

MANIMATIC: A MATH VISUALISER AND EXPLAINER

An AI model that can generate animated videos to dynamically illustrate and explain mathematical concepts, enhancing student understanding and engagement.

Shivek Daswani

SVKM's NMIMS Nilkamal School of Mathematics, Applied Statistics and Analytics

ABSTRACT

I present MANIMATIC, a generative AI model that explains mathematical concepts with visualizations and voice. This LLM incorporates the open-source LLM Llama-2 7B model, the python library Manim, developed by Grant Sanderson for generating visualizations for mathematics, and the text-to-speech python library to create a video explainer based on the prompt given by the user. This model successfully generates the Manim code and also converts the explanation into a speech file for later merging of the audio and video files.

1. INTRODUCTION

Mathematics is abstract yet beautiful. But the beauty of mathematics is understood only by a handful of people in the whole world. The last known official number of total mathematicians in the world was ~57000 out of a total world population of 6.31 billion people. Like I said, that's just a handful of the people. Even if we count theoretical physicists and math enthusiasts, the number still wouldn't reach even 1% of the world population of that time. Although the number might have grown in 2024, it still wouldn't be close to even 1% of the current 8 billion population we have. The awareness of mathematics has grown, but it still is not significant enough. There are no jobs that require a math degree, except a teaching job with miniscule pay in the initial years. Stipends are low for mathematical research, and hence, people don't have enough motivation to become a math researcher.

Although the discoveries made by mathematicians have had a significant role in development of theories in pure sciences, we still have very little pay in contrast to the big scope mathematics has. Why is there such a disparity? These are a few problems which develop during our graduation years. Although the problem has its roots deep down. I have taught students from 9th grade to 12th grade, and have seen a fear of mathematics in the majority of the students. Even those who already have great cognitive abilities, have this fear towards mathematics. The reason is simple, they cannot visualize the abstract nature of mathematics. Humans have a tendency to fear what they do not know, and what they cannot understand. A simple example would be of a child having a fear of fire, simply because it cannot understand what it is and it hurts when touched. This fear dissolves as we grow, since we then know the properties of fire and use it effectively in our daily lives.

This same analogy can be applied to fear of mathematics. Children fear mathematics because they don't know how numbers work, and thus, want to run away from understanding it. The properties and careful use of fire is nicely taught in childhood, thus eliminating the fear. The same can't be said about mathematics. Since childhood, we have been taught counting numbers, and just memorizing the tables and arithmetic operations, without ever knowing the backstory behind the working of numerical systems. In your high school years, you are just supposed to memorize the steps for calculating HCF and LCM, and not understand how they are related to each other. In Junior College, we are just supposed to memorize the list of all the basic integrals and derivatives, without understanding what it actually does and how it would look on the graph. This method of memorizing without in-depth understanding has instilled fear in students, which is why they tend to dislike mathematics and are reluctant to work on it.

