



# Zero-Shot Text-to-Image Generation for Housing Floor Plans

By

Shannon Phu

Shiv Kumar Ganesh

Kumuda B G Murthy

Raghava D Urs

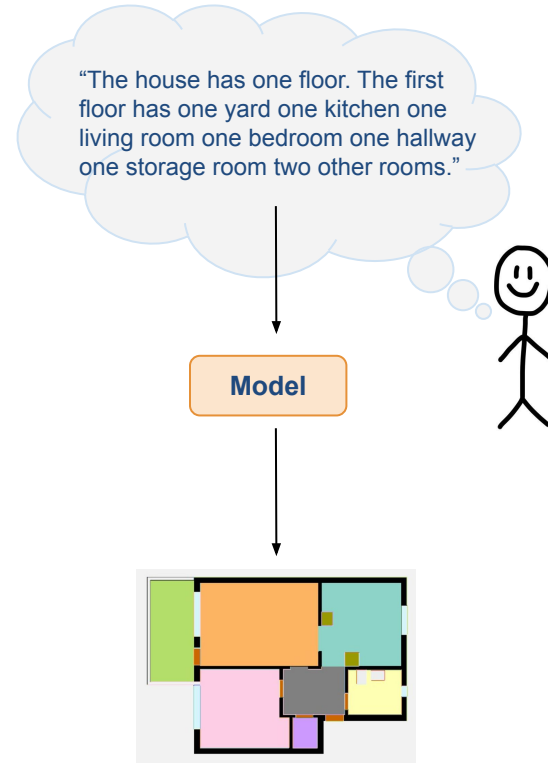


# Goal

Floor plans help you envision a space and understand how it will look when construction or renovations are complete.

Interior designers, pro builders, and real estate agents can use these floor plans when they are looking to design or sell a new home or property.

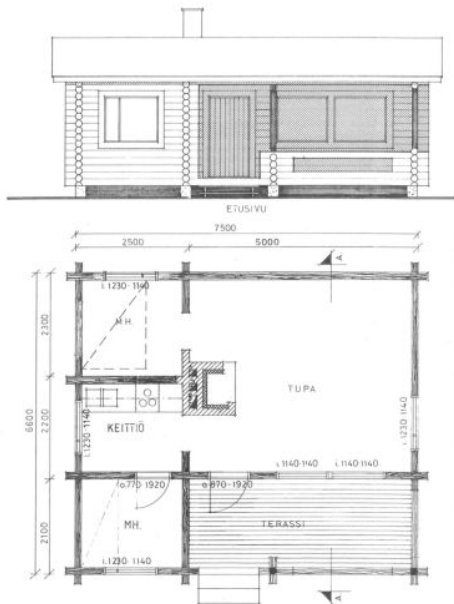
Such users can just enter the text description of floor plan and our model should give out the image representing the text entered.



# Data Collection

Source : [CubiCasa Dataset](#)  
Data Size : 5k housing floor plan images  
Data Format : Floor plan Images in SVG format

