A Neuro-Symbolic Incremental Learner Model *for the* Visual Question Answering Task.

Penelope Johnston

September 2023

Thesis submitted for the degree of
Doctor of Philosophy

to the

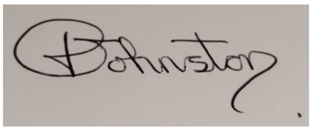
Faculty of Natural Science
Department of Computing Science and Mathematics
University of Stirling

“In search of an accurate World Model -
When a resilient system is perturbed,
it must adapt its constructs and
improve its alignment with its environment."

by Penny Johnston

Declaration

I confirm that I have authored this thesis and that I have not previously submitted the same work or findings for any other academic degree. I believe that the ideas not credited to others here are solely my own.



Signature

Date: 20th February 2024

Abstract

This research is motivated by the challenge of providing accurate and contextually relevant answers to natural language questions about visual scenes, particularly in support of individuals with visual impairments. Neural-Symbolic computing aims to unlock the potential of both the robust learning capabilities found in neural networks and the reasoning and interpretability of symbolic representation through their integration. This thesis introduces a Neuro-Symbolic Incremental Learner designed specifically for the Visual Question Answering Task. The system incrementally learns visual classes and symbolic facts to answer natural language questions about visual scenes. Using Deep Learning, a feature space is created from which visual classes are learnt as independent probability distributions. This allows for the easy addition of new classes even with limited data, mitigating the catastrophic forgetting typical of traditional neural networks. The incorporation of classification by category allows visual classes to not be limited to just objects but can also include other categories such as attributes. A knowledge graph stores facts about regions of interest, detailing; objects, attributes, actions, locations, and inter-relations, facilitating the incremental addition of knowledge. This allows facts to be stored explicitly and added incrementally. Leveraging a large language model, the system translates natural language questions into knowledge graph queries, ensuring a fluid visual question-answering experience.

Acknowledgements

I am grateful to my husband, whose maverick spirit and love for innovation have never ceased to amaze me. To my boys, who ground me daily, 'I hope my journey inspires in you a passion for continual learning, an open mind, and the belief that you can try anything'.

I'd like to thank Sam Nelson who delivered excellent 24 x 7 IT Support. My 2nd supervisor, Keiller Nogueira. My Viva panel, Jason Adair, Phil Bartie and Rachel Norman. For being there; Simona Hapca, Nora Tanner, Penne Jackson, Sandy Brownlee, Carron Shankland and Gavin Abernethy. My deepest appreciation goes to my 1st supervisor, Kevin Swingler for his consistency and dependability, endless patience, timely feedback and giving me the freedom to carve my own path. I have come to trust and value his insights.

Thank you Stirling Uni CSM for the opportunity.
Best Wishes for an awesome future!

Publications

This thesis has produced **Two** published peer-reviewed Journal papers.

1. Title: **NS-IL: Neuro-Symbolic Visual Question Answering using Incrementally Learnt, Independent Probabilistic Models for Small Sample Sizes.**
   Authors: Penny Johnston, Keiller Nogueira, Kevin Swingler.
   Journal: IEEE ACCESS, Volume: 11, Pages: 141406-141420
   Published: 7th Dec 2023
   Article DOI: 10.1109/Access.2023.3341007
   Electronic ISSN: 2169-3536
   Chapter 6 is directly based on this paper which can be considered as a distilled version of the thesis.

2. Title: **GMM-IL: Image Classification using Incrementally Learnt, Independent Probabilistic Models for Small Sample Sizes.**
   Authors: Penny Johnston, Keiller Nogueira, Kevin Swingler.
   Journal: IEEE ACCESS, Volume: 11, Pages: 25492-25501
   Published: 10th March 2023
   Article DOI: 10.1109/ACCESS.2023.3255795
   Electronic ISSN: 2169-3536
   Chapter 4 is directly based on this paper.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

AI Artificial Intelligence.

API Application Programming Interface.

BIC Bayesian Information Criterion.

CF Catastrophic Forgetting.

CLIP Contrastive Language-Image Pre-Training.

CNN Convolution Neural Nets.

DL Deep Learning.

DNN Deep Neural Network.

EM Expectation Maximisation.

GMM Gaussian Mixture Models.

HVS Human Visual System.

JSD Jensen- Shannon Divergence.

KG Knowledge Graph.

KLD Kullback-Leibler Divergence.

LLM Large Language Model.

LVM Language Vision Model.

ML Machine Learning.

MLE Maximum Likelihood Estimation.

NLQG Natural Language Query Generator.

NN Neural Nets.

NS-IL Neuro-Symbolic Incremental Learner.

ROI Region Of Interest.

VAE Variational Autoencoder.

VQA Visual Question Answering.

System Terminology

Unifying the Neural Net and Symbolic paradigms, also means their terminology must be aligned. The following terms are used throughout this thesis.

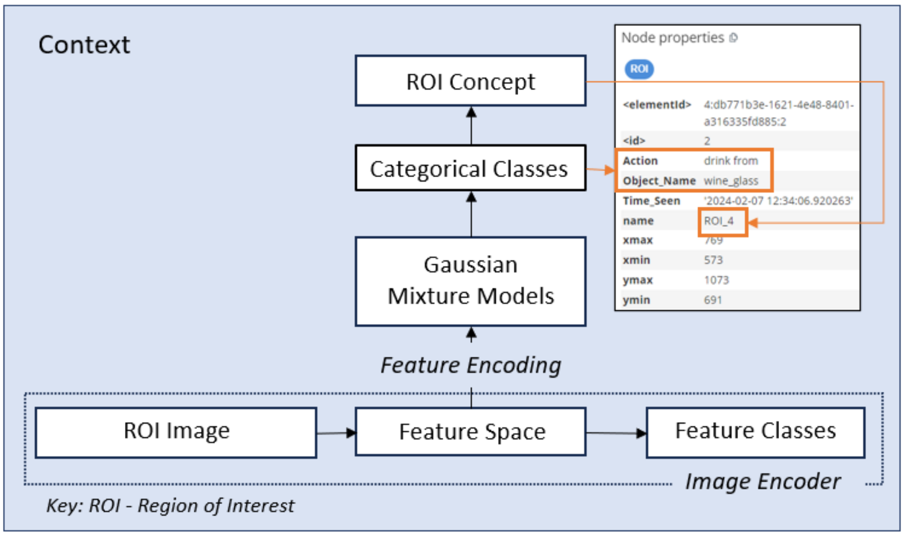


Figure 1: System Terminology.

- **ROI Image**: The image found within a bounding box.
- **Feature Space**: Generated by deep learning algorithms designed to produce a structured space suitable for encoding the features in an image by which it can be classified.
- **Feature Classes**: These are the classes used to train the image encoder model and are purely related to gaining a useful structure in the feature space.
- **Image Encoder**: The model that is trained on images that are/are not annotated by feature classes, producing a latent feature space.
- **Feature Encoding**: The feature encodings are created at inference time, through the selection of an embedding layer in the neural net.
- **Categorical Classes**: A Categorical class is a Class that belongs to a specific Category. A category can be thought of as an independent classifier. Categories are associated with ROIs and are used to group categorical classes.

- **ROI Concept**: Describes a 'Region of Interest' (ROI) which are the contents of a bounding box. This is described in terms of many categorical classes.
- **Context**: The environment in which data is collected, including images of scenes to be interpreted and human natural language questions.

1
Introduction

1.1 Introduction

'How do you put meaning to what you see in the visual world?' or, in other words, 'How do the Terms we use define the things we see ?' This thesis delivers a potential solution, called the Neuro-Symbolic Incremental Learner Model. The approach taken is to build a safe AI system, rather than making an AI system and then having to make that safe after the fact.

