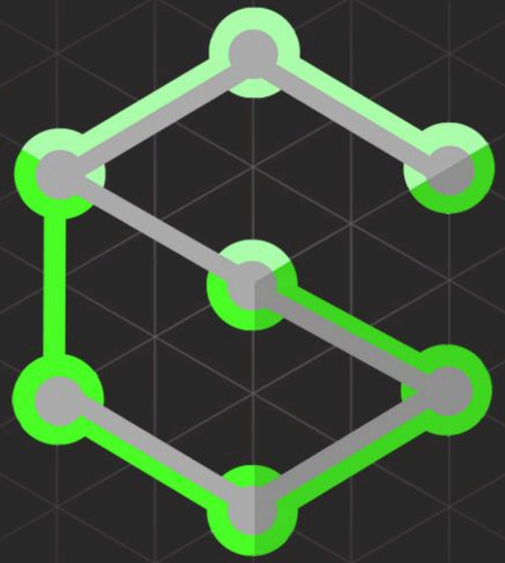
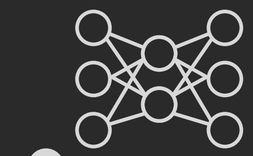


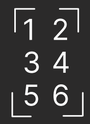
Graph Space

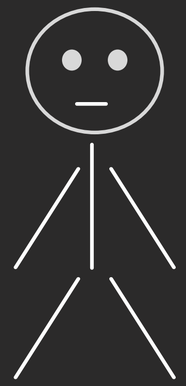
Effortless AI

The Problem

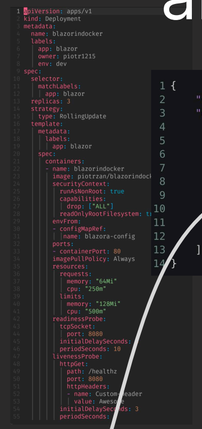
AI Fundamentals

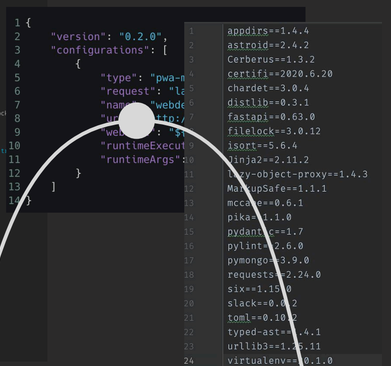




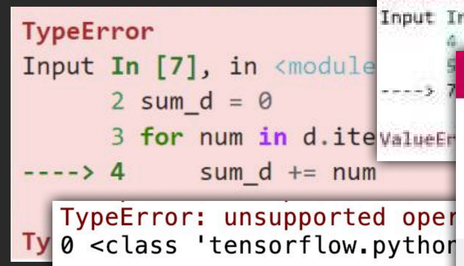


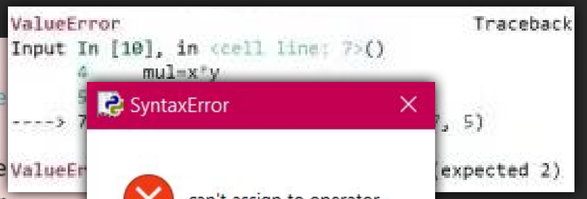
Environment and Configs

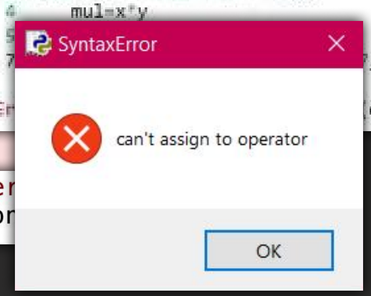




Debugging







Working AI Model





Frameworks, Libraries, and Tools

Coding

```
import tensorflow as tf
import numpy as np

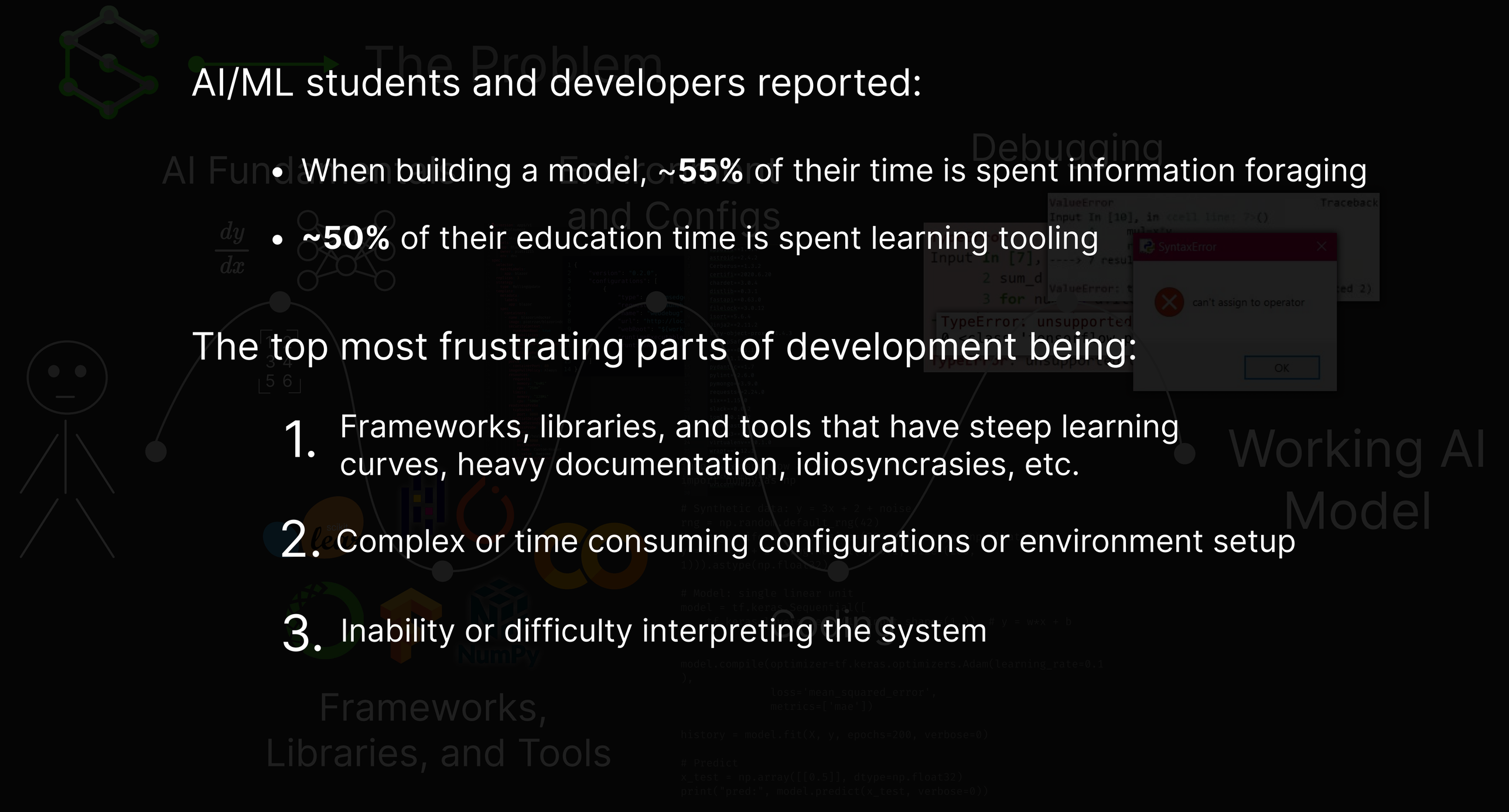
# Synthetic data: y = 3x + 2 + noise
rng = np.random.default_rng(42)
X = rng.uniform(-1, 1, size=(200, 1)).astype(np.float32)
y = (3 * X + 2 + rng.normal(0, 0.1, size=(200,
1))).astype(np.float32)

# Model: single linear unit
model = tf.keras.Sequential([
    tf.keras.layers.Dense(1, input_shape=(1,))  # y = w*x + b
])

model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.1
),
              loss='mean_squared_error',
              metrics=['mae'])

history = model.fit(X, y, epochs=200, verbose=0)

# Predict
x_test = np.array([[0.5]], dtype=np.float32)
print("pred:", model.predict(x_test, verbose=0))
```



