bar graphs, dot plots, line graphs, and scatter plots. Each figure is annotated with the chart type, axis values, and natural language summaries.

Some preprocessing steps were required, including filtering out figures with missing annotations and resizing images to a consistent resolution. data augmentation such as change hue, saturation and value, and shift RGB values are done, also applied standard normalization to the input image. We split the data into training, validation and test sets using an 80/10/10 ratio.

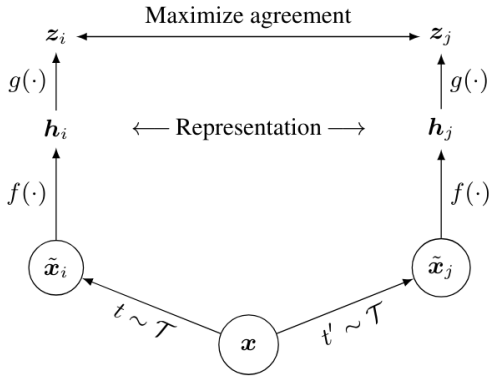


Figure 3: A simple framework for contrastive learning of visual representations[2]

4. Methods

Matcha is a standard encoder decoder model which does Image Patching, Patch Embedding and add Positional Embeddings, Encoding with transformer blocks, decoding with transformer blocks. We've showed the Vision Transformer encoder block in Figure 1 and the transformer decoder block which takes the hidden states as input and generate text in Figure 2.

4.1. Saliency Map Analysis

To understand how well it performs on existing chart to image task and the limitations, we first generated the saliency maps for input images. The concept of saliency maps was introduced by Simonyan [15], where they proposed a technique called "gradient-based saliency maps,". It computes the gradient of the output class score with respect to each input pixel. The magnitude of the gradient indicates the importance of each pixel for the model's prediction. Overall Saliency map is a visualization technique that highlights the regions of an input image that have the greatest influence on the model's output or prediction. It helps to understand which parts of the image are most important for the model's decision-making process. For transformer blocks one can look through the attention weights to get a hint on what the model is looking at. However, this approach can only check with respect to input patch size, and doesn't provide information over all transformer layers. Here instead we use the occlusion approach, where to generate a set of occluded images from the input image (10x10 patches here) and run the occluded image through the network. By comparing the loss difference relative to the original loss, we are able to visualize which regions caused the largest changes in loss when masked-off.

4.2. Performance Tuning Through Contrastive Learning

Our first approach to address the task was through self-supervised contrastive learning on the ViT encoder using a SimCLR [2] objective. SimCLR, which stands for Simple Contrastive Learning of Visual Representations, is a self-supervised learning framework for learning visual representations from unlabeled images. Although we have labeled data here, the decoder model itself has over 1 million paramters which incurs additional time to train. That why we consider only training the encoder module with self-Supervised Learning. Similar to the original paper, Our SimCLR approach is formulated with 6 steps 3.

We first generated augmented views of each chart image through cropping, scaling, and rotations up to 180 degrees (to preserve axis label orientation). The goal is to create two different views of each image, referred to as the "positive pair; Then Encoding using the VIT encoder captured from Matcha model; A projection head, which is a linear layer to map the encoder output to a one dimension vector; Then calculates the contrastive Loss over the batch of augmented data considering all the positive pairs within the batch. The contrastive loss function is a normalized temperature-scaled cross-entropy loss. For each positive pair of augmented views, the loss encourages their projected representations to be similar while pushing apart the representations of different images. The loss is then backward propagated to update the weights of the encoder, forcing the encoder to produce similar representations for different augmented views of the same image and dissimilar representations for different images. Lastly we update the Matcha's encoder module with this newly trained weights.