For most people, vision is a natural ability, which enables an effortless interpretation of our surroundings. However, for individuals with visual impairments, assistance is often required to bridge the gap between perception and understanding. In such instances, an Assistant, when queried, provides insights into the environment around them. When this Assistant takes the form of a computer, we then need to break the task down and explicitly code the actions. Upon receiving a question, the Digital Assistant must first capture and analyse an image of the relevant scene before processing the inquiry and delivering an answer. This process is known as the Visual Question Answering (VQA) task. The core contribution of this thesis is the development of a Neuro-Symbolic Incremental Learner Model, which leverages a Gaussian Mixture Model Incremental Learner. During this research, both models were developed, each with an accompanying published paper. This innovative approach changes the way Terms are linked to the visual information that gives them meaning. This structure enables information to flow, enabling neural nets and symbolic approaches to be utilised in a manner that they were both designed to excel in.

Figure 1.1 illustrates the human-to-computer interface. Let's take two scenarios:

1. In Scenario One, purely taking into account the blue box entitled *human* which represents a human in their environment. The person processes incoming visual information through their senses, forming beliefs and perceptions that guide their outgoing actions.

2. In Scenario Two, we take into account both the human and a Digital Assistant. The red cross indicates insufficient visual information, indicating a visually impaired human. In this case, the human requires help. The human poses a question to the Digital Assistant, shown by the action of the human sending the question, 'What's in the image?', which the Digital Assistant receives. The Digital Assistant then obtains an image of a specific *Region Of Interest* and following inference, it generates an ROI node in the Knowledge Graph. Using this graph and explicit information the Digital Assistant has previously learnt, the computer generates a response, to send to the human as an Answer, in this case *Apple*. The human's beliefs are amended and perceptions are updated about the current environment based on this received information. The design assumes a shared semantic ontology is in use, specific to this current environment.

The Visual Question Answering (VQA) Task was formed to encapsulate the skills needed to enable a computer to respond to a question about an image with an applicable Answer. This question-and-answer mechanism is an ideal interface to connect humans and computers effectively. Figure 1.2 shows a high-level overview of the flow of data that has been developed during this research, which enhances the learning abilities within this task. The structure also enables the Human-In-The-Loop to continually learn new Categories and Terms within an Ontology, allowing the system to dynamically evolve. This research proposes a mechanism to extract semantic and contextual information from images, enabling natural language inquiries.

The Neuro-Symbolic Incremental Learner (NS-IL) Model is comprised of three components; neural visual techniques within a classification system, a structured

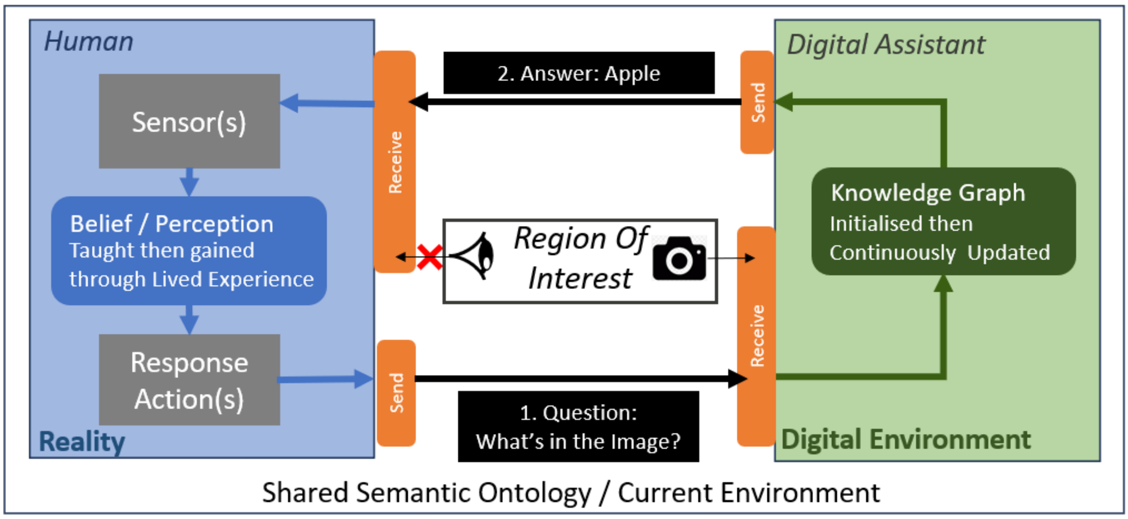


Figure 1.1: Visually impaired human receiving help from a Digital Assistant.

knowledge graph, and a large language model for smooth human-driven-natural-language-graph interactions. Further details for these main components are:

1. An extensible *Image Classification System* comprised of two main components.
    - A pre-trained Deep Neural Network (DNN) that maps pixel *Images* onto a visual feature space, this model is an image encoder that carries out *Feature Extraction* at inference time.
    - *Categorical Gaussian Mixture Models (GMM)s*, which are probability models to calculate an individual Terms likelihood of being *seen*. These models can be learned individually from small data sets and added to the system as needed.
2. A *Knowledge Graph (KG)* that stores *Region Of Interests* grounding information in the nodes of the graph, including temporal and spatial information. Other nodes hold categorical classes where explicit information is held, this information is inherited by the generated ROI nodes at inference time. Facts about relationships between categorical classes are stored in the KG edges and are propagated between the ROI nodes. Each GMM is associated with a categorical node in the KG. Questions are answered by traversing the graph to access the relevant ROI nodes and associated data.
3. A *Large Language Model (LLM)* is used to translate a human's question into code to interrogate the KG. The results from the query together with the

original question are then translated back by the LLM into an understandable *Answer* for a human.

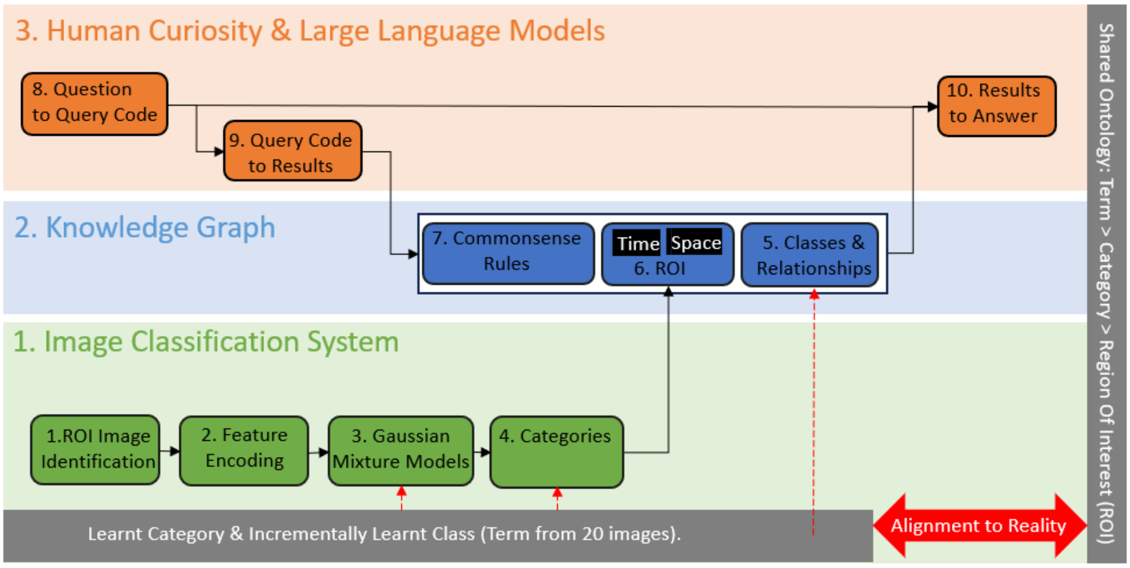


Figure 1.2: The Neuro-Symbolic Incremental Learner Data Flow for the Visual Question Answering Task.