Original Image



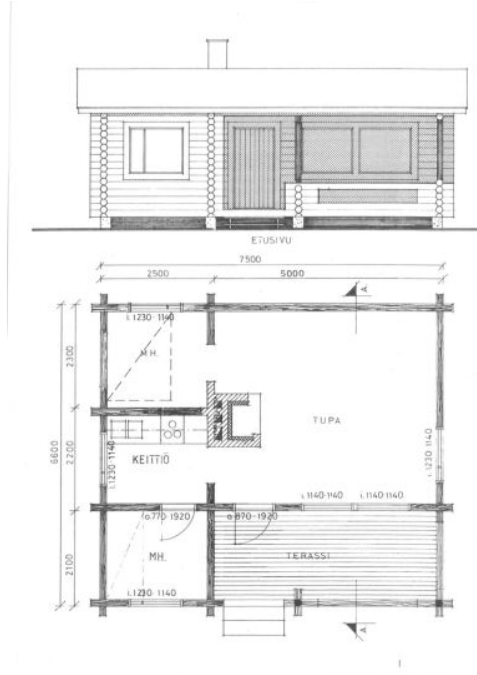
Original SVG

```
lc: /content/cubicasa5k/{instance_dir}/model.svg

<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" height="1369.6700439453125" ve
  <g class="BoundaryPolygon" fill="rgb(255, 255, 255)" stroke="rgb(0, 0, 0)">
    <polygon points="0.00,0.00 215.40,0.00 215.40,60.00 0.00,60.00 "/>
  </g>
  <g class="Direction" fill-opacity="0.9" style="display: none;">
    <polygon points="114.91,55.20 114.91,44.58 111.90,44.58 111.90,55.20 108.10,55.20 113.40,60.00 11
  </g>
  <g class="Name" fill-opacity="0.9" style="display: none;">
    <text transform="translate(3 12)" font-size="12" font-family="Verdana, Sans-Serif">CB</text>
  </g>
  <desc>Width:60 Height:90 Depth:60 Elevation:0</desc>
</g><g class="FixedFurniture DoubleSink" transform="matrix(1,0,0,1,221.0756,897.0606)" style="fill-opacit
  <g class="InnerPolygonLeft" fill="rgb(255, 255, 255)" stroke="rgb(0, 0, 0)">
    <polygon points="5.00,9.90 36.00,9.90 36.00,47.59 5.00,47.59 "/>
  </g>
  <g class="InnerPolygonRight" fill="rgb(255, 255, 255)" stroke="rgb(0, 0, 0)">
    <polygon points="39.23,10.00 69.85,10.00 69.85,47.68 39.23,47.68 "/>
  </g>
  <g class="OuterDrain" fill="rgb(255, 255, 255)" stroke="rgb(0, 0, 0)" stroke-width="0.8">
    <circle cx="21.170965431359104" cy="28.174781972274896" r="3.5"/>
    <circle cx="53.94834845960544" cy="28.174781972274896" r="3.5"/>
  </g>
  <g class="InnerDrain" fill="rgb(255, 255, 255)" stroke="rgb(0, 0, 0)" stroke-width="0.8">
    <circle cx="21.170965431359104" cy="28.174781972274896" r="2"/>
    <circle cx="53.94834845960544" cy="28.174781972274896" r="2"/>
  </g>
  <g class="Faucet" fill="rgb(255, 255, 255)" stroke="rgb(0, 0, 0)">
    <polygon points="34.39,4.25 40.39,4.25 40.39,18.25 34.39,18.25 "/>
  </g>
  <g class="Direction" fill-opacity="0.9" style="display: none;">
    <polygon points="39.24,47.42 39.24,36.80 36.23,36.80 36.23,47.42 32.42,47.42 37.73,52.72 43.03,47
  </g>
  <g class="Name" fill-opacity="0.9" style="display: none;">
    <text transform="translate(3 12)" font-size="12" font-family="Verdana, Sans-Serif">DSINK</text>
  </g>
  <desc>Width:70 Height:90 Depth:60 Elevation:0</desc>
</g><g class="FixedFurniture ElectricalAppliance IntegratedStove" transform="matrix(1,0,0,1,329.5773,897.
  <g class="BoundaryPolygon" fill="rgb(255, 255, 255)" stroke="rgb(0, 0, 0)">
```

# Image Pre-processing

Original Image



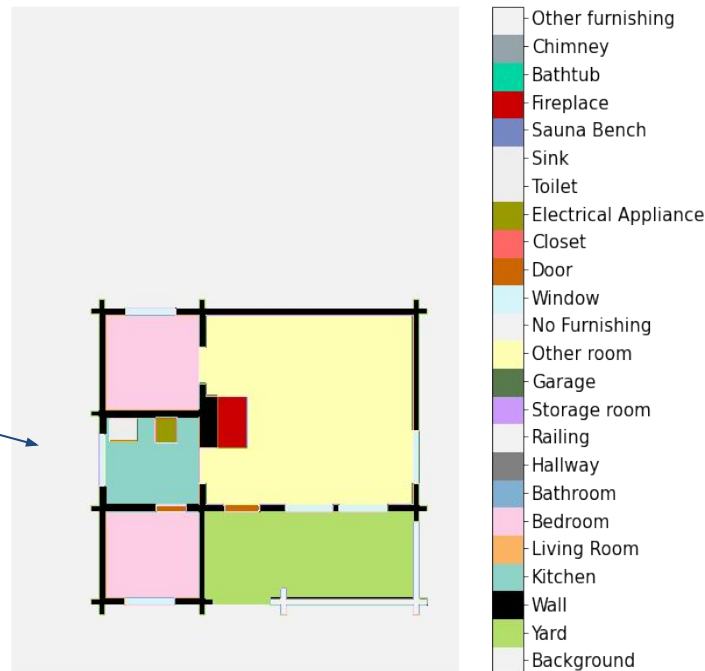
Generated annotated image



# Text Generation

- Original dataset did not provide natural text descriptions
- We parsed the SVG to generate the text descriptions
- Example
  - “The house has one floor. The first floor has one yard one kitchen two bedrooms one other room.”