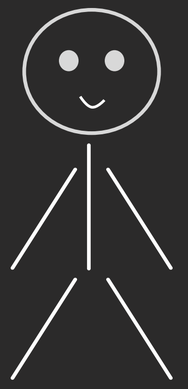
AI/ML students and developers reported:

- When building a model, **~55%** of their time is spent information foraging
- **~50%** of their education time is spent learning tooling

The top most frustrating parts of development being:

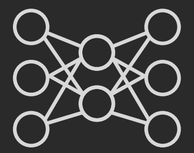
1. Frameworks, libraries, and tools that have steep learning curves, heavy documentation, idiosyncrasies, etc.
2. Complex or time consuming configurations or environment setup
3. Inability or difficulty interpreting the system

The Solution



AI Fundamentals

$$\frac{dy}{dx}$$



$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

A
m
con



Graph Space

lt
ly
n AI

Working AI Model

Example - Transformer

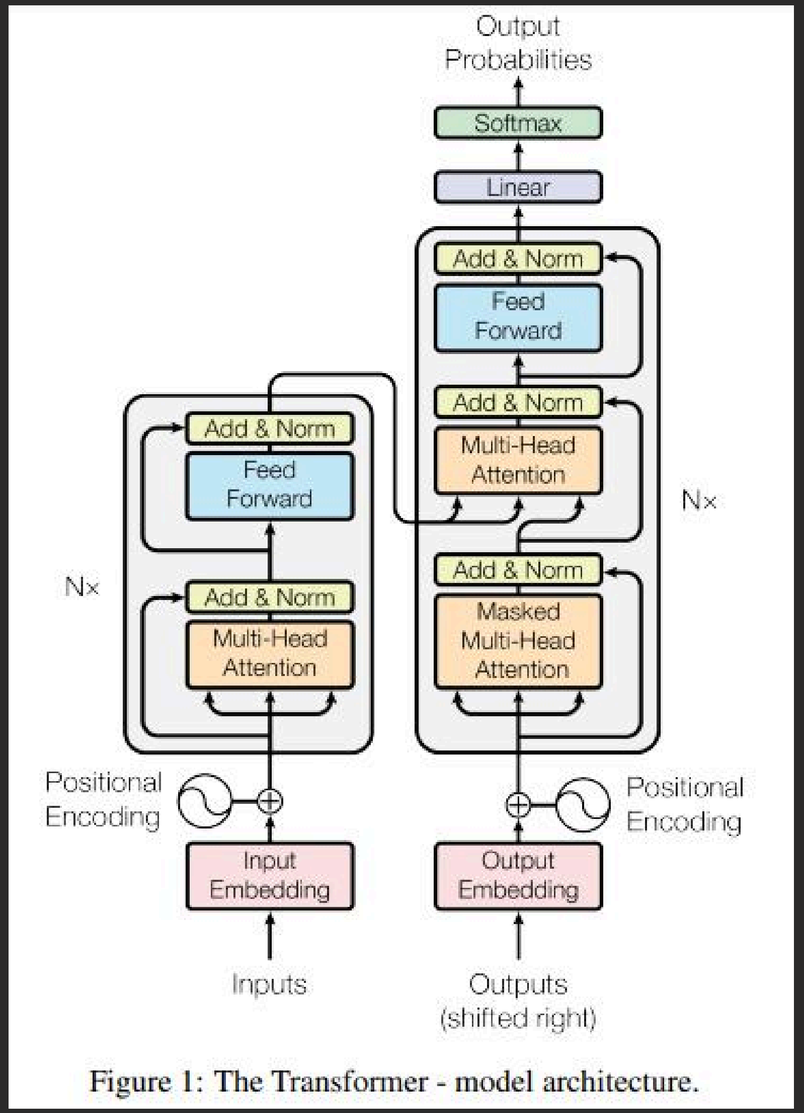
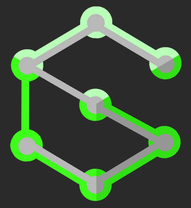


Figure 1: The Transformer - model architecture.