1.2 HYPOTHESIS

The central hypothesis in this thesis is as follows:

The use of independent Gaussian Mixture Models as a mechanism to link Neural Nets to a Symbolic memory, will lead to a more flexible Visual Question Answering Task.

This hypothesis is challenged by addressing the following research questions.

1. In traditional Neural Networks what model modifications are necessary to support independent visual class incremental learning?

2. How can Neural Networks effectively ground symbolic systems to enhance downstream understanding and reasoning.

1.3 Contributions

The presented complex Neuro-Symbolic System makes the following contributions to the field:

1. Development of an enhanced Neural Architecture: Introduction of a neural network architecture utilising GMMs for incremental visual class learning, overcoming the constraints of traditional architectures, such as catastrophic forgetting and the requirement for a large training dataset.

2. Neuro-Symbolic Integration Insights: Insights into the bottom-up encoding of information between neural nets, GMMs and symbolic reasoning, providing insights into fusion techniques.

3. Development of a Neuro-Symbolic Architecture: Improving VQA's contextual understanding through the use of *ROIs* comprised of multiple visual categories such as attributes and objects, together with the ability to apply commonsense rules over them.

1.4 Structure

The thesis is organised as follows:

- In Chapter 1, an Introduction is given, together with the **Hypothesis**, associated Research Questions, identified **Contributions** and how the thesis has been structured.

- The **Literature Review** contextualises and informs the thesis in Chapter 2. It starts with an overview of what is available for the visually impaired. It then covers applicable AI technologies such as; the frameworks used, current tasks and models, latent variable models, available types of computing memory, NLP, and LLM models. The *Visual Question and Answering* Task is looked at, together with current implementations. It concludes with what improvements have been identified to improve the current State-Of-Art for the VQA Task.

- An overview of **The Traditional VQA Model** is given in Chapter 3.

- A description of how the traditional model's limitations are overcome is described in Chapter 4. This is in the form of the Gaussian Mixture Model

Incremental Learner. This covers the research which resulted in the published **GMM-IL Model**. The main contribution of which was its ability to learn and add classes when needed using small data sizes.

- Chapter 5 forms the **Planning a Neuro-Symbolic Incremental Learner** section. Conclusions are drawn from the work carried out during the GMM-IL research, which are reflected upon and inform the research decisions for the creation of the NS-IL model.
- The **Neuro-Symbolic Incremental Learner** is described in Chapter 6. This covers the research carried out that resulted in the published **NS-IL Model**. In essence, a VQA system created from compositional machine learning, integrating neural nets, GMMs, a KG & a LLM.
- The overall **Summary & Conclusions** are formalised in Chapter 7.
- In Chapter 8, an evolved system is envisioned, detailing a **Human Centric Passive Assistant** to aid humans and carry out Visual Dialogue.

2

Literature Review

This literature review aims to offer a comprehensive understanding of this thesis's research field by exploring critical topics. It begins with an examination of applications designed for the visually impaired, followed by an analysis of the AI technologies that power these solutions. We discuss various frameworks available to developers, with a focus on those tailored for computer vision. The review delves into recognised tasks and models in the field, including Latent Variable Models, strategies for storing and querying information, Natural Language Processing, Referring Expressions, Large Language Models and Natural Language Question & Answers. A section on the notable VQA models, what benchmark datasets exist, and a detailed exploration of the NS-VQA model that Joshua B. Tenenbaum from MIT was involved in. Concluding with an overview of how the VQA Task could be enhanced.

2.1 The Visually Impaired

The Lancet Global Health document of 2017 [4] found, 'Globally, of the 7.33 billion people alive in 2015, an estimated 36.0 million (80% uncertainty interval [UI] 12.9-65.4) were blind.' This accounts for roughly 0.5% of the global population, and it highlights the potential for non-invasive assistance to improve the lives of these individuals. In the United Kingdom, the Royal National Institute of Blind People (RNIB) states, 'There are over 2 million people in the UK living with sight loss' and anticipates this number will increase due to an aging population. This number includes those who are registered as blind or partially sighted, as well as those whose vision is better than the levels that qualify for registration. The UK government's industrial strategy paper titled, 'The Grand Challenges', outlines key policy priorities for the future, with the first challenge being an 'Ageing society', followed by 'Artificial Intelligence (AI)'. The strategy emphasis is on the importance of enabling older citizens to lead independent and fulfilling lives while contributing to society.

2.2 Applications for the Visually Impaired

AI-powered assistive technologies for vision significantly enhance accessibility and independence for individuals with visual impairments. These technologies utilise advanced algorithms to interpret visual data, transforming images and text into audible information. Applications such as text-to-speech readers can scan printed material and read it aloud, while object recognition systems identify and describe objects in the environment, facilitating navigation and interaction for the visually impaired. Facial recognition software helps users identify acquaintances in social settings, enhancing their social interaction and independence. Navigation apps specifically designed for the visually impaired use AI to provide audio directions, warn of obstacles, and offer route suggestions in real-time. The factual benefits of these vision-focused AI technologies are profound, offering users not only assistance with daily activities but also a greater sense of autonomy, safety, and inclusion in society.

BeMyEyes is a low tech app created in 2015 that connects visually impaired people with sighted volunteers through live video calls. Users can ask for help with anything that requires sight, like reading labels or finding items, and a volunteer answers the call to assist them by looking through the user's phone camera. This product has evolved through a Microsoft partnership to incorporate extensions such as *specialised help*, which helps users to connect to the *right* people such as colleagues at work and specific customer support personnel. The tagline, 'Seeing the world together', reflects how this app enables access to another pair of human eyes. The fine-tuning of which human to pair with, based on what they know, highlights that vision is not enough without the ability to interpret what is seen. BeMyEyes main limitation is its dependence on an available human with the right domain knowledge.

Google Lookout is specifically engineered to process visual data autonomously, reducing the need for human assistance and enabling users to carry out daily tasks more independently. It is designed to help people identify information about their surroundings. It uses the phone's camera and AI to recognize objects, text, and people and then provides auditory feedback to the user. For example, it can read out text from labels, documents, or signs, identify products by their labels, and describe the environment, such as noting the presence of a chair or a door. It operates in 6 activity modes; Text, Document, Explore, Currency, Food Labels and the newly added Images which is in beta and allows question-and-answer functionality on an uploaded image. The latest feature allows users to simply point their camera, snap a photo, and ask a question about the captured scene. They then receive detailed information about the contents of the photo. The visually impaired person is determining the region of interest, by adjusting the zoom to either close in or pull back enhancing the level of detail provided. This system is limited by the human ability to point the camera in the right direction and is dependent on the system's knowledge and ability to reason. They ask a question and receive an answer much like the game, *Who am I?* where the player tries to narrow down the possible answers to the true answer.

OrCam MyEye is a camera that can be attached to glasses, imitating how humans capture visual information, through the natural movement of the head. The camera is voice activated in a 'Hey OrCam' type way and reads aloud text from books, screens, products, labels, street signs, identifies faces, products and currency.

Seeing AI is a Microsoft-developed app that assists users by supplying audio description about the environment they are in, by utilising the camera on their smartphone. The app uses AI to recognise and narrate the world around the user. It has several channels; Short Text, Documents, Products, Scenes, People, Currency, Colours, Handwriting, Light and Images from other apps. It uses audio to help guide the user to capture a useful region of interest, such as a barcode or page of text. An interesting human-to-computer interface is the functionality for a user to navigate the scene using their finger and then explore it based on descriptive feedback. The level of detail given can be adjusted and the user has question-and-answer functionality on the content in the documents.

