How Can Deep Learning & Reinforcement Learning Automate Media Content Creation & Optimize Interactive System Behaviors?

Group 7: Khang Do, Lilly Parham & Gracie Rehberg

Georgia State University | J. Mack Robinson College of Business

Robinson

Group Members

Khang Do

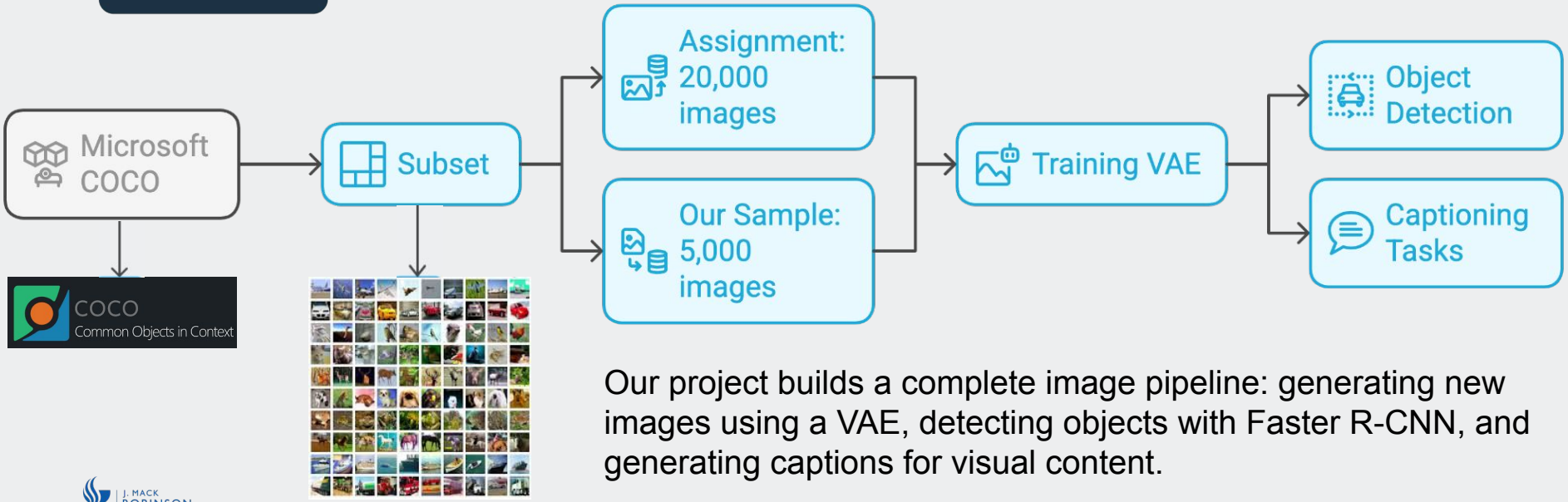
Lilly Parham

Gracie Rehberg

Part A: Image Generation, Object Detection & Image Captioning

Overview

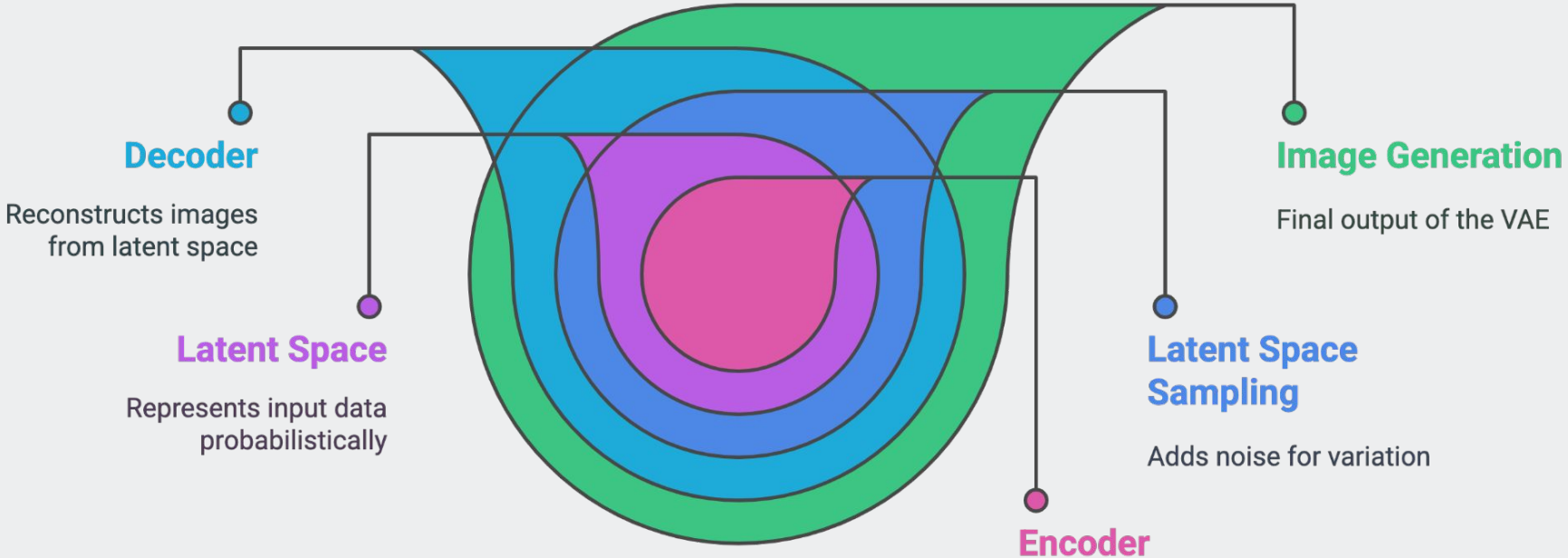
Purpose: To efficiently train a Variational Autoencoder (VAE) and support subsequent object detection and captioning models by selecting a diverse and manageable subset of Microsoft COCO images.



