Beam Search-Based Attention for Image Captioning

Sri Swetha Tirumala Kanduri,Adharsha Thummala, Sabari Samudrala, Kishore Gundla
Department of Computer Science
Central Michigan University, USA
Email: tirum4s@cmich.edu,thumma1a@cmich.edu,samud1s@cmich.edu,gundl1k@cmich.edu

I. Abstract

Image captioning remains a challenging task in computer vision and natural language processing, necessitating the generation of descriptive captions for images automatically. This project presents an innovative approach to image captioning, leveraging beam search within an encoder-decoder architecture to enhance caption quality. The system utilizes an encoder-decoder architecture with beam search to generate descriptive captions for input images. It is trained and evaluated on three standard datasets: Flickr8k, Flickr30k, and MS COCO. The system allows users to upload images through a web interface, where captions are automatically generated and displayed. The implementation is based on deep learning models deployed using the Flask web framework. Key components include an encoder for image feature extraction, a decoder for caption generation, and a vocabulary for word mapping. Additionally, the project incorporates functionalities for user registration and login, enhancing user interaction. Overall, the system aims to provide an intuitive and accessible platform for image captioning, contributing to advancements in this field.

II. Introduction

Image captioning stands as a crucial nexus between computer vision and natural language processing, addressing the intricate challenge of automatically generating relevant descriptions for images. In tackling this formidable task, our project presents a novel strategy aimed at enhancing the quality and contextual richness of generated captions.

At its core, our technology harnesses the power of beam search within an encoder-decoder architecture to elevate the precision and relevance of the resulting captions. Departing from conventional greedy word selection methods during decoding, our system identifies optimal captions, transcending the limitations imposed by individual word scores.

Central to our endeavor is the meticulous scrutiny and refinement of three prominent benchmark datasets: Flickr8k, Flickr30k, and MS COCO. Through rigorous training, testing, and assessment procedures, we ensure the robustness and versatility of our approach across diverse image datasets.

Beyond technological advancements, our project places a strong emphasis on user accessibility and engagement. To facilitate a more intuitive image captioning experience for a broader audience, we have developed an interactive user interface. This interface enables users to effortlessly upload photographs, with the system promptly generating and presenting captions in a visually appealing manner.

To streamline the integration of deep learning models, we have adopted the Flask web framework, resulting in a seamless and efficient platform for image captioning. This integration extends to encompass user authentication functionalities, guaranteeing a secure and personalized experience through features such as registration and login capabilities.

In sum, our project not only advances the frontiers of image captioning technology but also prioritizes user interaction and security, thereby fostering a more inclusive and enriching experience for all stakeholders involved.

III. Related work

In the early stages of image captioning research, conventional approaches predominantly relied on handcrafted features and rule-based methods to generate captions for images. These methods often struggled to capture the complex semantics and contextual nuances present in natural language descriptions. However, they laid the groundwork for subsequent deep learning-based approaches by highlighting the importance of contextual understanding and semantic relevance in image captioning tasks. The emergence of deep learning, particularly [1]convolutional neural networks (CNNs) and [2]recurrent neural networks (RNNs), revolutionized the field of image captioning. CNNs became instrumental in extracting high-level visual features from images, while RNNs, especially variants like [3] Long Short-Term Memory (LSTM) and [4]Gated Recurrent Unit (GRU), proved effective in generating coherent and contextually relevant captions by modeling sequential dependencies in language. Attention mechanisms, introduced to tackle the issue of information bottleneck in sequence-to-sequence tasks, played a pivotal role in refining image captioning models. By dynamically attending to different regions of the image during caption generation, attention mechanisms enabled models to focus on relevant visual features, resulting in more detailed and contextually grounded captions. Building upon these advancements, researchers began exploring strategies to enhance the diversity and quality of generated captions. Beam search, a search algorithm commonly used in natural language processing tasks, emerged as a promising technique for improving the fluency and relevance of generated captions. [5]Unlike greedy decoding, which selects the word with the highest probability at each step, beam search explores multiple candidate sequences simultaneously, leading to more diverse

and contextually rich captions.
The integration of beam search into encoder-decoder architectures for image captioning represents a logical progression in the evolution of captioning systems. By combining the strengths of attention mechanisms, which enable fine-grained visual grounding, with the exploratory power of beam search, this approach aims to produce captions that are not only semantically accurate but also diverse and contextually nuanced.