While all these apps aim to enhance independence for visually impaired people, the choice among them depends on the user's preference for interaction (human vs. automated), the type of assistance needed (environmental awareness, text reading or social interaction), and the convenience of use (wearable vs. smartphone app). Except for BeMyEyes, the ability of the application to deliver intelligent answers is dependent on the accuracy of the underlying algorithms.

2.3 AI Technologies

Artificial Intelligence (AI) has experienced several phases of evolution. Initially, the focus of AI was predominantly on Machine Learning (ML). At this stage, the main concern was about the *how*, the *methods* and *processes* that enabled machines to learn from data. This focus became stronger as the field developed various learning algorithms, forming the basis for machines to make predictions or decisions without them needing to explicitly program for a task.

As the field progressed, a more specialised branch known as Deep Learning (DL) emerged. This wasn't just an extension of traditional machine learning, it marked a significant shift in focus. Instead of the broader mechanisms of learning, deep learning homed in on the intricacies of features within data and the architecture of the neural networks. These neural networks, especially deep ones, allowed for the *automatic extraction of intricate patterns and representations* from vast amounts of data, pushing the boundaries of what machines could recognise and comprehend.

Today, the landscape of AI is shifting once more. The trend is now on the comprehensive functionalities that foundational models can offer. Instead of solely focusing on the mechanics of learning or the intricacies of data features, the current trend is to create adaptable models that can be fine-tuned for a broad array of tasks. These foundational models act as *foundations* for developing and improving various specific AI applications.

2.3.1 *AI General Frameworks*

There are several frameworks and libraries that provide developers with powerful tools to quickly build and deploy machine learning applications across various fields like text analysis, image recognition, and speech processing. They make cutting-edge

AI accessible, allowing developers to use pre-trained models which saves time and effort that would otherwise be spent on developing and training models from the ground up. This is a list of some of them that are currently supported and innovated:

- Hugging Face: Is a comprehensive library for natural language processing (NLP), it offers a wide range of pre-trained models for tasks like text classification, translation, and question answering. Hugging Face want to become the place with the largest collection of models and datasets to democratise AI for all, through open-source (OS) code and technologies.

- Google Brain: The Tensorflow Model Garden provides a collection of models and algorithms implemented in TensorFlow for a variety of machine learning tasks, including vision, image classification, Object Detection and Segmentation, Video Classification, and Natural Language Processing.

- Facebook Artificial Intelligence Research (FAIR): Their TorchHub is a repository within PyTorch for sharing pre-trained models contributed by the community, covering image classification, object detection, and more. Governed by Pytorch Foundation which is a subsidiary of the Linus Foundation.

- Intel: The OpenVINO is a cross-platform deep learning toolkit developed by Intel. The name stands for 'Open Visual Inference and Neural Network Optimization'. Part of the OpenVINO toolkit, includes pre-trained models optimized for performance on Intel hardware for tasks like face recognition and object detection.

- Salesforce: The LAVIS repository is a one-stop Library for Language-Vision Intelligence.

2.3.2 *Computer Vision Frameworks*

Libraries specifically for Computer Vision are:

- OpenCV (Open Source Computer Vision Library): While primarily a library for computer vision, OpenCV also supports the use of deep learning models, particularly for image and video analysis tasks. It includes functionalities for processing and analysing images and videos, and it can be used to implement and deploy computer vision and deep learning applications.

- Facebook Artificial Intelligence Research (FAIR): Their Detectron2 is an advanced library that they say delivers ‘object detection, segmentation algorithms, panoptic segmentation, Densepose, Cascade R-CNN, rotated bounding boxes, PointRend, DeepLab, ViTDet, MViTv2’.

The DL Model landscape is complex due to the programming languages used, the hardware requirements, ownership of hardware and software rights, and the quest for market dominance. Comparing these models requires the definition of specific tasks and benchmark datasets on which to quantify their accuracy.

2.3.3 *Computer Vision Tasks and Models*

Computer Vision encompasses a broad range of tasks aimed at enabling computers to interpret and understand visual information from the world. The main recognised tasks in Computer Vision include:

- Object Detection: Identifying objects within an image and drawing bounding boxes around them.
- Image Classification: Assigning a label to an entire image based on its content.
- Semantic Segmentation: Classifying each pixel of an image into a predefined category.
- Instance Segmentation: Similar to semantic segmentation, but it also differentiates between instances of the same class.
- Edge Detection: Identifying the edges in an image to understand object boundaries.
- Face Recognition and Detection: Detecting and recognising human faces within images.
- Optical Character Recognition (OCR): Converting text in images into machine-encoded text.
- Pose Estimation: Determining the position and orientation of objects or human figures.
- Depth Estimation: Estimating the distance between objects in an image and the camera.

- Image Generation: Creating new images or modifying existing ones (e.g., through Generative Adversarial Networks or GANs).
- Video Analysis: Understanding, interpreting, and manipulating video content.
- Action Recognition: Identifying actions or activities in videos.
- Object Tracking: Tracking the movement of objects across frames in videos.
- Scene Reconstruction: Creating a 3D model of a scene from images or video.
- Anomaly Detection: Identifying unusual patterns or outliers in visual data that do not conform to expected behaviour.

From these basic building blocks more complex tasks are created. These tasks often require understanding and processing visual information at a higher level, integrating multiple techniques and methodologies. Here are some notable composite computer vision tasks:

- Visual Question Answering (VQA): Given an image and a question in natural language about the image, the task is to provide an accurate answer. This combines elements of image understanding with natural language comprehension and generation.
- Visual Relationship Detection: Involves identifying relationships between objects in an image, such as 'person riding bicycle' or 'cat under table'. This task combines object detection with the understanding of the interactions or spatial relationships between detected objects.
- Scene Understanding: Encompasses a comprehensive analysis of an entire scene, including object detection, segmentation, and recognition, as well as understanding the spatial layout and the interactions between elements within the scene.
- Image Captioning: Requires both the recognition of objects, attributes, and actions in images and the ability to describe these visually perceived elements in natural language. This task bridges the gap between computer vision and natural language processing (NLP).

- Panoptic Segmentation: Merges semantic segmentation (where the aim is to classify each pixel into a category) with instance segmentation (which differentiates between individual objects of the same category). This task provides a comprehensive understanding of an image by labeling every pixel not just by its class, but also distinguishing between different instances of the same class.

- 3D Object Reconstruction: Involves creating 3D models of objects from one or more images. This task combines object detection, depth estimation, and sometimes motion analysis to reconstruct the shape and appearance of objects in three dimensions.

- Person Re-identification: The aim is to recognise individuals across different scenes or camera angles, often combining object detection, facial recognition, and feature matching techniques.

- Action Detection in Videos: Goes beyond action recognition by also localizing the action temporally and spatially within a video. This requires analysing sequences of frames to understand when an action starts and ends and where it occurs in the scene.

- Multi-object Tracking: Tracks multiple objects as they move across frames in a video. This task combines object detection with motion analysis to maintain the identity of each object over time.

2.3.4 *Latent Variable Models*

Latent Variable Models (LVMs) are a class of statistical models used to describe relationships in data by introducing latent (hidden) variables. These hidden variables capture the underlying, unobserved processes that influence the observed data. LVMs are particularly useful for dealing with complexities in data, such as when the data has underlying patterns or structures that are not directly observable. Finding the optimum combination of the following algorithms, in transforming an image into a new orthogonal coordinate system, is the balance and effectiveness that AI aspires to achieve.

- Optimally describes the variance in the training dataset.

- Will generalise to new unseen and evolving tasks for out-of-distribution data. What we mean by this is a desire to generalise from training to application data and also from sample to population data.