Example - Transformer (in code)

```python
def block(x, scope, *, past, hparams):
    with tf.variable_scope(scope):
        nx = x.shape[-1].value
        a, present = attn(norm(x, 'ln_1'), 'attn', nx, past=past, hparams=hparams)
        x = x + a
        m = mlp(norm(x, 'ln_2'), 'mlp', nx*4, hparams=hparams)
        x = x + m
        return x, present

def past_shape(*, hparams, batch_size=None, sequence=None):
    return [batch_size, hparams.n_layer, 2, hparams.n_head, sequence, hparams.n_embd // hparams.n_head]

def expand_tile(value, size):
    """Add a new axis of given size."""
    value = tf.convert_to_tensor(value, name='value')
    ndims = value.shape.ndims
    return tf.tile(tf.expand_dims(value, axis=0), [size] + [1]*ndims)

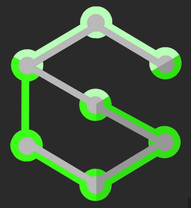
def positions_for(tokens, past_length):
    batch_size = tf.shape(tokens)[0]
    nsteps = tf.shape(tokens)[1]
    return expand_tile(past_length + tf.range(nsteps), batch_size)


def model(hparams, X, past=None, scope='model', reuse=False):
    with tf.variable_scope(scope, reuse=reuse):
        results = {}
        batch, sequence = shape_list(X)

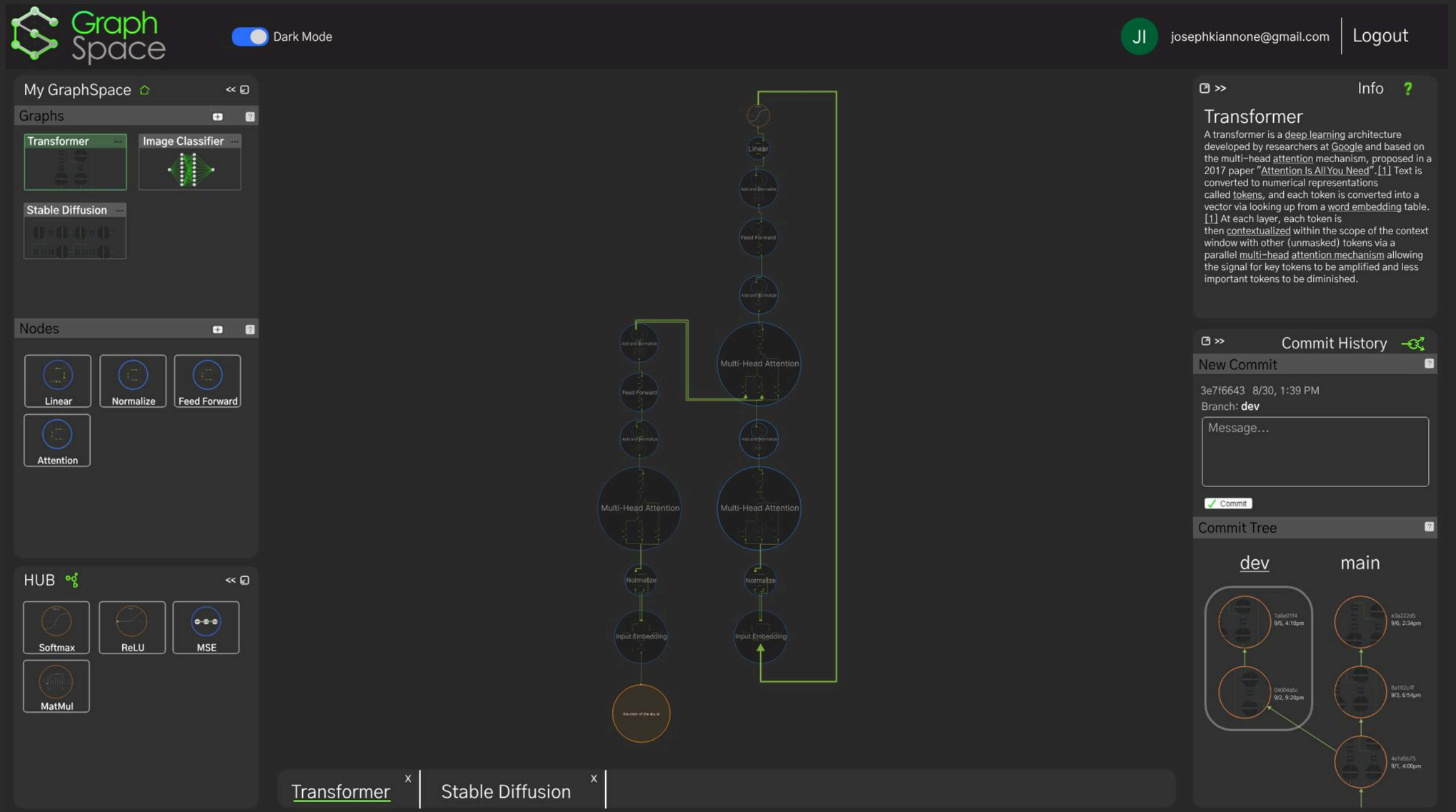
        wpe = tf.get_variable('wpe', [hparams.n_ctx, hparams.n_embd],
                              initializer=tf.random_normal_initializer(stddev=0.01))
        wte = tf.get_variable('wte', [hparams.n_vocab, hparams.n_embd],
                              initializer=tf.random_normal_initializer(stddev=0.02))
        past_length = 0 if past is None else tf.shape(past)[-2]
        h = tf.gather(wte, X) + tf.gather(wpe, positions_for(X, past_length))

        # Transformer
        presents = []
        pasts = tf.unstack(past, axis=1) if past is not None else [None] * hparams.n_layer
        assert len(pasts) == hparams.n_layer
        for layer, past in enumerate(pasts):
            h, present = block(h, 'h%d' % layer, past=past, hparams=hparams)
            presents.append(present)
        results['present'] = tf.stack(presents, axis=1)
        h = norm(h, 'ln_f')

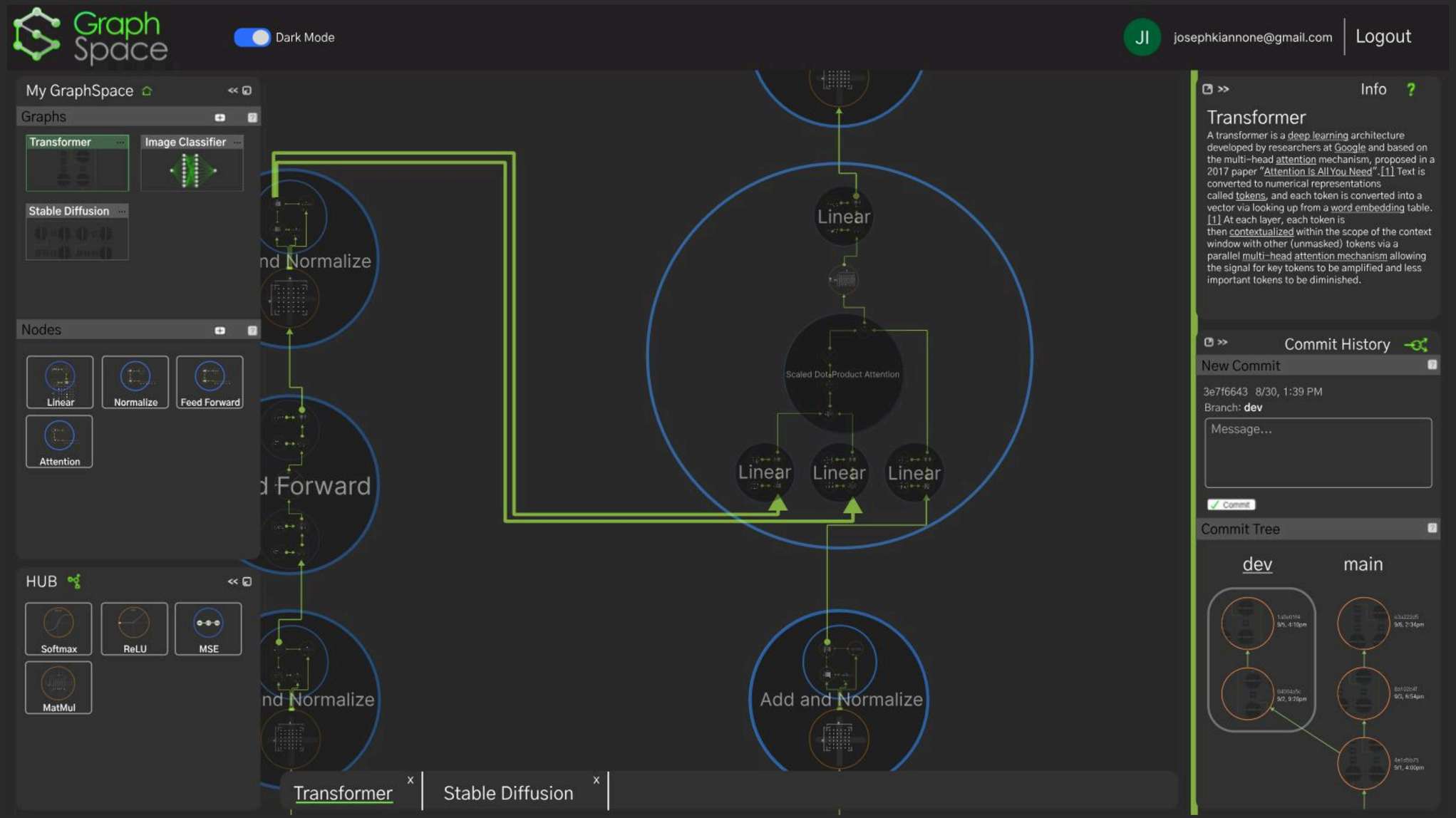
        # Language model loss.  Do tokens <n predict token n?
        h_flat = tf.reshape(h, [batch*sequence, hparams.n_embd])
        logits = tf.matmul(h_flat, wte, transpose_b=True)
        logits = tf.reshape(logits, [batch, sequence, hparams.n_vocab])
        results['logits'] = logits
        return results
```