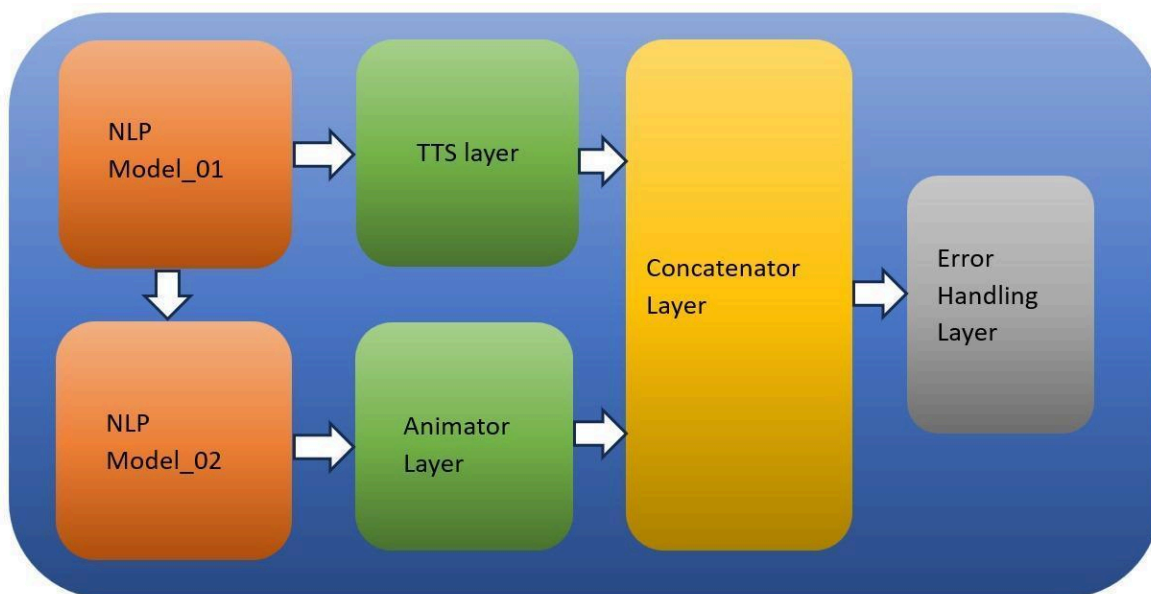
A solution to this problem is to visualize the working of mathematics behind any concept. The aim of developing this model is to instill mathematical intuition in learners of any age group. Since it can generate visualization for many complex topics in mathematics, learners of any age group can utilize this to learn mathematics, Pure or Applied. If used on younger age groups, there is potential of developing their liking towards mathematics at an early stage, and also improve their academic performance.

2. APPROACH

In this work, I used a pre-trained open-source LLM model for the task of text generation. There are a few reasons for this. First, To generate an intuitive explanation of the prompt given, I need capabilities of understanding and generating user inputs. To understand which model gives the best explanation while being constrained to my hardware limitations, I decided to experiment on various open-source LLMs capable of running on my hardware, i.e. GTX 1650 4 GB VRAM and Ryzen 4600H 16GB RAM. Some of the models which I tried were: Llama 7B and 13B chat, Llama 2 7B and 13B chat, Mistral 7B , Neural-chat 7B, Gemma 7B and Falcon 7B. I noticed that although Llama and Llama 2 7B are easier on generation, their output was not as satisfactory as Llama and Llama 2 13B variants, although the output was generated at a significantly slower rate. The only model that gave the output on par with Llama 2 13B variant was Mistral 7B's chat model, Neural-chat 7B,

The open-source model Mistral 7B is capable of generating the desired outputs. To generate the Manim code, I wanted to use Code Llama, but due to GPU and CPU memory constraints, I could only utilize 1 large language model. So I decided to completely rely on my Mistral 7B model for code generation too.

The flowchart below demonstrates the inner framework of my MANIMATIC model:



For **NLP Model_01**, **NLP Model_02** and **Animator Layer**, I used the Neural-chat 7B, a fine-tuned version of Mistral 7B for text generation and code generation. The reason I used Mistral 7B is because it outperforms many 13B models and is capable of generating code on par with Code Llama 7B while working in lower computational environments.

Each model is initialized from the Neural-chat 7B model. Since the model is pre-trained on mathematical texts, there is no need for further fine-tuning on that dataset. But the model can be fine-tuned on the Manim code dataset, for much more efficient output. Due to less computational capabilities, I decided to just rely on the capabilities of the model to write the code. A major downside of this was that the model was generating Manim code of different versions without specifying which version it was giving code for. This resulted in the model generating Manim code of different versions each time, causing version conflicts. To mitigate this issue, I gave an instruction in the system prompt of generating the code for a specified version of Manim. This worked properly.

For every model layer, I had to give a detailed system prompt according to the needs. NLP Model_01 is an explanation generator layer. Thus, I had to prompt it in such a way that it generated an explanation on how a teacher would teach in a classroom. The prompt I gave went through many trials and errors.

NLP Model_02 is an instruction generator. It generates instructions on how to visualize a certain part of the explanation. Thus it takes the output from the previous NLP model as input, analyzes it and then generates visualization for every point in the input. These instructions on how to visualize will then help the Animator layer to develop a Manim code that generates the visualizations as instructed by NLP Model_02.