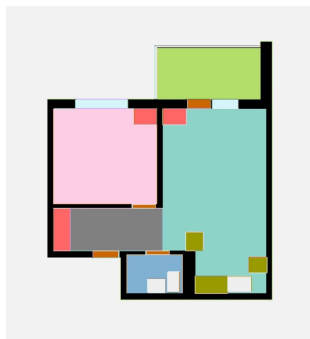
Generated annotated image



# Dataset Filtration

- We used only houses with one story to simplify the problem

1 floor



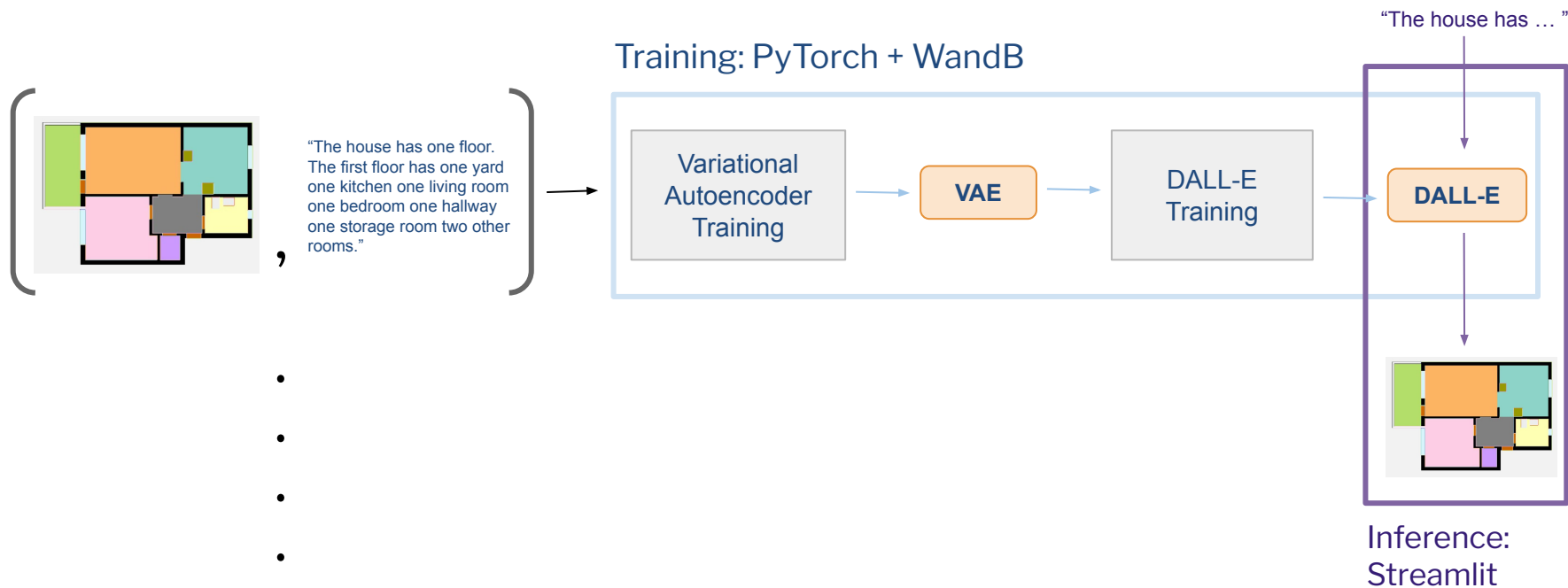
2 floors



3 floors

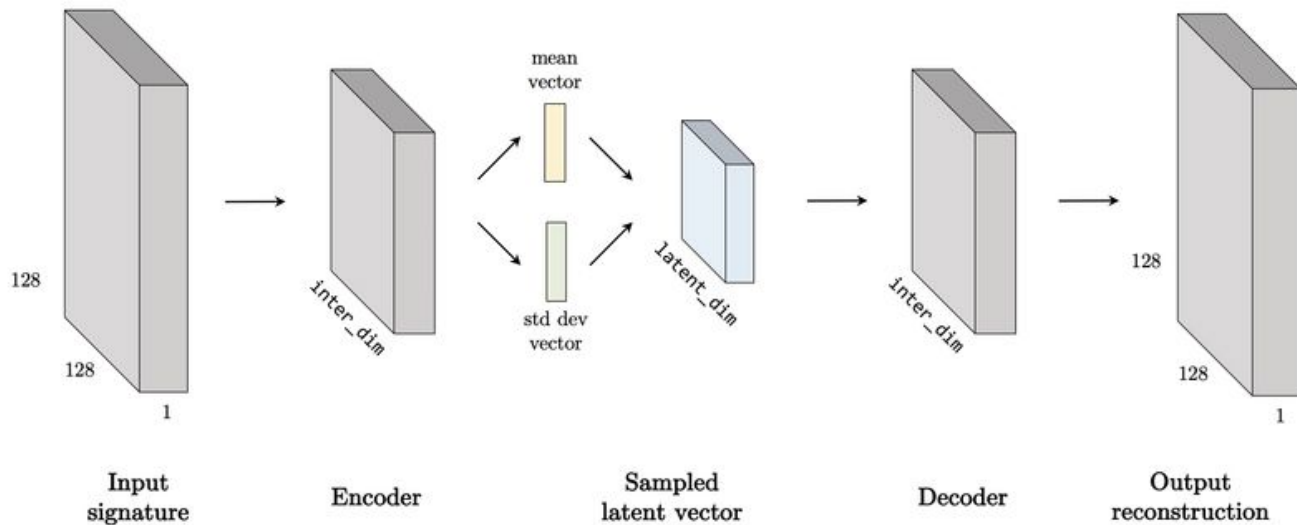


# Model Architecture



# What is VAE

A variational autoencoder is an architecture composed of both an encoder and a decoder and that is trained to minimize the reconstruction error between the encoded-decoded data and the initial data.

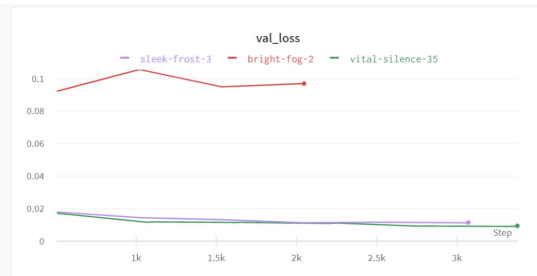


System diagram of VAE architecture ( source: <https://www.researchgate.net> )



