Multimodal Models for Understanding and Generating Technical Diagrams

Manish Gupta

Jul 2026

Generating & Understanding Technical Visuals

- Manually tedious, time-consuming, and error-prone: Authoring architecture diagrams and writing patent-figure descriptions. Slows down software documentation, scientific communication, and patent drafting.
- Technical visuals ≠ natural images: Architecture diagrams and patent figures demand strict semantic alignment, structural coherence, and fine-grained precision.
- Labeled data is the core barrier.
- Multimodal knowledge agents
  - Structure-aware encoders
  - Domain-adapted LLMs
  - Novel graph-level evaluation
- Broader goal: agents that retrieve, reason, and generate knowledge across modalities to solve real-world problems

Text2Arch: A Dataset For Generating Scientific Architecture Diagrams From Natural Language Descriptions

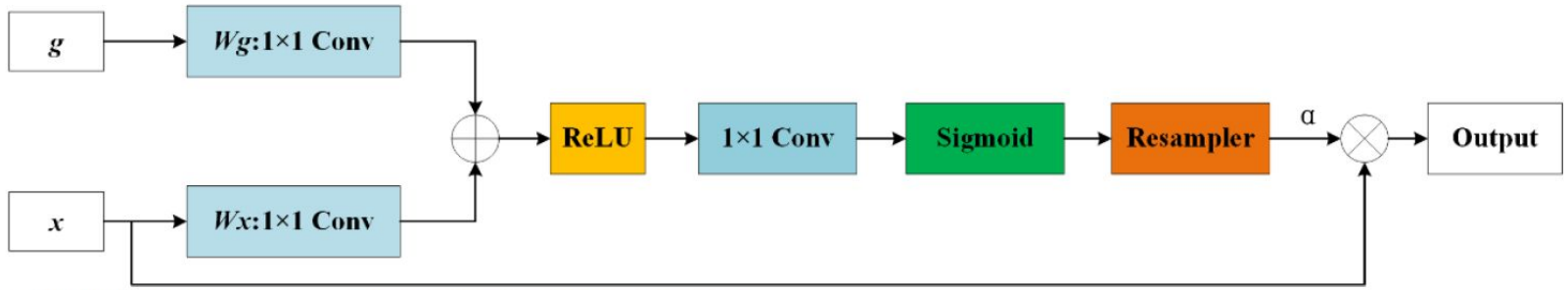
Shivank Garg
IIT Roorkee, India
shivank_g@mfs.iitr.ac.in

Sankalp Mittal
Google, India
sankalpmittal123@gmail.com

Manish Gupta
Microsoft, India
gmanish@microsoft.com

ICLR 2026

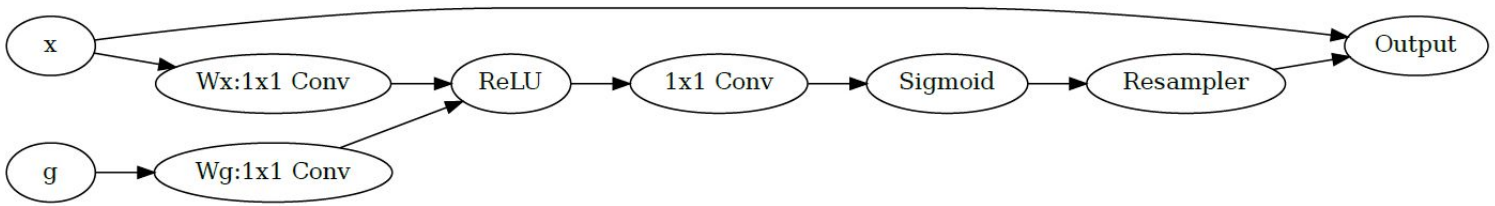
Text2Arch



(a) **Original figure**: Fig. 5 from https://arxiv.org/pdf/2110.01014v1

(b) **Figure description**: The image is an architecture diagram depicting an attention gate mechanism. The diagram starts with two inputs: g (decoding matrix) and x (encoding matrix). Both inputs undergo a 1×1 convolution operation separately, denoted as Wg:1×1 Conv and Wx:1×1 Conv respectively. The outputs of these convolutions are then added together at corresponding points. This sum is then passed through a ReLU activation function, followed by another 1×1 convolution operation. The resulting output is then processed through a Sigmoid function to generate an attention coefficient, denoted as α. This attention coefficient is used to resample the input encoding matrix x through a Resampler module. Finally, the resampled output is multiplied by the attention coefficient α to produce the final output of the attention gate.

(c) **DOT code**: digraph {
0 [label="Sigmoid"]; 1 [label="Output"]; 2 [label="1x1 Conv"]; 3 [label="Resampler"]; 4 [label="Wx:1x1 Conv"]; 5 [label="g"]; 6 [label="ReLU"]; 7 [label="x"]; 8 [label="Wg:1x1 Conv"];
5 → 8; 7 → 4; 8 → 6; 4 → 6; 6 → 2; 2 → 0; 0 → 3; 3 → 1; 7 → 1;
}



(d) **Generated figure using TEXT2ARCH.**

Why Convert Text → Architecture Diagrams?

- Scientific systems are often described only in text, which leads to:
  - Hard to understand complex pipelines
  - Ambiguous or incomplete visual representation
  - Manual diagram creation is time-consuming
- Goal
  - Automatically generate accurate architecture diagrams from natural language descriptions
- Potential Applications
  - Scientific papers
  - Software architecture documentation
  - AI system design
  - Education and learning material

Text2Arch Problem Definition

- Input: Natural language description of a system architecture
- Output: A structured architecture diagram
- Our approach
  - Text description
  - Generate DOT graph code
  - Render diagram using Graphviz
- Advantages of intermediate DOT code
  - Structured
  - Editable
  - Deterministic rendering
  - Easier evaluation

Challenges for Text2Arch

- Unlike natural image generation
  - Strict semantic alignment required
  - Precise structure of nodes and edges
  - Correct labels and flow direction
  - Long technical descriptions
- Existing text-to-image models fail because:
  - limited input context
  - poor text rendering
  - weak structural reasoning

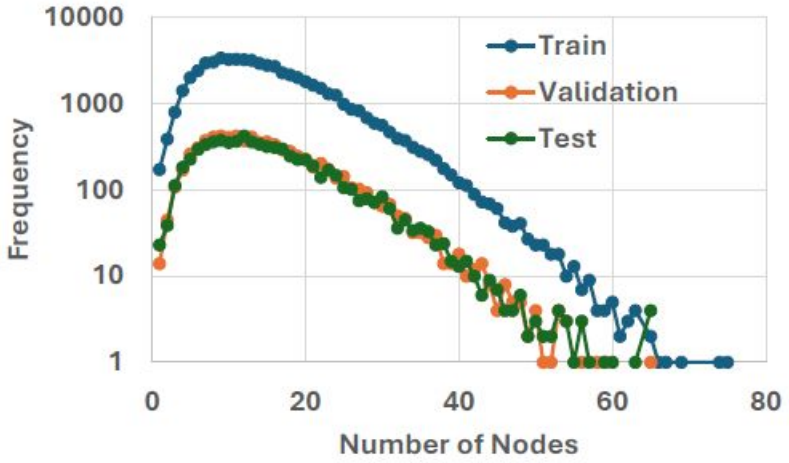
Key Contributions

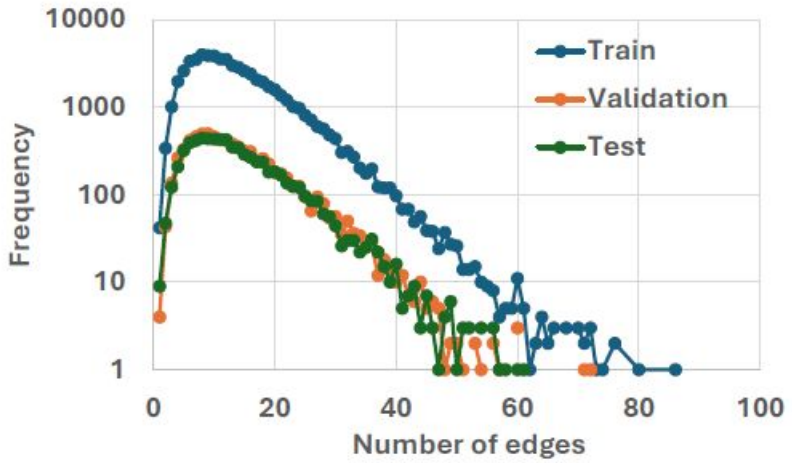
- Introduce Text2Arch task: Generate architecture diagrams from text.
- Release Text2Arch dataset
  - 75K samples
  - text description
  - architecture image
  - DOT code
- Fine-tune open-source LLMs: Llama-3, Qwen-2, DeepSeek
- Propose graph-based evaluation metrics to measure structural correctness.

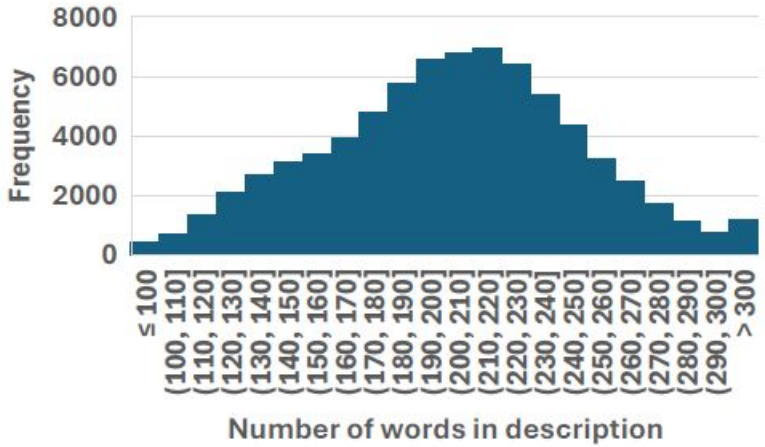
Text2Arch Dataset

- Total samples: 75,127
  - Train: 60,519
  - Validation: 7,565
  - Test: 7,043
- Manual annotation set: 99 samples.
- Each sample contains
  - Architecture diagram image
  - Clean textual description
  - DOT graph representation

- Average statistics:
  - 15.2 nodes; 13.9 edges
  - ~203 words per description
  - Around 54.3% samples are easy, 32.4% are medium and the rest are of hard complexity.