Animator Layer will also use the Neural-chat 7B model to generate the code. Since it is fine-tuned for chat purposes on the base Mistral 7B model, it performed on-par with Code Llama in coding capabilities. The problem of it generating different versions of Manim code persisted, although I was able to remove that problem by specifying which version of Manim to use. Below is a list of a few successful outputs I got based on the prompt provided. For the text-to-speech layer, I just used the python library GTTS, although an open-source text-to-speech model can be utilized for this purpose too.

3. RESULT

Enter your prompt: concept of Riemann Integrals

Stage 01 (NLP Model_01)	<p>Riemann Integrals are a mathematical concept used to calculate the definite integral of a function. The logic behind Riemann Integrals is that they are based on the idea of dividing the area under a curve into smaller rectangles and then summing up the areas of these rectangles. The book definition of Riemann Integrals states that it is a function that assigns a value to each element of a interval and then uses these values to calculate the definite integral of a function over that interval.</p> <p>Key Concepts:</p> <ul style="list-style-type: none">*Division of area under a curve into smaller rectangles*Summing up the areas of these rectangles to find the definite integral of a function.*The Riemann Integral is a function that assigns a value to each element of an interval and then uses these values to calculate the definite integral of a function over that interval. <p>Examples:</p> <p>*Example 1: Find the definite integral of the function $f(x) = x^2$ from $x = 0$ to $x = 2$ using Riemann Integrals.</p> <p>Solution: Divide the area under the curve into smaller rectangles as shown below:</p> <p> 0 1 2 </p> <p>Area of each rectangle = (height x width) = $(1 \times 1) = 1$</p> <p>Total area of n rectangles = $(1 \times 2) + (2 \times 3) + (3 \times 4) = 1 + 6 + 12 = 29$</p> <p>Definite integral = sum of areas = 29</p> <p>*Example 2: Find the definite integral of the function $f(x) = 3x^2 + 2x - 5$ from $x = 0$ to $x = 4$ using Riemann Integrals.</p>
--------------------------------	--

	<p>Solution: Divide the area under the curve into smaller rectangles as shown below:</p> <pre> 0 1 2 3 4 </pre> <p>Area of each rectangle = (height x width) = $(3 \times 1) + (2 \times 2) + (1 \times 3) + (5 \times 4) = 3 + 4 + 6 + 20 = 33$</p> <p>Total area of n rectangles = $(1 \times 4) + (2 \times 8) + (3 \times 12) + (4 \times 20) = 4 + 16 + 36 + 80 = 166$</p> <p>Definite integral = sum of areas = 166</p>
Stage 02 (TTS Layer)	Audio saved to: response_tts.mp3
Stage 03 (NLP Model_02)	<p>To create a detailed instruction for creating animations using Manim for the given query, we will break down the process into several steps:</p> <p>Step 1: Introduction</p> <p>In this step, we will introduce the concept of Riemann Integrals by providing a brief overview of the topic. We will explain what Riemann Integrals are, how they work, and why they are important in mathematical calculations.</p> <p>We will also provide a brief explanation of the logic behind Riemann Integrals, including the idea of dividing the area under a curve into smaller rectangles and summing up the areas of these rectangles to find the definite integral of a function.</p> <p>Step 2: Division of area under a curve into smaller rectangles</p> <p>In this step, we will demonstrate how to divide the area under a curve into smaller rectangles using Manim. We will use the given query as an example and divide the area under the curve into smaller rectangles as shown in the solution:</p> <p>We will use the Manim module for animating the rectangles and show how each rectangle is created with a specific</p>