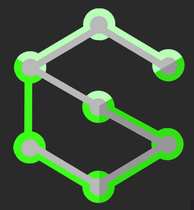


v1.0

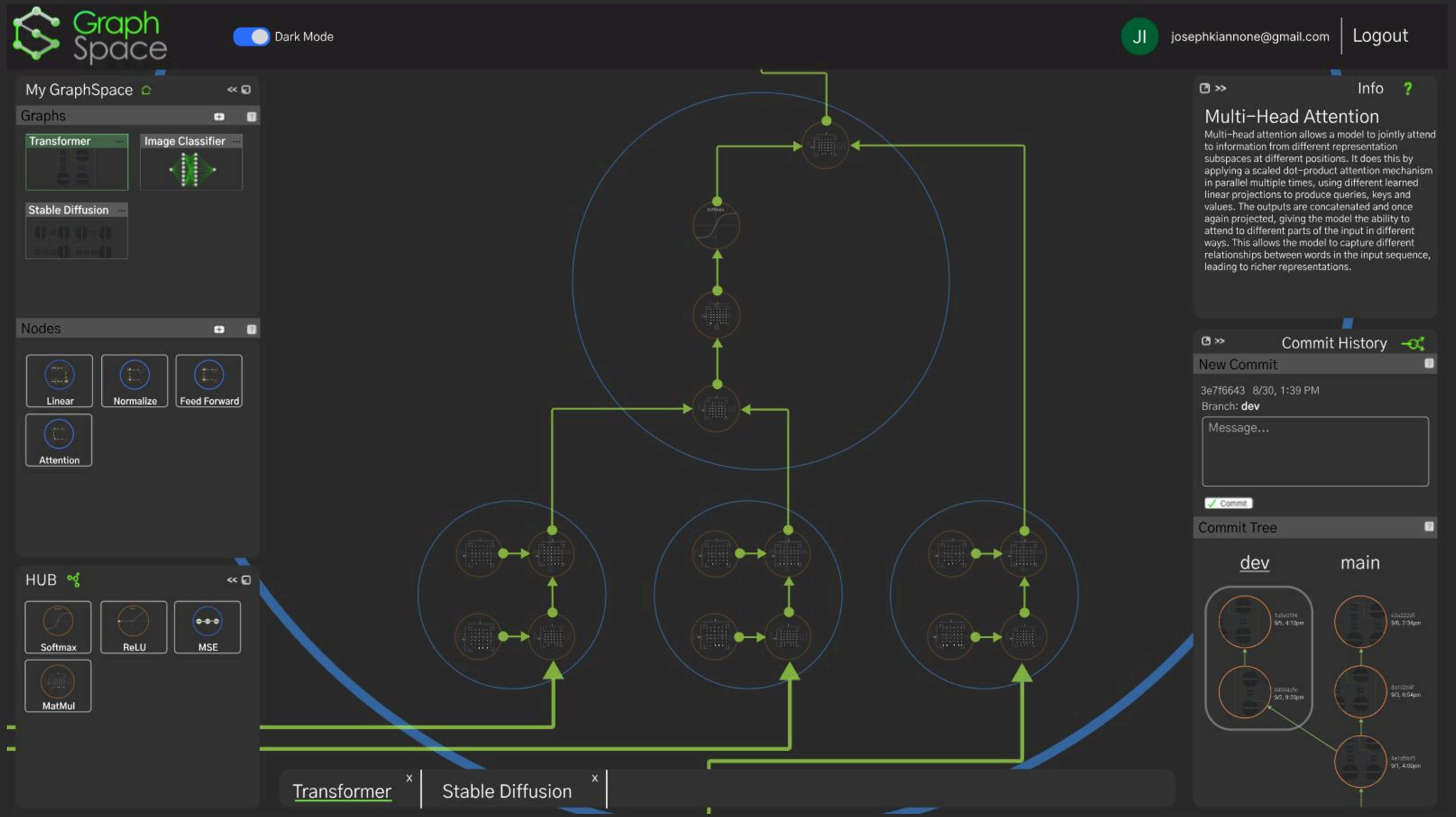

Graph Space
Dark Mode
JI
josephkiannone@gmail.com
Logout
My GraphSpace
Graphs
Transformer
Image Classifier
Stable Diffusion
Nodes
Linear
Normalize
Feed Forward
Attention
HUB
Softmax
ReLU
MSE
MatMul
Linear
Add and Normalize
Feed Forward
Add and Normalize
Multi-Head Attention
Add and Normalize
Feed Forward
Add and Normalize
Multi-Head Attention
Multi-Head Attention
Normalize
Normalize
Input Embedding
Input Embedding
Info
Transformer
A transformer is a deep learning architecture developed by researchers at Google and based on the multi-head attention mechanism, proposed in a 2017 paper "Attention Is All You Need". [1] Text is converted to numerical representations called tokens, and each token is converted into a vector via looking up from a word embedding table. [1] At each layer, each token is then contextualized within the scope of the context window with other (unmasked) tokens via a parallel multi-head attention mechanism allowing the signal for key tokens to be amplified and less important tokens to be diminished.
Commit History
New Commit
3e7f6643 8/30, 1:39 PM
Branch: dev
Message...
Commit
Commit Tree
dev
main
Transformer
Stable Diffusion

v1.0


Graph Space
Dark Mode
JI
josephkiannone@gmail.com
Logout
My GraphSpace
Graphs
Transformer
Image Classifier
Stable Diffusion
Nodes
Linear
Normalize
Feed Forward
Attention
HUB
Softmax
ReLU
MSE
MatMul
Linear
Scaled Dot-Product Attention
Linear
Linear
Linear
Add and Normalize
Info
Transformer
A transformer is a deep learning architecture developed by researchers at Google and based on the multi-head attention mechanism, proposed in a 2017 paper "Attention Is All You Need".[1] Text is converted to numerical representations called tokens, and each token is converted into a vector via looking up from a word embedding table. [1] At each layer, each token is then contextualized within the scope of the context window with other (unmasked) tokens via a parallel multi-head attention mechanism allowing the signal for key tokens to be amplified and less important tokens to be diminished.
Commit History
New Commit
3e7f6643 8/30, 1:39 PM
Branch: dev
Message...
Commit
Commit Tree
dev
main
Transformer
Stable Diffusion



