NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

Data Science and Information Technologies

MSc THESIS

Object-centric generative modeling: merging unsupervised segmentation learning with image synthesis

Konstantinos G. Konstantinidis

Supervisor: **Ioannis Emiris**, Professor, Department of Informatics and Telecommunications, National and Kapodistrian University of Athens
and General Director, Athena Research Center
Nikos Komodakis, Professor, Department of Computer Science, University of Crete and Senior Collaborating Researcher, Archimedes Research Unit Athena Research Center

ATHENS

OCTOBER 2024

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Επιστήμη Δεδομένων και Τεχνολογίες Πληροφορίας

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αντικειμενοστραφής παραγωγική μοντελοποίηση: Συγχώνευση μάθησης τμηματοποίησης χωρίς επίβλεψη με σύνθεση εικόνας

Κωνσταντίνος Γ. Κωνσταντινίδης

Επιβλέπων: **Ιωάννης Εμίρης**, Καθηγητής, Τμήμα Πληροφορικής και Τηλεπικοινωνιών, Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών και Γενικός Διευθυντής Ερευνητικού Κέντρου Αθηνά
Νίκος Κομοντάκης, Καθηγητής, Τμήμα Πληροφορικής, Πανεπιστήμιο Κρήτης και Ανώτερος Συνεργαζόμενος Ερευνητής, Ερευνητική Μονάδα Αρχιμήδης του Ερευνητικού Κέντρου Αθηνά

ΑΘΗΝΑ

ΟΚΤΩΒΡΙΟΣ 2024

MSc THESIS

Object-centric generative modeling: merging unsupervised segmentation learning with image synthesis

Konstantinos G. Konstantinidis
S.N.: 7115152200026

SUPERVISOR: **Ioannis Emiris**, Professor, Department of Informatics and Telecommunications, National and Kapodistrian University of Athens and General Director, Athena Research Center
Nikos Komodakis, Professor, Department of Computer Science, University of Crete and Senior Collaborating Researcher, Archimedes Research Unit Athena Research Center

EXAMINATION COMMITTEE: **Ioannis Emiris**, Professor, Department of Informatics and Telecommunications, National and Kapodistrian University of Athens and General Director, Athena Research Center
Vassilis Katsouros, Professor, Department of Informatics and Telecommunications, National and Kapodistrian University of Athens and Director of ILSP Athena Research Center
Nikos Komodakis, Professor, Department of Computer Science, University of Crete and Senior Collaborating Researcher, Archimedes Research Unit Athena Research Center

Examination Date: 22 October, 2024

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αντικειμενοστραφής παραγωγική μοντελοποίηση: Συγχώνευση μάθησης τμηματοποίησης χωρίς επίβλεψη με σύνθεση εικόνας

Κωνσταντίνος Γ. Κωνσταντινίδης
Α.Μ.: 7115152200026

ΕΠΙΒΛΕΠΩΝ: **Ιωάννης Εμίρης**, Καθηγητής, Τμήμα Πληροφορικής και Τηλεπικοινωνιών, Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών και Γενικός Διευθυντής Ερευνητικού Κέντρου Αθηνά
Νίκος Κομοντάκης, Καθηγητής, Τμήμα Πληροφορικής, Πανεπιστήμιο Κρήτης και Ανώτερος Συνεργαζόμενος Ερευνητής, Ερευνητική Μονάδα Αρχιμήδης του Ερευνητικού Κέντρου Αθηνά

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ: **Ιωάννης Εμίρης**, Καθηγητής, Τμήμα Πληροφορικής και Τηλεπικοινωνιών, Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών και Γενικός Διευθυντής Ερευνητικού Κέντρου Αθηνά
Βασίλης Κατσούρος, Καθηγητής, Τμήμα Πληροφορικής και Τηλεπικοινωνιών, Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών και Διευθυντής Ερευνητικού Κέντρου ΙΕΛ Αθηνά
Νίκος Κομοντάκης, Καθηγητής, Τμήμα Πληροφορικής, Πανεπιστήμιο Κρήτης και Ανώτερος Συνεργαζόμενος Ερευνητής, Ερευνητική Μονάδα Αρχιμήδης του Ερευνητικού Κέντρου Αθηνά

Ημερομηνία Εξέτασης: 22 Οκτωβρίου 2024

ABSTRACT

In this project, we approach the problem of image synthesis and how this can be improved by using object-centric segmentation. Slot-based auto-encoders stand out in the field of object-centric segmentation and the masked auto-encoders have shown impressive results in image generation. In our approach we use the following techniques. A slot based auto-encoder, which uses attention based training to distill attention maps from the decoder. A Masked Generative encoder, in order to achieve image synthesis, with the aid of the attention maps from the first step. We also perform clustering on the distilled attention maps and perform image generation. We focus on improving real-world image reconstruction, our approach demonstrates the potential of slot-based auto-encoders to enhance practical applications in object-centric representation learning, addressing the specific challenges posed by real-world imaging scenarios.

SUBJECT AREA: Computer Vision

KEYWORDS: Computer Vision, Image Generation, Segmentation

ΠΕΡΙΛΗΨΗ

Στην παρούσα εργασία, προσεγγίζουμε το πρόβλημα της σύνθεσης εικόνων και πώς αυτό μπορεί να βελτιωθεί με τη χρήση αντικειμενοκεντρικής τμηματοποίησης. Οι αυτοκωδικοποιητές με βάση τις υποδοχές(slots) ξεχωρίζουν στον τομέα της αντικειμενοκεντρικής τμηματοποίησης και οι αυτοκωδικοποιητές με τη χρήση μασκών έχουν δείξει εντυπωσιακά αποτελέσματα στη δημιουργία εικόνων. Στην προσέγγισή μας χρησιμοποιούμε τις ακόλουθες τεχνικές. Έναν αυτόματο κωδικοποιητή με βάση τις υποδοχές, ο οποίος χρησιμοποιεί εκπαίδευση με βάση την την απόσπαση των αντιστοιχίσεων προσοχής από τον αποκωδικοποιητή για κάθε εικόνα. Έναν δημιουργικός αυτόματος κωδικοποιητής με τη χρήση μασκών, προκειμένου να επιτευχθεί η σύνθεση εικόνας, με τη βοήθεια των αντιστοιχήσεων από το πρώτο βήμα. Πραγματοποιούμε επίσης ομαδοποίηση στις εξαγόμενες αντιστοιχίσεις προσοχής για τη δημιουργία εικόνων. Επικεντρωνόμαστε στη βελτίωση της ανακατασκευής εικόνων πραγματικού κόσμου, η προσέγγισή μας καταδεικνύει τις δυνατότητες των αυτόματων κωδικοποιητών που βασίζονται σε υποδοχές για την ενίσχυση των πρακτικών εφαρμογών στη μάθηση αντικειμενοκεντρικής αναπαράστασης, αντιμετωπίζοντας τις ειδικές προκλήσεις που θέτουν τα σενάρια απεικόνισης του πραγματικού κόσμου.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Όραση Υπολογιστών

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Όραση Υπολογιστών, Δημιουργεία εικόνων,Τμηματοποίηση

ACKNOWLEDGMENTS

First and foremost, I would like to express my gratitude to my parents for their support throughout this journey. I am grateful to my professor, Nikos Komodakis, for his guidance, and expertise, which have been fundamental to the completion of this work. I would also like to extend my heartfelt thanks to Spyros Gidaris and Ioannis Kakogeorgiou for their substantial contributions and support during the project. I would also like to thank Mr. Ioannis Emiris and Vassilis Katsouros for their collaborative efforts, with this research. Finally, I am thankful to the Athena Research Center for providing access to their computational resources, which were crucial for the successful implementation of this work.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

PREFACE

This thesis represents the culmination of my master's degree program in Data Science and Information Technology at the National and Kapodistrian University of Athens, documenting my research conducted from January to September 2024.

To complete this project, I integrated existing studies on object-based image segmentation and image generation, immersing myself in various articles and research papers. Conducted in Athens, this research journey was a profound learning experience.

Completing this project has provided with knowledge both on the artificial intelligence field and how to approach a big research project. This accomplishment marks the conclusion of my master's degree and the beginning of a new and exciting chapter in my personal and professional life. I have acquired significant expertise and am confident in my abilities moving forward.

1. INTRODUCTION

Object-centric learning is a promising approach in machine learning that enhances the understanding and representation of objects within visual data. This study explores the application of object-centric learning to improve image reconstruction and generation, utilizing techniques from the SPOT [1] and MAGE [2] frameworks, as well as generative methodologies using clustering techniques and slot generation.

Slot-based autoencoders represent another significant advancement in object-centric learning, particularly for image generation. These models segment an image into distinct ”slots,” each representing different objects or parts of the scene [3]. By learning to encode and decode these slots separately, the model can generate images with coherent and well-defined objects, improving the quality and realism of the generated images. This approach not only aids in image synthesis but also enhances the interpretability of the learned representations.

We employ several advanced methods for object-centric learning. One key technique is the use of autoregressive (AR) decoders, which surpass traditional MLP-based decoders by leveraging a more expressive capacity to process learned representations [4]. This enables the AR decoders to better handle complex spatial relationships and object structures within images, crucial for generating coherent, object-centric outputs. However, AR decoders can sometimes overfit, particularly when relying heavily on past ground-truth tokens, which weakens the learning of object-specific features [5]. To counter this, we adopt the patch-order permutation strategy from the SPOT framework [1], where image patches are rearranged during self-training. This approach forces the model to rely more on object-centric slot features, improving the robustness of object-wise predictions and enhancing the model’s overall spatial understanding.

Masked autoencoders play a crucial role in learning robust representations for image reconstruction and synthesis. In the MAGE framework, images are partially masked, prompting the model to reconstruct the missing parts [6]. This technique compels the model to understand the underlying structure and semantics of the image, leading to high-quality reconstructions. The generative encoder in MAGE unifies representation learning and image synthesis within a single model, improving efficiency and effectiveness in both tasks [7].

A crucial addition to the slot-based autoencoder approach is the use of clustering used for slot and image generation. We apply K-means clustering to group slots into semantically meaningful concepts. This allows the model to identify and group objects or components that share similar attributes. When generating new images, we can select slots from these concept libraries, which aims that the objects in the generated images have coherent visual properties. Moreover, this clustering mechanism avoids severe occlusions by rejecting slots with overlapping segmentation masks.

This project aims to demonstrate how combining these object-centric learning techniques can affect image reconstruction and generation. By leveraging the masked autoencoders

from MAGE, and the autoregressive decoders and patch-order permutations from SPOT, as well as slot-based autoencoders and clustering, the study seeks to show how these methods can collectively improve the performance and accuracy of image-based tasks.

The primary objectives of this study are to:

1. Explore object-centric learning in slot-based autoencoders by introducing a two-stage training process. In the first stage, we distill attention maps during image segmentation, capturing important structural details for better object representation.
2. Reconstruct images by leveraging the attention maps from the first stage and utilizing masked autoencoders to generate high-quality images, even in challenging real-world scenarios.
3. Explore the benefits of slot-based autoencoders and clustering of slots in generating high-quality images.

This research is significant because it addresses key challenges in computer vision, such as reconstructing images and generating synthetic images. By advancing our understanding of object-centric learning and its practical applications, this study aims to contribute to the field of computer vision.