	<p>height and width. We will also explain how the height and width of each rectangle are calculated based on the function being integrated.</p> <p>Step 3: Summing up the areas of each rectangle</p> <p>In this step, we will demonstrate how to sum up the areas of each rectangle using Manim. We will explain how to calculate the area of each rectangle by multiplying the height by the width and then adding up the areas of all the rectangles to find the definite integral of a function.</p> <p>We will use the Manim module for animating the summing up of areas of each rectangle and show how each rectangle is animated with its own area value. We will also explain how the area values of all the rectangles are added up to find the definite integral of a function.</p> <p>Step 4: Conclusion</p> <p>In this step, we will conclude the animation by summarizing the key concepts covered in the animation. We will explain how Riemann Integrals work, how they are calculated, and why they are important in mathematical calculations.</p> <p>We will also provide some additional tips for using Manim to create animations for other mathematical concepts, such as using different shapes for different functions and animating the graph of a function to show its behavior over time.</p> <p>Overall, this animation will provide a clear and detailed explanation of Riemann Integrals, including their logic and application in mathematical calculations. It will also demonstrate how Manim can be used to create interactive and dynamic animations for mathematical concepts, making learning more engaging and fun.</p>
Stage 04 (Animator layer)	<p>Here is a sample Manim code for creating an animation that explains the concept of Riemann Integrals:</p> <pre>import manim</pre>

```

def rectangles(height, width):
    # Create a rectangle with specified height and width
    rect = Rectangle(height=height, width=width)
    # Add the rectangle to the scene
    scene.add(rect)
    # Create n rectangles with different heights and widths
    for i in range(n):
        rect.set_height(height*i)
        rect.set_width(width)
        scene.add(rect)
        # Animate the rectangles moving and merging to form a curve
        anim = Animation(duration=5)
        for i in range(n):
            rect.animate(scale=0.9, rotation=(30*i),
easing=Easing.in_out)
            scene.add(anim)
        # Animate summing up the areas of each rectangle to find the definite integral
        anim = Animation(duration=5)
        total_area = 0

        for rect in scene:
            total_area += rect.area()

        scene.add(anim)
    # Display the definite integral value
    text = Text(f"Definite integral = {total_area}")
    text.set_font(font="Arial")
    text.set_color(color=Color.red)
    scene.add(text)

```



```

# Show the animation
anim = Animation(duration=5)
scene.add(anim)

def example1(height, width):
    # Create a rectangle with specified height and width
    for example 1
        rect = Rectangle(height=height, width=width)
        # Add the rectangle to the scene
        scene.add(rect)
        # Find the definite integral of  $f(x) = x^2$  from  $x = 0$  to
         $x = 2$  using Riemann Integrals
        def integral(f, a, b):
            return sum(f(x) * rect.area() for x in range(a, b))

        result = integral(f(x), 0, 2)

    # Display the definite integral value
    text = Text(f"Definite integral = {result}")
    text.set_font(font="Arial")
    text.set_color(color=Color.red)
    scene.add(text)
    # Show the animation
    anim = Animation(duration=5)
    scene.add(anim)

def example2(height, width):
    # Create a rectangle with specified height and width
    for example 2
        rect = Rectangle(height=height, width=width)
        # Add the rectangle to the scene
        scene.add(rect)
        # Find the definite integral of  $f(x) = 3x^2 + 2x - 5$ 
        from  $x = 0$  to  $x = 4$  using Riemann Integrals

```

	<pre> def integral(f, a, b): return sum(f(x) * rect.area() for x in range(a, b)) result = integral(f(x), 0, 4) # Display the definite integral value text = Text(f"Definite integral = {result}") text.set_font(font="Arial") text.set_color(color=Color.red) scene.add(text) # Show the animation anim = Animation(duration=5) scene.add(anim) manim.run() </pre>
--	---

The above table shows the output from a series of neural-chat model outputs pipelined. The output of generating an animated video is achievable by running the Manim code from the model. To mitigate the problem fully, the model either has to be fine-tuned on a large dataset of Manim code for specific version, or few-shot prompting can be used for this purpose.

The model asks for a prompt at the start. For this, we can either give a prompt like "concept of ____" or just write the name of the concept we want to understand.

Then in stage 1, The first model will generate an explanatory output, explaining the logic, definition, key points, with examples for each of the above things. To get a more definitive and reliable answer, I set the temperature of the model to 0.2, which enables the model to not be extremely creative and generate a good explanation without deviating too much from the topic. This output explains mostly everything regarding a concept with examples for everything, just like how a good teacher would explain a concept.

In Stage 2, I used a small python code which utilizes the python library GTTS to convert the output of stage 1 into voice. Although the voice is a bit computerized. To give it a more human touch, open-source text-to-speech models can be utilized from hugging face.