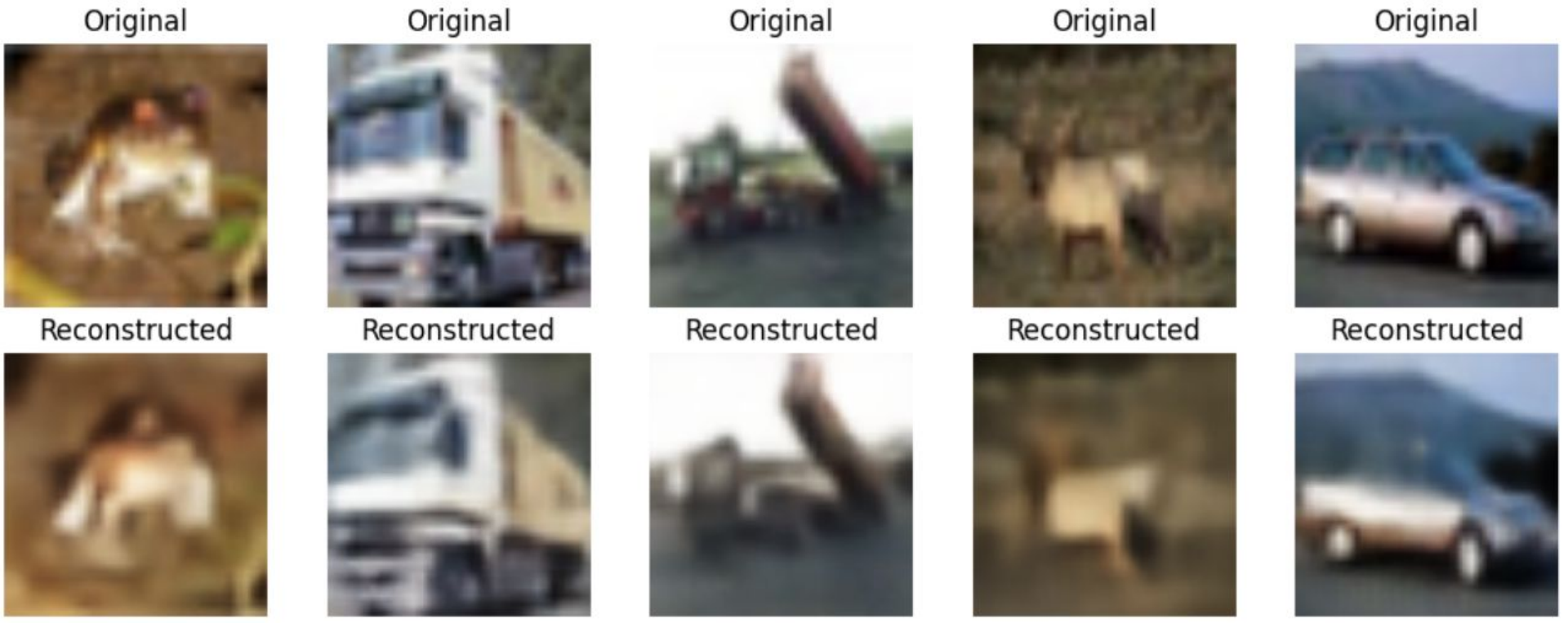
Our project builds a complete image pipeline: generating new images using a VAE, detecting objects with Faster R-CNN, and generating captions for visual content.

Image Generation
Variational Autoencoder (VAE)

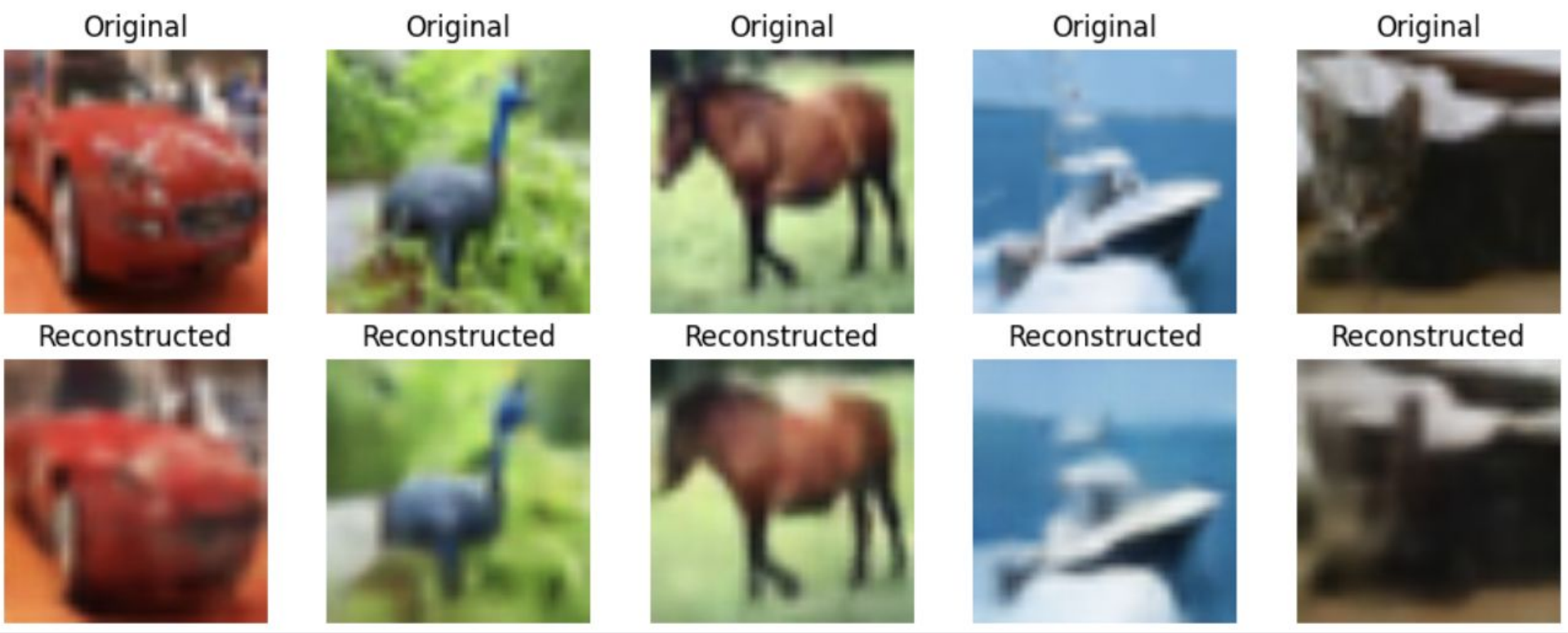
* Trained on resized CIFAR-10 images (128×128×3)
* Visual comparison shows the model successfully preserves key spatial and color features.