2. RELATED WORK

Unsupervised Object-centric learning aims to break down scenes containing multiple objects into distinct and meaningful components, referred to as slots, with slot-attention bottlenecks[3]. This research area has advanced significantly with various approaches improving the accuracy and stability of object segmentation and representation. Slot Attention stands out as a key method, employing an iterative attention mechanism to partition images into individual slots.Although previous models struggled with, real-world scenes [8, 9]. To overcome these issues, recent methods have enhanced the encoder's slot-attention module through techniques like bi-level optimization [10] and structural changes. Moreover, the development of decoders that facilitate effective decomposition has led to significant improvements, with autoregressive transformer decoders used in SLATE [11] and diffusion-based methods [12, 13] being particularly noteworthy. Utilizing self-supervised pre-trained features, as seen in DINOSAUR [14], has also been beneficial for tackling complex scenes. The SPOT framework [1] has introduced novel strategies, such as attention-based self-training and patch-order permutation, further enhancing object segmentation and reconstruction. Our research builds upon these developments by combining static image-based object-centric learning with improved image synthesis techniques.

Self-Supervised Learning for Vision has become a cornerstone in computer vision, enabling models to learn from unlabeled data by solving pretext tasks. Techniques such as masked image modeling (MIM)[15] have been particularly effective, drawing inspiration from natural language processing. For example, BEiT [16] uses a BERT-style approach to recover discrete visual tokens from masked inputs, while SimMIM [17] employs a simpler framework that predicts raw pixel values of masked image patches, demonstrating strong representation learning capabilities. Masked Autoencoders (MAE) [6] treat MIM as a denoising task, reconstructing pixel-level details from masked images, and CMAE[18] combines this approach with a contrastive loss to enhance feature separability. Despite their success, these methods often prioritize the quality of learned representations for downstream tasks over the fidelity of reconstructed images.

Generative Models for Image Synthesis have seen substantial advancements, with deep learning techniques such as Generative Adversarial Networks (GANs) [19] leading the way in creating realistic images. GAN-based models have been used across various domains but face challenges like training instability and mode collapse. To address these issues, two-stage approaches like VQVAE-2 [20] tokenize images into latent spaces before applying maximum likelihood estimation. This method, along with advancements like ViT-VQGAN [21], which leverages Vision Transformers for tokenization and autoregressive generation, has improved the diversity and quality of synthesized images. Diffusion models [22] have also emerged as powerful alternatives, achieving impressive results in image synthesis by iteratively refining noisy images into high-fidelity outputs.

However, many generative models struggle to simultaneously excel in image synthesis and representation learning. Our model utilizes the object-centric learning in order to

excel in image synthesis. By integrating techniques from object-centric learning, such as autoregressive decoders and slot-based autoencoders, our approach not only enhances the understanding and segmentation of objects within images but also improves the quality and coherence of generated images. This unified framework leverages the strengths of both methodologies, demonstrating that it is possible to achieve high performance in both image generation and representation learning.

3. BACKGROUND WORK

3.1 General Information

3.2 Computer Vision

Computer vision is a field of artificial intelligence that focuses on enabling machines to interpret and understand visual information from the world, much like humans do. It involves developing algorithms and models that can process, analyze, and make sense of images and videos. The goal of computer vision is to replicate the human visual system's ability to perceive and comprehend the environment. This technology primarily relies on pattern recognition, where computers are trained on vast amounts of visual data to learn the features and patterns that define different objects. A computer vision system learns to identify the common features of images and builds a model to recognize them in new images. These models can then be used to detect and classify objects in various applications, such as facial recognition, autonomous driving, and medical image analysis. Through this process, computer vision systems can interpret visual data, enabling machines to interact with and understand the visual world around them.

3.3 Image Generation

Image generation refers to the process of creating new images from scratch or from abstract representations using machine learning models. This area of research involves leveraging algorithms like Generative Adversarial Networks (GANs), diffusion models, and other generative frameworks to produce realistic and high-quality images. These models learn from a dataset of real images to understand the underlying patterns and features, enabling them to generate new images that are visually similar to the training data. Applications of image generation include creating art, enhancing video game graphics, and producing synthetic data for training other AI models.

3.4 Object-centric image segmentation

Object-centric image segmentation is a crucial task in computer vision that involves partitioning an image into distinct regions corresponding to different objects. This technique aims to accurately identify and delineate each object within an image, providing detailed and precise boundaries. Advanced segmentation models, such as those based on convolutional neural networks (CNNs) and transformers, are trained on annotated datasets where each pixel is labeled according to the object it belongs to. Object-centric segmentation is fundamental in various applications, including autonomous driving, where it is essential for detecting pedestrians and other vehicles, medical imaging for identifying

anatomical structures, and augmented reality for integrating virtual objects seamlessly into real-world scenes.

3.5 K-means Clustering

K-means is a widely-used clustering algorithm that partitions a dataset into K distinct, non-overlapping clusters. Given a dataset $X = \{x_1, x_2, \ldots, x_n\}$, the goal is to assign each data point x_i to one of K clusters, minimizing the intra-cluster variance. The algorithm proceeds by initializing K centroids $\mu_1, \mu_2, \ldots, \mu_K$ and iteratively updating them. At each step, the data points are assigned to the nearest centroid based on the Euclidean distance, as defined by:

$$d(x_i, \mu_k) = \|x_i - \mu_k\|_2$$

where $\|\cdot\|_2$ represents the Euclidean norm. After assigning each data point to the nearest cluster, the centroids are updated by computing the mean of the assigned points.

The optimization objective of K-means clustering is to minimize the sum of squared distances between the data points and their corresponding centroids. This can be expressed as the following objective function:

$$J = \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}(z_i = k) \|x_i - \mu_k\|_2^2$$

where z_i represents the cluster assignment for data point x_i, and $\mathbb{1}(\cdot)$ is the indicator function that is 1 if x_i belongs to cluster k, and 0 otherwise. The algorithm alternates between assignment and update steps until convergence.

3.5.1 Mini-batch K-means

Mini-batch K-means is a variant of the standard K-means algorithm that improves computational efficiency, particularly when dealing with large datasets. Instead of using the entire dataset for each iteration, Mini-batch K-means processes small, randomly selected subsets (mini-batches) of the data at each step. This approach significantly reduces the computational complexity while still maintaining good clustering performance. Let B represent a mini-batch of size m sampled from the dataset X.

At each iteration, the mini-batch $B = \{x_{i_1}, x_{i_2}, \ldots, x_{i_m}\}$ is used to update the centroids. The assignment and update steps are performed similarly to K-means, but the updates are computed using only the mini-batch, which makes the algorithm much faster:

$$\mu_k^{t+1} = \mu_k^t + \eta \left(\frac{1}{m} \sum_{x_i \in C_k \cap B} (x_i - \mu_k^t) \right)$$

where η is the learning rate, and $C_k \cap B$ represents the subset of the mini-batch assigned to cluster k.

Mini-batch K-means trades off a small loss in clustering accuracy for a significant gain in speed, making it particularly well-suited for large-scale clustering problems where processing the entire dataset in each iteration would be computationally expensive.

3.5.2 K-means Algorithm Pseudocode

The pseudocode for the K-means algorithm can be written as follows:

Algorithm 1 K-means Clustering

Input: Data points $X = \{x_1, x_2, \ldots, x_n\}$, number of clusters k
Output: Cluster centroids $\mu_1, \mu_2, \ldots, \mu_k$, and assignments of data points to clusters

Initialize: Randomly choose k points from X as the initial centroids $\mu_1, \mu_2, \ldots, \mu_k$
repeat
    for each data point $x_i \in X$ **do**
        Assign x_i to the closest centroid μ_j using a distance metric (e.g., Euclidean distance)
    end for
    for each centroid μ_j **do**
        Update the centroid μ_j by computing the mean of all points assigned to cluster j
    end for
until the centroids $\mu_1, \mu_2, \ldots, \mu_k$ do not change (or change is smaller than a threshold)

Return: Final cluster centroids $\mu_1, \mu_2, \ldots, \mu_k$ and assignments of data points to clusters

3.6 Computer Vision Models

3.6.1 Encoder and Decoder Structures

Encoder: An encoder consists of multiple identical layers, typically structured in a stack. Each layer generally comprises two main sub-components: a multi-head self-attention mechanism[4] and a position-wise feed-forward network. The role of the encoder is to convert an input sequence into a continuous representation, effectively capturing the essential information of the input. Residual connections [23], combined with layer normalization [24], help stabilize and optimize the training process, ensuring that the layers' outputs are added back to the input of that layer. This integration maintains the output dimensions consistent throughout the layers, typically specified by a predefined model dimension.

Decoder: Similar to the encoder, the decoder is built from multiple identical layers. Each decoder layer includes the same two sub-components as the encoder: a multi-head self-attention mechanism[4] and a position-wise feed-forward network, along with an additional sub-layer that performs multi-head attention over the encoder's output. The decoder's primary function is to generate an output sequence from the continuous representation produced by the encoder. There are various types of decoders.

3.6.2 Attention Mechanism

An attention mechanism is a method that enhances the capability of models to focus on specific parts of the input sequence when producing each part of the output sequence. It functions by mapping a query and a set of key-value pairs to an output. Both queries and keys, as well as values and outputs, are vectors. The output is a weighted sum of the values, where each weight reflects the relevance of the corresponding key to the query.

Scaled Dot-Product Attention A common type of attention mechanism, scaled dot-product attention, computes the dot products of a query with all keys, scales the result by the square root of the key dimension, and applies a softmax function to obtain the weights on the values. This method allows the model to dynamically focus on different parts of the input sequence for each output element [4]. The attention mechanism can be computed efficiently using matrix multiplication, which is highly optimized in many deep learning frameworks.

Practically, we compute the attention function for a set of queries simultaneously, packed into a matrix Q. The keys and values are similarly packed into matrices K and V. The matrix of outputs is computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.1}$$

The most commonly used attention functions are additive attention [25] and dot-product (multiplicative) attention. Dot-product attention aligns with our method, except for the scaling factor $\frac{1}{\sqrt{d_k}}$. Additive attention uses a feed-forward network with a single hidden layer for the compatibility function. While similar in theoretical complexity, dot-product attention is faster and more space-efficient in practice, benefiting from highly optimized matrix multiplication code.

3.6.3 Vision Transformer (ViT) Models

Vision Transformer (ViT) is a deep learning model that applies transformer architecture, initially designed for natural language processing (NLP), to computer vision tasks. In contrast

to convolutional neural networks (CNNs), which rely on convolutional layers to capture local features in images, ViT operates by dividing an image into smaller patches and treating these patches as a sequence of tokens, akin to words in NLP. Each patch is flattened into a vector and linearly embedded into a fixed dimension, forming the input sequence for the transformer. These patch embeddings are then processed through standard transformer layers to model long-range dependencies between patches. Formally, for an image I, divided into N patches, the input sequence is:

$$z_0 = [x_1 E; x_2 E; \ldots; x_N E] + E_{\text{pos}}$$

where E is the patch embedding matrix, and E_{pos} are the positional encodings that introduce spatial information.

A key advantage of ViT is its ability to model global relationships across the entire image early in the network. Unlike CNNs, which progressively expand the receptive field through successive layers, ViT can attend to any part of the image from the very first layer. This allows the model to capture both local and global dependencies more efficiently. The self-attention mechanism in ViT computes relationships between patches by calculating attention scores based on the pairwise similarity of patch embeddings. Given query Q, key K, and value V matrices for the patches, the attention scores are computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

where d_k is the dimensionality of the key vectors. This attention mechanism enables ViT to capture complex spatial dependencies that may be challenging for CNNs.