4.3. Swin Transformer as Encoder

When analyzing a chart to extract and interpret the underlying data, humans typically employ a systematic approach that focuses on several critical aspects. Initially, attention is directed towards the numerical values and textual labels on the axes, which provide essential information about the scale and context of the data being represented. This step is crucial for establishing a foundational understanding of the chart's scope and the range of values being depicted. Subsequently, the focus shifts to the visual elements within the chart, such as bars, lines, or markers, which represent the individual datapoints. By assessing the length, height, or position of these elements relative to the axes, one can discern the relative magnitudes and relationships between different datapoints. This visual comparison allows for a quick grasp of the overall trends and patterns present in the data. Finally, to determine the precise values associated with each datapoint, a process of interpolation is employed. By considering the position of a datapoint in relation to the labeled values on the axes, one can estimate its exact value. This interpolation step involves mentally subdividing the intervals between the axis labels and aligning the datapoint with the corresponding subdivisions. These observations means the task needs to capture the detailed information on the axises, measure position of points compared to axises in the chart, and understand the values for the posistion. This involve fine grained local details, cross communication among different set of patches and a global view of the chart.

Vision Transformer architecture, although has shown impressive performance on various computer vision tasks, has a limited capability of capturing fine-grained visual details. This is because ViT processes images as a sequence of fixed-size patches and relies on global self-attention to model the relationships between these patches. In the ViT architecture, an image is divided into a grid of fixed-size patches (e.g., 16x16 pixels), and each patch is linearly projected into a token embedding. These token embeddings are then processed by a stack of transformer encoder layers, which apply global self-attention to model the dependencies between all pairs of patches. While this global self-attention mechanism allows ViT to capture long-range dependencies and global context, it may struggle to capture fine-grained local details within each patch.

Swin Transformer architecture[11] addresses this issue and captures fine-grained visual details through a set of modifications to the ViT architecture. That includes Hierarchical Feature Representation, where it starts with small patches (e.g., 4x4 pixels) and gradually merges them into larger patches in deeper layers. This hierarchical representation allows the model to capture visual details at different scales and resolutions4; And local Self-Attention, where each image is divided into non-overlapping windows, and

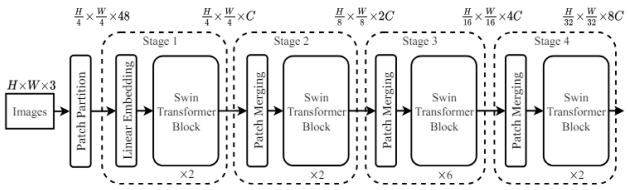


Figure 4: The architecture of a Swin Transformer[11]

self-attention is computed only within each window. This enables the model to focus on capturing fine-grained details within each local region of the image; Shifted Window Partitioning, where the windows are shifted by half the window size in both horizontal and vertical directions to allow for capturing dependencies between neighboring windows and maintain the global connectivity of the image; And window shuffling across different positions in the feature map to capture the dependencies among distant windows.

The Swin transformer architecture should be ideal for our task from this logical standpoint.

4.4. Loss Function

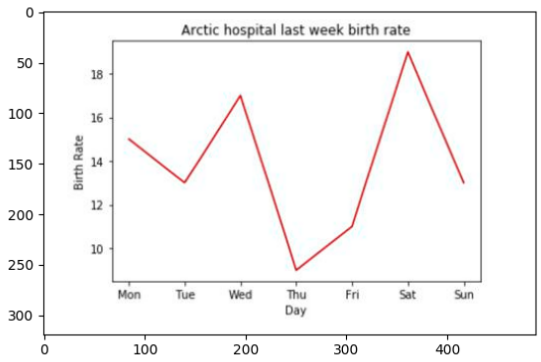
The loss is computed by taking the cross-entropy between the model's predicted token scores (logits) and the ground truth labels. The logits represent the raw scores or log-probabilities over the vocabulary at each position in the sequence, while the labels contain the target tokens the model should predict. Cross-entropy measures how close the model's predicted distribution is to the true distribution.