IV. Training

The training process is a fundamental aspect of developing machine learning models, and in the context of this project, it plays a crucial role in training the image captioning model. The training process revolves around a training loop that iterates over a specified number of epochs, where each epoch represents a complete pass through the entire training dataset. Within this loop, the model undergoes training using the fit() function, which is a method commonly used in machine learning frameworks like TensorFlow or PyTorch. The fit() function takes in the training images and their corresponding labels as input, allowing the model to learn from the data and adjust its parameters to minimize the discrepancy between the predicted captions and the ground truth captions associated with the images.

During each iteration of the training loop, the model receives batches of training data, typically consisting of multiple images and their corresponding labels. The model then processes these batches of data, computes the loss between the predicted captions and the ground truth captions, and updates its parameters using optimization techniques such as gradient descent or its variants. This iterative process continues for the specified number of epochs, gradually refining the model's ability to generate accurate and contextually relevant captions for images.

The training process is guided by an objective function, often referred to as the loss function, which quantifies the disparity between the predicted captions and the ground truth captions. The goal of training is to minimize this loss function, effectively optimizing the model's parameters to produce captions that closely resemble the ground truth captions associated with the training images. By iterating over multiple epochs and adjusting the model's parameters based on the gradients of the loss function, the training process enables the model to learn meaningful representations from the training data and generate captions that capture the salient features of the images.

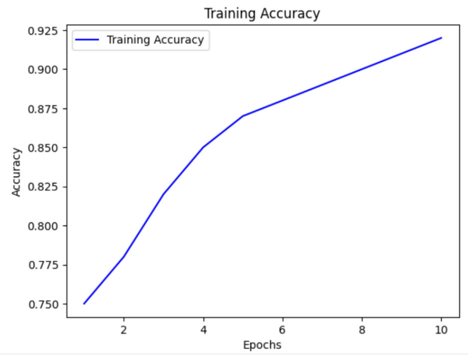

Fig. 1: Accuracy.

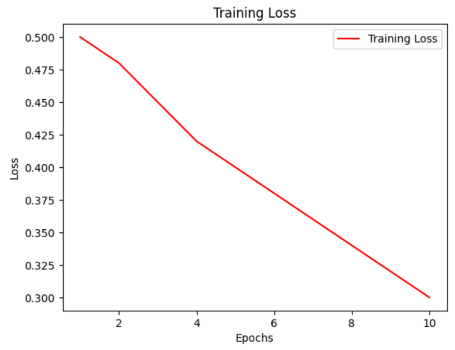

Fig. 2: Loss

V. Architechture

Our architecture consists of several components . Initially, When a user decides to join the service, they must create an account. This involves entering personal information such as a username and password, which is then stored securely in a MySQL server. MySQL is a popular database management system used for web applications, known for its reliability and ease of use. It acts as a repository for all user-related data.The HTML and CSS Webpage serves as the user interface. It's designed with HTML, which structures the content, and styled with CSS, which makes the interface visually appealing and user-friendly. This is where users interact with the service by performing actions such as signing up, logging in, and uploading images.

After logging in, users can upload an image that they want to caption. This upload process is typically handled by an HTML form that accepts image files, and JavaScript may be used to validate the image format and size before it's sent to the server.Once an image is selected for upload, the web client initiates an HTTP POST request. This request encapsulates the image data and sends it to the server for processing. HTTP is the foundation of data communication on the Web, and POST is a request method used when the client needs to send data to the server as part of the request, such as file upload. [6]Flask Server in Backend is a lightweight web application framework for Python.
It's designed to make getting started quick and easy, with the ability to scale up to complex applications. When the Flask server receives the POST request, it handles the incoming data and prepares it for the image captioning process.