. At its simplest these are the tools used to translate what the image shows into a Term or Terms. Key characteristics and uses of Latent Variable Models include:

- Dimensionality Reduction: In AI, dimensionality reduction techniques are invaluable for simplifying complex data, enhancing computational efficiency, and improving model performance. These methods, ranging from linear approaches like Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA), which are adept at handling datasets with straightforward linear relationships, to non-linear strategies such as; t-distributed Stochastic Neighbour Embedding (t-SNE), Uniform Manifold Approximation and Projection (UMAP), and Autoencoders, which excel in dealing with intricate data structures. Independent Component Analysis (ICA) excels in distilling multivariate signals into independent components, making it ideal for applications requiring the separation of mixed signals. Factor Analysis dives into datasets laden with variables to unearth latent dimensions, thereby simplifying data interpretation in fields like psychology and market research. Isomaps preserve the global geometry of data, adeptly handling non-linear relationships by maintaining the intrinsic geometric structure, which is crucial for accurate modeling in tasks like facial recognition. Multidimensional Scaling (MDS) visualises the similarities or dissimilarities among data points as spatial distances, facilitating pattern recognition and outlier detection. They not only reduce the computational load by minimising the number of features that models need to process but also help in uncovering hidden patterns within the data, making it easier to analyse and visualise. By distilling the essence of data while discarding redundant or irrelevant information.

- Feature Extraction: By modeling hidden factors, LVMs can extract meaningful features from the data that contribute to its generation or structure. This is particularly useful in tasks like speech recognition or image processing.

- Handling Missing Data: LVMs can infer missing values in a dataset by estimating the latent variables that would likely result in the observed data, providing a principled approach to impute missing data.

- Clustering and Classification: Some LVMs, like Gaussian Mixture Models (GMMs), can discover clusters within data, assigning data points to groups based on the underlying latent variables.

- Generative Modeling: LVMs are often used to build generative models, which can generate new data instances similar to the observed data. Examples include Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs), though GANs are a more complex case involving game theory alongside latent variables.

- Discovering Hidden Structures: In fields like natural language processing, LVMs can uncover latent structures in text, such as topics in collections of documents using models like Latent Dirichlet Allocation (LDA).

The approach to modeling with LVMs involves specifying a mathematical relationship between the observed variables, the latent variables, and the parameters of the model. Inference in LVMs typically focuses on learning the model parameters and estimating the hidden variables given observed data, often employing techniques like Expectation-Maximisation (EM), Markov Chain Monte Carlo (MCMC) simulations, or variational inference. Here are some of the techniques, although some have already been mentioned, here they are defined from a mathematical point of view:

- Gaussian Mixture Models (GMMs): A probabilistic model that assumes all the data points are generated from a mixture of several Gaussian distributions with unknown parameters.

- Principal Component Analysis (PCA): A technique for dimensionality reduction that identifies the directions (principal components) that maximise the variance in high-dimensional data.

- Latent Dirichlet Allocation (LDA): A generative statistical model that describes collections of discrete data such as text corpora by assuming a set of topics that generate words in documents.

- Hidden Markov Models (HMMs): Used for temporal data, HMMs model the observed data as a sequence of outputs generated by transitions between hidden states according to certain probabilities.

- Variational Autoencoders (VAEs): A generative model that learns to encode input data into a condensed latent space and then reconstruct the input from this space, useful for tasks like image generation and denoising.

2.3.5 *Computing Memory Types*

VQA systems analyse images to identify objects, attributes, and their relationships, then they store this information in a structured format. This allows the VQA system to reference the stored scene information when processing questions, ensuring that it can accurately interpret the visual content and provide coherent answers based on the combination of visual analysis and symbolic reasoning. Databases enable this structured data to be efficiently retrieved and manipulated for answering questions about visual content. Databases support the indexing of images, annotations, and metadata, facilitating quick access to relevant information, an example of a graph database is *Neo4J*. Databases also allow for the integration of additional knowledge bases such as *ConceptNet* that can enrich the system's understanding, enhancing its ability to interpret images and generate accurate, contextually relevant answers.

Neo4J's graph structure is designed around nodes, relationships, and properties. Nodes represent entities or objects, while relationships provide the connections between these nodes, with direction and type that describe how nodes are related. Properties are key-value pairs attached to both nodes and relationships, allowing for the storage of additional information. This structure is highly flexible and efficient for modeling complex networks of data, making it ideal for applications that require the representation of intricate relationships, such as the interconnected data within the VQA task. This graph structure naturally represents the interconnectedness of scene elements, facilitating quick traversal and query execution. This allows for dynamic reasoning over visual content and supports the integration of diverse data sources for enriched context.

ConceptNet is a semantic network that provides an extensive knowledge base of common sense, enabling systems to understand and interpret human language in a more nuanced manner. By integrating ConceptNet into a VQA system, these systems gain access to a rich set of relationships and properties across diverse concepts, enhancing their ability to process and answer questions with enriched context. This integration allows VQA systems to leverage background knowledge and inferential reasoning, improving their performance on tasks requiring understanding beyond the visual information present in the image alone.

2.3.6 *Natural Language Processing (NLP)*

Natural Language Processing (NLP) is a field at the intersection of computer science, artificial intelligence, and linguistics. It focuses on the interaction between computers and humans through natural language. The goal is to enable computers to understand, interpret, and generate human languages in a valuable way. NLP encompasses a range of technologies for processing and analysing text and speech, such as language translation, sentiment analysis, chatbots, and voice recognition systems. It plays a crucial role in facilitating seamless communication between humans and machines, driving advancements in search engines, digital assistants, and automated customer service. Referring expressions in linguistics and natural language processing (NLP) are phrases or words used to identify or describe entities within a given context. They enable speakers or writers to refer to objects, people, locations, or abstract concepts in communication.

In a VQA system, referring expressions are used to pinpoint and describe specific parts of an image in response to a question. The system analyses the image and the query, identifying relevant entities and their attributes. It then applies an understanding of referring expressions to generate accurate answers that correctly identify and describe these entities within the visual context. This involves interpreting the semantics of the query, mapping it to visual elements, and employing referring expressions to articulate the answer, enhancing the system's ability to provide precise and contextually appropriate responses.

2.3.7 *Large Language Model (LLM)*

Large Language Models (LLMs) in the context of natural language processing are advanced AI systems trained on vast amounts of text data. These models, such as Generative Pre-trained Transformer GPT[5] and Bidirectional Encoder Representations from Transformers (BERT) [6], have a deep understanding of language patterns, grammar, and context. They can generate coherent text, answer questions, translate languages, and more, mimicking human-like language abilities. LLMs have transformed NLP applications, enabling more accurate and contextually aware machine understanding and generation of natural language. Over the last 2 years, IBM has changed its focus from rule-based systems to LLMs as evidenced throughout their Summer school. These methods required significant domain expertise and were often task-specific using statistical methods like n-grams, analysis of word occurrence frequencies for predicting text sequences. These previous systems lacked the complexity to grasp language nuances fully, lacking the generalisability of the current LLMs.

The Large Language Models, like T5 [7], BLOOM [8], and GPT-3 [5], have been advancing rapidly and could be an ideal module to deliver the query answering functionality required by the VQA task. Among them, ChatGPT, based on Instruct-GPT [9], stands out for its ability to maintain conversation contexts effectively. Visual-ChatGPT, introduced by Wu et al. [10], combines ChatGPT with Visual Foundational Models, enabling ChatGPT to handle complex visual tasks.

