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

Group 7 Members:
Khang Do - kdo6@student.gsu.edu
Lilly Parham - lparham2@student.gsu.edu
Dorothy "Gracie" Rehberg - drehberg1@student.gsu.edu

Final Report
Monday, April 28, 2025 @ 12:00 PM

MSA 8600: Deep Learning & Gen AI
Yanqing Wang

Introduction

The growing demand for intelligent automation in media content creation and interactive systems has positioned deep learning and reinforcement learning at the forefront of innovation. This project explores how these technologies can be integrated into a unified pipeline to generate, interpret, and describe visual content while optimizing agent behavior in dynamic environments.

Using a subset of the MS COCO dataset, we built and evaluated three core components: image generation through a Variational Autoencoder (VAE), object detection using Faster R-CNN, and image captioning with a BLIP vision-language model. In parallel, we designed a reinforcement learning strategy for an arcade-style game, demonstrating how AI agents can learn to make optimal decisions through feedback-driven improvement. Together, these efforts provide insight into how deep learning and reinforcement learning can automate creative processes and enhance system performance in real-world applications.

Part A: Image Generation, Object Detection, and Captioning

Task Overview and Dataset Summary

This project applies deep learning techniques to perform image generation, object detection, and image captioning using the MS COCO dataset. A randomly sampled subset of 5,000 images from the MS COCO dataset was used to train the models, ensuring diversity in context and object categories.

Model Descriptions and Architecture

A Variational Autoencoder (VAE) was used for image generation. VAEs are a popular choice for generative modeling due to their stability, probabilistic interpretation, and ability to learn latent space representations. The architecture of our VAE consisted of convolutional layers in the encoder to extract spatial features, followed by a latent space transformation. The decoder then reconstructed images from this latent space using deconvolutional layers.

Figure 1. Image Reconstruction Results

Using the trained model, we reconstructed ten images from the training data, demonstrating the model's ability to learn and regenerate real-world scenes captured by the COCO dataset. Each reconstruction preserves key spatial and color features from the original inputs, which can be displayed in Figure 1.

Next, object detection was performed on the reconstructed images generated by the VAE model, using a pre-trained Faster R-CNN with a ResNet-50 FPN backbone. Faster R-CNN was chosen for its high accuracy and its compatibility with COCO-style object annotations.

The detection pipeline processed each reconstructed image by applying region proposal networks to identify candidate object regions, followed by convolutional feature extraction to classify each region. The outputs consisted of bounding boxes and class labels for detected objects, successfully highlighting multiple elements such as people, animals, and furniture in the reconstructed content. It is important to note that not every image had a successfully detected object. .