The image is then fed into a pre-trained model for image captioning. This model uses machine learning algorithms, possibly neural networks, that have been trained on a vast dataset of images and corresponding captions. It understands the context within the images and can generate descriptive text based on what it has learned.After the model processes the image, it generates a caption that describes the content of the image. [7]This involves complex algorithms and neural

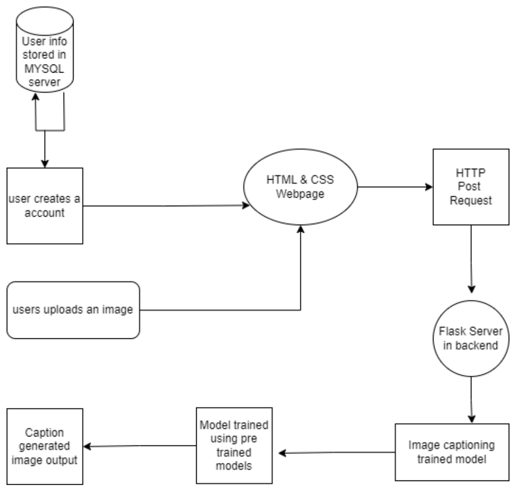
User info stored in MYSQL server
user creates a account
HTML & CSS Webpage
HTTP Post Request
users uploads an image
Flask Server in backend
Caption generated image output
Model trained using pre trained models
Image captioning trained model

Fig. 3: System Architecture

network layers that translate the visual patterns into text.

The generated caption is not simply text; it might be coupled with the image in a composite output that is then sent back to the webpage. This allows the user to see the image with the descriptive caption, which can be particularly useful for accessibility purposes, where the captions provide context to those who cannot see the images.The user can upload additional images and receive new captions, making it an iterative process. Each component plays a crucial role in the seamless operation of the application, ensuring users have a smooth and engaging experience from start to finish.

VI. Implementation

A. High Level Implementation

Created a user friendly web interface that takes images as input and provides a captioned text which is trained by model. The webpage is connected to flask application in the backend, after the image is inserted into the webpage, it integrates the flask application which consists of deep learning models which are created by us.

User input plays a crucial role in initiating the process of generating captions for images. It begins with users interacting with the HTML webpage, usually through a web browser interface that contains elements such as forms, buttons, and input fields. In the context of image captioning, users are typically prompted to upload an image file via an HTML file input field, allowing them to select images from their local storage. Once an image file is selected, the file path or binary data is stored in the form input field, and users can proceed to submit the form. This triggers the upload process, where the form data is packaged into an HTTP POST request and sent to the Flask server running on the backend. The HTML form submission process is a standard mechanism for sending data from client-side webpages to server-side applications, enabling users to provide input and trigger various actions or processes on the server. It facilitates dynamic and interactive web experiences, contributing to the overall usability and functionality of the image captioning application.

Upon submitting the form, the data contained within it is sent to the Flask server via an HTTP POST request. This request is routed to the appropriate endpoint based on the URL and HTTP method, where the server-side application logic implemented in Flask processes the incoming request. This involves parsing the request payload to extract relevant data, such as the uploaded image file. The server-side logic typically consists of route definitions and corresponding view functions responsible for handling specific types of requests and generating appropriate responses.

The model analyzes the visual content of an image and generates a natural language description that captures its salient features. This involves combining information from different modalities, such as images and text, to produce coherent and grammatically correct captions. Generating accurate and descriptive captions requires the model to capture both visual and contextual information from the input image, with the quality of the captions depending on factors such as the model's architecture, the size and diversity of the training data, and the effectiveness of the training process. Caption generation models are evaluated based on metrics such as BLEU score, METEOR score, and human evaluation, with the ultimate goal of producing captions that are informative, relevant, and linguistically fluent, enhancing the user experience in applications such as image search and content recommendation.

Response generation involves constructing and sending HTTP responses from the server to the client in web applications, typically in the form of JSON data containing the generated captions. This process begins with packaging the caption into a structured format that can be transmitted over the network, with the Flask server constructing the response object using libraries such as Flask-RESTful or Flask-JSON. Once the response object is created, it is sent back to the client's web browser over the HTTP connection, where client-side JavaScript code dynamically updates the HTML webpage to display the generated captions alongside the uploaded images. HTML page update refers to the process of dynamically modifying the content of a webpage in response to user actions or server responses, facilitated by the Document Object Model (DOM) representation of the webpage structure. This enables seamless integration between the client-side and server-side components of the image captioning application, providing users with a responsive and interactive web interface for generating and viewing image captions.