In communication, we often use brief language, expecting others to infer the unsaid. This process involves guessing the extent and context of terms we use, which isn't immediately obvious to the listener. They must interpret the unspoken parts of our message. Similarly, in developing larger language models, machine learning and data-driven approaches aim to uncover the underlying meaning not directly visible in the data, striving to decode the implicit content and fill in the gaps of understanding.

2.3.8 *Natural Language Question and Answers*

Integrating Large Language Models (LLMs) with Knowledge Graphs (KGs) enables natural language question answering. This combination addresses two main areas; logical reasoning with knowledge graphs and using prompts for reasoning in large language models.

Adapted from Pan et al [1] Figure 2.1 offers a view of the research landscape concerning the integration of LLMs and KGs. It presents different ways to combine Knowledge Graphs (KGs) with Large Language Models (LLMs), including KG-enhanced LLMs, KG-augmented LLMs, and LLMs working in synergy with KGs. Our current research aligns with the areas highlighted in blue, though there is potential to expand into the other sectors.

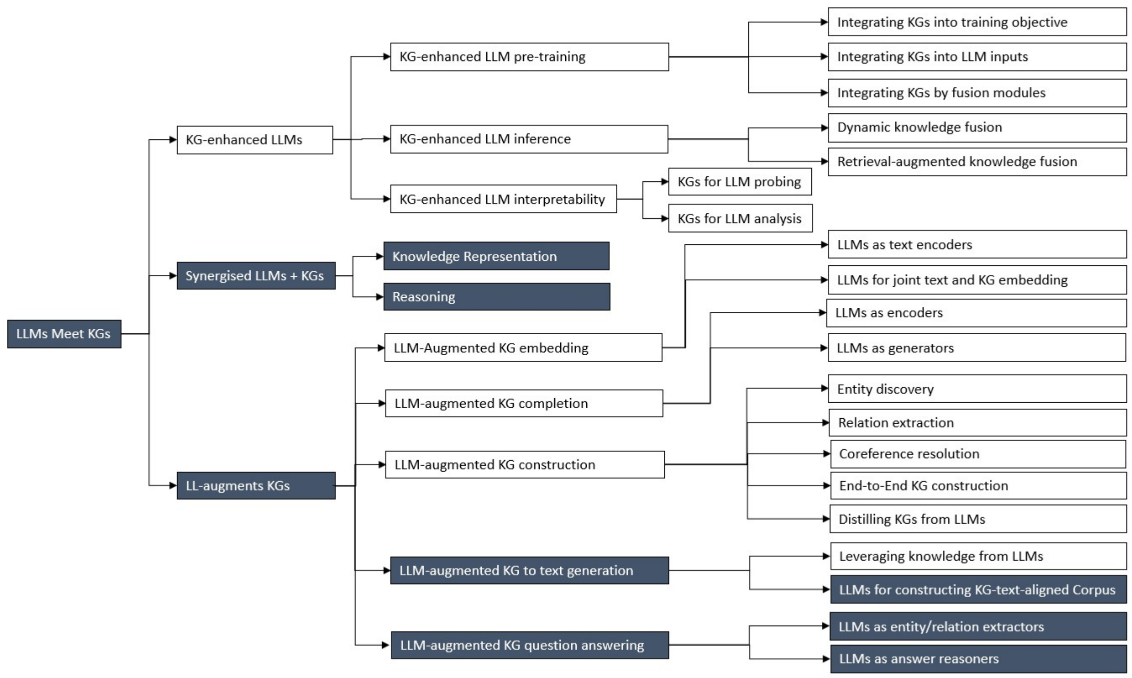


Figure 2.1: Research categories in LLM-KG integration. NS-VQA requirements align with blue areas. Image Credit [1].

LLMs are reshaping the landscape of knowledge representation and there's a clear trend towards blending explicit knowledge like KGs, with parametric knowledge from LLMs. Large language models (LLMs) are considered one of the most remarkable models in AI research, having an ability to harvest the internet artifact as an embodiment of societal knowledge. However, the issue of hallucinations, sometimes referred to as confabulations remains a significant hurdle before it can be trusted and there is no current clear path towards resolution.

There is an ongoing need for a framework that incorporates encoded meaning. A comprehensive understanding of the combined capabilities of Knowledge Graphs and Large Language Models is discussed in-depth by Pan et al. [1]. There is a trend towards Knowledge Computing, which expands reasoning across various knowledge formats. Now, KGs, a standard for explicit knowledge, are being merged with Transformer-based LLMs such as BERT [11], RoBERTa [12], GPT series [5], and LLaMA [13]. Some research augments KGs with LLM for tasks like knowledge extraction, while others leverage KGs to enhance LLMs for training or knowledge augmentation.

Knowledge graphs (KGs) offer an organised way to represent information, making it easier for reasoning and inference. Some critics such as Bender et al [14] argue that the knowledge in LLMs relies more on statistical patterns than genuine comprehension. Advocates, such as those behind ChatGPT, underscore its ability in generalising from vast datasets, demonstrating extensive information capture and impressive language comprehension. However, Zhang et al [15] discuss how LLMs might produce plausible but incorrect responses, due to the absence of explicit knowledge representation. Li et al [16] conclude that LLM become, 'incompetent directional entailment learners, in contrast to entailment graphs'. Building KGs can be resource-intensive, and while training LLMs is costly, they offer immediate utility for various applications.

There's an emerging trend of using LLMs to distil knowledge for end users. Some research focuses on harmonising KGs with prompts to amplify LLM effectiveness and trustworthiness. Studies tap into KGs to refine prompt creation, achieving enhanced volume, quality, and diversity over traditional methods. Knowledge graphs are used in both single and multi-turn prompts, offering efficient traversal paths with minimal authoring effort, and promoting meaningful learning patterns. LLMs have also excelled in multi-step reasoning, various methods were used to achieve this. Wei et al [17] in 'Chain-of-Thought', allowed the model to breakdown problems into intermediate steps, similar to human thought processes, which not only improved their problem-solving capabilities but also provided transparency into how the model arrived at it's conclusions. Chowdhery et al PaLM model [18] used a combination of model scaling and innovative prompting techniques, which suggested the emergence of new capabilities as models reach sufficient size. and Ye et al propose InstructGLM (Instruction-finetuned Graph Language Model) [19] which uses natural language to describe the multi-scale geometric structure of graphs and then fine-tuning a LLM accordingly. It adds flexibility, scalability, and compatibility to multi-step reasoning in graph machine learning, allowing for efficient handling of complex graph tasks without the need for specialised graph attention mechanisms or token representations.

2.4 The Visual Question Answering Task

The Visual Question Answering (VQA) task is chosen as the task that this research will focus on. Its selection was based on its functionality to help the visually impaired by combining seeing and understanding images with answering questions in natural language. This technology acts as an eyesight substitute, allowing visually impaired users to ask questions and receive answers about their surroundings, making daily activities more accessible. VQA offers real-time help, enabling users to understand their environment instantly through smartphones or wearable tech. It's versatile, answering a wide range of questions from simple object identification to complex scene descriptions. The technology can be customised for individual needs, improving its relevance and effectiveness in personal scenarios. Other terms that are in use with similar functionality are Vision Language Models (VLM) and when these models use frozen image encoders as priors (pre learnt knowledge) they're called Vision Language Pretraining models (VLP).

2.4.1 *Significance of Deep Learning*

Deep learning has significantly impacted Visual Question Answering (VQA) by introducing end-to-end neural networks that utilise self-supervised Vision and Language Pre-training (VLP). These networks excel at learning from vast image-text data but often falter in logical reasoning task, as discussed in Malinowski et al [47]. Yu et al [48], Chen et al [49], Zhang et al [50] and Zhou et al [51] discuss methods using visual attention, this is a key component in many VQA tasks. Research suggests that when a model learns to pay close attention to both images and text simultaneously, it can enhance how it represents images and questions. This, in turn, leads to better predictions when answering questions because the model can focus on crucial parts of the visuals and important words in the text to improve its understanding. There's a growing trend towards using both image and textual data (rather than single labels) for training, with models like Contrastive Language-Image Pre-Training (CLIP) suggesting the potential of open-set challenges through linguistic domains.