ViT has demonstrated competitive performance on several benchmark datasets, such as ImageNet, with fewer inductive biases than CNNs. However, one of the challenges of training ViT models is the need for large amounts of data due to the lack of locality priors that CNNs naturally exploit. To address this, ViT models are often pre-trained on large datasets (such as JFT-300M) and then fine-tuned on smaller, task-specific datasets. ViT has opened new avenues for applying transformer-based architectures to vision tasks, showing that with sufficient data and computational resources, transformer models can achieve or even surpass the performance of traditional convolutional models.

3.6.4 Masked language modeling

Masked language modeling and its autoregressive variants, such as BERT [26] and GPT [27, 28, 29], have proven to be highly effective pre-training methods in NLP. These techniques involve masking parts of the input sequence and training models to predict the missing content, demonstrating excellent scalability [29]. Substantial evidence indicates that these pre-trained representations generalize well to a wide range of downstream tasks.

3.6.5 Autoencoding

Autoencoding is a traditional technique for learning representations, employing an encoder to map inputs to latent representations and a decoder to reconstruct the inputs. PCA and k-means serve as examples of autoencoders [30]. Denoising autoencoders (DAEs) [7] specifically corrupt the input signal and train to reconstruct the original, uncorrupted version. Many methods can be viewed as generalized DAEs under various corruptions, such as masking pixels [7, 31, 32] or removing color channels [33]. Our approach, MAE, aligns with denoising autoencoding but diverges in several significant ways.

3.6.6 Self-supervised learning

Self-supervised learning has gained significant interest in computer vision, often focusing on different pretext tasks for pre-training [34, 35, 36, 33, 37]. Recently, contrastive learning [35, 38] has become popular, with methods like [35, 36, 39, 40] modeling image similarity and dissimilarity (or only similarity [39, 40]) between two or more views. These methods heavily depend on data augmentation [35, 39, 40]. Autoencoding follows a different conceptual path and exhibits distinct behaviors, as we will present.

3.6.7 Unsupervised learning

Unsupervised learning is a type of machine learning where the model is trained on data without explicit labels, allowing it to discover hidden patterns and structures within the dataset. This approach contrasts with supervised learning, where models are trained using labeled data. Unsupervised learning techniques, such as clustering, dimensionality reduction, and generative models, are particularly valuable in scenarios where labeled data is scarce or expensive to obtain. Clustering algorithms like k-means [41] and hierarchical clustering [42] group data points based on similarity, facilitating the discovery of inherent groupings in the data. Dimensionality reduction techniques such as Principal Component Analysis (PCA) [43] and t-Distributed Stochastic Neighbor Embedding (t-SNE) [44] reduce the number of variables under consideration, enabling more efficient data visualization and analysis. Generative models, including autoencoders and Generative Adversarial Networks (GANs) [19], learn to generate new data samples that resemble the training data, offering powerful tools for data augmentation and synthesis.

3.6.8 Methodologies for Selecting Patch Tokens to Mask

In transformer-based architectures, particularly Vision Transformers (ViT), images are divided into patch tokens before being processed by the model. In masked image modeling tasks, a subset of these patch tokens is masked, meaning that their information is removed or obscured, and the model is trained to predict the missing patches. Selecting which patches to mask is crucial, as it can influence the model's learning and ability to

generalize. A common strategy is to randomly select patches to mask. This approach provides a uniform coverage across the image, ensuring that the model does not overfit to specific regions while learning to reconstruct the masked tokens.

Another methodology for masking patch tokens is based on attention or importance scores. In this approach, the model computes the significance of each patch token through a preliminary attention mechanism. Tokens with lower attention scores or less significance are more likely to be masked, as the model can focus on learning to reconstruct less salient parts of the image. Conversely, masking more informative or highly attended patches can make the task more challenging and potentially improve the model's ability to learn contextual relationships between patches. This technique leverages the idea that not all parts of the image are equally important for reconstruction.

A more sophisticated masking strategy involves selective masking based on image structure or semantic information. For instance, patches corresponding to important regions like object boundaries, textures, or high-contrast areas may be masked to encourage the model to learn meaningful visual features. This structured masking is often informed by pre-existing image segmentation algorithms or gradient-based saliency maps. By strategically masking important areas, the model is challenged to leverage contextual clues from surrounding patches, improving its ability to handle complex visual patterns and improve downstream performance in tasks like segmentation or object recognition.

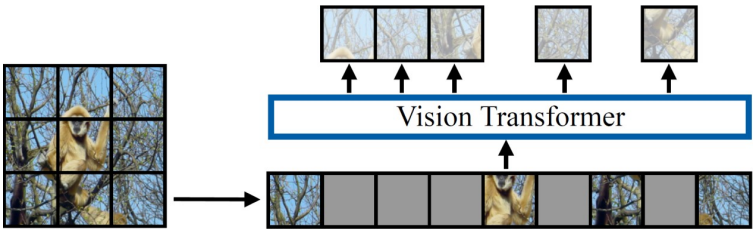


Figure 3.1: Simple example of patch token masking

3.6.9 Auto-regressive transformer decoders.

The autoregressive decoder is a fundamental component in sequence generation models, particularly in natural language processing and computer vision tasks. It models the conditional probability of each element in a sequence given the previous elements. Formally, for a sequence of tokens $x = (x_1, x_2, \ldots, x_T)$, the joint probability is factorized in an autoregressive manner as:

$$p(x_1, x_2, \ldots, x_T) = \prod_{t=1}^{T} p(x_t \mid x_1, x_2, \ldots, x_{t-1})$$

This factorization ensures that each token x_t is generated conditioned on all the previously generated tokens, making the model sequential in nature.

The autoregressive property can be seen in decoders such as those used in Transformers and RNNs. The decoder generates the output sequence one step at a time, where each step involves predicting the next token. During training, teacher forcing is often employed, where the ground truth sequence is fed as input, but during inference, the model relies on its own generated output as context. The probability of generating the next token is computed using a softmax over the vocabulary:

$$p(x_t \mid x_1, x_2, \ldots, x_{t-1}) = \text{softmax}(W h_t)$$

where h_t is the hidden state at time step t, and W is the learned weight matrix.

A key challenge with autoregressive decoders is the accumulation of errors during inference. If an incorrect token is generated at any step, it becomes part of the context for generating future tokens, potentially leading to further mistakes. However, autoregressive decoders remain highly effective for many tasks due to their strong ability to capture temporal dependencies in sequence data. The sequential nature of these models contrasts with non-autoregressive approaches, where all tokens are predicted independently or in parallel.

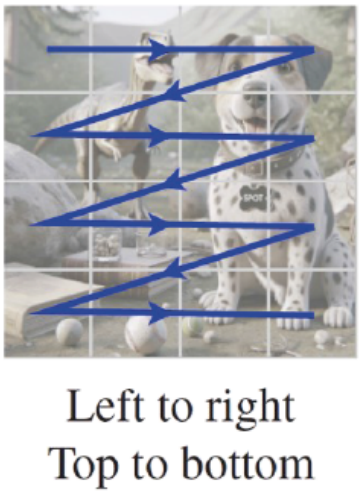


Figure 3.2: AR Decoder in ViT

3.6.10 Generative Adversarial Networks (GANs)

Generative Adversarial Networks, introduced by Ian Goodfellow et al. in 2014 [19], consist of two neural networks: a generator and a discriminator, which are trained simultaneously through adversarial training.

Architecture The architecture of GANs includes:

- **Generator:** This network takes random noise as input and generates images.

- **Discriminator:** This network evaluates the authenticity of the images, distinguishing between real and generated images.

The generator aims to produce images that are indistinguishable from real images, while the discriminator tries to correctly classify real and fake images.

Training Process The training process of GANs involves a minimax game:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{3.2}$$

where G represents the generator, D represents the discriminator, $p_{data}(x)$ is the distribution of real data, and $p_z(z)$ is the distribution of the input noise.

Diffusion Models Diffusion models are a class of generative models that iteratively refine images from noise. They have gained popularity due to their simplicity and effectiveness in generating high-quality images.

Architecture Diffusion models work by modeling the data distribution as a reverse diffusion process. This involves:

- **Forward Process:** Gradually adding noise to the data to transform it into a Gaussian distribution.
- **Reverse Process:** Learning to reverse the noise addition process to generate new data samples.

Training Process The training of diffusion models involves learning the reverse process through a parameterized model, typically a neural network. The objective is to minimize the difference between the model's output and the original data distribution. Mathematically, this can be represented as:

$$L(\theta) = \mathbb{E}_{t,x_0,\epsilon}\left[\|\epsilon - \epsilon_\theta(x_t, t)\|^2\right] \tag{3.3}$$

where x_t represents the noisy data at step t, ϵ is the added noise, and ϵ_θ is the model's prediction of the noise.

3.6.11 Variational Autoencoders (VAEs)

Variational Autoencoders, introduced by Kingma and Welling in 2013 [45], are a type of generative model that combine principles from variational inference and deep learning. They are designed to learn a probabilistic mapping from data to a latent space and back, enabling the generation of new data samples.

Architecture The architecture of VAEs includes:

- **Encoder:** This network maps input data x to a latent representation z. Instead of directly outputting z, the encoder outputs parameters of a distribution (typically Gaussian), from which z is sampled.
- **Decoder:** This network reconstructs the data from the latent representation z. It aims to produce outputs that are as close as possible to the original input data.

The encoder and decoder are trained jointly to maximize the evidence lower bound (ELBO) on the likelihood of the data.

Training Process The training process of VAEs involves optimizing the ELBO, which consists of two terms: the reconstruction loss and the KL-divergence between the learned latent distribution and the prior distribution. This can be mathematically expressed as:

$$\mathcal{L}(\theta, \phi; x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \mathrm{KL}(q_\phi(z|x)||p(z)) \tag{3.4}$$

where $p_\theta(x|z)$ is the decoder network, $q_\phi(z|x)$ is the encoder network, and $p(z)$ is the prior distribution over the latent variables.

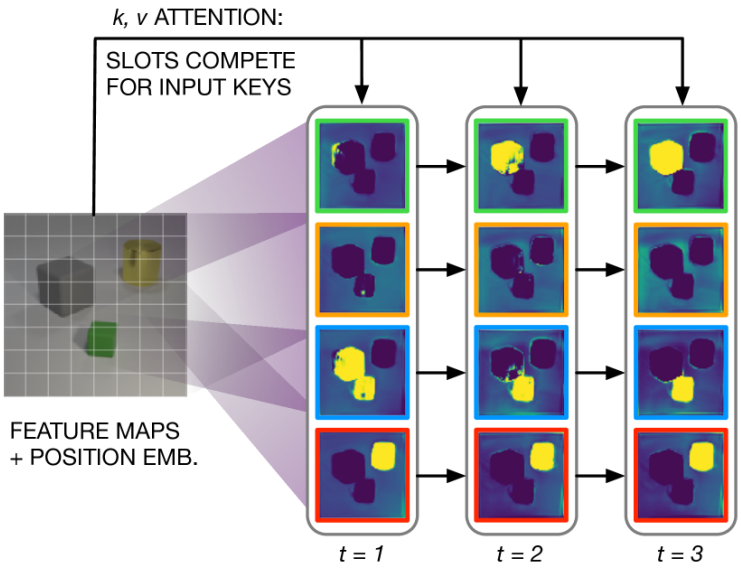
3.7 Previous Work

3.7.1 Slot Attention

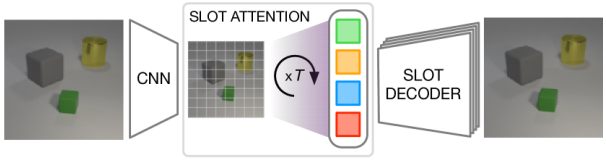
The Slot Attention module [3] maps a set of N input feature vectors to K output vectors, referred to as slots, where each slot typically represents an object or entity in the input.