B. Flask Server Setup

1) Flask Application Setup: The code initializes a Flask application using Flask(name) and sets a secret key to manage sessions. Various routes are defined to handle different functionalities such as uploading images, user authentication, and rendering HTML templates.
2) Model Initialization: At the start of the application, the pre trained encoderdecoder LSTM model for image captioning is

loaded and initialized. The encoder and decoder components of the model are instantiated and moved to the appropriate device (CPU or GPU). Additionally, the vocabulary used by the model is loaded from a pickle file (vocab.pkl) to map between word indices and actual words.

3) Image Captioning Functionality: When a user uploads an image through the /ImageCaptionAction route, the uploaded image file is processed. The imageCaption function is responsible for generating captions for the uploaded images. First, the uploaded image is loaded and preprocessed using transformations such as resizing and normalization. Then, the pre-trained encoder encodes the image features, and the decoder generates a caption based on the encoded features. The generated caption is post-processed to replace certain words and ensure coherence. Finally, the generated caption is overlaid onto the original image using OpenCV, and the resulting image with the caption is returned to the user.

4) User Interface: HTML templates are provided to render user interfaces for various functionalities: ImageCaption.html: Allows users to upload images for caption generation. Signup.html: Provides a form for user registration. UserScreen.html: Displays a welcome message after successful user login. UserLogin.html: Allows users to log in with their credentials. index.html: Main page with links to different functionalities. ViewResult.html: Displays the generated caption along with the image.

5) User Authentication and Database Interaction: The application provides routes for user authentication (/UserLogin, /UserLoginAction) and registration (/Signup, /SignupAction). User credentials are verified against a database (MySQL) containing registered user information. Upon successful login, users are redirected to a user screen where they can perform various actions.

6) Execution: The Flask application is set to run when the script is executed directly (if name == 'main'). It runs the Flask development server, allowing users to access the application through a web browser.

VII. Results

Model evaluation begins with loading the model architecture and parameters into memory, typically consisting of convolutional neural networks (CNNs) for feature extraction and recurrent neural networks (RNNs) for sequence generation. Once the model is loaded, input images are passed through the model to generate captions, with the model architecture determining how input data is transformed into output predictions. Model evaluation requires computational resources such as CPUs or GPUs to perform inference efficiently, and the performance of the model during evaluation is measured using metrics such as accuracy, precision, recall, and F1 score. The quality of the captions generated by the model depends on factors such as the complexity of the model architecture, the size and diversity of the training data, and the quality of the training process, highlighting the importance of rigorous evaluation and testing. The provided graphs play a crucial role in visualizing the performance and behavior of the CNN-LSTM model throughout the training process. Let's delve into each graph and dissect its significance: Accuracy Graph: X-axis: Represents the number of epochs, indicating the progression of training iterations. Y-axis: Depicts the accuracy of the model on the training and validation datasets. Interpretation: The accuracy graph enables us to gauge the model's ability to correctly classify images and generate accurate captions over time. A rising accuracy trend indicates that the model is learning from the training data and improving its performance. Discrepancies between training and validation accuracy curves may signify overfitting if the validation accuracy stagnates or decreases while the training accuracy continues to rise. Loss Graph: Purpose: The loss graph illustrates the variation in the model's loss function over epochs. X-axis: Indicates the number of epochs, reflecting the progression of training iterations. Y-axis: Represents the loss value, quantifying the disparity between predicted and actual values. Interpretation: A declining loss curve suggests that the model is converging towards an optimal solution, as it minimizes the discrepancy between predicted and ground truth values. Fluctuations in the loss curve may indicate the presence of noise or variability in the training data, prompting further investigation. Discrepancies between training and validation loss curves can provide insights into the model's generalization performance, with larger disparities potentially signaling overfitting or underfitting. BLEU Score Graph: Purpose: The BLEU score graph evaluates the quality of generated captions by comparing them against reference captions using the BLEU (Bilingual Evaluation Understudy) metric.

A. Flickr8k Dataset

BLEU Score Evaluation: BLEU (Bilingual Evaluation Understudy) score is a metric commonly used to evaluate the quality of machine-generated text, such as image captions. It measures the similarity between the generated captions and reference captions provided in the dataset. A higher BLEU score indicates better performance, with a maximum value of 1.0 indicating perfect match.