In Stage 3, I used the same Neural-chat model for generation of instructions on how to generate visualization of each point in the explanation. This makes it easier for the subsequent model to understand, analyze and generate the Manim code accordingly. Here, I set the temperature of the model to 0.6 for it to be more creative in explaining the visualizations for each point in the explanation.

In Stage 4, using the neural-chat model, I used the outputs of Stage 1 and Stage 3 as inputs and gave the model an instruction to generate a Manim code based using both the inputs. This helps the model to generate an effective Manim code that uses snippets of explanation as well as gives figures and graphs for visualization, making it a very good experience for the user.

5. LIMITATIONS

This research project had its fair share of limitations, mainly being hardware for low computational abilities. This model was built and tested on my laptop with:

CPU: Ryzen 5 4600H

RAM: 16 GB RAM with 20 GB Swap Memory

GPU: Nvidia GeForce GTX 1650; 4 GB VRAM

Due to this setup, I could only load an LLM once, and thus, I had an array of LLMs pipelined, it became even more difficult for testing purposes. A way to improve on this is using cloud GPU services, which come at a cost.

6. CONCLUSION

I present MANIMATIC, a model designed to visualize and explain complex mathematical concepts with just a prompt. I have shown this model's capability in generating explanations and the code for visualization of that explanation. This model can further achieve better outputs once fine-tuned on specific datasets of Manim code. I hope that this model can become a useful foundation for further development of models in various other areas of education, and for imbibing mathematical intuition.

7. ACKNOWLEDGEMENTS

I would like to thank Prof. Dr. Yogesh Naik for mentoring me in this humongous project with their valuable guidance. I would also like to thank my parents for supporting me and providing motivation during the development of this project.

8. REFERENCES

Azerbayev, Zhangir, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. "Llemma: An open language model for mathematics." *arXiv preprint arXiv:2310.10631* (2023).

Drori, I., Zhang, S., Shuttlesworth, R., Tang, L., Lu, A., Ke, E., Liu, K., Chen, L., Tran, S., Cheng, N. and Wang, R., 2022. A neural network solves, explains, and generates university math problems by program synthesis and few-shot learning at a human level. *Proceedings of the National Academy of Sciences*, 119(32), p.e2123433119.

Drori, I., Tran, S., Wang, R., Cheng, N., Liu, K., Tang, L., Ke, E., Singh, N., Patti, T.L., Lynch, J. and Shporer, A., 2021. A neural network solves and generates mathematics problems by program synthesis: Calculus, differential equations, linear algebra, and more. *CoRR*, abs/2112.15594.

Poola, I. and Božid, V., 2023. Guiding AI with human intuition for solving mathematical problems in Chat GPT. *Journal Homepage*: <http://www.ijmra.us>, 11(07).

Rane, Nitin (2023) Enhancing Mathematical Capabilities through ChatGPT and Similar Generative Artificial Intelligence: Roles and Challenges in Solving Mathematical Problems. <http://dx.doi.org/10.2139/ssrn.4603237>

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

Kliziene I, Paskovske A, Cizauskas G, Augustiniene A, Simonaitiene B, Kubiliunas R. The Impact of Achievements in Mathematics on Cognitive Ability in Primary School. *Brain Sciences*. 2022; 12(6):736. <https://doi.org/10.3390/brainsci12060736>

<https://www.elastic.co/what-is/large-language-models#/what-is/large-language-models>

Jiang, A.Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D.S., Casas, D.D.L., Bressand, F., Lengyel, G., Lample, G., Saulnier, L. and Lavaud, L.R., 2023. Mistral 7B. *arXiv preprint arXiv:2310.06825*.

Roziere, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., Tan, X.E., Adi, Y., Liu, J., Remez, T., Rapin, J. and Kozhevnikov, A., 2023. Code Llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S. and Bikel, D., 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

<https://github.com/epfLLM/Megatron-LLM>

<https://docs.manim.community/en/stable/examples.html>

https://www.reddit.com/r/manim/comments/11z2p4n/generative_manim_an_experiment_to_generate_manim/

https://www.reddit.com/r/manim/comments/18g4px7/manim_studio_a_gui_for_manim_to_make_easier_your/

Wang, Y., Wang, W., Joty, S. and Hoi, S.C., 2021. Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. *arXiv preprint arXiv:2109.00859*.