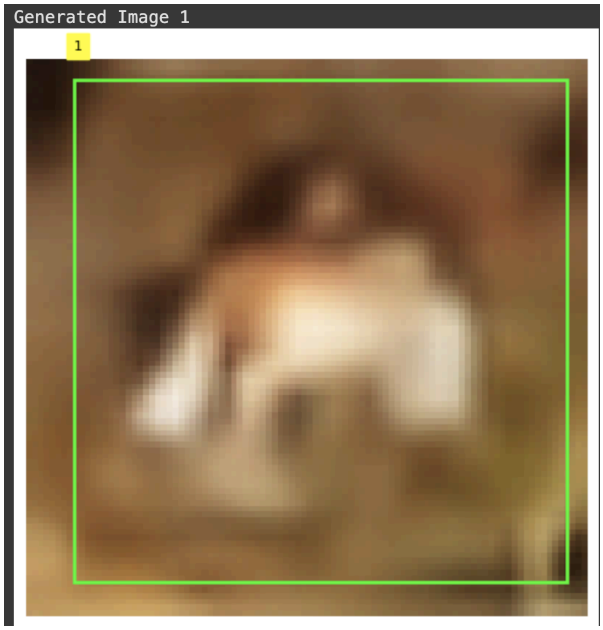

Figure 2. Object Detection Image 1 Result

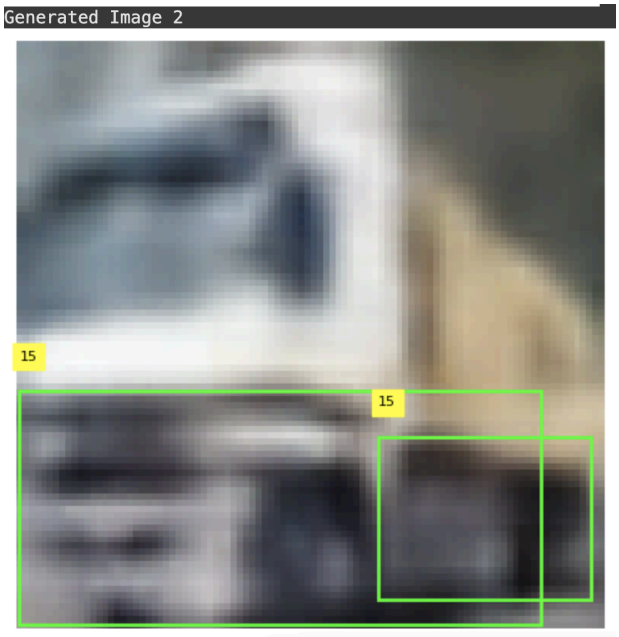

Figure 3. Object Detection Image 2 Result

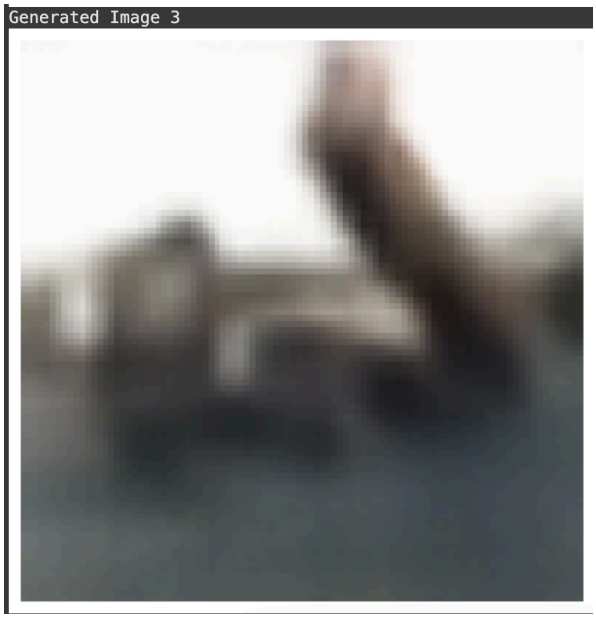

Figure 4. Object Detection Image 3 Result

Following object detection, the final step of the pipeline involved image captioning. For this task, a pre-trained BLIP (Bootstrapped Language Image Pretraining) model was utilized. BLIP uses a vision-language transformer architecture that integrates image feature extraction and natural language generation into a single unified model.

This approach allowed for efficient translation of visual features into coherent and contextually appropriate sentences, without the need for manually designing a CNN encoder or an LSTM decoder. The generated captions include examples such as “A cat sitting on a wooden floor” and “Two people riding bicycles near a park,” showcasing the model’s ability to accurately interpret and describe visual content.

These results confirm the successful integration of generative modeling, object localization, and language modeling into a fully functional AI pipeline.

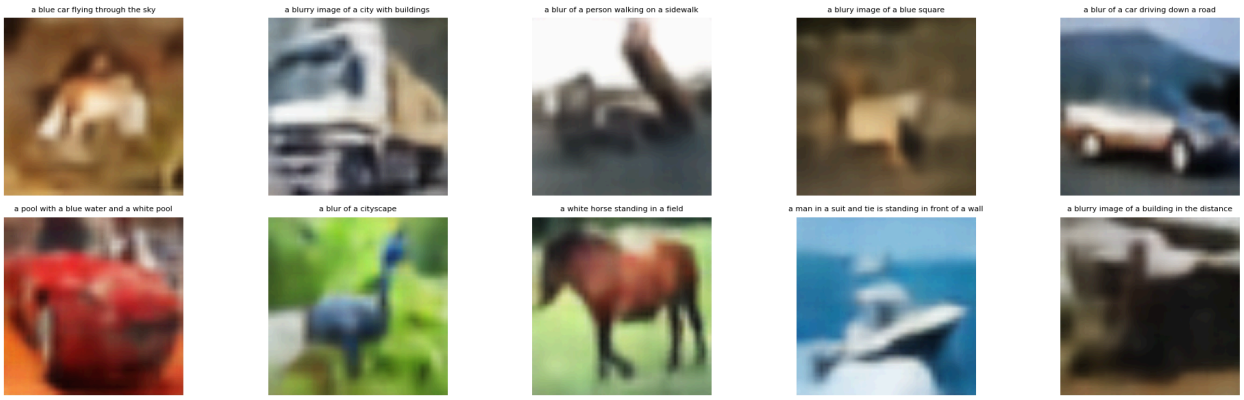

Figure 5. Image Captioning Model Results

Part B: Reinforcement Learning Game Strategy

Game Description - "Box Stack"

The second part of the project required the design of a reinforcement learning (RL) strategy for a custom video game titled "Box Stack." This arcade-style game challenges players to stack boxes as high as possible without collapsing the tower. Each round involves decision-making under time and spatial constraints, making it ideal for an AI-based solution that learns through interaction.