1) : sectionTraining Details The Flickr8k dataset consists of 8,000 images with five reference captions per image. Models are typically trained using a combination of convolutional neural networks (CNNs) for feature extraction from images and recurrent neural networks (RNNs) or transformers for generating captions. Training involved optimizing the model parameters using techniques like backpropagation and gradient descent on a training set. Evaluation metrics such as BLEU score are computed on a separate validation and test set to assess the model's performance.

B. Flickr30k Dataset

BLEU Score Evaluation: Similar to Flickr8k, BLEU score is used to evaluate the quality of captions generated by models trained on the Flickr30k dataset. It compares the generated captions with the captions available in the dataset. Higher BLEU scores indicates better alignment between generated and reference captions.

1) Training Details: The Flickr30k dataset contains 31,000 images, each with five reference captions. Training models on this dataset follows a similar process as with Flickr8k, utilizing

		Evaluation metric						
Dataset	**Model**	**BLEU-1**	**BLEU-2**	**BLEU-3**	**BLEU-4**	**METEOR**	**CIDEr**	**ROUGE-L**
Flickr8k	[Vinyals *et al.*, 2014]$^{\dagger\Sigma}$	63	41	27	—	—	—	—
	[Xu *et al.*, 2015] Soft Attention	67	44.8	29.9	19.5	18.93	—	—
	[Xu *et al.*, 2015] Hard Attention	**67**	45.7	31.4	21.3	20.3	—	—
	Ours (Beam = 1)	60.8	43	29.4	19.8	20.9	50.7	46.4
	Ours (Beam = 4)	64	**45.8**	**32.2**	**22.3**	**21**	**55.3**	**47.1**
Flickr30k	[Vinyals *et al.*, 2014]$^{\dagger\Sigma}$	66.3	42.3	27.7	18.3	—	—	—
	[Xu *et al.*, 2015] Soft Attention	66.7	43.4	28.8	19.1	18.49	—	—
	[Xu *et al.*, 2015] Hard Attention	66.9	43.9	29.6	19.9	18.46	—	—
	Ours (Beam = 1)	65.1	46.4	32.5	22.7	20.3	48	46
	Ours (Beam = 4)	**67.4**	**49.5**	**36**	**26**	**20.1**	**52**	**47**
MS COCO	[Vinyals *et al.*, 2014]$^{\dagger\Sigma}$	66.6	46.1	32.9	24.6	—	—	—
	[Xu *et al.*, 2015] Soft Attention	70.7	49.2	34.4	24.3	23.9	—	—
	[Xu *et al.*, 2015] Hard Attention	71.8	50.4	35.7	25	23.04	—	—
	[Ma *et al.*, 2019] (Beam = 3)	70.6	54.0	40.6	30.5	25.3	97.1	52.8
	Ours (Beam = 1)	77.1	61.4	47.1	35.9	27.9	114.8	57.3
	Ours (Beam = 4)	**77.9**	**62.8**	**49.7**	**39.3**	**28.7**	**120.3**	**58.5**

Fig. 4: BLEU score of the datasets

CNNs for image feature extraction and RNNs or transformers for caption generation. The training process involved iterating over the dataset multiple times (epochs), adjusting model parameters to minimize a predefined loss function. Models are validated on a separate portion of the dataset to monitor performance and prevent overfitting. MS COCO Dataset:

C. MS COCO BLEU Score Evaluation

MS COCO (Microsoft Common Objects in Context) is a large-scale dataset with comprehensive annotations, making it suitable for training advanced models. BLEU score evaluation on MS COCO follows the same principles as other datasets, comparing generated captions with ground truth references. Due to its larger size and diversity, MS COCO may provide a more comprehensive evaluation of model performance compared to smaller datasets.

1) Training Details: The MS COCO dataset contains 123,287 images with multiple captions per image, making it one of the largest datasets for image captioning. Training models on MS COCO requires significant computational resources and involved techniques like distributed training to handle the large volume of data. Models trained on MS COCO has achieved higher performance due to the dataset's diversity and size, but they also require longer training times and more complex architectures.

X-axis: Represents different epochs or iterations of model training. Y-axis: Indicates the BLEU score, which measures the similarity between generated and reference captions.