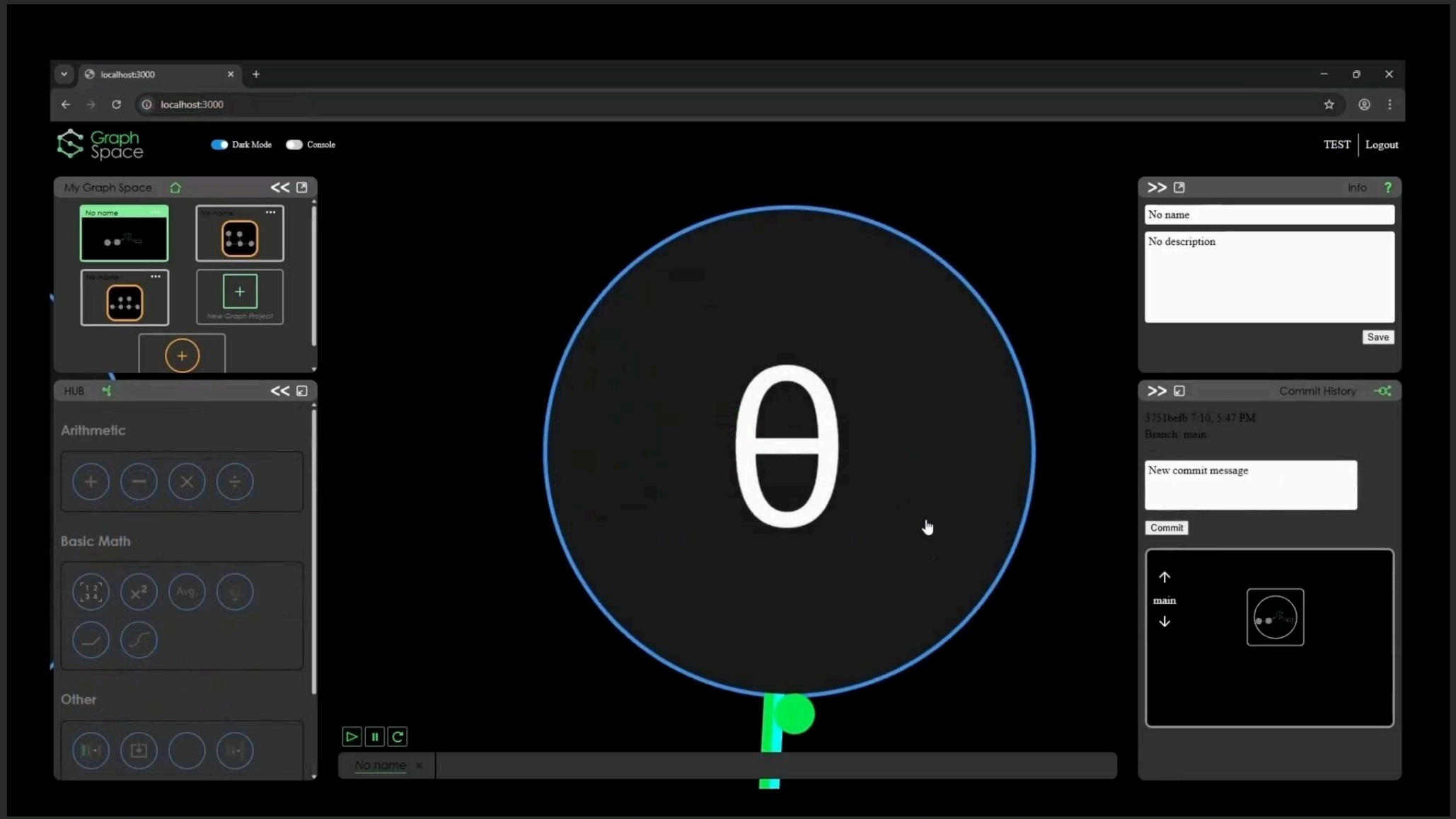
v1.0


Graph Space
Dark Mode
JI
josephkiannone@gmail.com
Logout
My GraphSpace
Graphs
Transformer
Image Classifier
Stable Diffusion
Nodes
Linear
Normalize
Feed Forward
Attention
HUB
Softmax
ReLU
MSE
MatMul
Info
Multi-Head Attention
Multi-head attention allows a model to jointly attend to information from different representation subspaces at different positions. It does this by applying a scaled dot-product attention mechanism in parallel multiple times, using different learned linear projections to produce queries, keys and values. The outputs are concatenated and once again projected, giving the model the ability to attend to different parts of the input in different ways. This allows the model to capture different relationships between words in the input sequence, leading to richer representations.
Commit History
New Commit
3e7f6643 8/30, 1:39 PM
Branch: dev
Message...
Commit
Commit Tree
dev
main
Transformer
Stable Diffusion