Reconstructed Image Results (1-5)



Reconstructed Image Results (6-10)



Object Detection Model

Model

Pre-trained Faster R-CNN with ResNet-50 FPN backbone

Framework

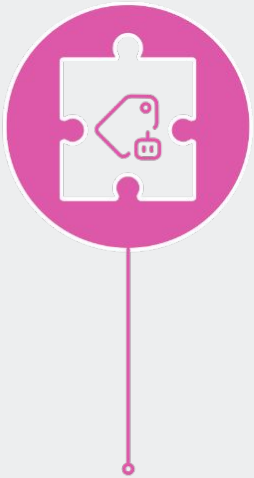
Utilizes PyTorch & TorchVision libraries

Input

VAE-Reconstructed Images (Converted from TensorFlow to PyTorch format)

Purpose

Detect & label objects in generated images



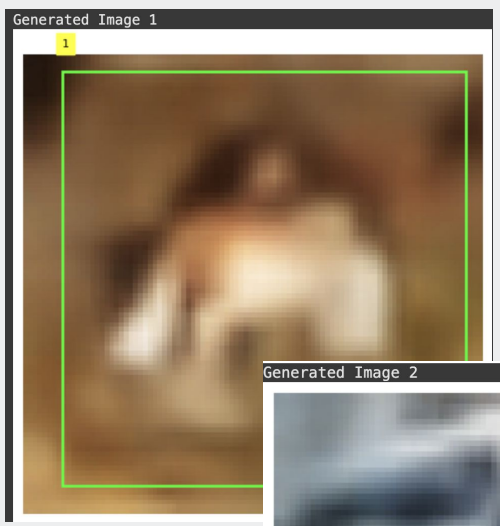
Detection Threshold

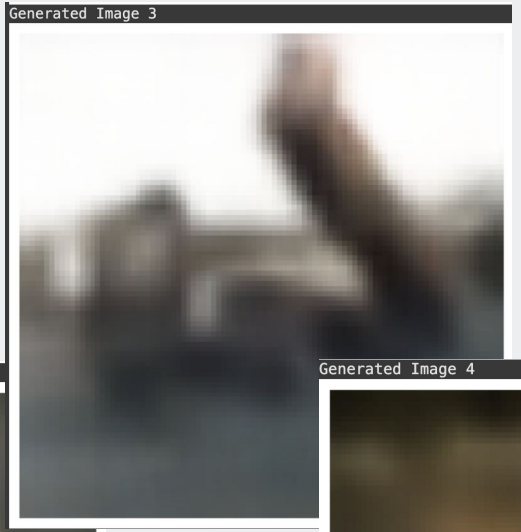
Confidence score > 0.4 to filter predictions

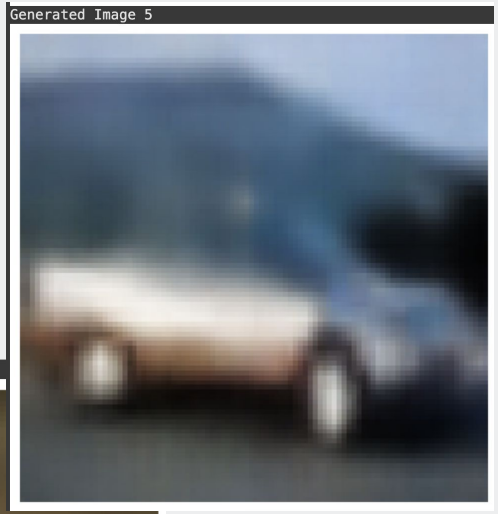
Output

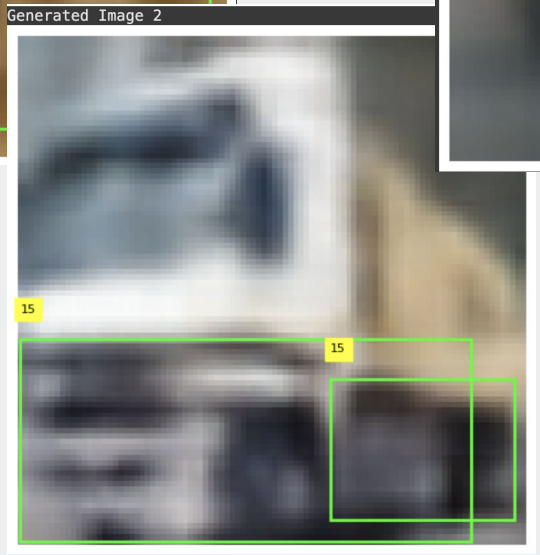
Bounding boxes with class labels (e.g., people, animals, etc.)