2.4.2 *Notable Models*

GLIP [20] redefines object detection as a phrase grounding problem. Their method merges phrase grounding and object detection. In this perspective, object detection becomes a specific type of phrase grounding, while phrase grounding is a context-driven object detection, offering a new perspective on both tasks.

DeepMind's *Flamingo* belongs to this class of models and is designed to process visual data interleaved with text using *Gated cross attention dense layers* where the keys and values come from the vision input. It generates free-form text as its output. *BLIP2* [21], which is short for *Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models* is also an example of a VLM and currently the top variant called *BLIP-2 ViT-G Flan T5 XXL* (Zero-shot) scores 65.2% on the VQA v2 test-dev dataset as recorded by the web site *PapersWithCode*. This is a model that takes two frozen models; an image encoder and LLM, and then trains a Querying-Transformer (Q-Transformer) in two passes to create a bridge between them. Its focus is on aligning image features to questions rather than the referent in the question. BLIP2 and Flamingo authors say that the freezing of the image encoder stops catastrophic forgetting, which occurs when they trial fine-tuning in ablation studies. All these models add a disclaimer stating they are limited when using Foundational models, since their outputs could be unexpected, inaccurate or even inappropriate.

LLaVa [22] short for *Large Language and Vision Assistant* that appeared in Arxiv after our NS-IL had been accepted by IEEE, uses the LLM ability to align with the users question to give an appropriate answer. It has a structure that links an Image Encoder through a transformer to a LLM. PaLI [23] by Google Research is straightforward and designed for scalability. It employs an encoder-decoder Transformer structure, incorporating a high-capacity Vision Transformer (ViT) element for handling image data. It is currently (12/2/2024) ranking 1st on the VQA v2 Test-Dev dataset with 84.3% accuracy, The best BLIP model ranks 41st with a 65% accuracy and Flamingo ranks 48th with a 56% accuracy. Drilling into these structures shows the accuracy is directly dependent on the usefulness of the image encodings in relation to the task and dependent on the dataset. PaLI used a Vision Transformer (ViT) architecture named ViT-e to create the image encoding. ViT-e has the same architecture and used the same training recipe as the 1.8B parameter ViT-G model [24]. OpenFlamingo, LLaVa and many more use a model called CLIP [25] for image encodings.

The CLIP model *(Contrastive Language–Image Pre-training)* by OpenAI leverages a method centred on contrastive learning to effectively link visual and textual data. By training on a vast dataset of image-text pairs, it uses separate encoders to create embeddings for both modalities, learning to associate images with corresponding texts by bringing correct pairs closer and pushing incorrect ones apart. This foundational step allows for the dynamic creation of dataset classifiers directly from textual labels, transforming class descriptions into embeddings that can be compared against image representations. CLIP achieves zero-shot prediction capabilities, enabling it to classify images into unseen categories by matching the image's embedding with the closest textual label embedding.

To use a frozen Large Language Model, aligning visual features with the textual space is necessary. Various approaches, such as the 'Frozen' [26] model and the 'Flamingo' [27] model, have addressed this challenge differently. Downstream activities to make these foundational models more relevant include:

- Fine Tuning image encoders - Image encoders, which transform visual inputs into a feature-rich, machine-readable format, are crucial for enabling LLMs to 'understand' visual content. Fine-tuning these encoders ensures that the visual features are more aligned with the text representations used by LLMs. This fine-tuning process involves adjusting the encoder's parameters specifically for the task at hand, which could be identifying objects in an image, understanding scenes, or answering visually grounded questions. The goal is to optimise the encoder's ability to generate representations that are both comprehensive and compatible with the textual modality processed by the LLM.

- New Layers in the LLM - These new layers are designed to process and integrate the feature vectors generated by the image encoders, effectively bridging the gap between the visual and textual modalities. This could involve layers that perform specific transformations on the input features, layers that facilitate attention mechanisms between words and visual elements, or even layers that enable more complex interactions between the different types of data.

Tiong et al [28] propose a zero-shot VQA State-of-the-art model which can quickly adapt to new Visual Question Answering (VQA) tasks called *Plug-and-Play VQA*. These models effectively incorporate external context, bridging vision and language pre-trained models, handling multimodal inputs, and achieving zero-shot learning.

Jin et al [29] explore prompt-based learning models to reduce dependence on VQA training data while maintaining accuracy, making effective use of both image and textual context. Recognising the limitations of Large Language Model-based reasoning, recent proposals have emerged to integrate language models with external symbolic plug-ins such as Knowledge Graphs (KG) or toolkits.

2.4.3 *Benchmarking Datasets*

Datasets are vital in AI and ML, serving as the base for building and training models by providing examples to learn from. They are key for benchmarking, allowing for the objective evaluation and comparison of AI models' performances. High-quality datasets ensure models can handle real-world scenarios effectively, making AI solutions more robust and versatile. Diverse datasets drive innovation, pushing researchers to solve a broad array of problems and develop new AI techniques. They also play a crucial role in identifying and reducing biases, ensuring AI technologies are fair and equitable. As a result of President Biden's Executive Order 14110 in October 2023, the *National Artificial Intelligence Research Resource Pilot (NAIRR)* was formed with the aim to create resources that advance trustworthy AI, protects privacy, civil rights, and civil liberties. A focus on independent datasets would enable a true comparison of domain models. What is in the dataset and what that represents for society is a much bigger question to answer. It is a nuanced question as to whether datasets should be *open* so learning can be assessed and stay transparent. The importance of the dataset cannot be understated, whether it reflects what currently the world *is* or what we think the world *ought* to be and whose point of view that *ought* was created from is the driving force behind many political discussions in the pursuit of control.

Some of the VQA benchmarks are listed here, Figure 2.2 shows where the cross overs are of these datasets. The year is shown to help understand the dataset evolution.

- COCO [30]: The Microsoft COCO dataset provides 91 object types and approximately 2.5 million labeled instances in around 328,000 images. It focuses on identifying objects, understanding their precise boundaries, and describing images with text.

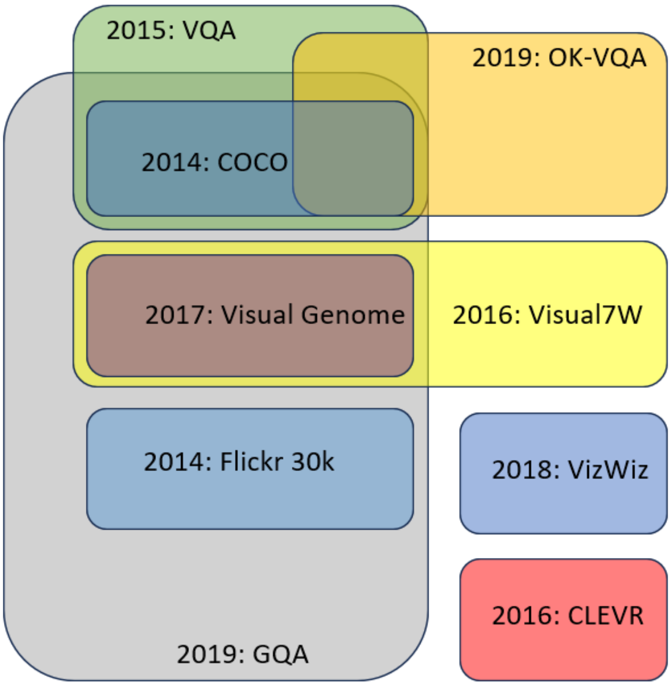


Figure 2.2: Dataset synergies.