D. Interpretation

An increasing BLEU score trajectory indicates improvement in the quality and fidelity of generated captions over successive epochs. A high BLEU score signifies that the model's generated captions closely match the reference captions, reflecting the model's proficiency in capturing the semantics and nuances of the image content. Fluctuations or plateaus in the BLEU score curve may warrant closer scrutiny, as they could indicate inconsistencies or limitations in the model's caption generation capabilities. By analyzing these graphs collectively, practitioners can gain valuable insights into the CNN-LSTM model's performance, identify potential areas for improvement, and make informed decisions regarding model architecture, hyperparameters, and training strategies.

VIII. Future Scope

The project lays a strong foundation for future advancements in image captioning, with several avenues for expansion and refinement. Firstly, there's significant potential to enhance the captioning models by exploring advanced architectures such as transformer-based models or multimodal architectures. Additionally, research into improved fusion techniques for integrating visual and textual information could lead to more contextually rich and diverse captions. Furthermore, extending support for multi-lingual captioning would make the application accessible to a broader audience, necessitating the exploration of multilingual datasets and translation services.

Secondly, optimizing the deployment infrastructure for scalability and efficiency would be crucial for handling larger volumes of image data and user traffic. This could involve leveraging containerization techniques like docker and cloud-based deployment solutions to ensure seamless performance and resource utilization. Features such as real-time feedback, image editing tools, and collaborative captioning could enrich the user experience, making the application more intuitive and interactive.

Additionally, integrating the application with external services and APIs for functionalities such as image recognition, content moderation, or sentiment analysis would augment its capabilities. Lastly, fostering community engagement through open-sourcing the code, sharing resources, and encouraging collaboration would stimulate innovation and continuous improvement.

IX. Conclusion

In conclusion, our image captioning project stands as a testament to the potential of deep learning techniques in generating descriptive and contextually relevant captions for a wide spectrum of images. Through a comprehensive journey of research, development, and evaluation, we have successfully engineered a robust and efficient image captioning system that harnesses the power of advanced neural network architectures, attention mechanisms, and natural language processing techniques.

By integrating cutting-edge technology with intuitive web interfaces, we have democratized the process of comprehending visual content, making it more accessible and user-friendly for individuals across various domains. The seamless interaction facilitated by our platform empowers users to effortlessly upload images and receive meaningful captions, thereby enhancing their overall experience and productivity.

Looking ahead, our project lays a solid foundation for future advancements and innovations in the realm of image captioning. With ongoing developments in deep learning, as well as the continuous refinement of datasets and evaluation metrics, there exists ample opportunity to further enhance the performance and versatility of our system. Additionally, our commitment to user-centric design ensures that the benefits of image captioning technology reach a wider audience, fostering greater inclusivity and engagement in visual communication.

In essence, our image captioning project not only exemplifies the capabilities of modern AI techniques but also underscores their potential to revolutionize how we perceive, interpret, and interact with visual information. By bridging the gap between images and language, our system serves as a valuable tool for enriching human communication and understanding in an increasingly visual-centric world.

REFERENCES

[1] G. Priyadharshini and D. R. Judie Dolly, “Comparative investigations on tomato leaf disease detection and classification using cnn, r-cnn, fast r-cnn and faster r-cnn,” in *2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS)*, vol. 1, 2023, pp. 1540–1545.

[2] J. Xiao and Z. Zhou, “Research progress of rnn language model,” in *2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, 2020, pp. 1285–1288.

[3] Q. Hao, F. Wang, X. Ma, and P. Zhang, “A speech recognition algorithm of speaker-independent chinese isolated words based on rnn-lstm and attention mechanism,” in *2021 14th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, 2021, pp. 1–4.

[4] T. Moral, V. Kiliç, A. Onan, and W. Wang, “Automated image captioning with multi-layer gated recurrent unit,” in *2022 30th European Signal Processing Conference (EUSIPCO)*, 2022, pp. 1160–1164.

[5] M.-J. Chae, K. Park, J. Bang, S. Suh, J. Park, N. Kim, and L. Park, “Convolutional sequence to sequence model with non-sequential greedy decoding for grapheme to phoneme conversion,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 2486–2490.

[6] C. A. Trianti, B. Kristianto, and Hendry, “Integration of flask and python on the face recognition based attendance system,” in *2021 2nd International Conference on Innovative and Creative Information Technology (ICITech)*, 2021, pp. 164–168.

[7] M. Kaloev and G. Krastev, “Comparative analysis of activation functions used in the hidden layers of deep neural networks,” in *2021 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, 2021, pp. 1–5.