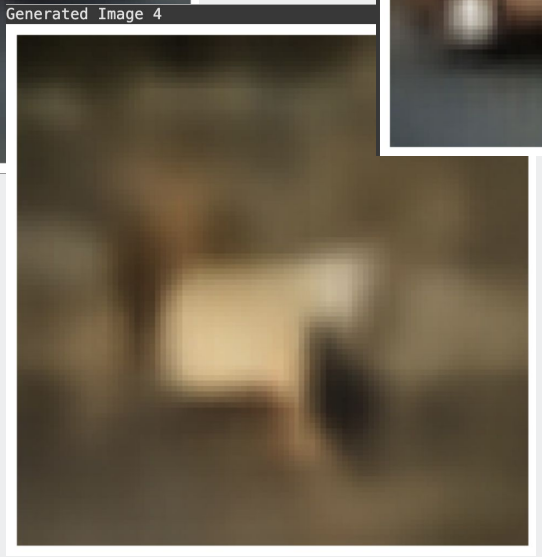
Object Detection Model Results











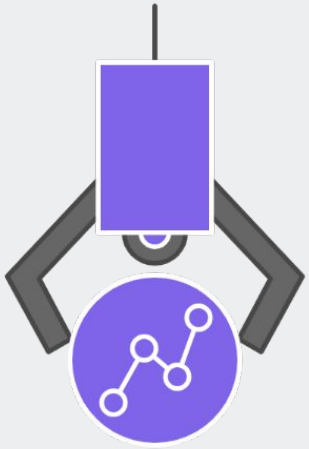
Georgia State University | J. Mack Robinson College of Business