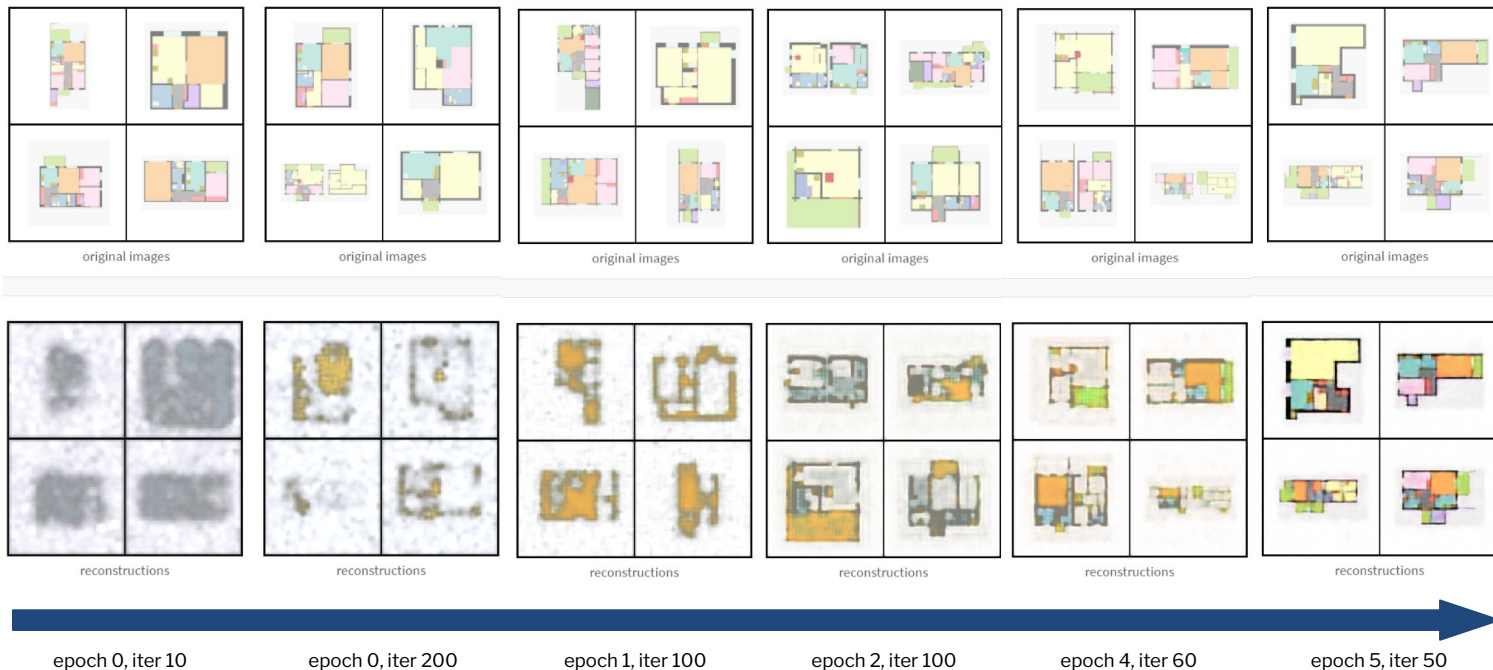
# VAE Training

- Train/test: 90/10 split
- Resize images to 128 px
- Image augmentations
  - *sweeping resize*
- Epochs: 6
- Batch size: 6
- Learning rate: 1e-3
- # layers: 3
- Embedding # dimensions: 512
- Loss function: reconstruction loss and KL divergence loss



Experiments	Training Loss	Validation Loss
6 epochs, 512 dim	0.012	0.009
5 epochs, 512 dim	0.010	0.012
4 epochs, 256 dim	0.081	0.107

# VAE Results



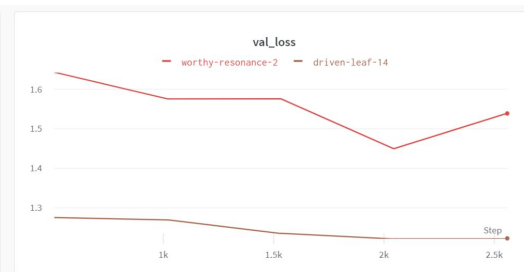
# What is DALL-E

DALL-E is an extension of GPT-3 by creating images from text descriptions for a wide range of concepts expressible in natural language.

DALL-E is an autoregressive transformer decoder model. It accepts concatenated encoded text tokens with the encoded image embeddings to train the transformer auto-regressively. The training minimizes cross-entropy loss for a combination of the text and image representations.

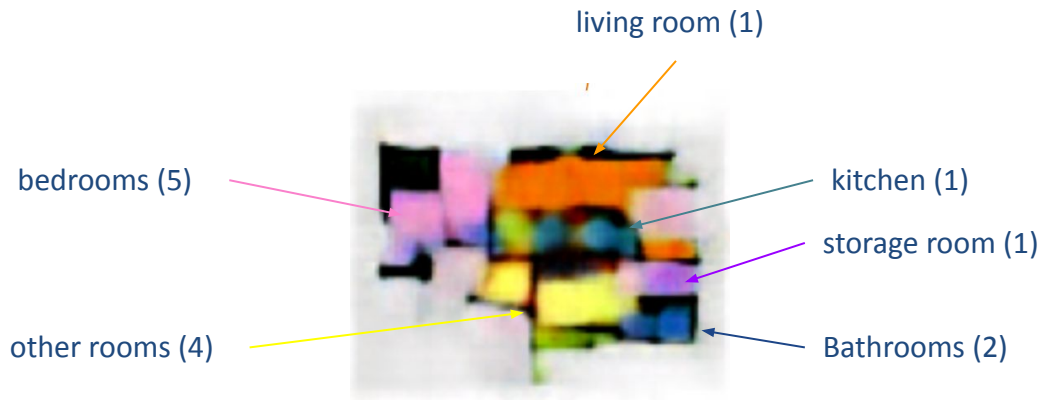
# DALL-E Training

- Resize images to 128 px
- Train/test: 90/10 split
- Image augmentations
  - *sweeping resize*
- Epochs: 5
- Batch size: 4
- Learning rate:  $1e-3$
- Seq length: 256 tokens
- # heads: 4
- Embedding # dimensions: 512
- Loss function: cross-entropy weighted  $\frac{7}{8}$  image loss +  $\frac{1}{8}$  text loss