5. Experiments

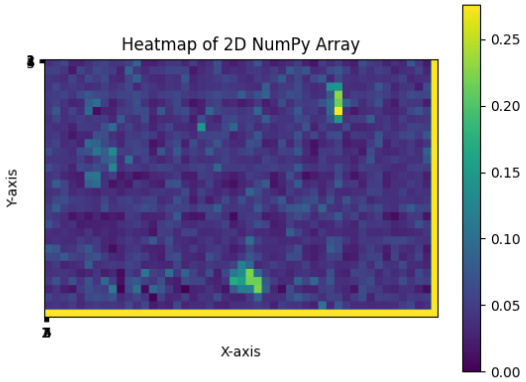
Our model development is done on google colab cpu nodes, and test run is done on a single NVIDIA A100 80G GPU. the size of the parameter is reason to chose for a 80G GPU. although the Matcha network is a 2.8 Million parameter network, which converts to 1.12GB with everything running fp32. All the intermediate values required for back propagation easily makes the model occupy 12GB data with batch size of 2, and grows to over 40GBs with a batch size of 8.

5.1. Saliency Map Analysis for Matcha Model

Figure 5 presents a line chart image and its corresponding saliency map generated using the pretrained Matcha model. The saliency map illustrates the difference in loss values between the occluded and non-occluded regions of the input image. The model successfully captures the dip on Thursday and the spike on Saturday, as well as the birth rate values of 14 and 16 on the y-axis and some "Day" information on the x-axis. However, the model fails to accurately capture values 10, 12, Saturday, and Sunday. Ideally, all



(a) A Line Chart Image from Dataset



(b) SaliencyMap of the Line chart Image

Figure 5: a line chart image and its corresponding saliency map on Matcha pretrained Model.

x-axis and y-axis values should be precisely captured since each value has a corresponding datapoint in close proximity. Out of the total 5 values on the y-axis, 7 values on the x-axis, and 7 datapoints, the Vision Transformer (ViT) captures at most 7 of them. The remaining values are either not captured or have minimal influence on the model's output. This observation highlights the inherent limitation of ViT in capturing a sufficient amount of detailed information that is spatially distributed in the input image.

5.2. Analysis of SimCLR on ViT Encoder

During the experimentation with the SimCLR framework on the ViT encoder, several challenges were encountered in guiding the model to learn from the augmented dataset. One notable issue was that even when the contrastive loss decreased, the model's performance deteriorated.

This phenomenon could be attributed to the limited batch size that our GPU could accommodate. Despite the encoder portion containing only approximately 1.5 million parameters, its output has a dimension of 1000x768. To compress this output to a manageable dimension for loss calculation, a linear layer with a weight matrix of size 1000*768 to 512 was employed, resulting in an additional 393 million parameters in the forward pass. Given the memory constraints of an 80GB GPU, a batch of 4 pairs of augmented data (i.e., 8 inputs) would require 6.4GB of memory (0.4G * 2Byte * 8), leaving limited space for the weights and gradients of the encoder network. Consequently, the model could only be trained with a batch size of 4 pairs of data. Limited batch size means limited number of negative samples. It would not provide a stable and consistent reference for the model to learn from, as their representations also change quickly over time.

Furthermore, the SimCLR architecture has an inherent instability issue due to the use of an unstable model for the negative pairs. Here the same encoder is used for all positive and negative samples, meaning a different encoder is used to generate the other negative samples. The inconsistency in the encoded representations can make it challenging for the model to learn stable and meaningful representations of the input data. We should consider using the MOCO work[4] instead, where a slowly moving average of the query encoder were used for the negative samples to avoid jitter or even none-convergence behavior. It is also worth noting that self-supervised learning might face fundamental challenges in chart-to-table conversion tasks. While rotation and scaling transformations might be easily interpretable for humans, they pose significant difficulties for the model to learn since it needs to capture the relationships between different image patches.

5.3. Analysis of Swin Transformer as Encoder

The initial experiments involving the replacement of the Vision Transformer (ViT) with a Swin Transformer backbone in the chart-to-text conversion task yielded a significantly high loss value during the first 10 epochs of training. This suboptimal performance can be attributed to several factors. Firstly, the pretrained Swin Transformer model was originally trained on natural image datasets, such as COCO, which exhibit substantially different visual characteristics and patterns compared to charts and graphs. Charts are composed of distinct elements, including lines, key points, and numerical values along the axes, which are not com-

monly found in natural images. Consequently, the Swin Transformer requires extensive fine-tuning to adapt to these novel visual features, necessitating a considerable number of training iterations to effectively capture and represent the unique properties of charts.