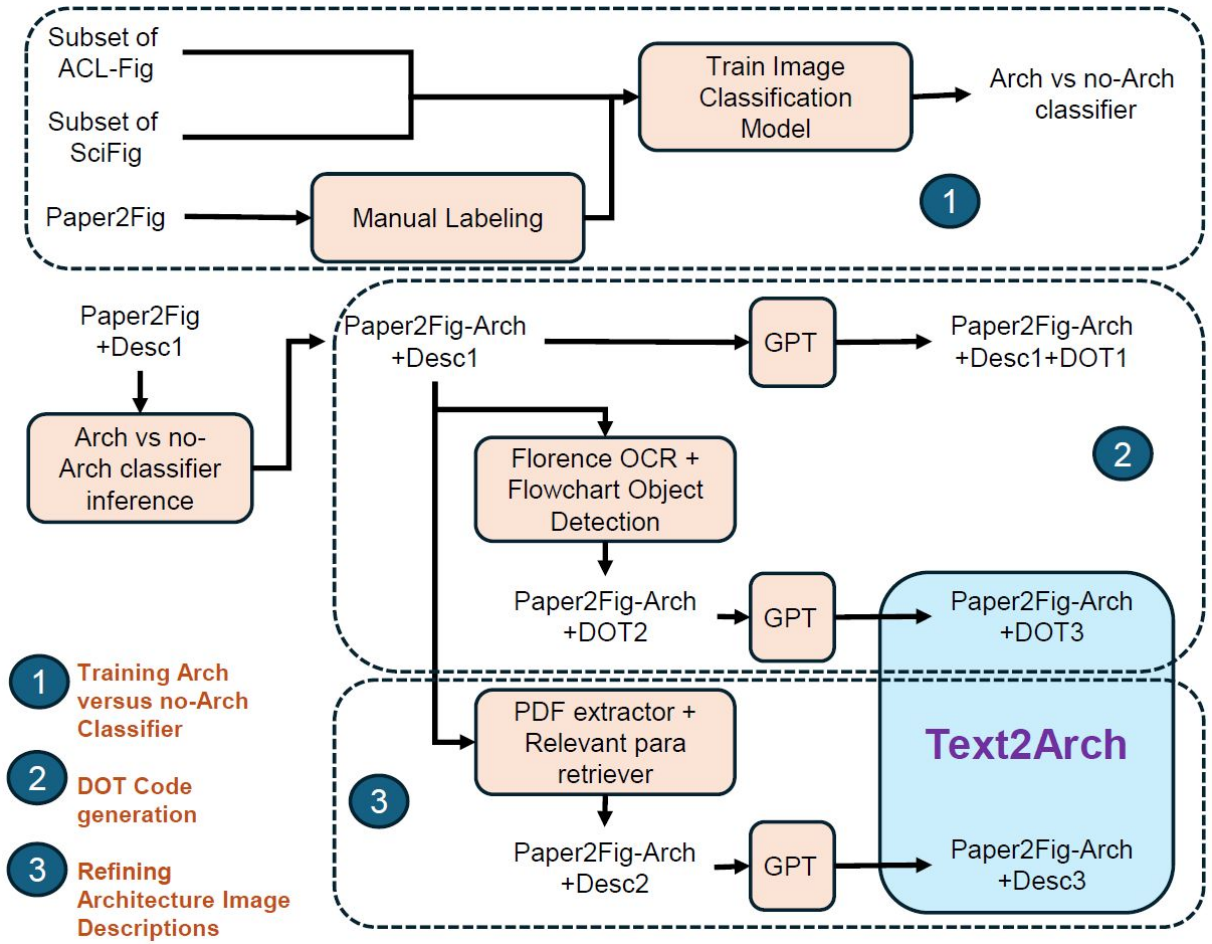






Dataset Curation

- Architecture Diagram Detection
  - Train classifier: Arch vs Non-Arch diagrams
  - Models tested: CLIP, ViT, BEiT, ResNet
  - Best model: CLIP (83.45%)
- DOT Code Generation to obtain structured graph representation
  - DOT1: Generated by GPT-4o
  - DOT2: Extract nodes and edges using OCR (Florence2) and Object detection
  - DOT3: GPT refinement of DOT2
  - Result: significantly higher graph accuracy
- Description Refinement
  - Original dataset contained noisy text.
  - We improve descriptions via:
    - Extract relevant paragraphs from paper
    - TF-IDF similarity filtering
    - GPT refinement
  - Final output: Desc3
  - Human evaluation: Desc3 preferred >90% of the time.



Models evaluated

- Baselines
  - DiagramAgent
  - GPT-4o zero-shot
- Few-shot ICL
  - Llama-3-8B
  - Qwen-2-7B
  - DeepSeek-7B
- Fine-tuned models
- Same models trained on Text2Arch dataset.

Evaluation Metrics

- Text Similarity Metrics
  - ROUGE-L
  - CodeBLEU
  - Edit Distance
  - chrF
- Graph Structure Metrics
  - Node Precision / Recall / F1
  - Edge Precision / Recall
  - Jaccard similarity
  - Node PR-AUC
- These measure semantic + structural fidelity.

Results

		Text Metrics				Graph Metrics								
		ROUGE-L	Code BLEU	Edit Distance	chrF	Node Prec	Node Recall	Node F1	Node PR-AUC	Edge Prec	Edge Recall	Edge F1	Edge PR-AUC	Jaccard Sim.
	DiagramAgent	42.2	31.0	680	48.2	60.7	56.5	55.1	20.9	31.7	22.6	24.8	18.0	17.8
	GPT	30.8	17.7	730	42.9	**71.6**	56.5	60.7	**27.4**	**56.3**	**39.5**	**44.6**	**31.6**	**36.1**
Few-shot ICL	Llama-3-8B	34.9	21.5	709	41.0	69.6	56.8	59.7	24.2	41.5	29.0	32.5	22.0	24.6
Few-shot ICL	Qwen2-7B	27.4	19.4	811	30.4	64.9	48.1	52.0	19.8	32.8	21.2	24.3	15.9	17.2
Few-shot ICL	DeepSeek-7B	30.4	21.4	1079	31.4	54.1	41.3	43.5	17.4	25.0	13.6	16.6	13.0	11.5
Fine-tuned	Llama-3-8B	28.2	27.8	956	39.8	27.8	45.3	31.9	7.0	22.9	10.2	13.2	9.1	8.4
Fine-tuned	Qwen2-7B	35.0	30.7	975	45.3	33.3	49.8	36.8	8.1	21.8	10.8	13.7	8.8	8.8
Fine-tuned	DeepSeek-7B	**46.8**	**34.5**	**608**	**55.7**	66.2	**69.6**	**65.7**	21.5	46.4	34.2	38.0	23.7	28.6

test set

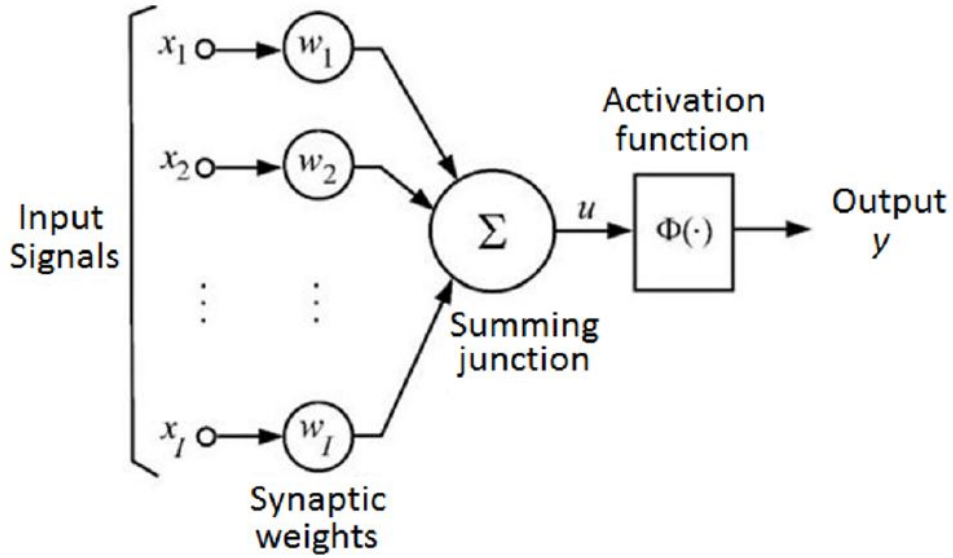
		Text Metrics				Graph Metrics								
		ROUGE-L	Code BLEU	Edit Distance	chrF	Node Prec	Node Recall	Node F1	Node PR-AUC	Edge Prec	Edge Recall	Edge F1	Edge PR-AUC	Jaccard Sim.
	DiagramAgent	49.1	40.9	959	55.3	55.0	59.8	54.3	12.4	32.0	22.8	25.3	16.9	17.8
	GPT	28.2	16.3	790	43.4	**69.7**	60.1	63.0	**28.0**	55.9	40.4	46.2	30.9	37.8
Few-shot ICL	Llama-3-8B	37.3	23.1	474	43.6	62.4	52.9	54.1	17.0	42.4	28.5	32.5	17.4	25.7
Few-shot ICL	Qwen2-7B	30.1	22.0	562	32.0	54.9	50.7	48.9	13.2	37.1	21.7	25.5	14.9	18.1
Few-shot ICL	DeepSeek-7B	36.9	28.6	872	35.8	52.1	42.2	42.9	13.8	27.5	17.5	20.4	13.9	15.4
Fine-tuned	Llama-3-8B	30.9	46.0	1024	44.0	21.3	46.4	27.5	4.9	27.7	11.9	15.2	14.3	10.1
Fine-tuned	Qwen2-7B	40.9	46.3	891	53.1	35.3	57.1	40.2	9.4	19.4	11.5	14.0	11.8	9.3
Fine-tuned	DeepSeek-7B	**55.2**	**49.3**	**407**	**66.6**	66.1	**78.1**	**69.4**	27.4	**59.4**	**44.6**	**49.1**	**35.1**	**39.8**

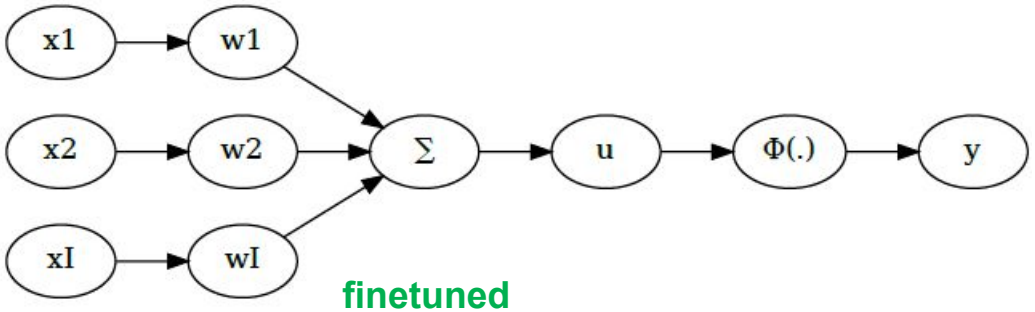
manual annotation set

- Fine-tuned models outperform few-shot models and GPT-4o.
- Best model: Fine-tuned DeepSeek-7B
- Close to GPT-4o perf while being open and lightweight.