Robinson

Image Captioning Model

Model

Pre-trained BLIP for image pretraining



Architecture

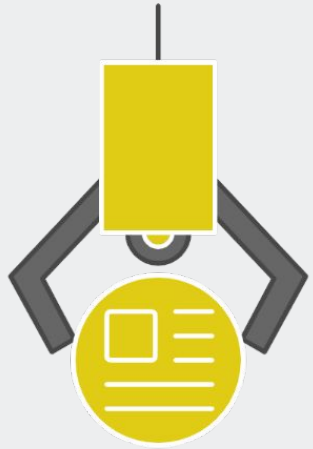
Vision-Language Transformer

Purpose

Translates visual features into language

Input

VAE-Reconstructed Images as input



Output

Generates coherent, context-aware captions

Image Captioning Results

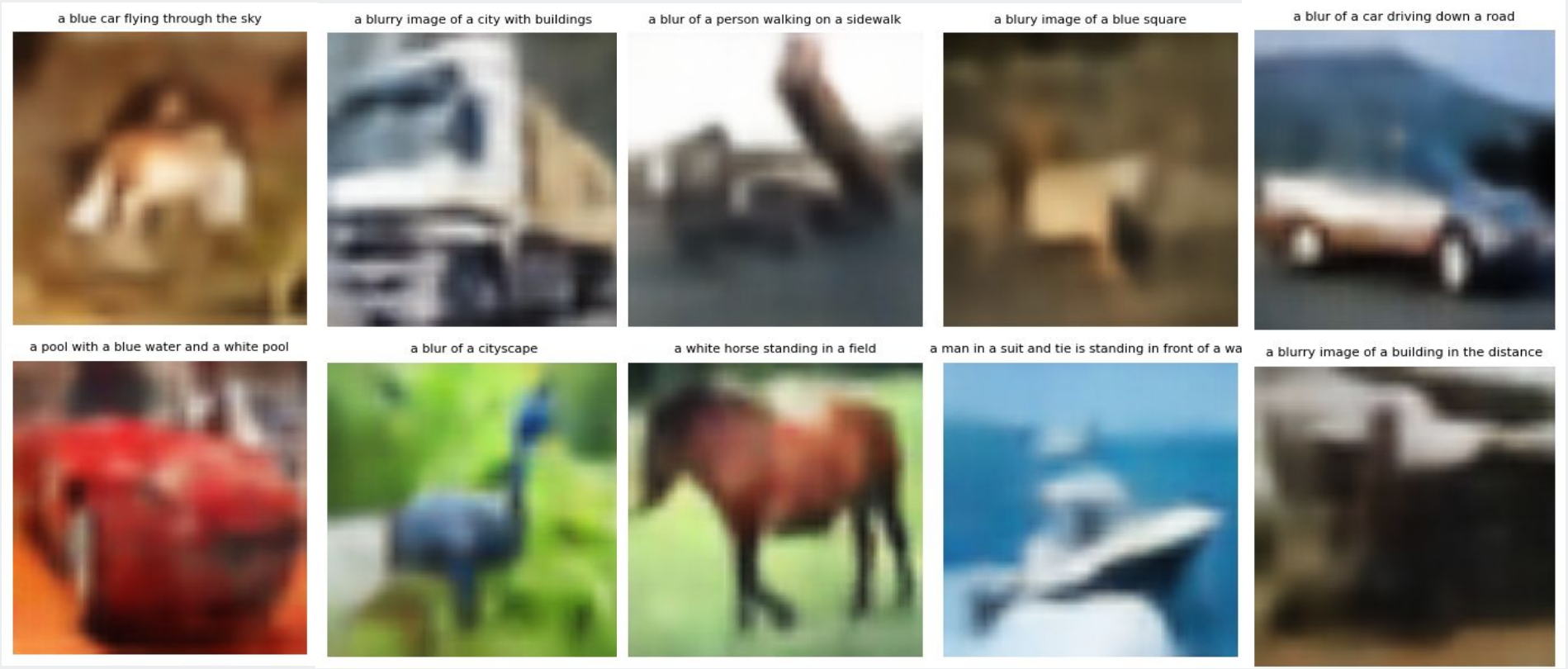
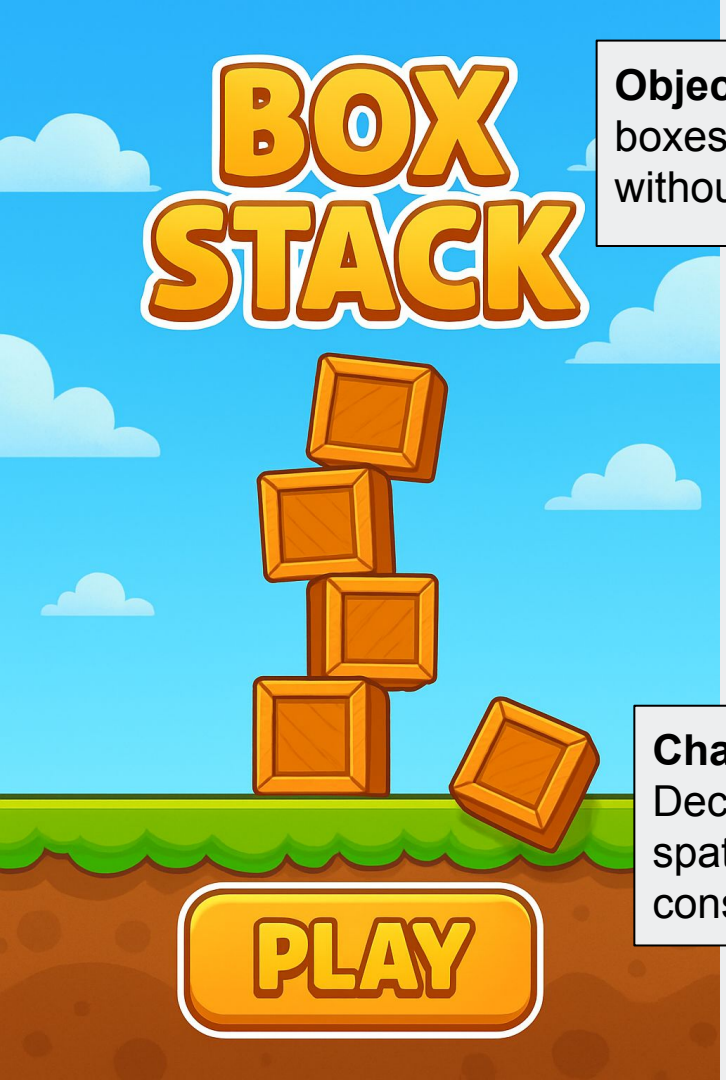


Image Captioning Text Results

*** Only 1/10 result accurate

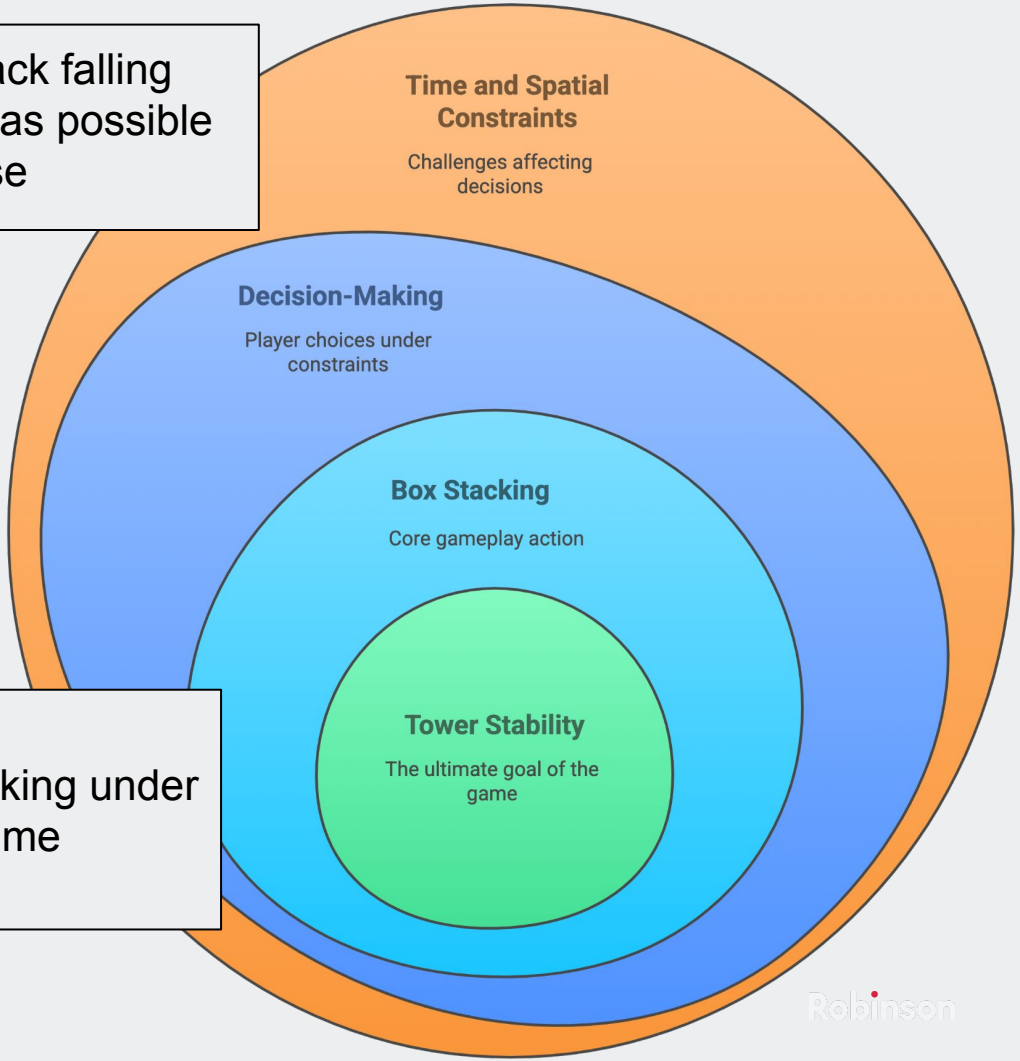
- Image 1: “a blue car flying through the sky”
- Image 2: “a blurry image of a city with buildings”
- Image 3: “a blur of a person walking on a sidewalk”
- Image 4: “a blurry image of a blue square”
- Image 5: “a blur of a car driving down a road”