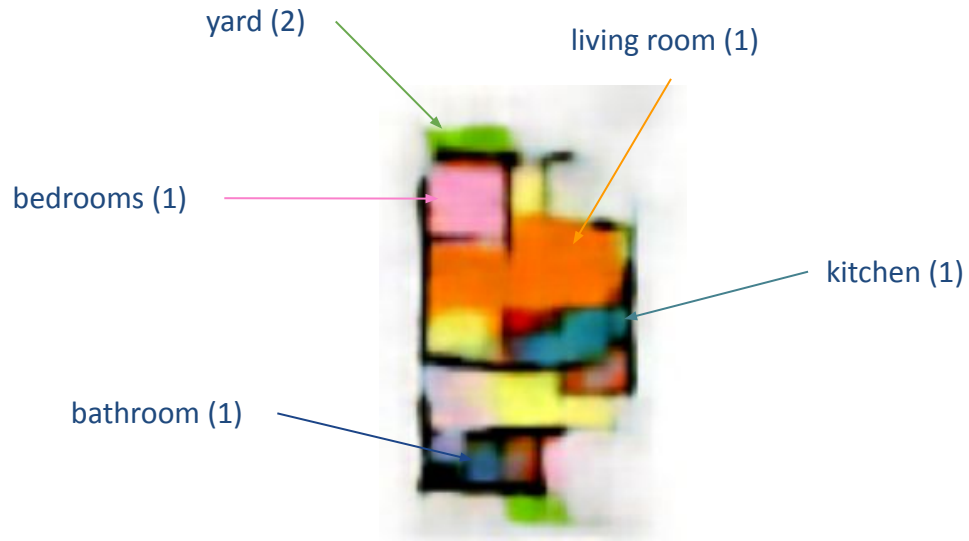
Experiments	Training Loss	Validation Loss
5 epochs, 512 dim	1.244	1.252
4 epochs, 256 dim	1.443	1.572

# Example 1



the house has one floor. the first floor has one yard one kitchen one living room five bedrooms two bathrooms three hallways one storage room four other rooms.

# Example 2



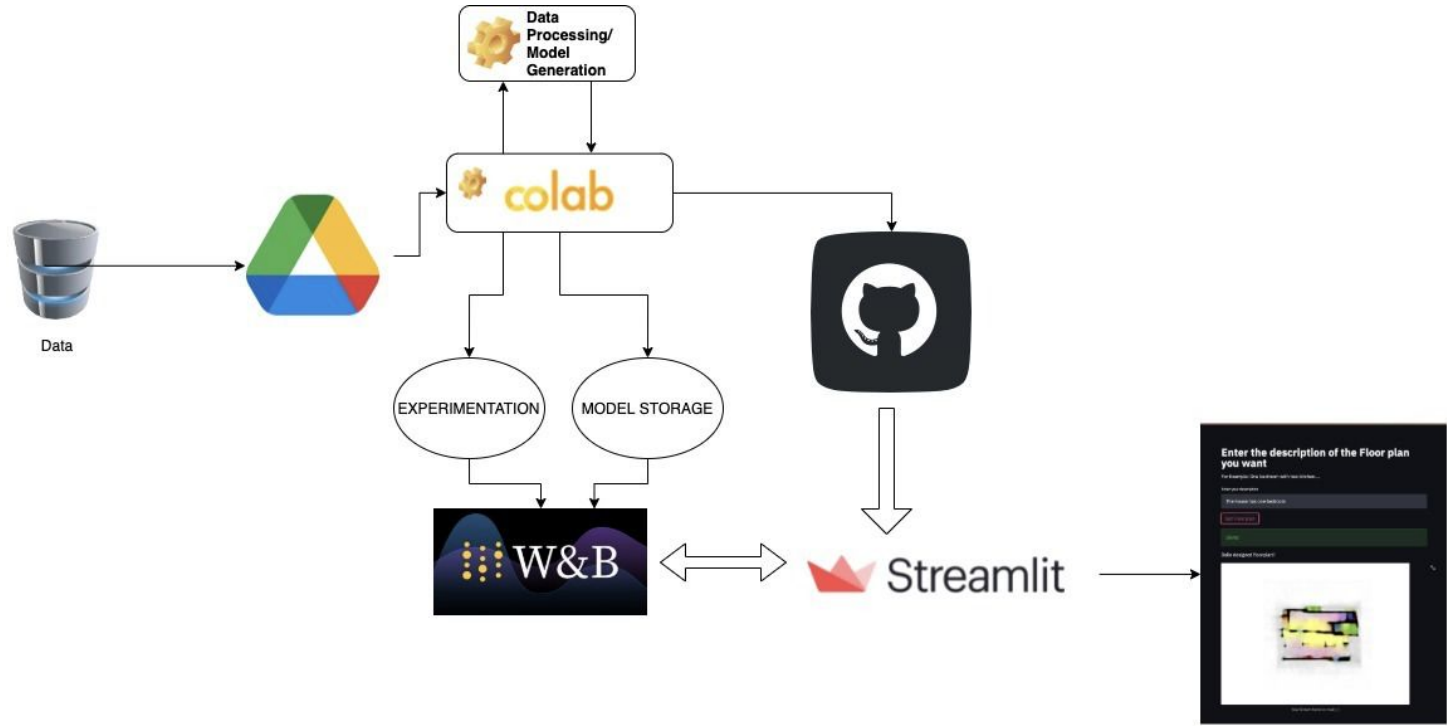
the house has one floor . the  
first floor has one yard one  
kitchen one living room one  
bedroom one bathroom .