The module employs an iterative attention mechanism to refine randomly initialized slots over T iterations, ensuring that each slot binds to specific parts of the input features. This process involves a competitive softmax-based attention mechanism, which normalizes the attention coefficients over the slots, thereby promoting competition among slots for explaining the input features. The attention mechanism utilizes dot-product attention, with attention coefficients normalized using a fixed temperature, defined by the input feature dimension D.

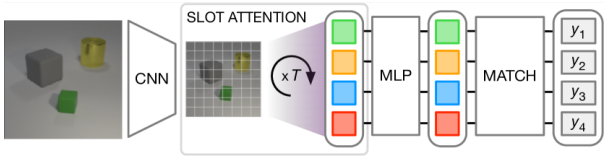
The slots are updated using a weighted mean of the input values, which enhances the stability of the attention mechanism. A Gated Recurrent Unit (GRU) with optional multi-layer perceptron (MLP) and residual connections updates the slots at each iteration. Layer normalization is applied to both the inputs and slot features to accelerate training convergence. The Slot Attention module is permutation invariant with respect to the input and permutation equivariant with respect to the slots, making it particularly suitable for set-based inputs. The time complexity of the module is $\mathcal{O}(T \cdot D \cdot N \cdot K)$.



(a) Slot Attention module.



(b) Object discovery architecture.



(c) Set prediction architecture.

Figure 3.3: (a) Slot Attention module and example applications to (b) unsupervised object discovery and (c) supervised set prediction with labeled targets

3.7.2 Spot

The Spot framework[1], excels in object-centric image segmentation, becoming the SOTA model for unsupervised segmentation.

Spot uses slot-based auto-encoders, a framework used in object-centric learning, where the model consists of an image encoder and a slot-attention module. The encoder extracts patch-wise features from an image, while the slot-attention module groups these features into a smaller number of latent vectors, or ”slots,” each representing an object in the image. The decoder then reconstructs the original image or another target signal from these slot vectors. This framework benefits from using self-supervised pre-trained feature encoders, such as DINO[46], to enhance the quality of the extracted features and reconstruction targets.

A key component in this architecture is the autoregressive transformer decoder, which predicts features sequentially using prior target features and slots, employing teacher-forcing during both training and testing. The primary objective is to decompose the input image into its constituent objects, which is evaluated using slot-attention maps derived

Algorithm 2 Slot Attention module

Input: inputs $\in \mathbb{R}^{N \times D_{\text{inputs}}}$, slots $\sim \mathcal{N}(\mu, \text{diag}(\sigma)) \in \mathbb{R}^{K \times D_{\text{slots}}}$
Layer params: k, q, v: linear projections for attention; GRU; MLP; LayerNorm (x3)
inputs = LayerNorm(inputs)
for $t = 0$ to T **do**
    slots_prev = slots
    slots = LayerNorm(slots)
    attn = Softmax($\frac{1}{D} k(\text{inputs}) \cdot q(\text{slots})^T$, axis='slots') ▷ norm. over slots
    updates = WeightedMean(weights=attn + ϵ, values=$v(\text{inputs})$) ▷ aggregate
    slots = GRU(state=slots_prev, inputs=updates) ▷ GRU update (per slot)
    slots += MLP(LayerNorm(slots)) ▷ optional residual MLP (per slot)
end for
return slots

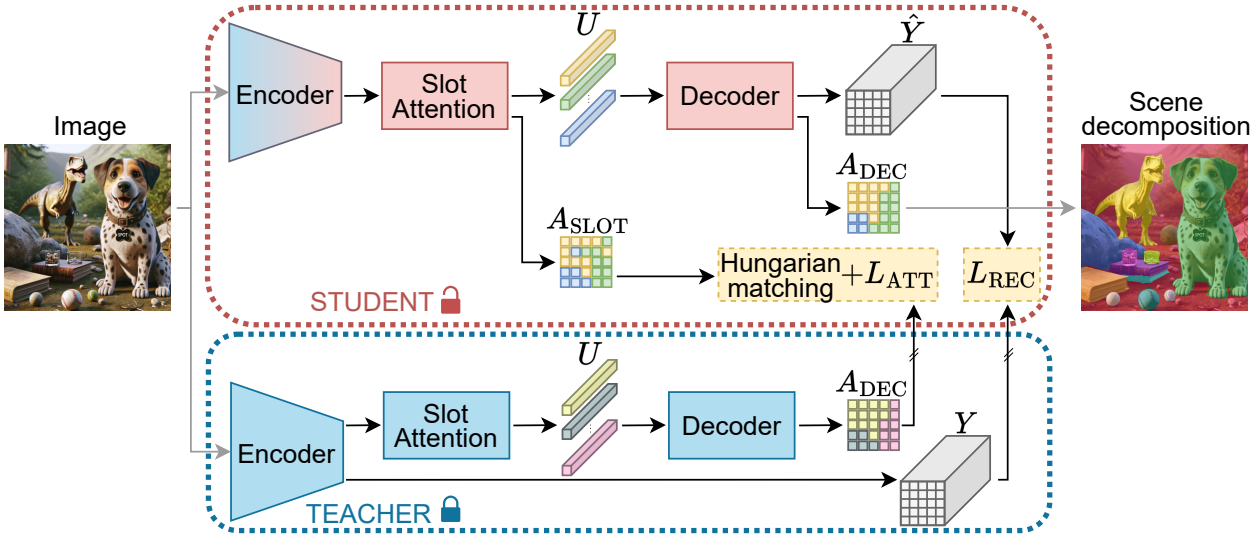


Figure 3.4: Spot Framework.

from the slot-attention module or the decoder.

A novel two-stage training method, SPOT, improves this process by using slot-attention distillation. In the first stage, the model is trained with a reconstruction loss. In the second stage, a teacher-student framework guides the student model with an additional attention distillation loss, enhancing the object-specific grouping capability of the slot-attention module. This method also stabilizes the training of self-supervised Vision Transformers, enabling better fine-tuning and maximized learning capacity.

Figure 3.5: Samples of annotated images in the MS COCO dataset.

3.7.3 Mage

MAGE unifies generative tasks and representation learning by first quantizing input images into semantic tokens using a pre-trained VQGAN model [47]. The model employs a variable masking ratio strategy, masking out some input tokens randomly, and then uses an encoder-decoder transformer architecture to predict the masked tokens. To enhance the learned representations, a contrastive loss similar to SimCLR [35] is added to the encoder's output, termed MAGE-C.

Pre-training

Tokenization: Input images are tokenized into a sequence of semantic tokens, enabling the model to operate on higher-level representations rather than raw pixels, which benefits both generation and representation learning.

Masking Strategy: The masking ratio m_r is sampled from a truncated Gaussian distribution, and a portion of the input tokens is masked and replaced with a learnable mask token. Further, a significant fraction of masked tokens is dropped to reduce pre-training time and memory consumption, inspired by findings in MAE [6].

Encoder-Decoder Design: The model uses a Vision Transformer (ViT) encoder-decoder structure. After masking and dropping tokens, a learnable class token is concatenated to the input sequence. The encoder processes this sequence, and its output is padded with the class token feature before being decoded to reconstruct the original tokens.

Reconstructive Training: The training objective is to reconstruct masked tokens from unmasked tokens using a cross-entropy loss. This loss is applied only to masked tokens to optimize both generation and representation performance, as supported by MAE's ob-

servations [6].

4. METHOD

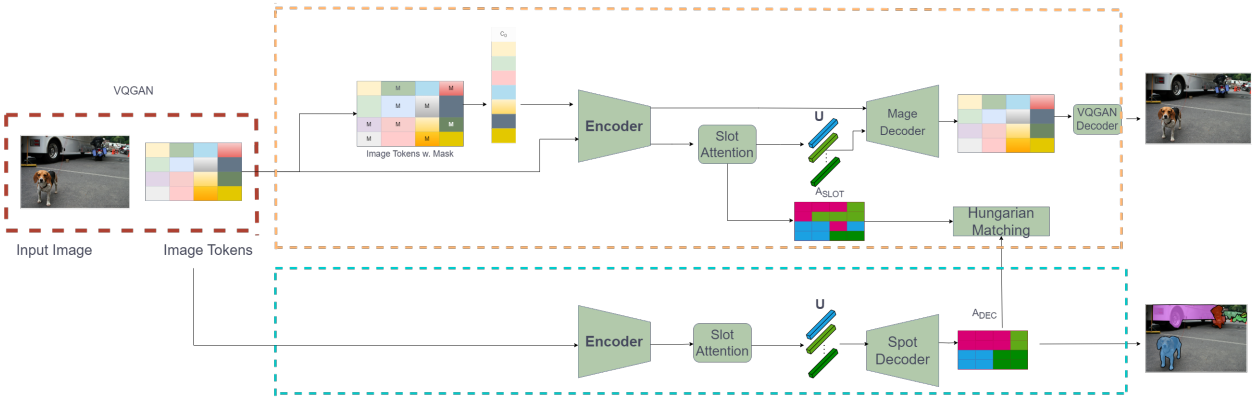


Figure 4.1: Enhancing image synthesis by utilizing unsupervised object-centric learning. Our two-stage approach starts with exclusive training in the initial stage(blue) by distilling the attention maps from the image. Then on the second stage(orange) we train the model for image synthesis, which we aid with the hungarian matching of the attention maps.

4.1 Model Framework

In our framework, we employ multiple stages in our training mechanism, in order to achieve both good object-centric image segmentation, as well as image reconstruction. Our starting point is the Mage[2] pretrained model. We start with the first stage by using the SPOT[1] framework. We distill the attention maps from the Spot decoder and save them. The purpose of the extracted masks is, on the second stage of training which focuses on image reconstruction from partially masked images, to sustain the object segmentation and also try to prove that image segmentation helps to in the context of reconstruction.

Slot-based auto-encoders. In our framework, we employ a slot-based auto-encoding approach, adapted from common practices in object-centric learning. This approach consists of two modules: an image encoder and a slot-attention mechanism. The encoder extracts n patch-wise features from image X, which are then grouped into $k < n$ latent vectors, or slots, $U = (u_1; \ldots; u_k) \in \mathbb{R}^{k \times d_u}$, where each slot represents a distinct object or part of the image. We employ the MAGE encoder which was trained to handle masked image reconstruction. The slot-attention module is responsible for clustering these patch-wise features into object-centric representations. Each slot is iteratively refined through an attention mechanism, which forces the slots to compete over the patches, ensuring that each slot captures information about a specific object in the scene. By utilizing the MAGE encoder, our method benefits from a more detailed and semantically meaningful feature extraction process, making the object segmentation more precise compared to other methods.

Defining Y, $\hat{Y} \in \mathbb{R}^{n \times d_y}$ as the target features and the predicted reconstructions, respectively, the model is trained by minimizing the reconstruction loss:

$$L_{REC} = \frac{1}{n \cdot d_y} \|Y - \hat{Y}\|_2^2.$$

Defining the reconstruction task with high-level features provides a valuable training signal for learning object-centric representations from real-world data.

Auto-regressive transformer decoders. For the decoding stage, we employ an autoregressive transformer as our decoder. This transformer predicts the target feature $\hat{y}_i$ at position i based on prior target features $Y_{<i}$ and the slot representations U. Specifically, the decoder is tasked with reconstructing high-level target features from the slots, guided by the attention mechanism. To achieve this, the decoder is composed of a series of transformer blocks, including causal self-attention layers and cross-attention layers that link each patch back to its corresponding slot.