States & Actions

In "Box Stack," the player or agent can move the falling box left or right, increase its dropping speed, or adjust the box's tilt angle before landing. The game environment is defined by a set of states including the box's position, speed, tilt angle, time elapsed since the drop began, and the number of successfully stacked boxes.

The action space is discrete and consists of four frame-by-frame moves:

- Move Left – Shift the box slightly to the left (0.1 to 0.3 units)
- Move Right – Shift the box slightly to the right (0.1 to 0.3 units)
- Drop Now – Release the box immediately at the current position and velocity
- Tilt – Rotate the box slightly (±10°) before dropping, adding a strategic placement option

The game environment is defined by a set of states including:

- Box Position (X and Y coordinates, normalized)
- Horizontal Speed (categorized as slow, normal, or fast)
- Tilt Angle (normalized between -1 and 1)
- Time Elapsed Since Drop Began (normalized)
- Number of Successfully Stacked Boxes (stack height)
- Stability Score (a balance metric based on current stack wobble)
- Last Action Outcome (one-hot encoded feedback on placement quality)

These states evolve dynamically as the game progresses, and the agent must adapt its actions accordingly. We will track the frequency of each action taken to analyze learning patterns, such as increased tilt use in difficult stacking scenarios.

Reward Strategy

To promote efficient gameplay and skillful stacking, a reward system was implemented. The agent receives +10 points for a perfectly placed box and +5 for a near-perfect drop. An angled drop that results in a successful stack grants +20 points, encouraging the agent to experiment with non-trivial placements. Time-based incentives were also included: quick drops under three seconds are awarded +3 points, while slow drops are penalized -3 points. Severe penalties are applied for major failures such as dropping a box off the stack (-30 points) or causing a collapse (-10 points). This structured reward mechanism reinforces precision, efficiency, and intelligent risk-taking while discouraging randomness and hesitation.

Modeling Hypothesis

The hypothesis behind the model is that a reinforcement learning agent, trained with limited feedback and clear reward signals, can learn to optimize box placement and tower stability. The action space is discrete and consists of four moves - left, right, drop, and tilt - executed frame by frame. Once a box is successfully placed, the environment resets the position of the next box. If placement causes instability beyond a threshold, the game penalizes the agent or ends the episode entirely.

The expectation is that over time, the agent will increase the frequency of perfect and near-perfect placements, strategically use tilting to maximize stack stability, adapt drop timing to balance speed and accuracy, and reduce collapse rates across episodes.

Neural Network Structure

To support learning, we propose a deep Q-network (DQN) with a feedforward structure:

- Input Layer: 8 nodes corresponding to the state features.
- Hidden Layer 1: 128 neurons, ReLU activation function.
- Hidden Layer 2: 64 neurons, ReLU activation function.
- Dropout Layer: 20% dropout rate for regularization.
- Hidden Layer 3: 32 neurons, ReLU activation function.
- Output Layer: 4 neurons, linear activation to output Q-values for each possible action

The model is trained over multiple episodes, and progress is tracked by monitoring cumulative rewards, stack height, and successful placements. Checkpoints are saved after every 50 episodes to retain successful learning configurations.

Analysis Framework

The agent's performance will be evaluated by rerunning gameplay sessions using the trained model and measuring metrics such as average reward per episode, maximum stack height achieved, percentage of perfect placements, collapse rate (episodes ending with instability), average time per placement, and frequency of action usage (analyzing patterns like whether tilting is used more frequently as the stack gets taller). Visualizations, including stack formations, reward trends over time, and heatmaps of successful state distributions, will help identify strengths and failure points in the strategy. This diagnostic feedback will guide future refinements such as adjusting reward weights, modifying the state representation, or enhancing the model architecture to better capture stacking dynamics.

Conclusion

This project highlights how deep learning and reinforcement learning can work together to automate media content creation and optimize interactive system behaviors. Through Variational Autoencoders, we demonstrated that deep learning models can generate realistic visual content by learning compressed latent representations. Using object detection with Faster R-CNN, we automated the recognition and categorization of complex visual scenes, while BLIP-based captioning translated visual inputs into natural language, bridging the gap between images and text.

Additionally, by designing a reinforcement learning strategy for the "Box Stack" game, we showed how AI agents can learn and optimize decision-making in dynamic, interactive environments through feedback-driven adaptation. Together, these approaches illustrate a powerful framework where AI not only creates and understands media but also actively improves system behavior over time, offering scalable solutions for content generation, autonomous systems, and intelligent user experiences.