Prototype - Building a Model



Prototype - Training



Prototype - Importing and Manipulating



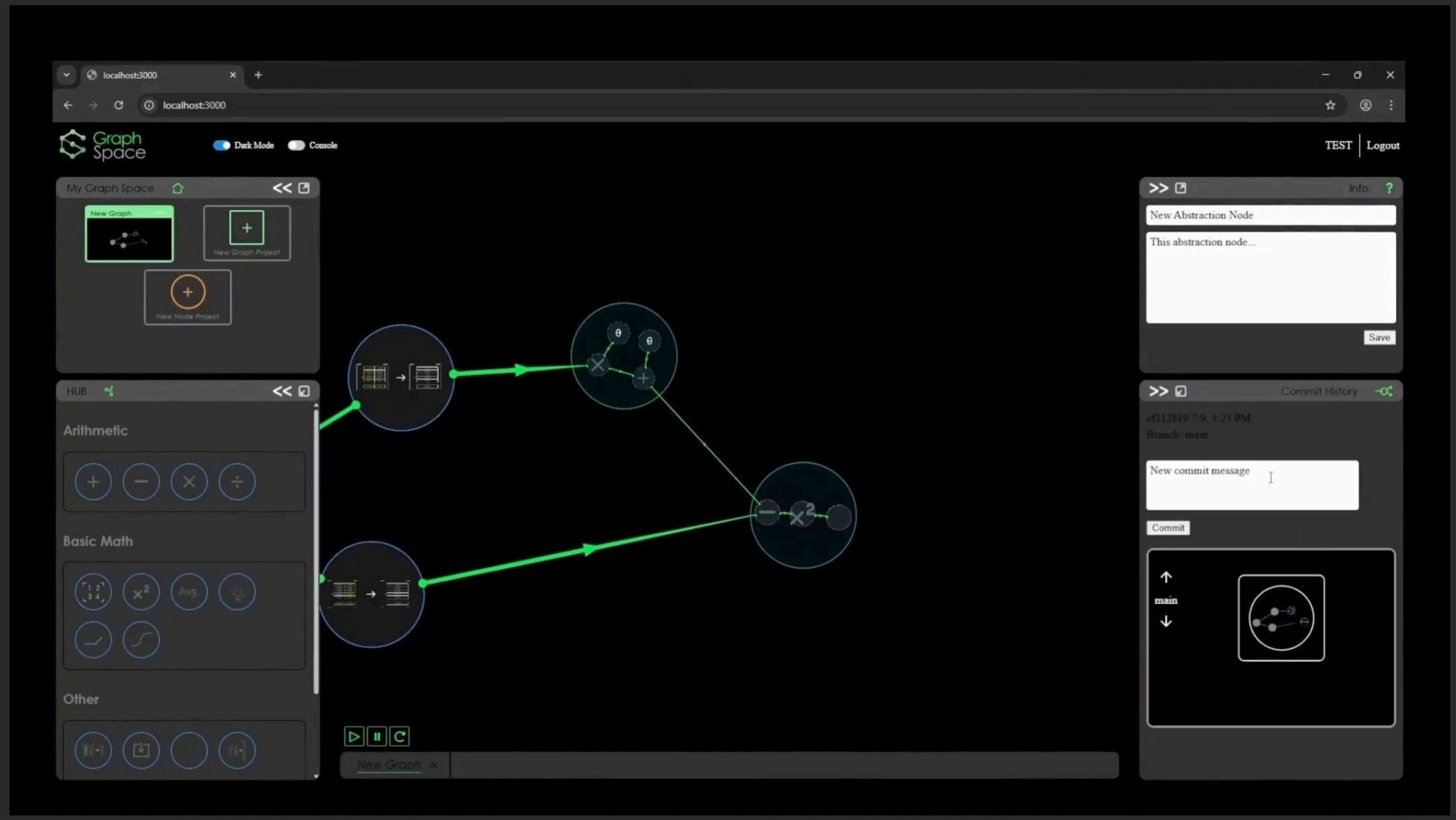
Prototype - Creating Parameters



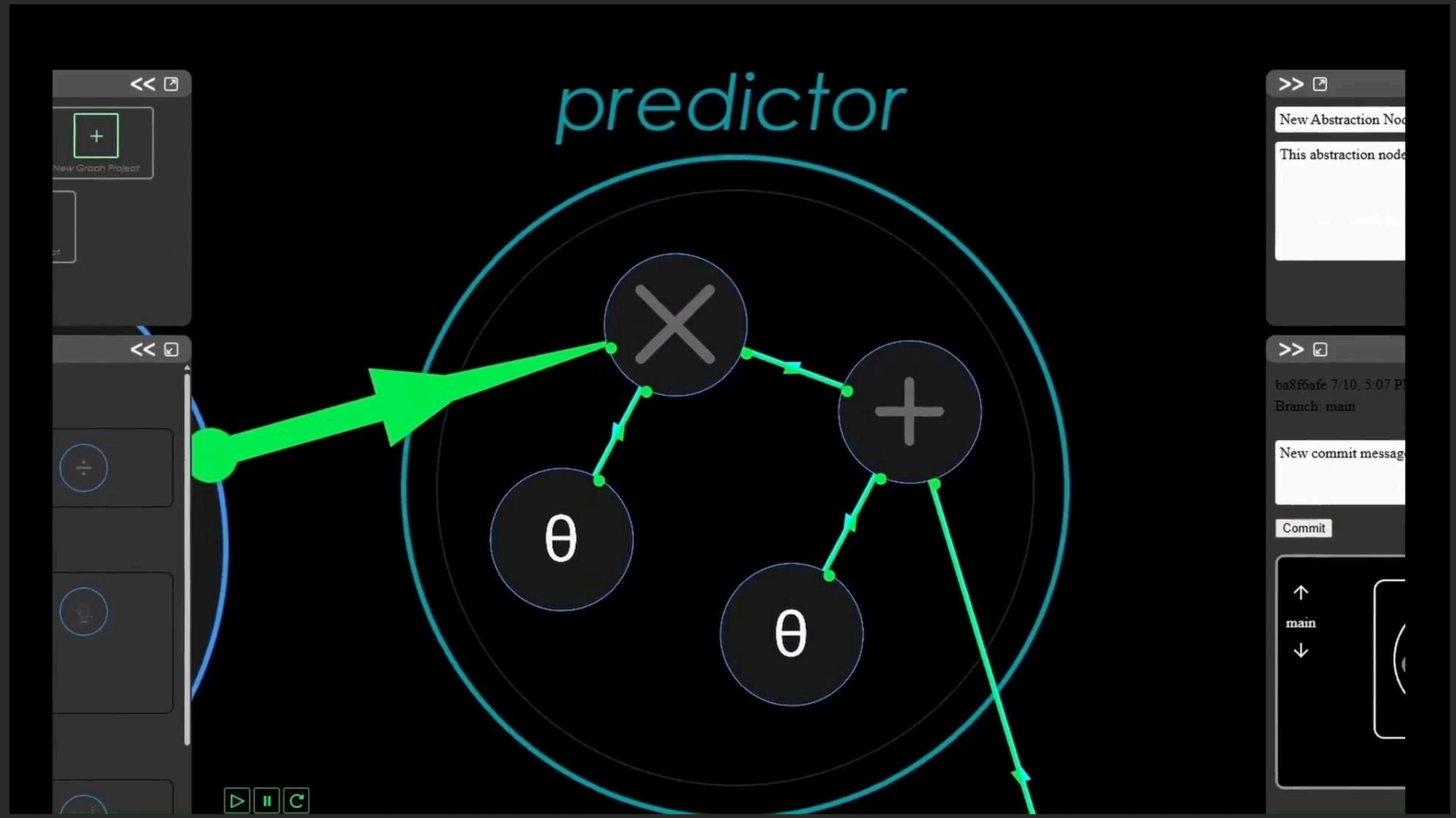
Prototype - Arrays



Prototype

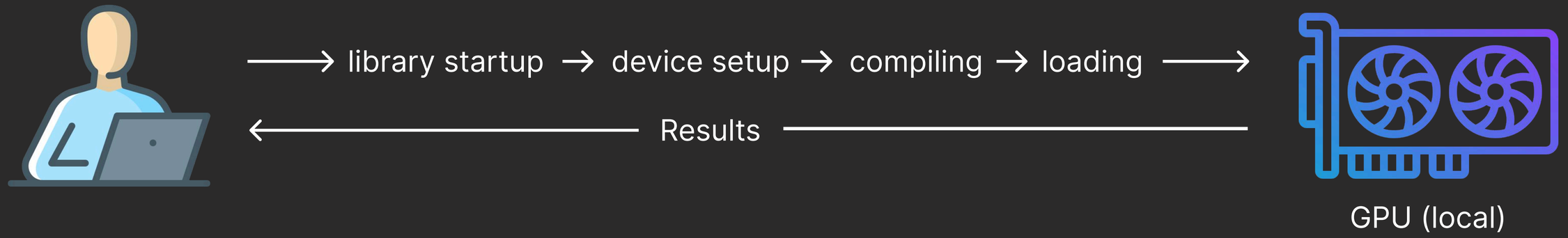

localhost:3000
Graph Space
Dark Mode
Console
TEST
Logout
My Graph Space
New Graph
New Graph Project
New Node Project
HUB
Arithmetic
Basic Math
Other
Info
New Abstraction Node
This abstraction node...
Save
Commit History
New commit message
Commit
main
New Graph

Prototype

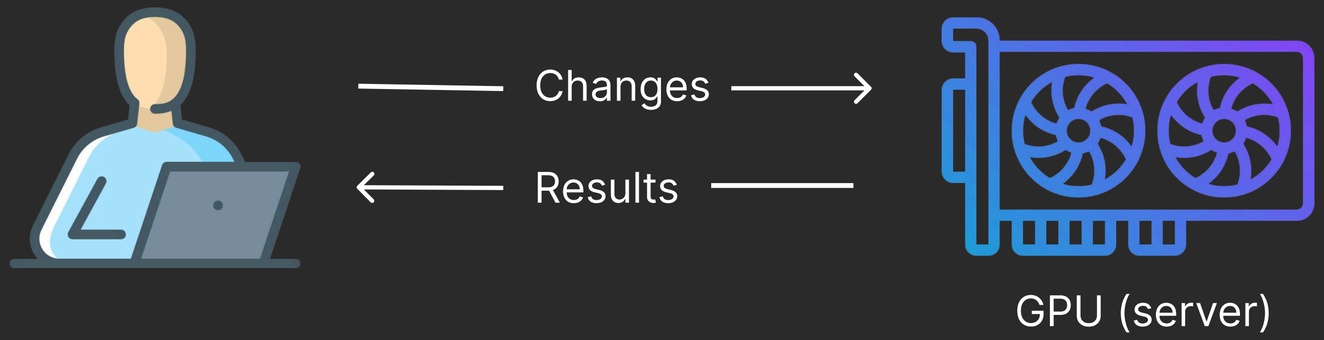

predictor
New Graph Project
New Abstraction Noc
This abstraction node
Branch: main
New commit messag
Commit
main
θ
θ

Additional Benefits

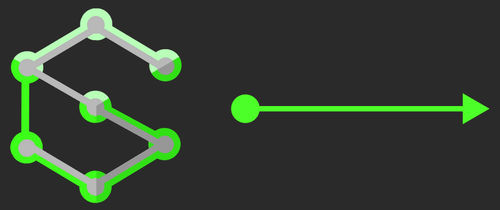
Standard Feedback Loop:



GraphSpace Feedback Loop:



- Compute Graph is already in GPU memory, and updated as the user makes changes
- **Almost zero latency** (a few milliseconds for information to travel from server to client)
- **Zero environment configuration** → choose hardware and go



Validation

After a demonstration to 15 engineers, researchers, and data scientists:

- On a 1 to 10 scale they rated it **8.6** for usefulness explaining AI/ML systems and **7.6** for building
- **83%** said they could imagine themselves using it
- **67%** signed up for beta access