The prediction is performed using teacher-forcing, where the input to the decoder is the target feature set $Y_< \in \mathbb{R}^{n \times d_y}$, which consists of target features right-shifted by one position (excluding the last token). A learnable Beginning-Of-Sentence (BOS) token is prepended: $Y_< = (y_{BOS}; y_1; \ldots; y_{n-1})$.

Slot-attention distillation

We distill the slot-attention maps from the decoder to the encoder to improve object segmentation. Attention maps $A_{\text{DEC}} \in \mathbb{R}^{n \times k}$, which are derived from the autoregressive decoder during the reconstruction phase, are transferred to the encoder. These masks provide valuable information for refining the slot representations.

Slot-attention distillation loss L_{ATT}. The attention maps A_T from the teacher's decoder are converted to hard-assignment masks by applying row-wise argmax:

$$A'_T = \text{one-hot}(\text{argmax}(A_T)) \in \{0, 1\}^{n \times k}.$$

To map the teacher's masks A'_T to the student's masks A_S, we use Hungarian matching based on the Intersection over Union (IoU) between the masks. Finally, the cross-entropy loss is applied between A'_T and A_S:

$$L_{\text{ATT}} = \frac{1}{n} \langle A'_T, \log(A_S) \rangle_F.$$

The total training loss for this stage is:

$$L = L_{\text{REC}} + \lambda L_{\text{ATT}},$$

where λ controls the weight of the distillation objective.

4.1.1 First Stage

In the first stage we use the SPOT[1] (Self-Training with Patch-Order Permutation) model, the focus is on training the slot-based auto-encoders using a reconstruction loss. First, We select image encoder and the slot-attention module.

Then, the slot-attention module uses an iterative attention-based approach that starts with initial query slot vectors $\tilde{U}$ and generates the final output slot vectors U. The focus of this module is a modified slot-to-patch cross-attention layer. The matrix A for the slot-attention module is derived from the cross-attention map obtained during the last iteration.

$$A_{\text{SLOT}} = \text{Softmax}\left(\frac{Q_E K_E^\top}{d_p}\right)^\top \in \mathbb{R}^{n\times k}, \tag{4.1}$$

where $K_E \in \mathbb{R}^{n\times d_p}$ are keys derived from the patch-wise features extracted by the image encoder from the input image X, and $Q_E \in \mathbb{R}^{k\times d_p}$ are queries computed from the slot vectors of the previous iteration. The Softmax function is applied along the slots dimension to enforce competition.

The decoder module employs transformer-based decoders that utilize patch-to-slot cross-attention layers to incorporate information from slot vectors U. In this setup, the attention maps A, denoted as $A_{\text{DEC}} \in \mathbb{R}^{n\times k}$, are derived by averaging the patch-to-slot cross-attention maps across H heads from the final transformer layer.

$$A_{\text{DEC}} = \frac{1}{H}\sum_{j=1}^{H} \text{Softmax}\left(\frac{Q_j K_j^\top}{d_h}\right), \tag{4.2}$$

The keys $K_j \in \mathbb{R}^{k\times d_h}$ are computed from the slot vectors U, and the queries $Q_j \in \mathbb{R}^{n\times d_h}$ are derived from the transformed decoder input $Y_<$ from previous layers. The Softmax function is applied along the slots dimension.

Afterwards, it proposes a self-training scheme that distills slot-based attention maps from the decoder to the encoder, and as a result advances the object segmentation captured by the slot attention.

4.1.2 Second Stage

In our model we use two branches, that use a slot based auto-encoder framework. We split our model in two branches. At the beginning for both branches we tokenize the images from the pixel to the latent space, using the VQGAN[19] model.

At the first branch, using the pretrained MAGE[2] encoder, we choose to mask a part of the tokens. Specifically, the masking ratio mr is initially determined by sampling from a

truncated Gaussian distribution with a mean of 0.55, a minimum of 0.5, and a maximum of 1. This follows the methodology used in the original Mage framework. Given the length l of the input sequence of tokens, we then randomly select mr×l tokens to mask, substituting them with a learnable mask token [M].

On the second branch, we choose to keep all the patches of the encoder and then pass it to the slot Attention module. We keep the hard mask pooled slots from this stage. Afterwords, the decoder tries to reconstruct the masked patch tokens of the first branch. We combine both the masked pooled slots and the encoded patches of the first branch and pass it through the decoder.

Reconstructive Training, Let $Y = [y_i]_{i=1}^{N}$ be the tokenized latent tokens, where N is the token sequence length, and $M = [m_i]_{i=1}^{N}$ being the binary mask that determines which tokens are to be masked. The training objective is to reconstruct the masked tokens from the unmasked tokens, with the addition of the slots extracted from the second branch. To achieve this, we add a cross-entropy loss between the ground-truth one-hot tokens and the output of the decoder. Specifically,

$$\mathcal{L}_{\text{reconstructive}} = -\mathbb{E}_{Y \in \mathcal{D}} \left(\sum_{i, m_i = 1} \log p(y_i \mid Y_M) \right), \tag{4.3}$$

where Y_M represents the subset of unmasked tokens in Y, and $p(y_i \mid Y_M)$ is the probability predicted by the slot-encoder and decoder network, conditioned on the unmasked tokens. We follow the approach of optimizing this loss exclusively on the masked tokens, like the MAE[6] approach.

4.1.3 Third Stage

Kmeans In the third stage of our framework we are clustering the slots of the training dataset, extracted from the trained model. Then from these slots we can do different experiments such as in the image synthesis, which will be discussed below, by replacing the slots in the evaluation dataset with the closest K-means clusters. With this method we can generate new images based on the K-means centroids.

Masked Slots The next phase of the model is train it to fully generate new slots. The methodology involves creating two possible occasions with a probability p during the training phase. One occasion is to keep the classic training method described in the second stage. The second occasion is to hide all the latent tokens and some slots from the decoder, and also replace the remaining unhidden slots with the closest centroids from the K-means algorithm. We add a cross-entropy loss which tries to predict the K-means ids of the hidden slots.

4.1.4 Post-training Evaluation

To generate images for generative model evaluation, we initiate the process with the slots extracted from the Slot Attention module, while all other tokens are masked like in Maskgit[48]. At each iteration, our model predicts the tokens for the remaining masked positions. Subsequently, we sample some of these predicted tokens, prioritizing those with higher predicted probabilities, and replace the corresponding masked tokens with the sampled predictions. The number of tokens replaced in each iteration is determined by a cosine function, starting with fewer replacements in the initial iterations and increasing the number towards the later iterations.

This iterative process is repeated for a total of 20 steps to fully generate an image. For representation learning, we apply global average pooling to the features output by the ViT encoder, using the pooled features as input for the classification head.

5. EXPERIMENTS

5.1 Setup

The COCO (Common Objects in Context)[49] dataset is a large-scale object detection, segmentation, and captioning dataset, with a diverse collection of real-world images, each featuring multiple co-occurring objects. It contains over 330,000 images, with more than 200,000 labeled images and 1.5 million object instances across 80 object categories. Each image in the dataset is annotated with object bounding boxes, segmentation masks, and detailed captions, making it a rich resource for various computer vision tasks.

In the realm of object detection, COCO is extensively used to train and evaluate models that identify and localize objects within images. Its challenging and diverse set of annotations helps in developing robust detection algorithms that can generalize well to real-world scenarios. For semantic scene labeling, COCO provides dense annotations that allow models to assign a class label to each pixel in an image, facilitating the development of algorithms capable of understanding complex scenes at a fine-grained level. The comprehensive nature of the COCO dataset has made it a benchmark standard for evaluating object detection and scene understanding models, driving significant progress in these areas.

Beyond object detection, COCO's extensive annotations enable advancements in image captioning, where models are trained to generate descriptive captions for images. By providing human-written captions for each image, COCO allows researchers to develop and evaluate models that bridge vision and language. This multimodal capability has catalyzed progress in tasks like visual question answering (VQA) and image generation. Furthermore, COCO's emphasis on complex, cluttered scenes featuring multiple interacting objects pushes models to understand context and relationships between objects, leading to more sophisticated and context-aware computer vision systems.

The dataset also plays a crucial role in developing models that handle object segmentation, where each object is precisely outlined using segmentation masks. COCO's instance segmentation annotations have been instrumental in the rise of powerful models like Mask R-CNN, which not only detect objects but also delineate their exact shapes. This level of granularity in object localization is essential for tasks such as autonomous driving, medical image analysis, and robotic perception, where precise understanding of object boundaries is critical.

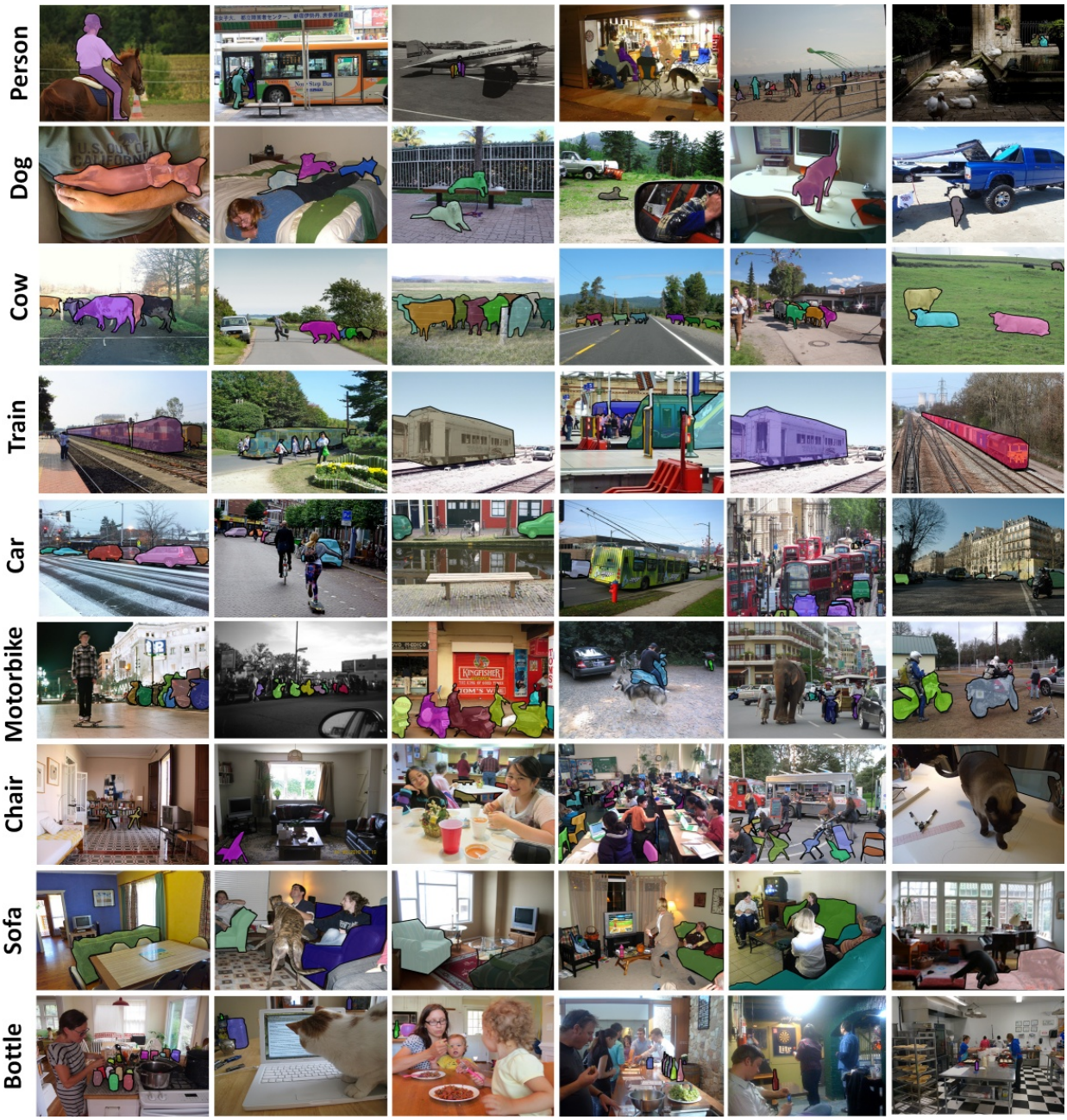


Figure 5.1: Samples of annotated images in the MS COCO dataset.