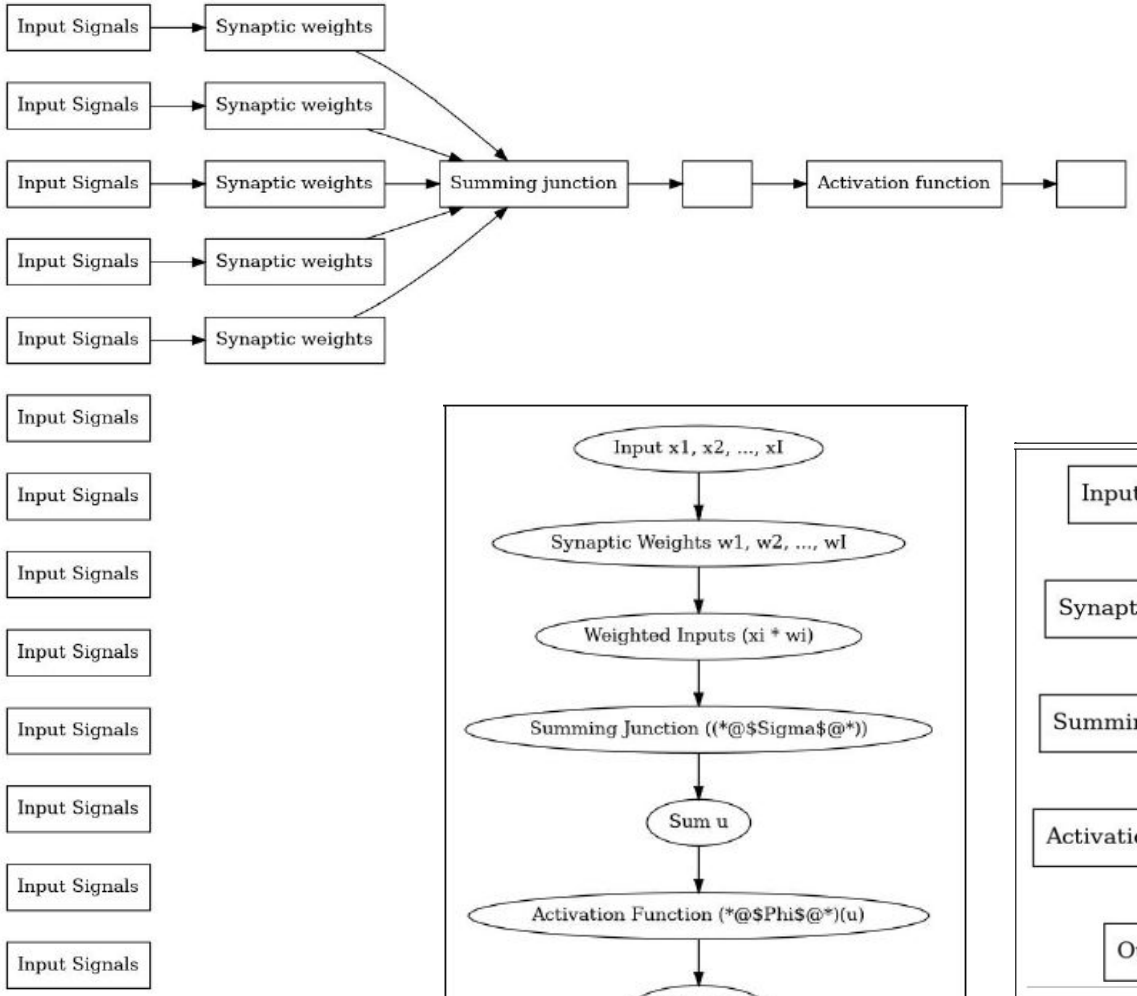
An Example of Text2Arch Results

(a) **Original figure** (Fig. 3 from https://arxiv.org/pdf/1701.07543v1)



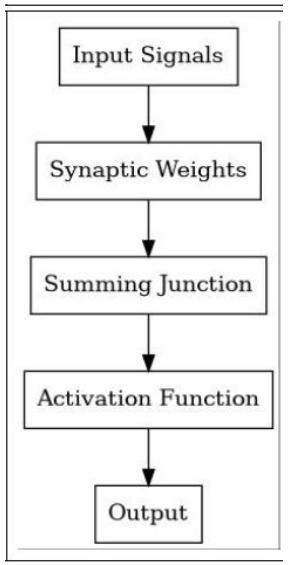


finetuned DeepSeek-7B



DiagramAgent

GPT



fewShot DeepSeek

Takeaways

- Introduced Text2Arch, the first large dataset for architecture diagram generation.
- Demonstrated LLM-based structured diagram synthesis.
- Fine-tuned DeepSeek-7B achieves best performance.
- Graph-based metrics enable rigorous structural evaluation.
- Dataset, models, and code are publicly released.
- Future Work
  - Multimodal training (text + image)
  - Direct diagram generation without intermediate code
  - Interactive diagram editing
  - Application to software engineering tools

PatentLMM: Large Multimodal Model for Generating Descriptions for Patent Figures

Shreya Shukla
IIT Jodhpur, India
shukla.12@iitj.ac.in

Nakul Sharma
IIT Jodhpur, India
sharma.86@iitj.ac.in

Manish Gupta
Microsoft, India
gmanish@microsoft.com

Anand Mishra
IIT Jodhpur, India
mishra@iitj.ac.in

AAAI 2025

Generating descriptions for patent images

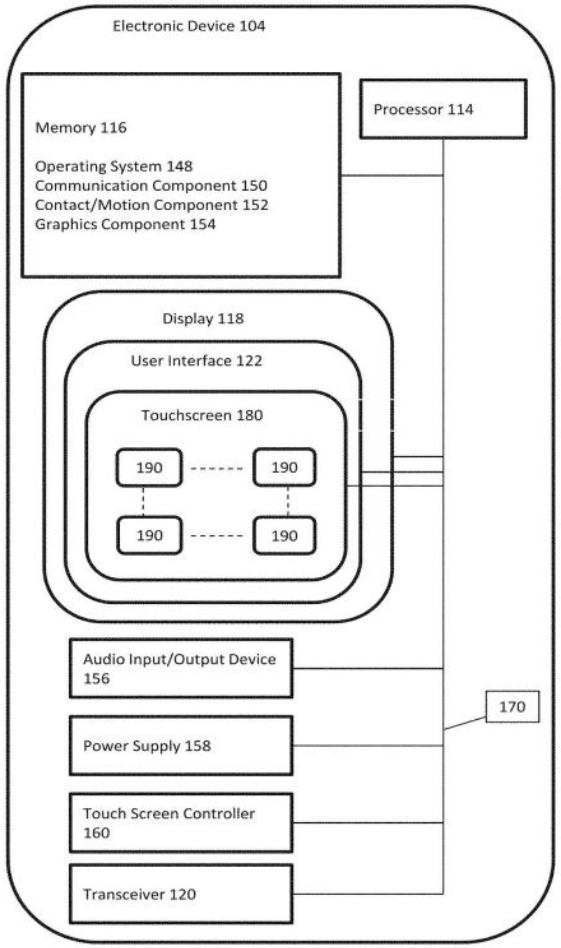


FIGURE 1

Generated Brief and Detailed Descriptions	Ground Truth Brief and Detailed Descriptions
FIG. 1 illustrates a mobile smartphone in accordance with an aspect of the invention.	FIG. 1 shows the details of an exemplary electronic device in accordance with aspects of the invention.
Referring now to FIG. 1, an exemplary electronic device 104 is illustrated. It should be understood that the electronic device 104 may be any one of a number of electronic devices including, but not limited to, a cell phone, a smart phone, a personal digital assistant, a laptop computer, a desktop computer, a netbook computer, a server computer, etc. The electronic device 104 includes a processor 114, a memory 116, a display 118, a user interface 122, a transceiver 120, and an optional touch screen 160. The processor 114 is configured to execute instructions received from the memory 116, and is, for example, a general purpose processor, a field programmable gate array, or any other suitable processor. The processor 114 is configured to execute instructions received from the memory 116, including instructions for displaying a user interface on the display 118. The processor 114 may also be configured to execute instructions received from the transceiver 120, including instructions for receiving data from the transceiver 120 and instructions for transmitting data from the transceiver 120.The memory 116 is configured to store data received from the processor 114 and instructions to be executed by the processor 114. The memory 116 may be, for example, a random access memory device, a solid state memory device, or any other suitable storage device. The display 118 is configured to visually display data received from the processor 114. The display 118 may be, for example, a liquid crystal display, an organic light emitting diode display, a cathode ray tube display, or any other suitable display. The user interface 122 is configured to allow a user to interact with the electronic device 104. The user interface 122 may include, for example, a keypad, a touchscreen 160, a microphone, a speaker, etc. The transceiver 120 is configured to send and receive…	FIG. 1 shows the details of an exemplary electronic device in accordance with aspects of the invention. The electronic device 104 includes a processor 114, memory 116, display 118, user interface 122, and the like. The processor 114 may be a central processing unit configured to execute instructions including instructions related to software programs. The display 118 may be a liquid crystal display having a backlight to illuminate the various color liquid crystals to provide a colorful display. The user interface 122 may be any type of physical input having buttons and further may be implemented as a touchscreen 180.The electronic device 104 may further include in the memory 116, an operating system 148, a communication component 150, a contact/motion component 152, a graphics component 154, and the like. The operating system 148 together with the various components providing software functionality for each of the components of the electronic device 104.The memory 116 may include a high-speed random-access memory. Also, the memory 116 may be a non-volatile memory, such as magnetic fixed disk storage, flash memory or the like. These various components may be connected through various communication lines including a data bus 170.Additionally, the electronic device 104 may include an audio input/output device 156. The audio input/output device 156 may include speakers, speaker outputs, and in the like, providing sound output; and may include microphones, microphone inputs, and the like, for receiving sound inputs. The audio input/output device 156 may include and analog to digital converter and a digital to audio converter for audio input and output functions respectively.When implemented as a wireless device, the electronic device 104 may include a transceiver 120 and the like. The electronic device 104 may provide radio and…

PatentDesc dataset

- 355K patent images + brief and detailed descriptions from 90K patents.
  - Amazon, Microsoft, LinkedIn, Google, Yahoo, etc.
- Vision Element Detection
  - FasterRCNN trained on 350 images
  - Node, Arrow, Text, Node labels, Figure labels

Table 1: PATENTDESC-355K: Dataset Statistics.