Market

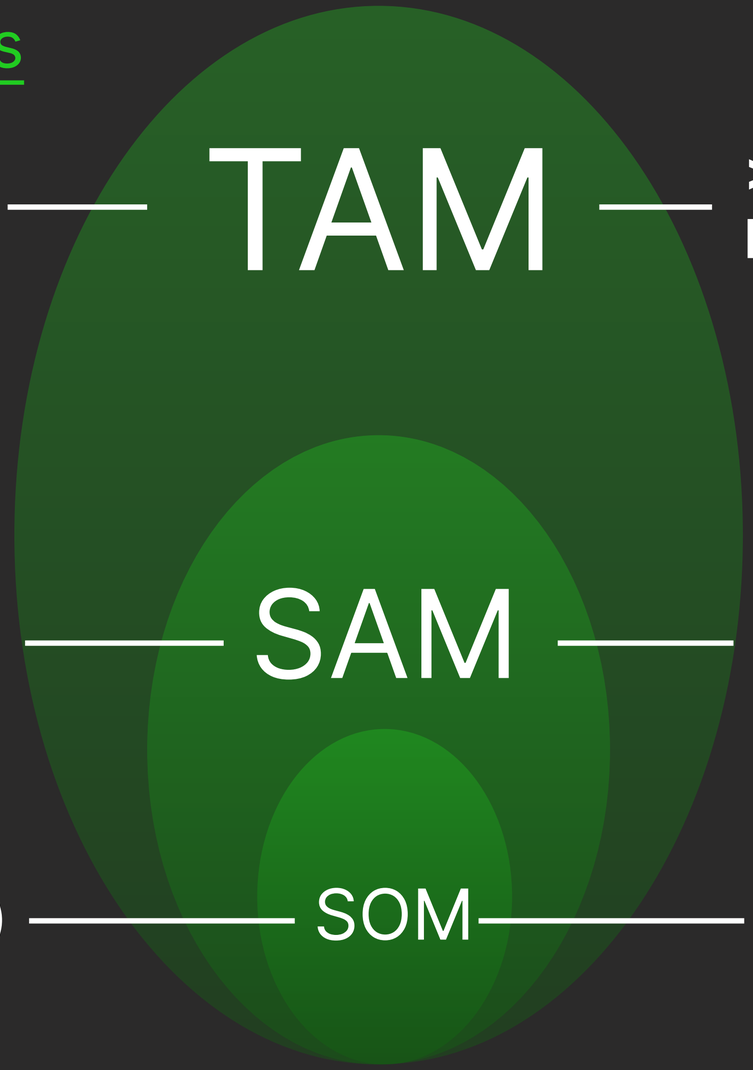
AI/ML Developer Platforms

Total market: ~$25 Billion @ ~33% CAGR

- Core frameworks
- Cloud services
- Tools

~$10 Billion (40%) from:

- Fast prototyping, no-code tools
- Visual development
- MLOps: $4.12B, AutoML: $2.7B



AI/ML Online Education

>50 million people actively seeking ML/AI education @ ~50% CAGR

- ~29 million ML course enrollments between Coursera, Udemy, and edX.
- Kaggle has >25 million members
- DataCamp has >11 million members

~25 million (50%) would find it more beneficial to learn visually and actively

- Coached through building models with visual representations

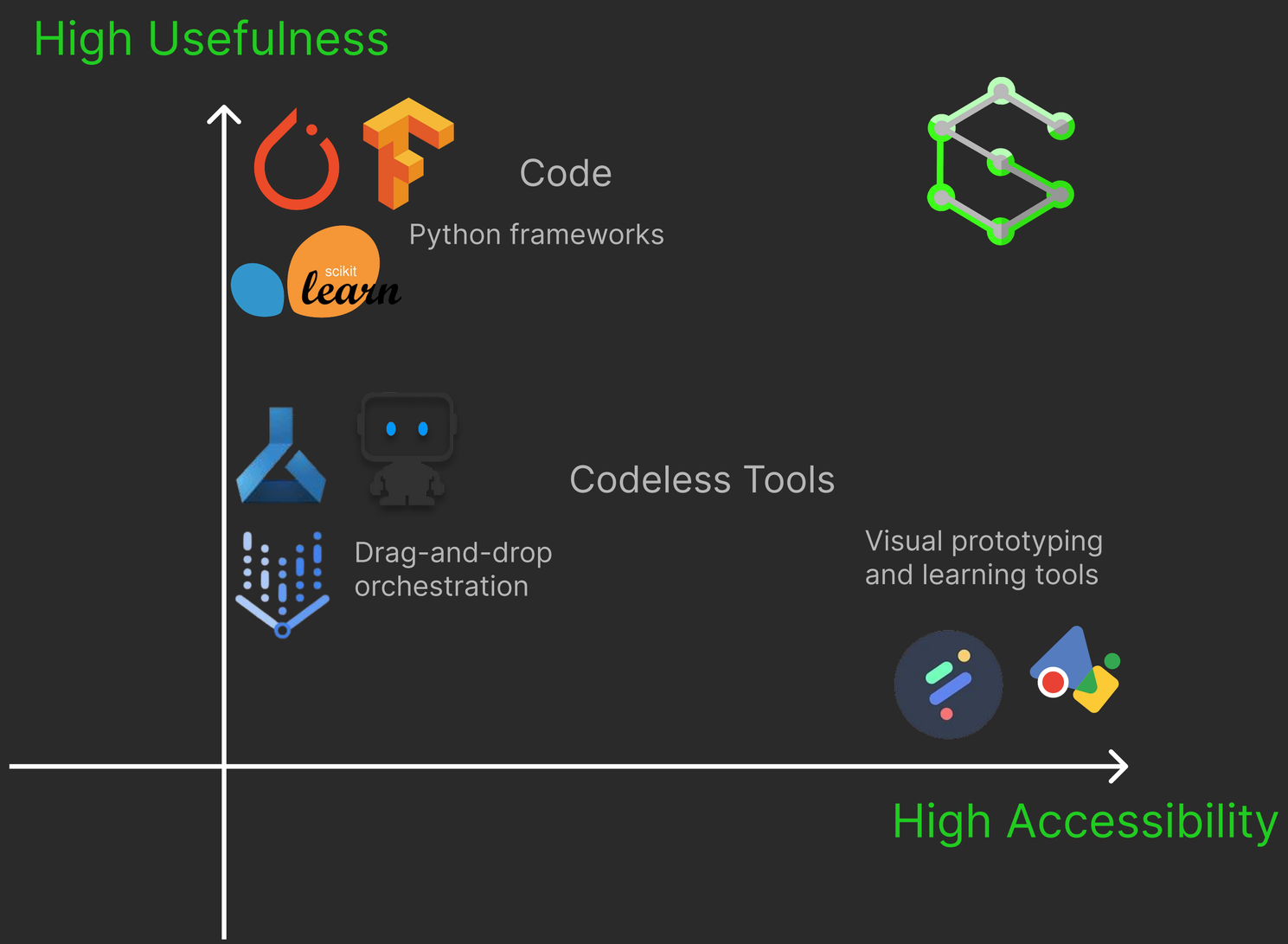
~$100M (1%)

~750k (3%) * $240 ARPU = $180M

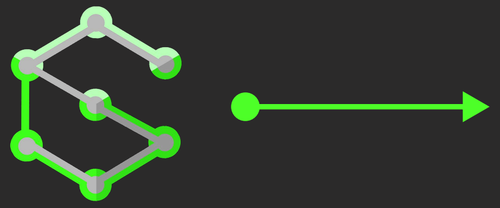
Sources:

- Machine Learning Platforms Market Size, Share, Trends, Industry Analysis Report (Polaris Market Research)

Competition



- **Codeless is the future.** But not like this.
- **One-size-fits-all** → Can’t address the nuance requirements of different domains
- **Fixed abstraction level (either high-level or low-level)** → Blocks off parts of the system which could be central to the user's problem space
- These platforms are codeless for the sake of being codeless → **don't add value** beyond what developers already use



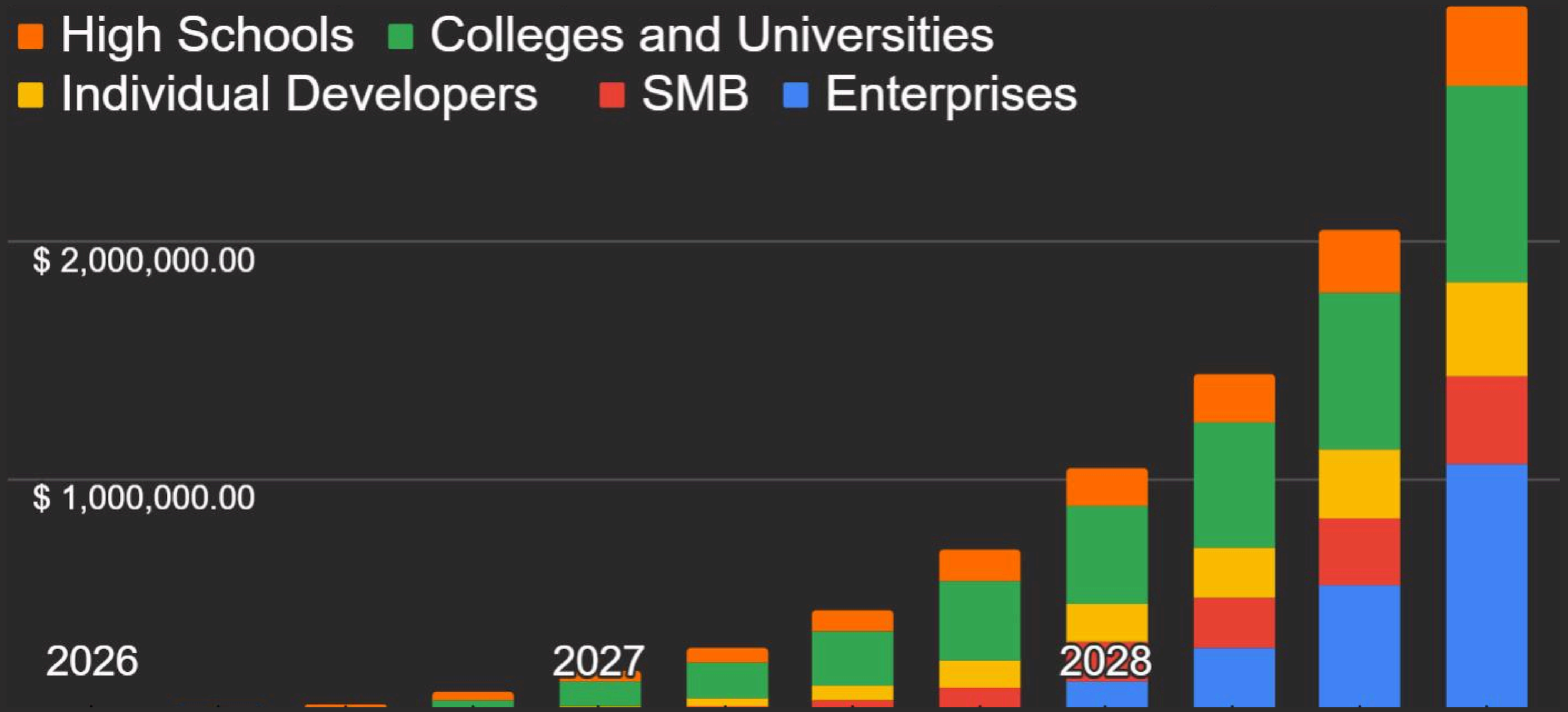
Business Roadmap

Strategy: Staggered market entry over the course of 3 years.

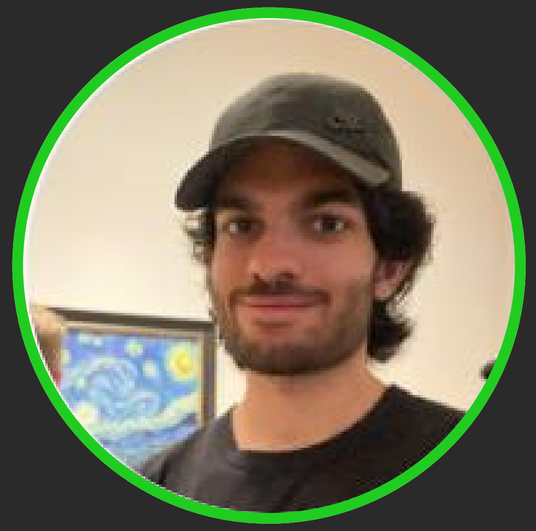
- Allows for rapidly iterating on the product in a market before advancing to a larger, more demanding market

	Year 1 (2026)			Year 2 (2027)			Year 3 (2028)		
Added Segment	**Colleges, Universities, High Schools**			**SMBs**	**Individual Developers, Learners**		**Enterprises**		
Pricing Monthly	$20/student			$500 + $50/seat	$20		$10k + $1k/seat		
Initial Target for Q1	20			30	1000		5		
Growth For years 1, 2, 3	2x	1.5x	1.3x	-	2x	1.5x	-	-	2x
Focus	• Visual, clear, and comprehensive UI • Basic functionality • Large models			• Data pipelines • Integrating cloud platforms • Copilot (LLM + text2model) • Private repositories			• Mechanistic interpretability tools • Converting legacy code • Collaboration in the same environment		

Business Roadmap (cont.)



About Me



Joseph Iannone

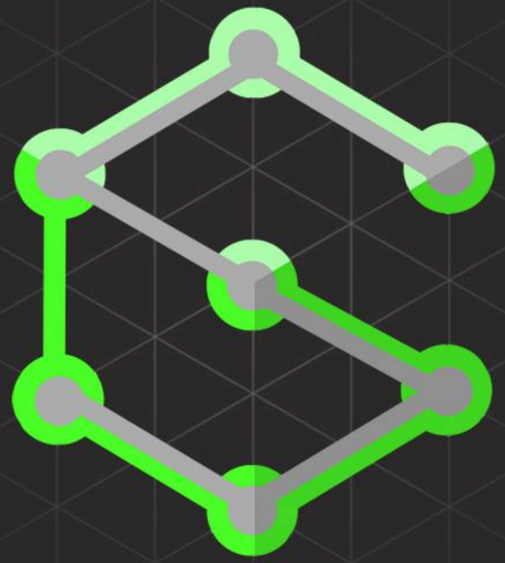
- Programming since ~13
- Coded the prototype for GraphSpace and two mobile apps
- 5+ years experience full-stack development experience
- Studied Economics and Computer Science at the University of Washington (graduated in 2023)
- Created the financial runway to dedicate myself full-time towards building GraphSpace through strategic investing

Ask

Seeking $500k

- Build a team of 3
- Launch a beta in 6 months
- Product/market fit and $100k in ARR in 12 months

Phase 1	Phase 2	Phase 3	Phase 4	Phase 5
• Hire ML engineer • Hire frontend dev • Hire backend dev	Develop product	Launch	Reach 100 educational institutions	~$100k ARR



GraphSpace

Contact:
joe@graphspace.dev
206-348-3202