5.1.1 Metrics

To evaluate object-centric learning, we utilize several metrics: Mean Best Overlap at the instance level (MBO^i), Mean Best Overlap at the class level (MBO^c), Foreground Adjusted Rand Index (FG-ARI), and Mean Intersection over Union (MIOU). MBO^i measures the best overlap for each ground truth mask, whereas MIOU uses Hungarian matching to create a one-to-one correspondence between predicted and ground truth segments. We emphasize MBO and MIOU metrics because they include background pixels, providing a comprehensive assessment of how well the masks align with the objects. On the other hand, FG-ARI, a cluster similarity metric, focuses solely on foreground pixels, which can potentially misrepresent segmentation quality by ignoring the accuracy of predicted masks and encouraging over-segmentation.

To evaluate segmentation and object-centric models in image-based tasks, several key metrics are used. The **Mean Intersection over Union (mIoU)** is defined as the average ratio of the intersection and union between predicted and ground truth segments across all classes:

$$\text{mIoU} = \frac{1}{C} \sum_{c=1}^{C} \frac{|P_c \cap G_c|}{|P_c \cup G_c|}$$

where P_c and G_c represent the predicted and ground truth sets for class c. The Mean Boundary Overlap for Instances (MBO^i) measures the overlap between instance boundaries, defined as:

$$\text{MBO}_I = \frac{1}{N} \sum_{i=1}^{N} \frac{|B(P_i) \cap B(G_i)|}{|B(P_i) \cup B(G_i)|}$$

where $B(\cdot)$ is the boundary operator, and N is the number of instances. Similarly, the Mean Boundary Overlap for Classes (MBO^c) focuses on class boundaries and is defined as:

$$\text{MBO}_C = \frac{1}{C} \sum_{c=1}^{C} \frac{|B(P_c) \cap B(G_c)|}{|B(P_c) \cup B(G_c)|}$$

The Foreground Adjusted Rand Index (FG-ARI) quantifies clustering performance by adjusting for chance, and is defined as:

$$\text{FG-ARI} = \frac{\sum_{i,j} \binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}}$$

where n_{ij} is the number of elements in the intersection of clusters i and j, and a_i, b_j are the sums of cluster sizes.

5.2 Segmentaion metrics

For the implementation we use AdamW optimization with betas b_1 0.9, b_2 0.95, weight decay of 0.05 and batch size 32. For the two training stages we use 50 and 30 epochs on the COCO dataset. We implement our model with the ViT-B/16 Mage encoder and decoder. On the slot attention module, we choose to create 7 slots. All models were trained on a single GPU with 24 Gbytes.

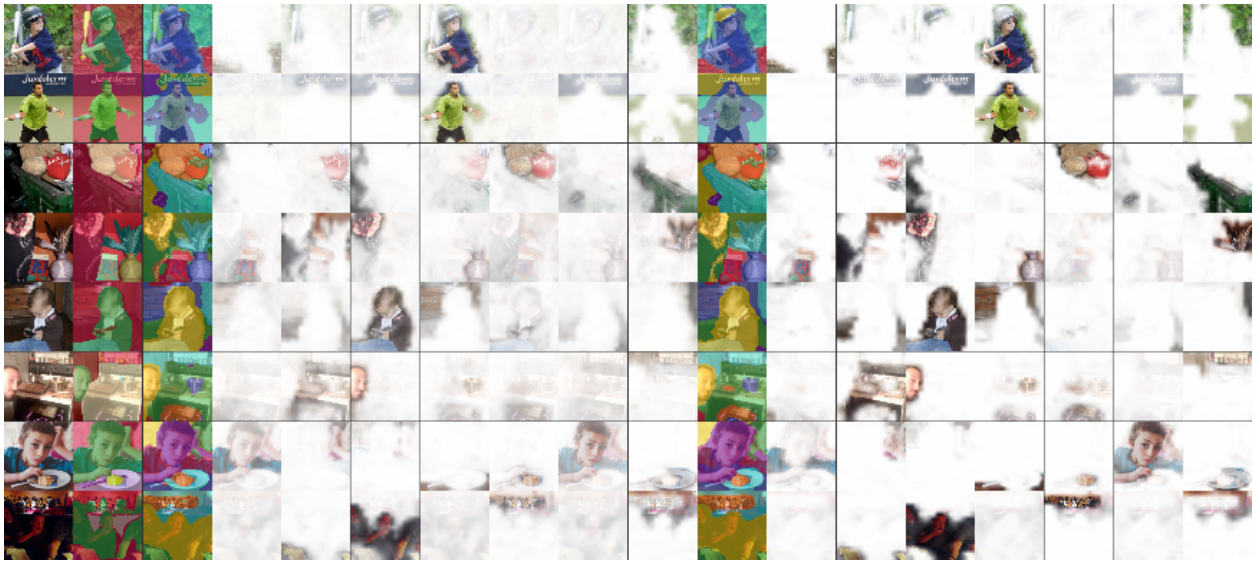


Figure 5.2: Image Segmentations using the second stage of our model

Model	mBO^i	mMBO^c	FG-ARI
SLATE[11]	29.1	33.6	-
Spot + DINO ViT	**35**	**44.7**	37
SlotDiffusion	31	35	37.2
Ours + Mage	29.7	38.6	**40.07**

Table 5.1: Evaluation metrics for object segmentation in the COCO Dataset compared to other SOTA models

5.3 Image Reconstruction

During the image synthesis experiments, we use the iterative process mentioned in the previous chapter. The images shown are from the validation of the MS COCO dataset. During this process, we have no latent tokens and start with only the slots from the slot attention module.

Figure 5.3: Reconstructed images using the COCO Dataset, these images were masked at random by at least 50 percent

We can see that the model struggles with the faces of people as well as letters, like in the stop sign. The majority of the images can be reconstructed starting only from the slot attention module.

5.4 Image Generation

In our attempt to create new images, we use the Kmeans approach to cluster the slots of the training dataset. Then, in the test set, using the iterative generation approach, we replace all the slots from the images with their closest centroids from the Kmeans model. The results look promising, even though the images are not yet looking real-life like.

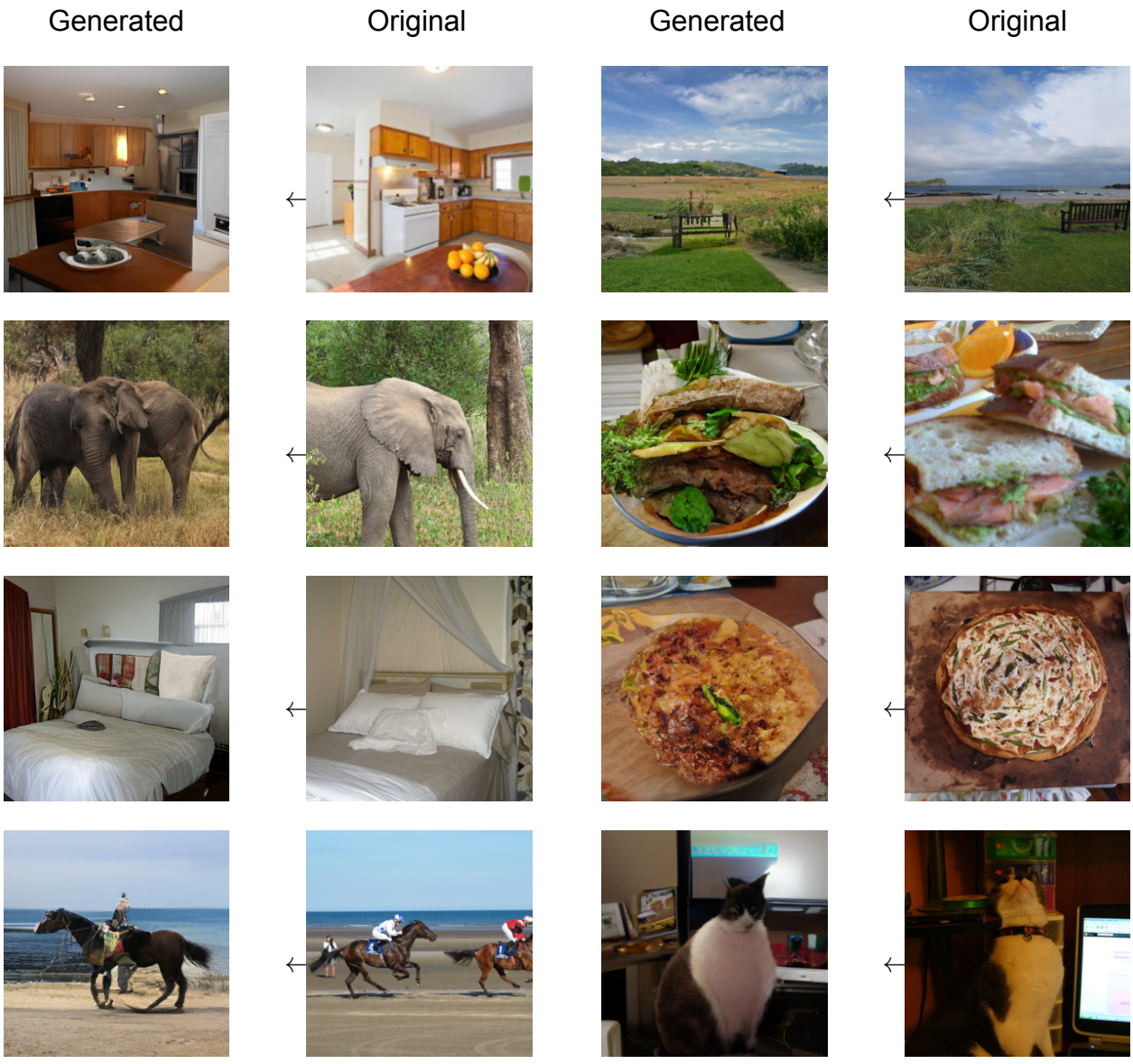


Figure 5.4: Images that create good enough results using our Kmeans approach in order to Generate new images

There were some hyper-parameters that needed tuning in the process such as the number of clusters that we chosen for the segmentation of our slots. At the beginning we chose around 100 clusters which did not perform well enough. A lot of the generated images where similar. After a not of testing the best approach was around 4 and 8 thousand clusters, so that the model can both generalize and also give detailed results in our images.

Also since the slots took a lot of memory, we did the clustering in batches. We concluded in 1024 slots per batch after testing.

For example we can see that we can generate different backgrounds that make sense and have similarity with our initial images. This approach is very interesting for image editing as well.

This methodology helps with creating a more explainable version of what our image generation methodology does.

5.5 Further analysis and experiments

During the testing process of finding the best possible setup, that tries to balance the trade off between the segmentation and the reconstruction metrics, we conducted multiple experiments. These varied from adding different modules to our setup to hyper-parameter tuning. Below we see some comparisons of the major experiments we run.