	Train	Validation	Test
Number of Images	320,717	17,286	17,336
Avg. number of tokens in brief descriptions	34.37	34.28	34.30
Avg. number of tokens in detailed descriptions	1,677.85	1,676.71	1,697.16
Number of Unique Patents	50,448	8,027	7,964
Avg. number of images per patent	6.36	2.15	2.18

PatentDesc dataset samples

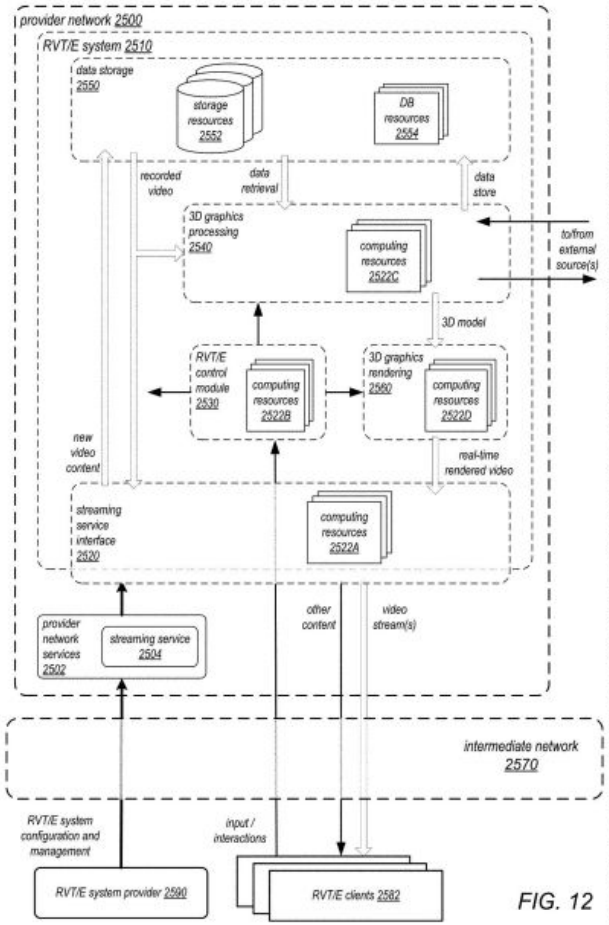


FIG. 12

Brief - FIG. 12 illustrates an example network-based RVE environment, according to at least some embodiments.

Detailed - Referring to FIG. 12, in at least some embodiments, data including but not limited to video content may be streamed from the streaming service interface 2520 to the RVT/E client 2582 according to a streaming protocol. In at least some embodiments, data including but not limited to user input and interaction may be sent to the streaming service interface 2520 from the RVT/E client 2582 according to the streaming protocol. In at least some embodiments, the streaming service interface 2520 may receive video content (e.g., rendered video frames) from a video playback module (not shown) and/or from a rendering 2560 module, package the video content according to the streaming protocol, and stream the video according to the protocol to…

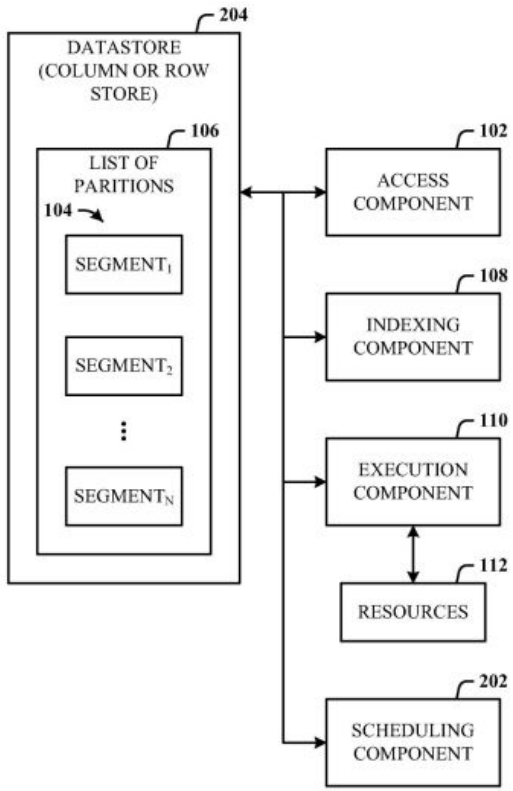


FIG. 2

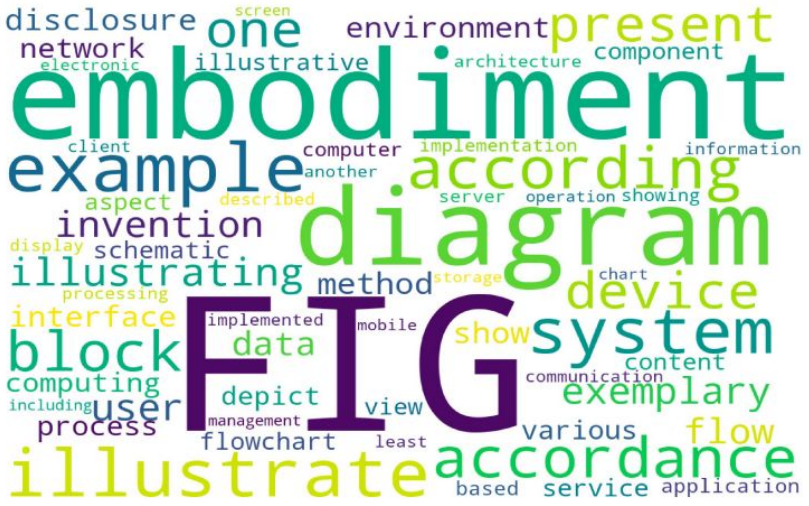
200

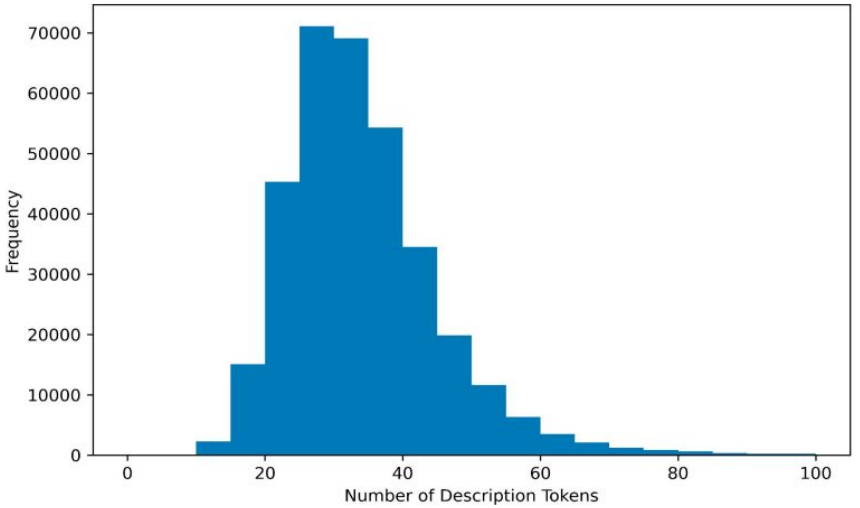
Brief - FIG. 2 illustrates an alternative embodiment of an indexing system that further includes a scheduling component for scheduling work to the resources.

Detailed - FIG. 2 illustrates an alternative embodiment of an indexing system 200 that further includes a scheduling component 202 for scheduling work to the resources. The scheduling component 202 assigns segments to worker threads according to load balancing across on the resources. The load balancing can be based on segment size and the number of the worker threads. The worker threads can work on the same or different segments concurrently.Put another way, the system 200 comprises the access component 102 that accesses a datastore 204 to obtain the list of partitions 106 and corresponding segments 104 as part of processing a query. The datastore 204 can be a column store and/or a row…