Part B: Reinforcement Learning Strategy Design

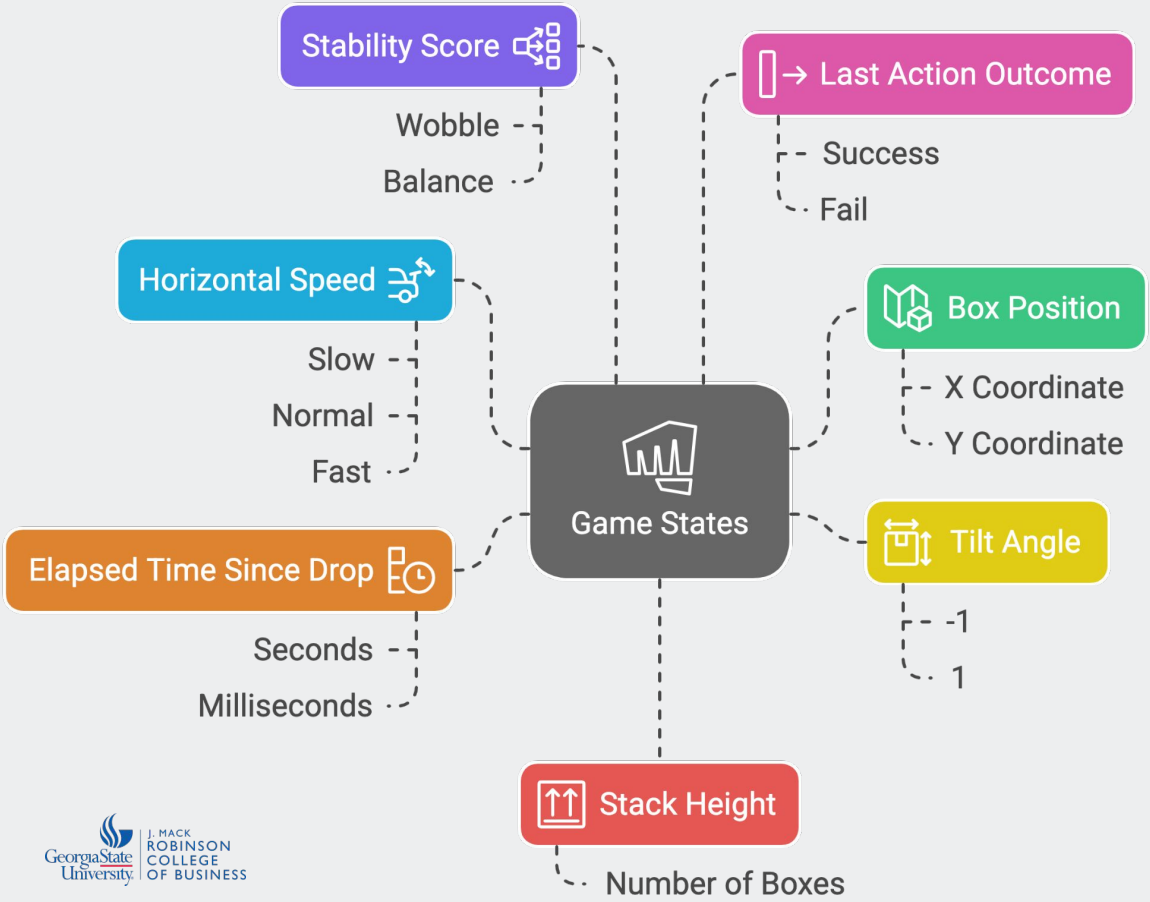


Objective: Stack falling boxes as high as possible without collapse

Challenge: Decision-making under spatial and time constraints



Game States & Actions



Move Left

Allows slight adjustment to the left for better positioning.

Move Right

Allows slight adjustment to the right for better positioning.