5.5.1 Encoder fine-tuning

Starting from the pre-trained Mage model, we had the choice of only training the slot attention module and the Decoder, or also fine-tuning the encoder as well. This proved to be a very interesting concept, since fine-tuning the encoder came out to be in of the biggest difference makers.

Model	mBO^i	$mMBO^i$(slots)	Reconstruction Loss
Frozen Encoder	**29.1**	**29.3**	4.3
Fine-tuned encoder	19	28.7	**1**

Table 5.2: Evaluation metrics for encoder fine-tuning. The lowest the reconstruction loss, the better the image reconstruction.

We can see from this experiment that the frozen encoder significantly increases the capabilities of the model to segment the images, where on the other hand

5.5.2 Slots vs Hard mask pooling vs Soft mask pooling

There are many ways to generate what we call 'slots'. In Slot Attention, the slots obtained directly from the module are learned representations that iteratively bind to distinct objects or parts of a scene, guided by attention maps. When applying soft mask pooling, the attention maps are continuous values (soft masks) that softly weight the latent space features, allowing each feature to contribute to multiple slots to varying degrees. In contrast, hard mask pooling uses binary attention maps (hard masks), where each feature is either fully assigned to one slot or excluded from others, creating a strict partition of the latent space without overlap.

Model	mBO^i	$mMBO^i$(slots)	Reconstruction Loss
Slots	28	28.4	4.5
Soft mask pooling	28.3	28.5	4.2
Hard mask pooling	**29.8**	**30**	4.2

Table 5.3: Evaluation metrics for slot methodologies. These results are from the second stage, using the same model shown at chapter 4, with a frozen encoder

These evaluation metrics clearly show that the hard mask pooling is the better way to extract our slots. In more experiments it proved its reliability to give stable results.

5.5.3 Spot Decoder Addition on second stage

We experimented with adding another decoder to the second stage of our model. The Spot decoder is an auto-regressive decoder, whose loss on the second stage was added to help the model in image segmentation.

Model	mBO^i	$mMBO^i$(slots)	Reconstruction Loss
Second Stage + Spot Decoder	**30**	**30.1**	5
Second Stage	29.8	30	**4.2**

Table 5.4: Evaluation metrics for the second stage of our model with the added loss of spot Decoder.

We can see from this table that the impact of the extra decoder in not signification in the segmentation metrics, while it reduces the reconstruction capabilities of the model. Moreover, it significantly slowed down our model in training times, so we chose to not use it.

5.5.4 Stage 2 only

We can compare how the model performed with only the second stage and the spot decoder, in comparison to the 2 stage approach that we show in our methodology.ns

Using only the second stage was our initial plan, but because of the higher segmentation metrics of stage one we chose to use it in order to guide the model to segment the objects more reliably.

Model	mBO^i	$mMBO^c$	Reconstruction Loss
Full architecture	**29.8**	**30**	4.2
Second Stage only	27	27.3	4.1

Table 5.5: Evaluation metrics for object segmentation in the COCO Dataset with and without the second stage of the model

6. CONCLUSIONS

This project has been an invaluable learning experience, offering useful insights into the complexities of image synthesis and object-centric image segmentation. Throughout the process, we developed a better understanding of how these tasks intersect, providing a roadmap for future research in the area. Although not all objectives were fully met, the project successfully demonstrated the feasibility of leveraging object-centric learning for image reconstruction, which, in turn, could pave the way for more advanced models in the future.

One of the most challenging aspects was balancing high-quality image synthesis with precise object-centric segmentation. Achieving this coexistence required addressing competing priorities: generating real world images while maintaining accurate object segmentation boundaries. This difficulty was reflected in the evaluation metrics, while our segmentation results did not reach state-of-the-art performance, they still achieved promising outcomes that highlight the potential of the methods used. Moreover, Vision Transformer (ViT) models made a significant impact on this project, as their ability to model global relationships in images early in the network can greatly enhance segmentation performance.

Slot Attention also proved to be highly useful in the context of image segmentation. Its iterative refinement of object slots allowed for more structured and interpretable representations, facilitating better segmentation, even in complex scenes. Furthermore, working with real-world images from the COCO dataset introduced significant challenges, particularly for image reconstruction. The diverse and cluttered nature of these images made it difficult to generate realistic reconstructions, adding complexity to the task but also making it more engaging and rewarding. In sum, this project has provided valuable takeaways and has laid a strong foundation for future work in object-centric image synthesis and segmentation.

ABBREVIATIONS - ACRONYMS

AR	Auto Regressive
AI	Artificial Intelligence
ViT	Vision Transformer
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network

APPENDIX A. FIRST APPENDIX

These are a few more of the reconstructed images. They were sampled from the Imagenet Dataset and are used to train most of the Image Generation models.

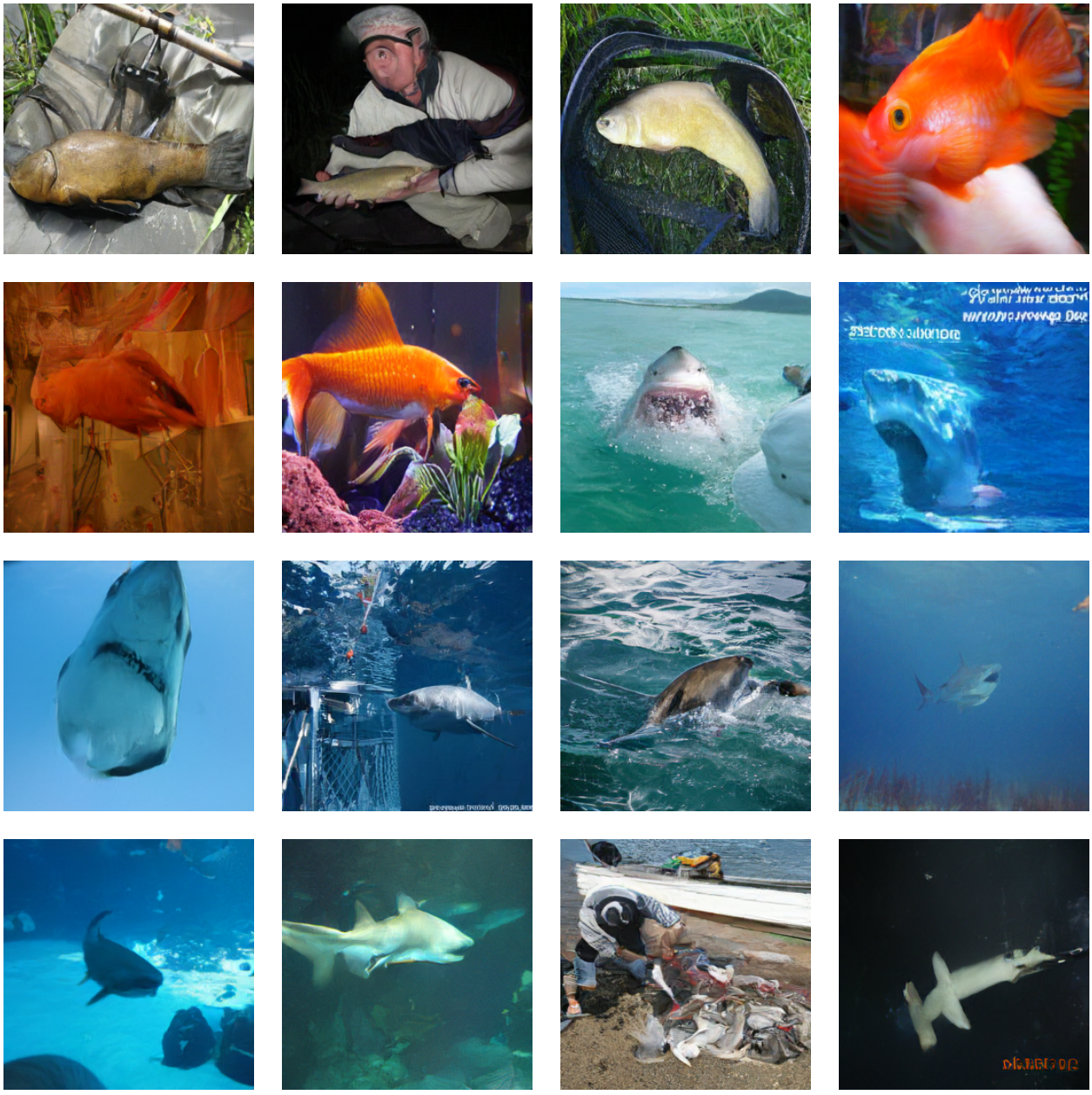


Figure A.1: Reconstructed images from the Imagenet Dataset.

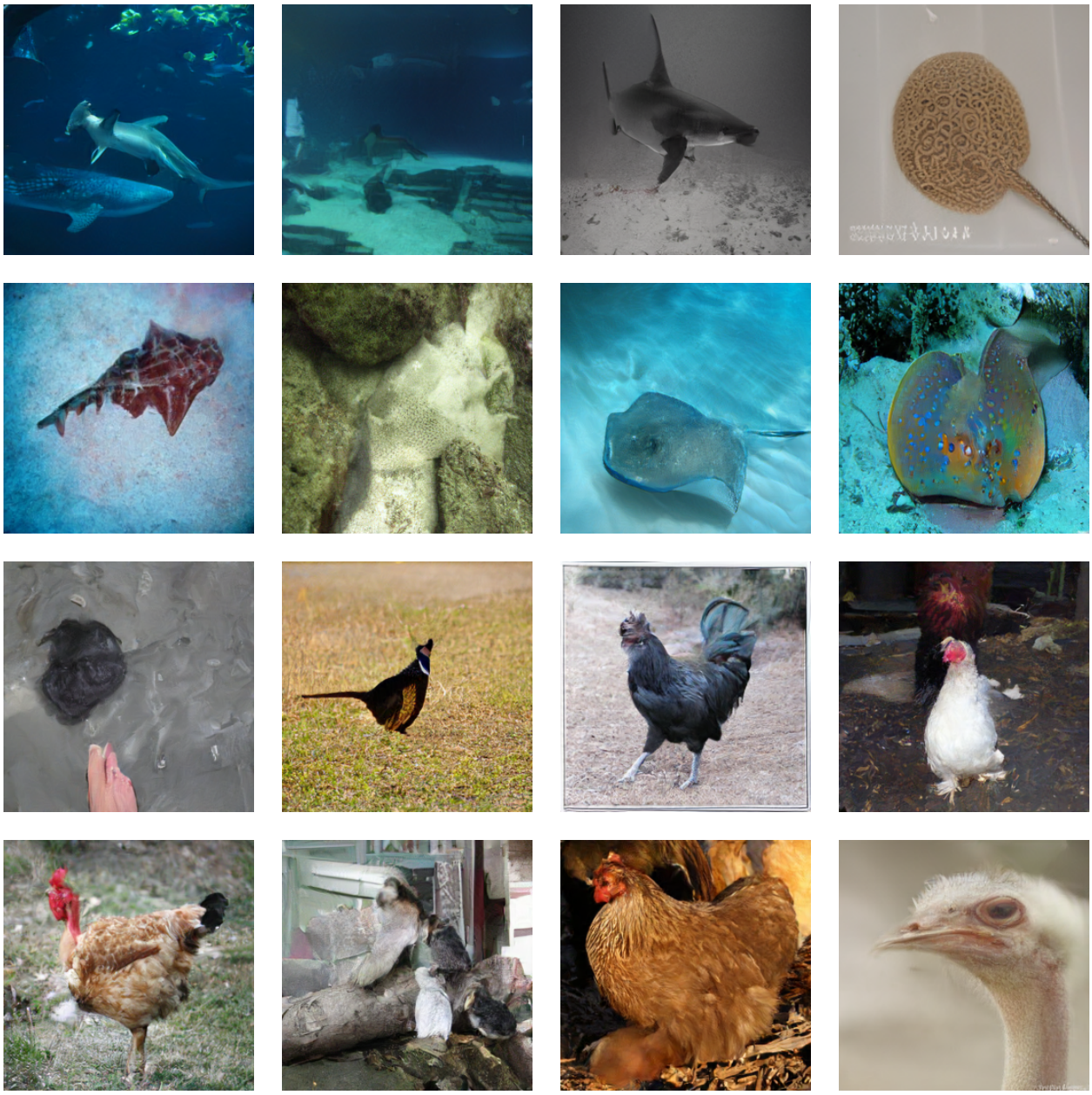


Figure A.2: Reconstructed images from the Imagenet Dataset.