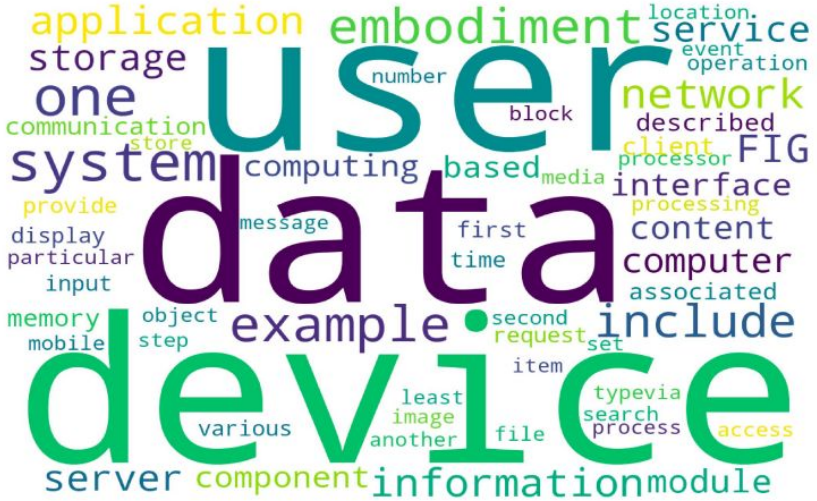
PatentDesc dataset

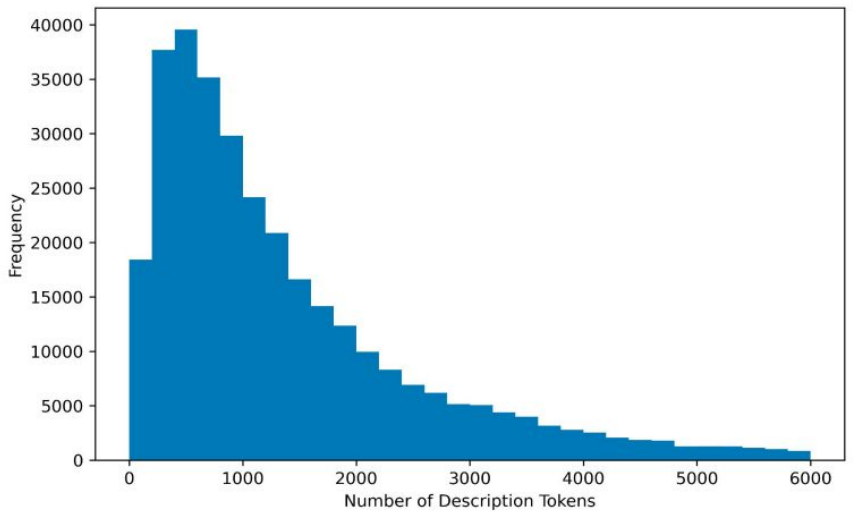
- Brief descriptions
  - ~34 tokens





- Detailed descriptions
  - ~1680 tokens





Vision Element Detection Example

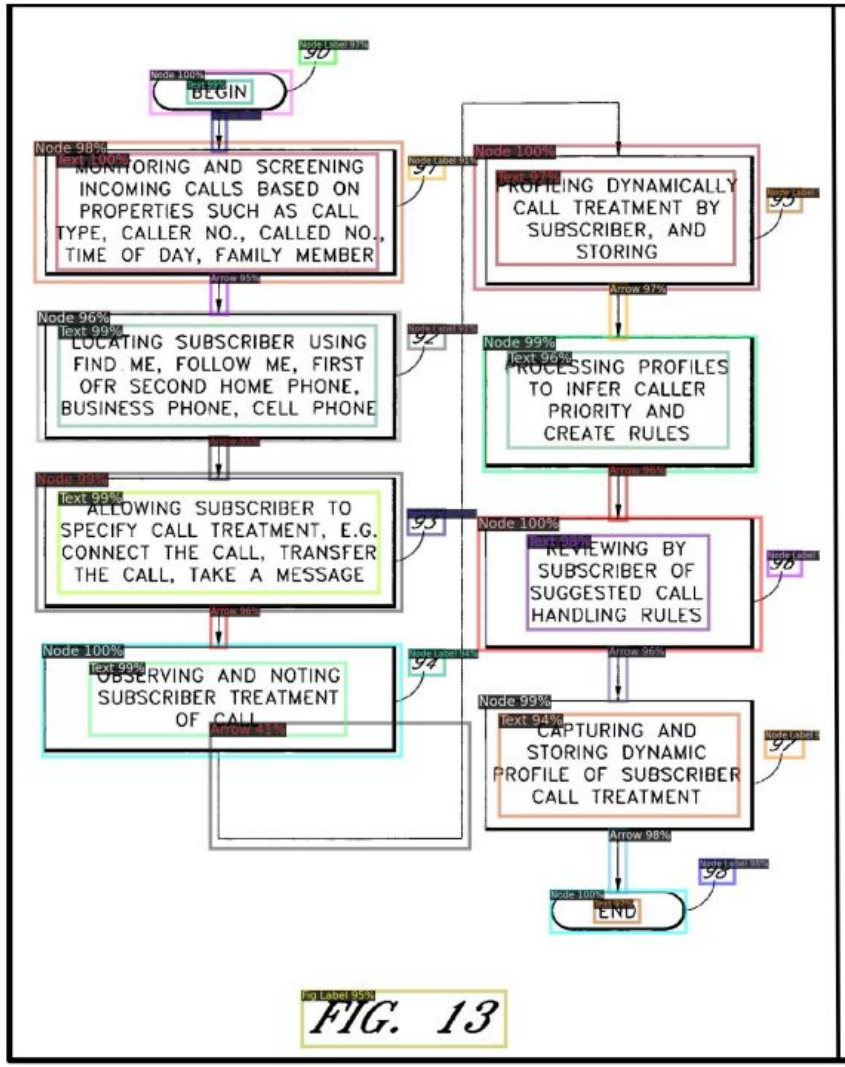


FIG. 13

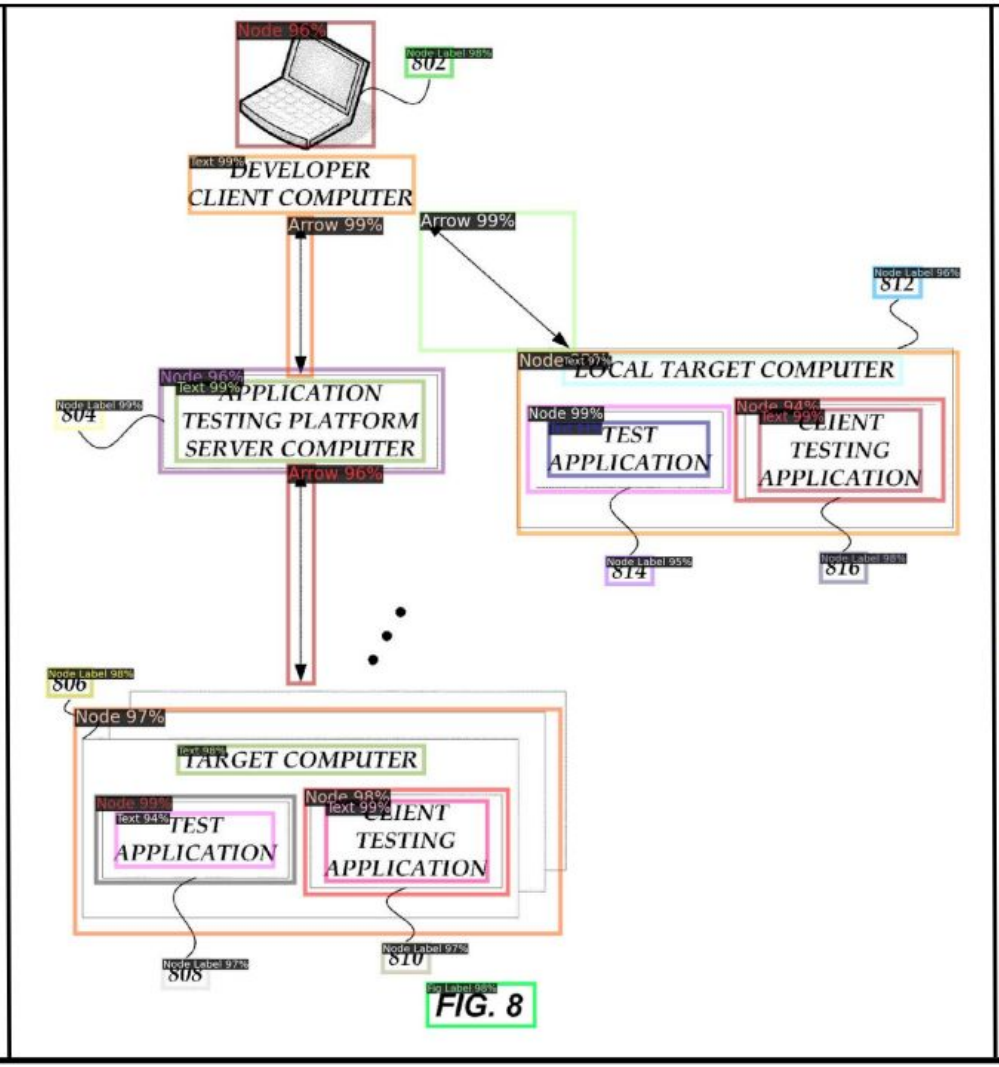


FIG. 8

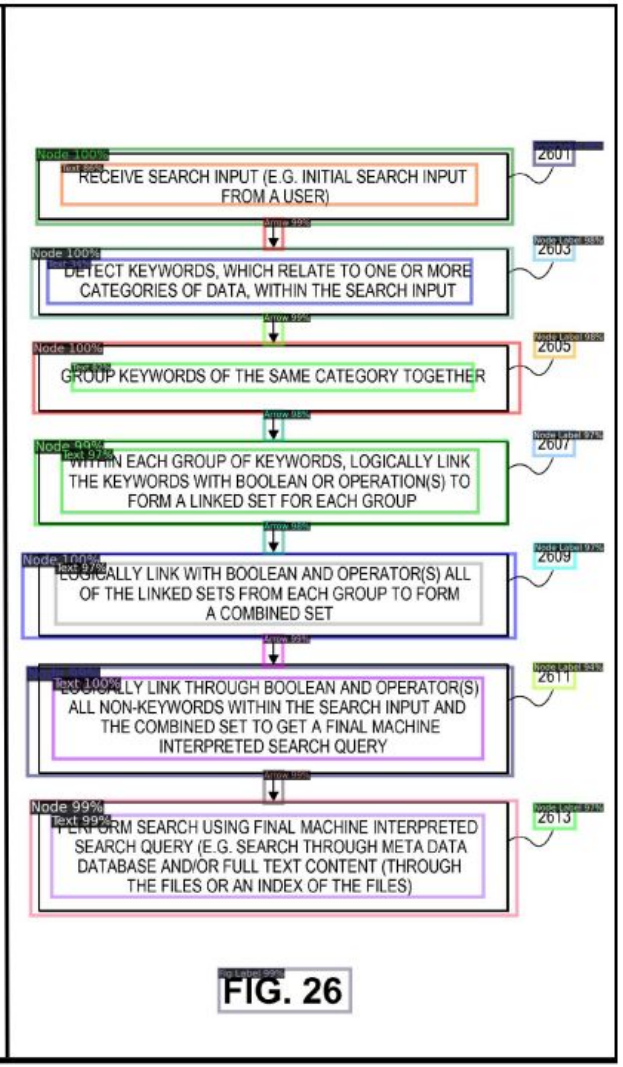
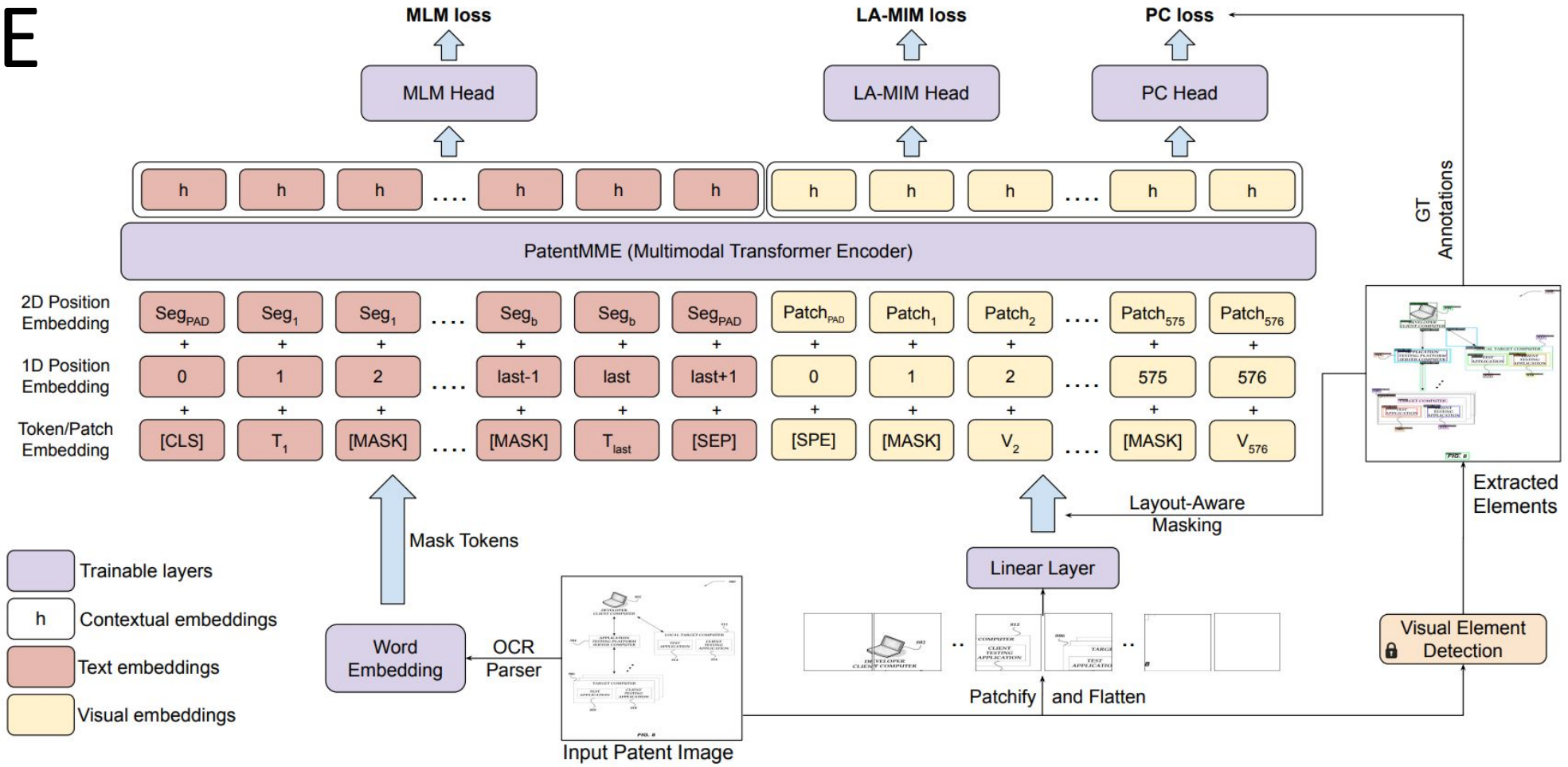


FIG. 26

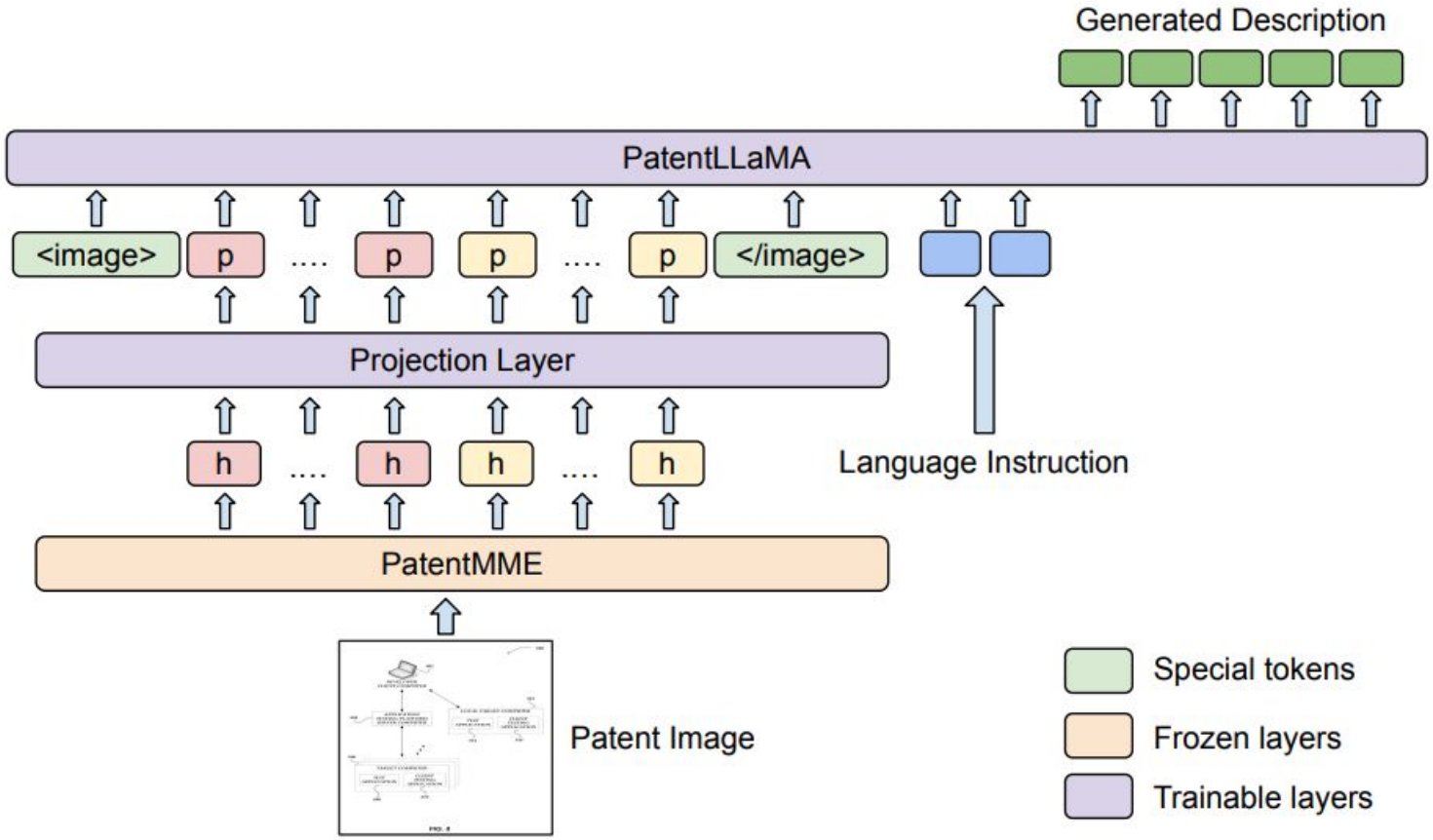
PatentMME



- Pre-train LayoutLMv3 on ~900K HUPD patent images

- MLM (unmask 30% OCR tokens)
- LA-MIM (layout aware MIM for 40% tokens with visual elements)
- Patch Classification (binary loss over 5 classes)

PatentLMM



- PatentLLaMA: Continue pre-train LLaMA2 7B with added LoRA adapters on the HUPD (Harvard USPTO Dataset) patent text for domain adaption.

- PatentLMM is trained in two stages:
  - Alignment using projection layer
  - Finetuning projection layer + PatentLLaMA w/ LoRA

How does PatentLMM perform compared to other models on patent description generation?