Drop Now

Immediate release for quick placement.

Tilt

Rotates the box for strategic placement.

Game Points

Perfect Placement +10

Awarded for ideal block positioning.

Near-Perfect Placement +5

Awarded for almost ideal block positioning.

Angled Placement +20

Awarded for successful angled block placement.

Fast Drop +3

Awarded for quick block placement.

Slow Drop -3

Penalty for slow block placement.

Box Misses -30

Penalty when a block misses the stack.

Tower Collapse -10

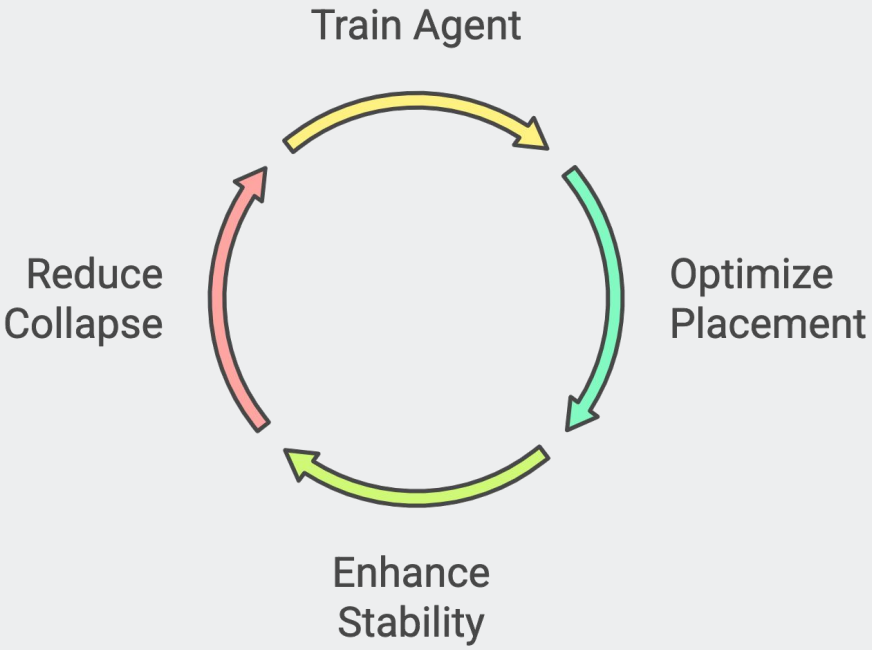
Penalty for causing the tower to fall.

Unnecessary Tilt -5

Penalty for using tilt when not needed.

Modeling Hypothesis

- **Agent can learn optimal box placement & tower stability**
- **Goal:** Increase precision, adapt tilt use & minimize collapse
- **Assumption:** Model is trained with clear feedback & cumulative rewards
- Note: Design is meant for external implementation (no coding here)



Modeling Hypothesis

Average Reward

Measures overall success in earning rewards per episode.

Max Stack Height

Indicates the highest level of stack achieved, reflecting skill.

Perfect Placements

Shows accuracy in placing blocks, crucial for high scores.

Collapse Rate

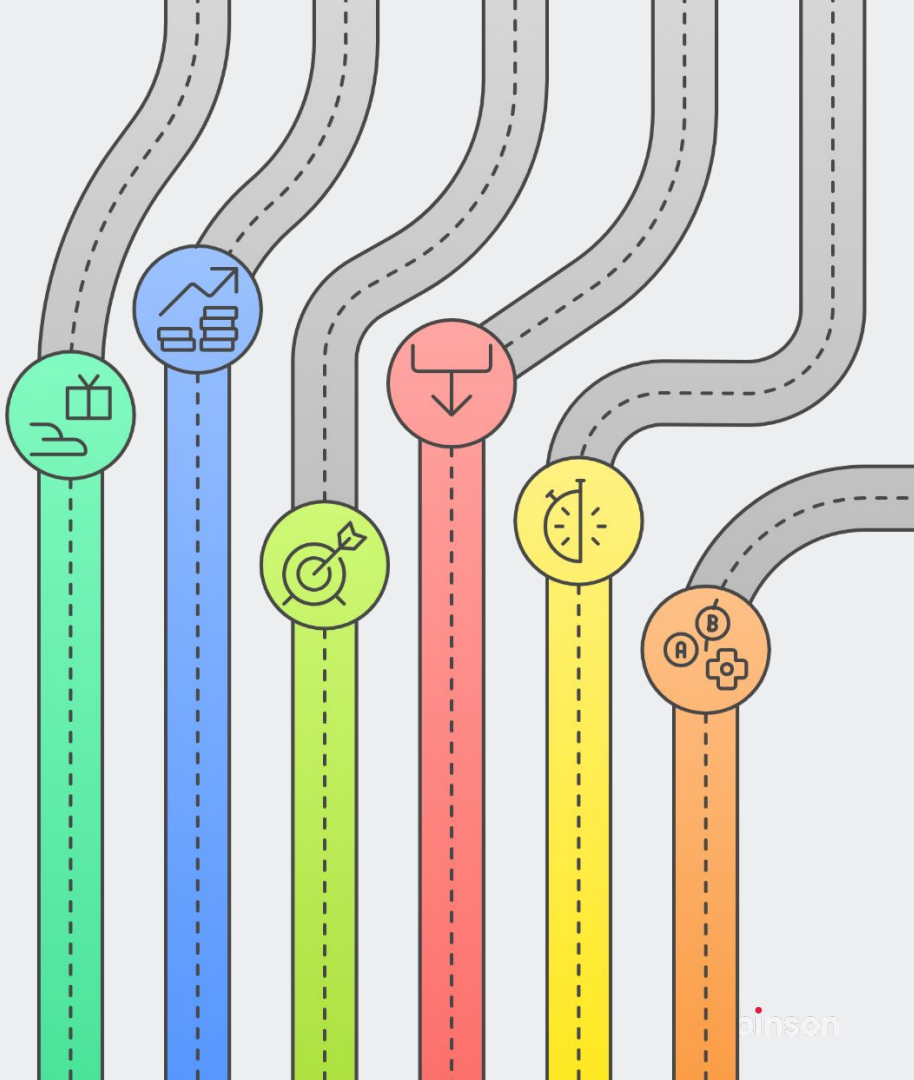
Highlights stability and risk management in gameplay.