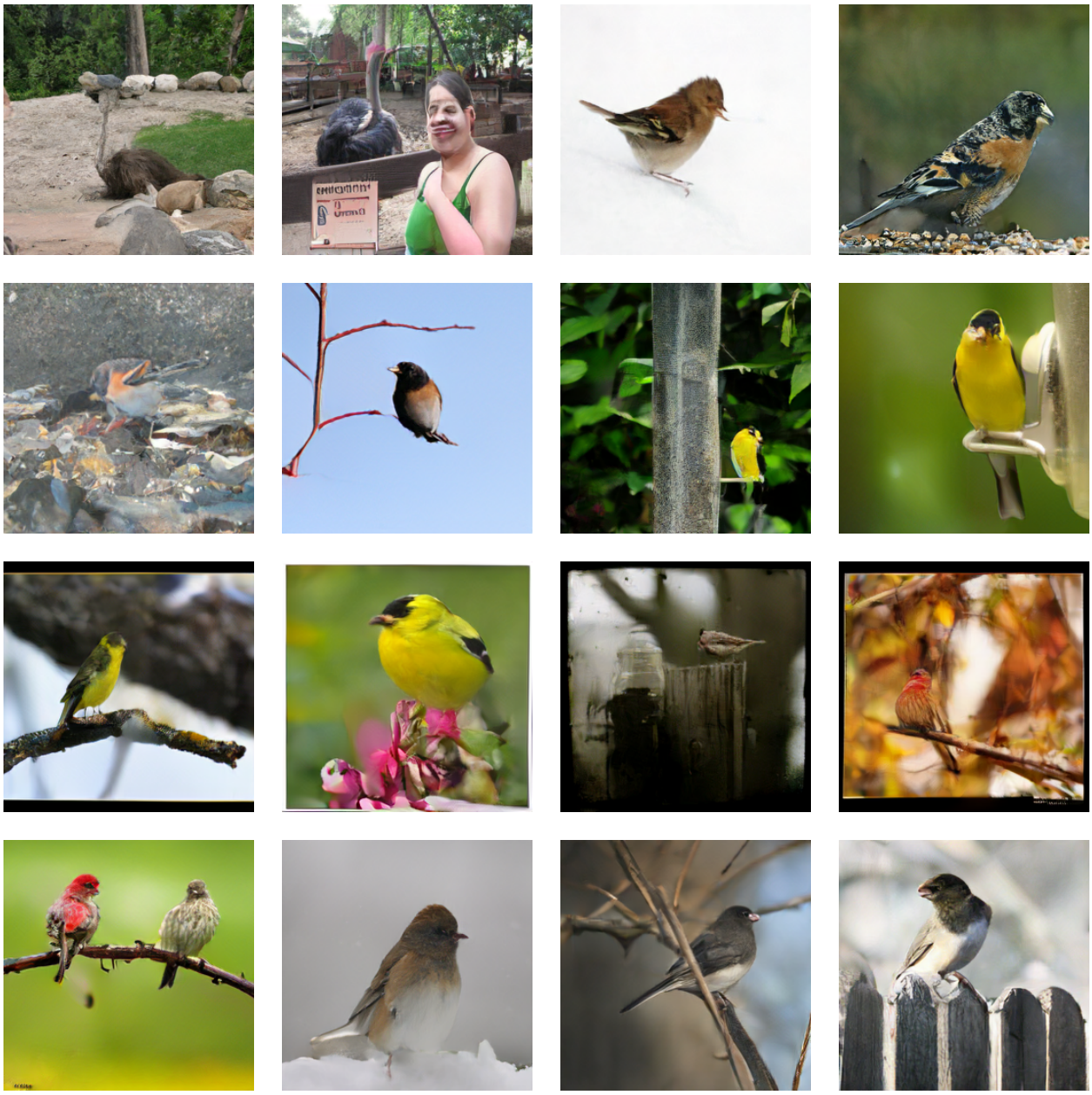


Figure A.3: Reconstructed images from the Imagenet Dataset.

Figure A.4: Reconstructed images from the Imagenet Dataset.

REFERENCES

[1] I. Kakogeorgiou, S. Gidaris, K. Karantzalos, and N. Komodakis, "Spot: Self-training with patch-order permutation for object-centric learning with autoregressive transformers," *arXiv preprint arXiv:2312.00648*, 2023.

[2] T. Li, H. Chang, S. Mishra, H. Zhang, D. Katabi, and D. Krishnan, "Mage: Masked generative encoder to unify representation learning and image synthesis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 2142–2152.

[3] F. Locatello, D. Weissenborn, T. Unterthiner, A. Mahendran, G. Heigold, J. Uszkoreit, A. Dosovitskiy, and T. Kipf, "Object-centric learning with slot attention," *Advances in neural information processing systems*, vol. 33, pp. 11 525–11 538, 2020.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, □. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[5] M. Tschannen, M. Kumar, A. Steiner, X. Zhai, N. Houlsby, and L. Beyer, "Image captioners are scalable vision learners too," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[6] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 16 000–16 009.

[7] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th International Conference on Machine Learning*, 2008, pp. 1096–1103.

[8] L. Karazija, I. Laina, and C. Rupprecht, "Clevrtex: A texture-rich benchmark for unsupervised multi-object segmentation," *arXiv preprint arXiv:2111.10265*, 2021.

[9] Y. Yang and B. Yang, "Promising or elusive? unsupervised object segmentation from real-world single images," *Advances in Neural Information Processing Systems*, vol. 35, pp. 4722–4735, 2022.

[10] B. Jia, Y. Liu, and S. Huang, "Improving object-centric learning with query optimization," *arXiv preprint arXiv:2210.08990*, 2022.

[11] G. Singh, F. Deng, and S. Ahn, "Illiterate dall-e learns to compose," *arXiv preprint arXiv:2110.11405*, 2021.

[12] Z. Wu, J. Hu, W. Lu, I. Gilitschenski, and A. Garg, "Slotdiffusion: Object-centric generative modeling with diffusion models," *Advances in Neural Information Processing Systems*, vol. 36, pp. 50 932–50 958, 2023.

[13] J. Jiang, F. Deng, G. Singh, and S. Ahn, "Object-centric slot diffusion," *arXiv preprint arXiv:2303.10834*, 2023.

[14] M. Seitzer, M. Horn, A. Zadaianchuk, D. Zietlow, T. Xiao, C.-J. Simon-Gabriel, T. He, Z. Zhang, B. Schölkopf, T. Brox *et al.*, "Bridging the gap to real-world object-centric learning," in *The Eleventh International Conference on Learning Representations*, 2022.

[15] H. Wang, Y. Tang, Y. Wang, J. Guo, Z.-H. Deng, and K. Han, "Masked image modeling with local multi-scale reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 2122–2131.

[16] H. Bao, L. Dong, and F. Wei, "Beit: Bert pre-training of image transformers," *arXiv preprint arXiv:2106.08254*, 2021.

[17] Z. Xie, Z. Zhang, Y. Cao, Y. Lin, J. Bao, Z. Yao, Q. Dai, and H. Hu, “Simmim: A simple framework for masked image modeling,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 9653–9663.

[18] Z. Huang, X. Jin, C. Lu, Q. Hou, M.-M. Cheng, D. Fu, X. Shen, and J. Feng, “Contrastive masked autoencoders are stronger vision learners,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[19] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks,” *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[20] A. Razavi, A. Van den Oord, and O. Vinyals, “Generating diverse high-fidelity images with vq-vae-2,” *Advances in neural information processing systems*, vol. 32, 2019.

[21] J. Yu, X. Li, J. Y. Koh, H. Zhang, R. Pang, J. Qin, A. Ku, Y. Xu, J. Baldridge, and Y. Wu, “Vector-quantized image modeling with improved vqgan,” *arXiv preprint arXiv:2110.04627*, 2021.

[22] J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[23] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[24] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization,” *arXiv preprint arXiv:1607.06450*, 2016.

[25] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *arXiv preprint arXiv:1409.0473*, 2014.

[26] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.

[27] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving language understanding by generative pre-training,” *arXiv preprint arXiv:1801.06146*, 2018.

[28] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language models are unsupervised multitask learners,” *OpenAI Blog*, 2019.

[29] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, “Language models are few-shot learners,” *arXiv preprint arXiv:2005.14165*, 2020.

[30] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[31] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, “Context encoders: Feature learning by inpainting,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2536–2544.

[32] M. Noroozi and P. Favaro, “Unsupervised learning of visual representations by solving jigsaw puzzles,” in *European Conference on Computer Vision*, 2016, pp. 69–84.

[33] R. Zhang, P. Isola, and A. A. Efros, “Colorful image colorization,” in *European Conference on Computer Vision*, 2016, pp. 649–666.

[34] S. Gidaris, P. Singh, and N. Komodakis, “Unsupervised representation learning by predicting image rotations,” *arXiv preprint arXiv:1803.07728*, 2018.

[35] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” in *International Conference on Machine Learning*, 2020, pp. 1597–1607.

[36] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Avila Pires, Z. D. Guo, M. G. Azar *et al.*, “Bootstrap your own latent: A new approach to self-supervised learning,” in *Advances in Neural Information Processing Systems*, 2020.

[37] C. Doersch, A. Gupta, and A. A. Efros, “Unsupervised visual representation learning by context prediction,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1422–1430.

[38] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum contrast for unsupervised visual representation learning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9729–9738.

[39] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, “Unsupervised learning of visual features by contrasting cluster assignments,” in *Advances in Neural Information Processing Systems*, 2020, pp. 9912–9924.

[40] X. Chen and K. He, “Exploring simple siamese representation learning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 15 750–15 758.

[41] J. MacQueen *et al.*, “Some methods for classification and analysis of multivariate observations,” in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.

[42] S. C. Johnson, “Hierarchical clustering schemes,” *Psychometrika*, vol. 32, no. 3, pp. 241–254, 1967.

[43] S. Wold, K. Esbensen, and P. Geladi, “Principal component analysis,” *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[44] L. Van der Maaten and G. Hinton, “Visualizing data using t-sne.” *Journal of machine learning research*, vol. 9, no. 11, 2008.

[45] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *arXiv preprint arXiv:1312.6114*, 2013.

[46] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, “Emerging properties in self-supervised vision transformers,” in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 9650–9660.

[47] P. Esser, R. Rombach, and B. Ommer, “Taming transformers for high-resolution image synthesis,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 12 873–12 883.

[48] H. Chang, H. Zhang, L. Jiang, C. Liu, and W. T. Freeman, “Maskgit: Masked generative image transformer,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 11 315–11 325.

[49] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 2014, pp. 740–755.