Setup	Method	# Parameters	B-2	B-4	Avg. B	R-1	R-2	R-L	M	B-2	B-4	Avg. B	R-1	R-2	R-L	M
			Brief							Detailed						
Zero-shot	BLIP-2	2.7B	1.01	0.03	1.62	15.43	1.70	12.47	6.72	0.00	0.00	0.01	3.24	0.49	2.84	1.00
	TextMonkey		0.91	0.11	1.12	13.00	4.60	12.16	7.14	0.10	0.03	0.10	6.18	2.38	4.83	2.64
	PEGASUS	568M	3.18	0.13	4.20	14.68	2.33	11.46	13.24	0.86	0.04	1.12	12.26	1.96	9.47	5.18
	mPLUG-owl2	7.45	3.64	0.36	4.47	21.47	5.10	19.41	13.07	3.65	0.49	3.40	23.75	5.39	14.85	11.83
	UReader		3.54	0.35	4.50	20.90	4.56	17.85	13.45	0.05	0.01	0.06	5.15	1.54	4.49	2.04
	LLaVA	7.39B	4.52	0.24	4.71	17.59	3.27	14.63	15.74	3.65	0.37	3.36	23.75	4.63	14.71	11.69
	GPT-4V	Unknown	20.74	8.56	18.68	36.07	15.65	31.89	32.88	19.61	6.05	18.26	39.95	12.14	20.16	27.31
Finetuned	Pegasus	568M	2.44	0.14	4.03	13.86	1.55	11.52	11.62	5.80	0.41	6.33	19.28	2.24	15.27	12.11
	GIT	681M	26.95	15.33	24.78	45.28	27.17	42.29	44.27	6.33	1.18	6.23	13.66	3.17	10.87	10.68
	BLIP	252M	24.62	12.52	22.40	42.59	23.78	39.16	42.84	5.45	1.05	5.31	12.42	2.89	9.46	9.55
	MiniGPT-4	7.83B (3.15M)	30.57	17.96	28.13	43.53	25.33	40.35	43.03	11.01	2.81	10.26	28.91	6.23	15.67	16.65
	OFA	472M	33.01	21.76	31.24	54.26	37.94	51.47	44.89	15.76	7.23	14.93	33.20	13.70	22.89	21.17
	LLaVA	7.39B (341M)	36.64	25.00	34.37	48.92	32.01	45.87	48.23	20.90	11.12	19.81	36.86	15.68	24.48	24.71
	PATENTLMM	7.44B (341M)	**46.40**	**36.66**	**44.59**	**56.68**	**42.63**	**54.18**	**56.44**	**25.42**	**15.02**	**24.24**	**40.70**	**19.27**	**27.54**	**28.39**

B=BLEU, R=ROUGE, M=METEOR

PatentLMM detailed results

Pre-training	B-2	B-4	Avg. B	R-1	R-2	R-L	M
Pretrained LayoutLMv3	42.81	32.50	40.86	53.68	38.88	51.07	53.34
w/ MLM + LAMIM	45.24	35.33	43.39	55.69	41.38	53.20	55.34
w/ MLM+LAMIM+PC	**46.39**	**36.65**	**44.59**	**56.68**	**42.62**	**54.18**	**56.44**

OCR in training?	OCR in Inference?	B-2	B-4	Avg. B	R-1	R-2	R-L	M
No	No	30.32	19.17	28.30	41.61	25.21	38.95	41.46
Yes	No	11.51	2.77	9.83	24.38	7.92	21.68	22.52
Yes	Yes	**46.40**	**36.66**	**44.59**	**56.68**	**42.63**	**54.18**	**56.44**

Ablations of PatentLMM on brief description generation.

Desc.	Relevance	Accuracy	Completeness	Coherence	Fluency	Coverage
Brief	1.442	1.184	1.168	1.909	1.999	1.146
Detailed	0.899	0.779	0.761	1.146	1.848	0.748

GPT-4V evaluation on a set of 1K samples.

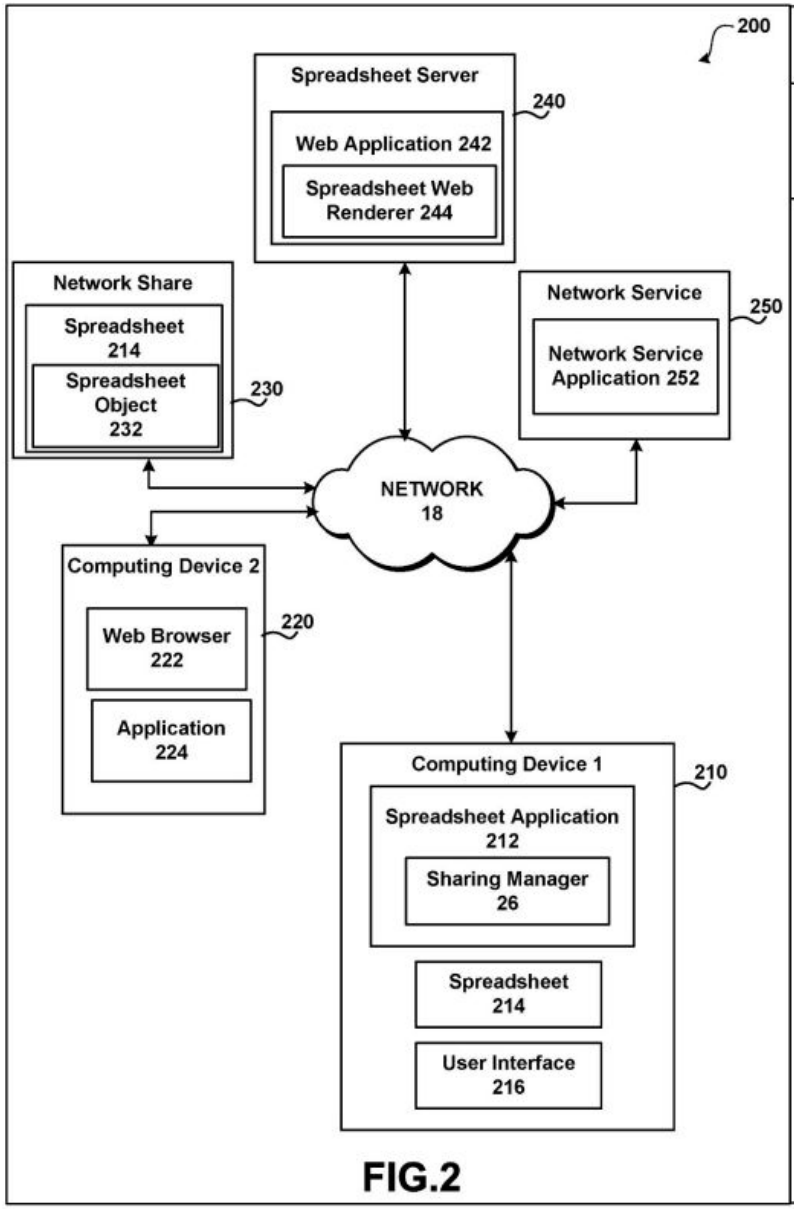


FIG.2

Generated Brief and Detailed Descriptions	Ground Truth Brief and Detailed Descriptions
FIG. 2 shows a system for using a network service for sharing spreadsheet objects;	FIG. 2 shows a system for codeless sharing of spreadsheet objects with a network service;
FIG. 2 illustrates an embodiment of a system for sharing spreadsheet content. The system 200 may include a computing device 1 210, a computing device 2 220, and a spreadsheet server 240. The computing device 1 210 may include a spreadsheet application 212, a sharing manager 26, and a user interface 216. The computing device 2 220 may include a web browser 222 and an application 224. The sharing manager 26 may be a software application that may manage sharing of spreadsheet content between the computing device 1 210 and the computing device 2 220. The sharing manager 26 may manage sharing of spreadsheet content using a web browser 222 and an application 224. The web browser 222 may be a web browser such as Internet Explorer, Mozilla Firefox, Safari, or other similar web browser. The application 224 may be a software application that may display and/or interact with spreadsheet content. The user interface 216 may be a user interface such as a graphical user interface.The computing device 2 220 may include a spreadsheet application 224, a web browser 222, and a user interface 216. The spreadsheet application 224 may be a software application that may display and/or interact with spreadsheet content. The web browser 222 may be a web browser such as Internet Explorer, Mozilla Firefox, Safari, or other similar web browser. The user interface 216 may be a user interface such as a graphical user interface.The spreadsheet server 240 may include a web application 242, a spreadsheet web renderer 244. A network service 250 may include a network service application 252. The web application 242 may be a software application that may provide spreadsheet content to the computing device 1 210 and the computing device 2 220. The spreadsheet…	FIG. 2 shows a system for codeless sharing of spreadsheet objects with a network service. As illustrated, system 200 includes computing device 1 (210), computing device 2 (220), network share 230, spreadsheet server 240 and network service 250.The computing devices may be any type of computing device that is configured to perform the operations relating to the sharing of one or more spreadsheet objects with a network service. For example, some of the computing devices may be: mobile computing devices (e.g. cellular phones, tablets, smart phones, laptops, and the like); desktop computing devices and servers. Some computing devices may be arranged to provide an online cloud based service (e.g. interacting with spreadsheets online), some may be arranged as data shares, some may be arranged in local networks, some may be arranged in networks accessible through the Internet, and the like.The computing devices are coupled through network 18. Network 18 may be many different types of networks. For example, network 18 may be an IP network, a carrier network for cellular communications, and the like. Generally, network 18 is used to transmit data between computing devices, such as computing device 1, computing device 2, network share 230, spreadsheet server 240 and network service 250.Computing device 1 includes spreadsheet application 212, spreadsheet 214 and user interface 216. As illustrated, computing device 1 is used by a user to select one or more objects to share with a network service, such as network service 250. The network service may be a social networking site, a blog, a web page hosted by a service, or some other network location.Spreadsheet 214 comprises different cells/spreadsheet objects that may be selected for sharing with a network service. The…