Secondly, to the best of our knowledge, the application of Swin Transformers to chart-to-text conversion tasks is a novel research direction that has not been extensively explored in the existing literature. As a result, there is a possibility that the architectural properties of the Swin Transformer may not be inherently well-suited for this specific task. However, to thoroughly investigate this hypothesis and determine the effectiveness of the Swin Transformer in capturing the numerical values and relationships between image patches, it is essential to analyze the saliency maps generated by the model after it has converged. The saliency maps will provide valuable insights into the model's attention distribution and its ability to focus on the relevant chart elements, such as numbers and key points, enabling a comprehensive assessment of the Swin Transformer's suitability for chart-to-text conversion.

Table 1 shows our preliminary results on a subset of the Chart-to-Text dataset:

Model	Cross-Entropy Loss
Matcha (ViT)	0.8922
Matcha (ViT) + SimCLR	0.92
Matcha (Swin)	13

Table 1: Training Loss results on Chart-to-Text dataset. Cross-entropy loss is computed between the model's predicted token scores and ground truth labels. Lower is better.

Future work will include 1) training the new swin-transformer network thoroughly, 2) conducting neural architecture search to optimize the Swin Transformer for chart data, and 3) evaluating our approach on additional datasets like ChartQA [13] and PlotQA [14] to assess generalization. Integrating with downstream RAG systems is another important direction to demonstrate the practical value of our work.

6. Conclusion

In this paper, we explored techniques for improving chart-to-text conversion in the Matcha framework through self-supervised learning and architecture search. Our experiments with Swin Transformers show promising results in terms of cross-entropy loss reduction and detail capture. However, due to limited time and computation resource constraints we are not able to improve on the final results.

This effort should further extends to work on better designed architectures for chart to text conversion tasks, neural architecture search, and the integration of chart understanding into downstream retrieval-augmented generation systems. By enabling machines to reliably understand the wealth of information stored in charts and graphs, this work steps toward on making these resources more accessible and useful for a variety of real-world applications.

References

[1] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[3] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[4] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[5] https://www.kaggle.com/competitions/benetech-making-graphs accessible/overview.

[6] H. Kato, M. Nakazawa, H.-K. Yang, M. Chen, and B. Stenger. Parsing line chart images using linear programming. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2109–2118, 2022.

[7] K. Lee, M. Joshi, I. R. Turc, H. Hu, F. Liu, J. M. Eisenschlos, U. Khandelwal, P. Shaw, M.-W. Chang, and K. Toutanova. Pix2struct: Screenshot parsing as pretraining for visual language understanding. In *International Conference on Machine Learning*, pages 18893–18912. PMLR, 2023.

[8] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[9] F. Liu, J. M. Eisenschlos, F. Piccinno, S. Krichene, C. Pang, K. Lee, M. Joshi, W. Chen, N. Collier, and Y. Altun. Deplot: One-shot visual language reasoning by plot-to-table translation. *arXiv preprint arXiv:2212.10505*, 2022.

[10] F. Liu, F. Piccinno, S. Krichene, C. Pang, K. Lee, M. Joshi, Y. Altun, N. Collier, and J. M. Eisenschlos. Matcha: Enhancing visual language pretraining with math reasoning and chart derendering. *arXiv preprint arXiv:2212.09662*, 2022.

[11] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

[12] J. Luo, Z. Li, J. Wang, and C.-Y. Lin. Chartocr: Data extraction from charts images via a deep hybrid framework. In

Proceedings of the IEEE/CVF winter conference on applications of computer vision, pages 1917–1925, 2021.

[13] A. Masry, D. X. Long, J. Q. Tan, S. Joty, and E. Hoque. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. *arXiv preprint arXiv:2203.10244*, 2022.

[14] N. Methani, P. Ganguly, M. M. Khapra, and P. Kumar. Plotqa: Reasoning over scientific plots. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1527–1536, 2020.

[15] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[17] J. Wang, Z. Yang, X. Hu, L. Li, K. Lin, Z. Gan, Z. Liu, C. Liu, and L. Wang. Git: A generative image-to-text transformer for vision and language. *arXiv preprint arXiv:2205.14100*, 2022.