# Example 3



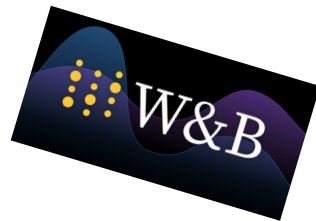
the house has one floor . the first floor has three yards one kitchen one living room three bedrooms two bathrooms one hallway one storage room three other rooms .

# MLOps Architecture

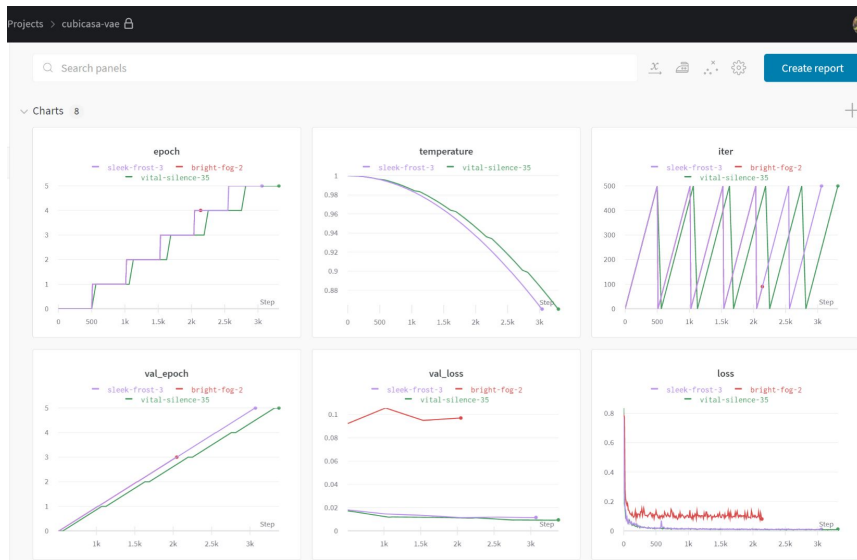




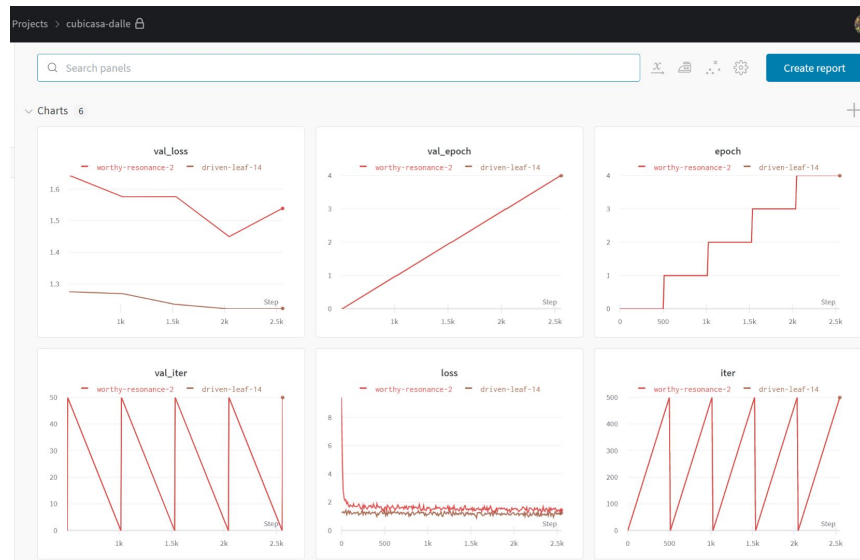
# MLOps - WandB



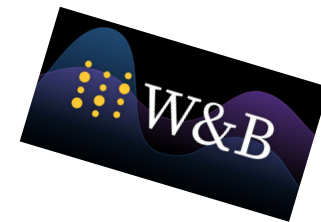
## VAE Training Pipeline Runs



## DALL-E Training Pipeline Runs



# MLOps - WandB



## Model Tracking/Logging

The screenshot shows the WandB Model Tracking/Logging interface. The breadcrumb path is: sjsu-cmpe-258-musketeers > Projects > cubicasa-dalle > Artifacts > model > trained-dalle > 186901ef6. The left sidebar shows a tree view with 'trained-dalle' expanded, listing versions v0, v1, v2, v3, v4, and 'v5 latest'. The main panel has tabs for Overview, API, Metadata, Files, and Graph view. The 'Overview' tab is active, displaying the following information:

- Version: trained-dalle:v5
- Digest: 865d55cdc539bb04fd4cf9a6fb7aaab2
- State: committed
- Created: April 28th, 2021 at 10:42:57 pm
- Aliases: latest v5 +
- Notes: What changed in this revision?
- Output by: worthy-resonance-2

Below this, the 'Used by' section shows a table of runs that used this model:

Run	Group	User	Created	Duration
eternal-feather-1		shivkumarga...	2021-04-30T22:52:38	2h 43s
dashing-salad-2		shivkumarga...	2021-05-02T06:22:24	10m 2s
lilac-pond-3		shivkumarga...	2021-05-02T06:32:28	2h 2m 49s

## Model Versioning/Repository

The screenshot shows the WandB Model Versioning/Repository interface. The breadcrumb path is: sjsu-cmpe-258-musketeers > Projects > cubicasa-dalle > Artifacts > model > trained-dalle > 186901ef6 > files. The left sidebar shows a tree view with 'trained-dalle' expanded, listing versions v0, v1, v2, v3, v4, and 'v5 latest'. The main panel has tabs for Overview, API, Metadata, Files, and Graph view. The 'Files' tab is active, displaying a file named 'dalle-final.pt' with a size of 334.0MB. A 'Compare' button is visible next to the 'v3' version in the sidebar.