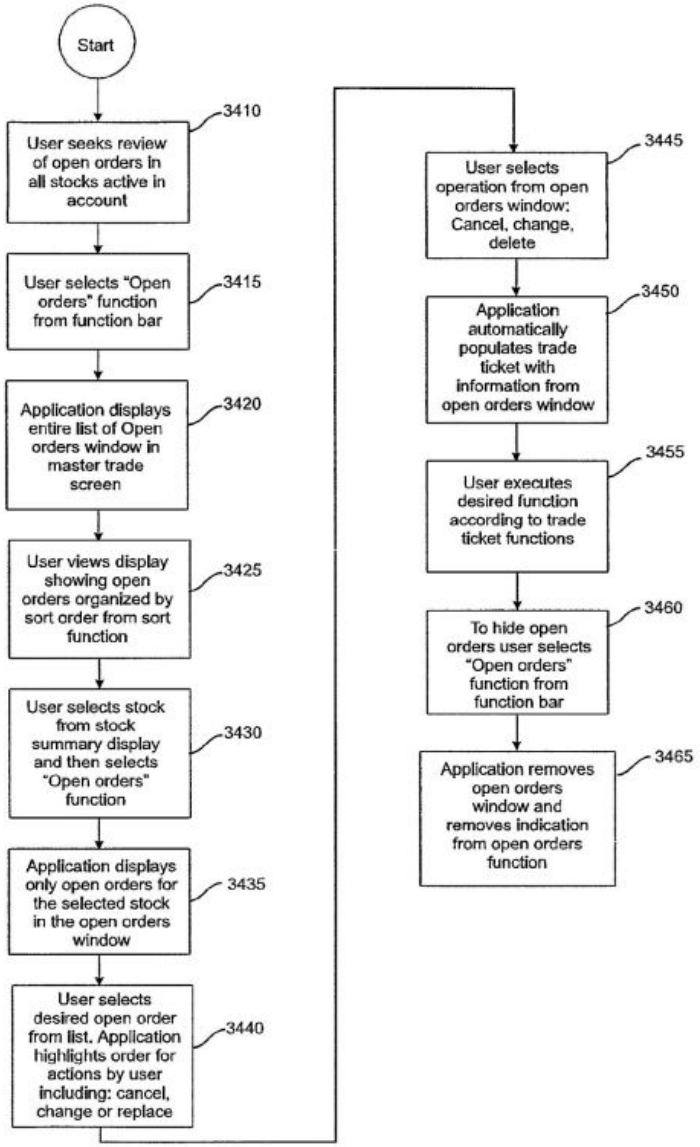


FIG. 34

Generated Brief and Detailed Descriptions	Ground Truth Brief and Detailed Descriptions
FIG. 34 illustrates a flow chart of a method for checking open orders in a stock;	FIG. 34 is a flow diagram illustrating software which enables a user to view the status of his open orders in a stock.
FIG. 34 is a block diagram that illustrates the function of the open orders window in the main trade screen, in accordance with an embodiment of the present invention. Step 3410. The user seeks to review the list of open orders in all stocks that are active in the account. The user selects the open orders function from function bar 2725. Step 3415. The application displays an entire list of the open orders window in the master trade screen. Step 3420. The user can then scroll through the open order list and can focus on a particular open order. The user can also select the open orders function from function bar 2725 to display the open orders window in the master trade screen. Step 3425. The user selects a particular stock from the stock summary display and then selects the open orders function from function bar 2725. Step 3430. The application displays only the selected open order in the open orders window. Step 3435. The user selects a desired open order from the list displayed in the open orders window. The application highlights the order for possible actions by the user, including: cancel, change, or replace. Step 3440. The user selects the desired open order from the list displayed in the open orders window. Step 3445. The user selects an operation from the open orders window. The application automatically cancels, changes, or deletes the selected open order. Step 3450. The application populates a trade ticket with information from the open order window. Step 3455. The user executes the desired function according to trade ticket functions. Step 3460. To hide the open orders window, the user selects the open orders function from function bar 2725. Step 3465. The application removes the open orders window and removes the indication of open orders from the open orders function.	The application software used in the practice of the invention is described in greater detail in conjunction with FIGS. 17-40 below. FIG. 34 is a flow diagram illustrating software which enables a user to view the status of his open orders in a stock. Initially, at step 3410, the user seeks to review his open orders in all stocks active in his account. At step 3415, the user selects the open orders function from the function buttons display (see FIGS. 5 & 15). At step 3420, the application displays the entire list of open orders in the open orders (see display in FIGS. 5 & 7) display of the master trade screen. At step 3425, the user views the display showing the open orders. The user selects a stock, at step 3430, from the stock summary display (see FIGS. 5 & 12) and selects the Open orders function. The application then, at step 3435, displays only open orders for the selected stock in the open orders display. The user selects, at step 3440, the desired open order from the list. The application highlights the order for actions by the user, such actions including cancel, change, or replace. At step 3445, the user selects an operation (e.g., cancel, change, or delete) from the open orders display. The application automatically populates the trade ticket, at step 3450 (see FIGS. 5 & 11), with information from the open orders display on the selected stock. At step 3455, the user executes the desired function as requested in the trade ticket. To hide open orders, the user selects, at step 3460, the Open orders function from the function button display (function bar). The application then removes the open orders display and removes the indication from the open orders function, at step 3465.

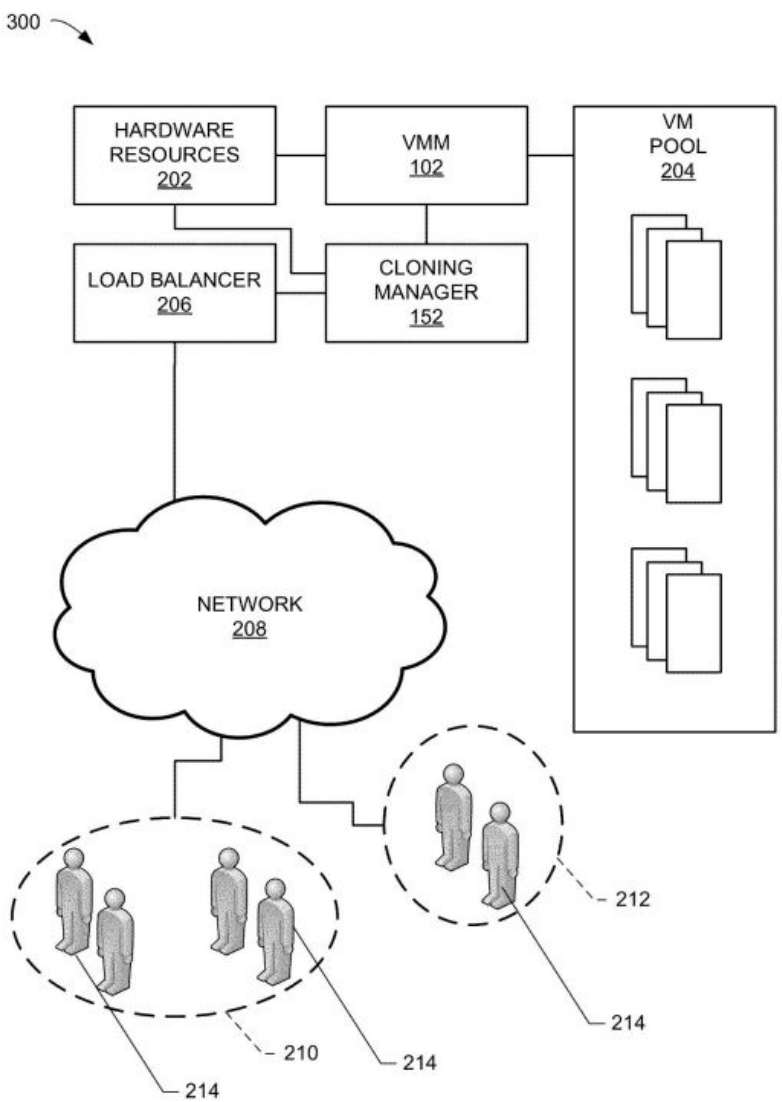


FIG. 2

Generated Brief and Detailed Descriptions	Ground Truth Brief and Detailed Descriptions
FIG. 2 is a block diagram illustrating a dedicated VMM pool of virtual machines with dedicated VMM hardware resources, according to various embodiments.	FIG. 2 is a schematic illustration of an example implementation of a cloning manager of the example system of FIG. 1B.
FIG. 2 is a block diagram of an example system 200 for maintaining a virtualized environment. The system 200 includes hardware resources 202, a load balancer 206, a pool 204 of virtual machine instances, a cloning manager 152, and a virtual machine migration service (VMMS) 102. The hardware resources 202 can include one or more servers. The one or more servers can include one or more CPUs, memory, storage, and/or networking resources. The hardware resources 202 can be virtualized, meaning the hardware resources 202 can be presented to virtual machines as multiple virtual machines having multiple virtualized CPUs, memory, storage, and/or networking resources. The hardware resources 202 can be virtualized by a hypervisor or virtual machine monitor (VMM) that manages the hardware resources 202. The hardware resources 202 can be virtualized by a single VMM and present to multiple virtual machines.The load balancer 206 can be a computer system or a server that manages a computer cluster or a cloud computer cluster. The load balancer 206 can be responsible for distributing resources, for example, workloads, input or output among a network of servers, for example, in a computer cluster. The load balancer 206 can be responsible for balancing the resources, for example, the number of client requests by a client application running on different servers, to optimize resource use, maximize throughput, minimize response time, and avoid overload. The load balancer 206 can be responsible for increasing or decreasing resources responsive to demand. The load balancer 206 can use a least connections algorithm to route client requests to servers. The pool 204 of virtual machine…	FIG. 2 is a schematic illustration of an example system 300 to provision virtual machine resources. In addition to the VMM 102 and the cloning manager 152 described above in connection with FIG. 1B, the illustrated example of FIG. 2 includes hardware resources 202, a VM pool 204, and a load balancer 206 communicatively coupled to a network 208. The example hardware resources 202 of FIG. 2 may include any number of CPUs, storage resources (e.g., hard disk drive(s), optical disk drive(s), flash disk drive(s), etc.) and/or any amount of memory, such as the example RAM 112 and/or nonvolatile memory. The illustrated example of FIG. 2 also includes a first customer 210 and a second customer 212, each of which may include any number of users 214. Although the illustrated example of FIG. 2 includes two customers, any number of customers may be serviced by example methods, apparatus, systems and/or articles of manufacture disclosed herein.In operation, the example cloning manager 152 of FIG. 2 invokes the example VMM 102 to instantiate a parent VM, such as the example VM1 104 of FIG. 1B, which results in an established (running) VM. As described above, the parent VM performs a number of disk accesses to acquire and store pages to physical memory 112. The parent VM may allocate other hardware resources 202, such as one or more CPUs and/or other processors dedicated to the parent VM. The example parent VM may be generally associated with a type of task and/or operation including, but not limited to OLAP, banking, ecommerce, and/or web services. In the event one or more task(s) are deemed dissimilar, additional parent VMs may be instantiated, as desired. When the parent VM…

Take-aways

- PatentDesc-355K - Large-scale dataset of (patent figure, brief description, detailed description) triplets.
- PatentMME - Novel training objectives for learning patent-image-specific representations.
- PatentLMM outperforms all the baselines including zero-shot GPT-4V.

- https://vl2g.github.io/projects/PatentLMM/

Summary

- Shivank Garg, Sankalp Mittal, Manish Gupta. Text2Arch: A Dataset for Generating Scientific Architecture Diagrams from Natural Language Descriptions [Slides] [Poster] [Video]. ICLR 2026.
- Shreya Shukla, Nakul Sharma, Manish Gupta, Anand Mishra. PatentLMM: Large Multimodal Model for Generating Descriptions for Patent Figures. [Poster] AAAI 2025

- LinkedIn: http://aka.ms/manishgupta
- HomePage: https://sites.google.com/view/manishg/
- manishg.iitb@gmail.com
- gmanish@microsoft.com