Average Time

Reflects efficiency in making placements.

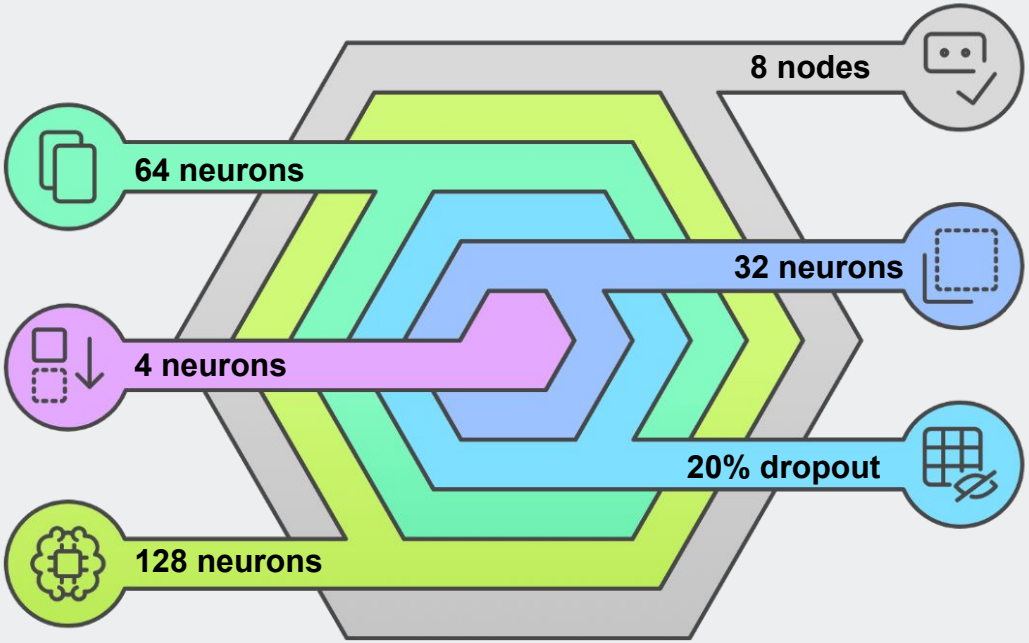
Action Usage

Reveals strategic trends in using actions like tilting.



Neural Network Structure

Deep Q-Network (DQN) Architecture



Input Layer

Receives state features as input

Hidden Layer 1

Initial feature processing with ReLU

Hidden Layer 2

Processes features with ReLU activation

Hidden Layer 3

Refines features for action selection

Output Layer

Generates Q-values for actions

Dropout Layer

Prevents overfitting through regularization

Let's Build…

Action-State-Reward Design

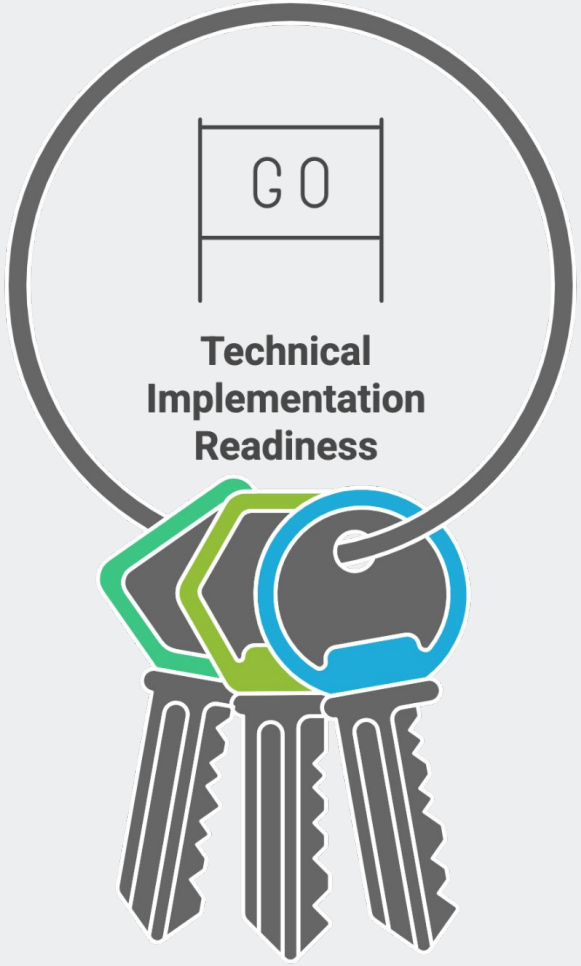
A clear design that guides actions to desired states and rewards.

Scalable Training Procedure

A training process that can be expanded to accommodate growth.

Robust Evaluation Framework

A strong framework for assessing the effectiveness of the implementation.




Thank
you!