# Demo in Streamlit

The inference can be viewed on providing text description of floor plan like below and get the annotated floor plan image as result

**Enter the description of the Floor plan you want**

For Example: One bedroom with two kitchen....

Enter your description, value=

House with one room and a living room and three kitchen with patio

Get Floor plan

Waiting for Model to Run...

**Enter the description of the Floor plan you want**

For Example: One bedroom with two kitchen....


Enter your description, value=

House with one room and a living room and three kitchen with patio

Get Floor plan

Done!

Dalle designed floorplan!!



# Technical Difficulty

- Difficulty parsing original noisy SVGs to generate sentences
- Difficulty coloring in vector graphics for custom colored image
- Experimented with [StackGAN](#), but ran into many problems with it
- Finding appropriate problem scope
  - *changed our dataset from original larger scope*
  - *restricted type of images to single story homes*
- RAM/memory usage
  - *paying for Colab Pro helped*

# Lesson Learnt

- Generative tasks require a lot of data and computing resources
- We needed to keep our scope small for what type of images to generate

# Teamwork

- Slack was used as the medium of communication across the team.
- We followed agile methodology for project management and its execution. We met as a team every wednesday at 9PM.
- The version control system used was git and the project was pushed to GitHub to be stored.
- Team communication played an important role in development of the project.

# Code and Version Control

Packages : dalle-pytorch, pytorch, matplotlib, wandb, PIL, streamlit  
Input : housing floor plan images, floorplan text description  
Output : generated floor plan image from given text description  
DL Model : Variational autoencoders (VAE), DALL-E

Tools Used : Google Colab Pro, Weights and Biases (WandB)  
Google Colab : [Floor Plan Generation Training](#)  
Version Ctrl : [GitHub](#)

MLOps: Weights and Biases (WandB)

Q & A



Thank You