- VQA Dataset (2015) [31]: This is one of the foundational datasets introduced for the VQA task. It contains open-ended questions about images, where the answers can be a word or a phrase. The dataset is split into multiple versions, with VQA v2 being a popular choice as it addresses some of the biases found in the earlier version by including complementary pairs of images with questions that have different answers.

- Visual7W (2016) [32]: This dataset extends the Visual Genome project by adding question-answer pairs to the images, categorised into who, what, where, when, why, and how questions, hence the name "7W." It provides both multiple-choice and open-ended questions.

- Stanford & Facebook AI Research's CLEVR dataset (2017) [33]: A dataset designed to evaluate a model's ability to understand complex reasoning about objects. CLEVR contains synthetic images of 3D shapes where questions require multi-step reasoning, making it significantly different from datasets based on natural images.

- Stanford's GQA Dataset (2019) [34]: GQA: Focused on real-world visual reasoning and compositional question answering, GQA offers a large number of visual questions that require understanding object properties, spatial relationships, comparisons, and more. It is designed to support more detailed and structured reasoning than other datasets.

- OK-VQA (2019) [35]: The Outside Knowledge VQA dataset, OK-VQA, is designed to assess a model's ability to use external knowledge (information not present in the image) to answer questions. This dataset challenges models to go beyond visual recognition to leverage broad world knowledge.
- VizWiz [36]: Unique among VQA datasets, VizWiz focuses on images taken by visually impaired users. The questions are sourced from real-world scenarios, making the dataset diverse and challenging due to the often poor quality of images and the real-life nature of the questions.

As a new era emerges, benchmarks such as *AgentBench* [37] have been created to evaluate Large Language Models (LLMs) acting as Agents in interactive environments, beyond traditional NLP tasks. It features 8 environments for assessing LLMs' reasoning, decision-making, and open-ended generation abilities. Testing across 27 API-based and open-source LLMs revealed significant performance differences, particularly highlighting commercial LLMs' superior capabilities. Challenges include poor long-term reasoning and instruction following. Enhancements in training, focusing on code and multi-turn alignment data, are suggested to improve LLM Agent performance. AgentBench provides datasets, environments, and an evaluation package for comprehensive analysis.

2.4.4 *Examples: NS-VQA & NS-CL Model*

The Neuro-Symbolic architectural approach separates reasoning from vision and language understanding. They merge structured image representation with symbolic programs from sentences, to reason and gain answers, integrating deep learning for visual and language recognition, with symbolic execution for reasoning, examples can be found in NS-VQA [38] and Cho et al [39]. The Neuro-Symbolic Concept Learner (NS-CL) [40] is a model that shares similarities with the Neuro-Symbolic Visual Question Answering (NS-VQA) architecture. NS-CL has a novel approach that combines symbolic programming and neural network processing to tackle complex tasks. It excels in breaking down questions into a series of programs, serving as step-by-step instructions for finding answers. What sets NS-CL apart is its ability to execute these programs on object features, which contain crucial information about the objects in the images. Vedantam et al [41] have introduced a novel approach to visual question answering using probabilistic neural-symbolic models. These models incorporate symbolic functional programs to represent how questions are answered based on visual data. What makes them unique is the addition of a stochastic latent variable, representing uncertainty in program generation and execution. During training, the model learns how visual input, questions, and programs are related. When confronted with new questions, it employs probabilistic methods to estimate possible values for the latent variable, allowing for different program variations. This flexibility enables the model to provide nuanced, probabilistic answers by generating and choosing answers from various program interpretations.

The Neuro-Symbolic Visual Question Answering model NS-VQA [38] is an example of combining the strengths of neural networks (excellent at processing raw, unstructured data) with symbolic AI (superior at logical reasoning and handling structured knowledge). This hybrid approach aims to improve AI's understanding and reasoning capabilities, allowing it to perform complex tasks that require both processing vast amounts of data and applying precise, rule-based reasoning.

The NS-VQA disentangles reasoning from vision and language understanding. It introduces a model that first interprets an image into a structured scene representation and then translates a question into a program trace, executing this program to generate an answer. Despite achieving high accuracy (99%) on the CLEVR dataset and showing promising results in scene understanding within Minecraft environments, challenges remain. The system uses predefined rules and knowledge bases to preserve logical consistency and relevance in the generated program or query, ensuring the output aligns with the semantic content of both the image and the question. The *questions are not in natural language.* The system has not been designed to *carry out incremental learning* or to adapt to new information with minimal data. If new segmentations are required, the *scene parser needs to be retrained using the full dataset.*

2.5 VQA Enhancements

This section introduces incremental learning in neural networks which addresses a critical challenge. Traditional systems lack the design to learn incrementally or adapt to new data with minimal resources. Often, to incorporate new classes, a complete retraining from the full dataset is required, pointing to a significant need for more flexible and efficient learning approaches. Incremental learning aims to solve these issues by enabling neural networks to update their knowledge continuously without the need for extensive retraining, making AI systems more adaptable and dynamic.

2.5.1 *Incremental Learning*

Catastrophic Forgetting (CF), as pinpointed by McCloskey and Cohen [42] over three decades ago, occurs when new learning disrupts old learning, leading to decreased accuracy. Ideally, maintaining the stability of a neural network's weights helps retain previously learned tasks, but excessive stability hampers the model's ability to acquire new tasks. Kirkpatrick et al [43] discuss the core challenge of the stability-plasticity dilemma, which lies in designing a balanced system that remains responsive to new inputs without being excessively perturbed to enable incremental learning.

Incremental Learning in Neural Nets involves gradually updating a trained model as it learns new Classes without forgetting the previously learned ones, DeLange et al [44] contrasts and evaluates multiple approaches. The Class Incremental Learning Task is a subset of Incremental Learning, it imposes constraints such as a limited memory or the absence of previously learned samples during training. These constraints artificially mirror the environment when there are practical considerations like storage and computing limitations, which prevents the retraining of the entire model for the addition of each new Class. It's important to note that Incremental Learning differs from Transfer Learning because it aims to maintain good performance in both old and new tasks. Evaluating an Incremental Learner involves assessing its classifier's performance on past and present tasks to ensure its adaptability to future, unseen Classes. Additionally, Lopez-Paz and Ranzato [45] introduced the concepts of Backward Transfer (BWT) and Forward Transfer (FWT), which measure how learning a new task influences the performance of previous and future tasks, respectively. Chaudhry et al [46] discuss these emerging metrics for *forgetting and intransigence* and try to strike a balance, between adaptability and retaining previously acquired knowledge.

Standard Incremental Learner models are usually constructed using a neural network framework, and this approach inherently brings about various challenges, including CF, memory limitations, and concept drift. Reviews and ontologies in this field, such as those found in Masana et al [47] categories the field into Regularisation, Exemplar and Task-based, Parisi et al [48] categories into Regularisation, Dynamic Architectures and Memory Replay and Complementary Learning Systems, Luo et al [49] categorise into Architectural, Regularisation (further divided into Weight Regularisation and Distillation), Rehearsal and Pseudo-Rehearsal. Additionally, they mention other approaches meta-learning and reinforcement learning-inspired methods. and De Lange et al [50] categorise as Replay, Regularisation and Parameter isolation. Figure 2.3, has been adapted from Liu et al's Taxonomy [2], which was chosen for its structured categories and provides a framework to drill into. They establish three primary categories to address challenges within DL Class Incremental Learning. These are; (1) Parameter Regularisation, (2) Knowledge Distillation, and (3) Dynamic Architecture. We'll cover each of these next.

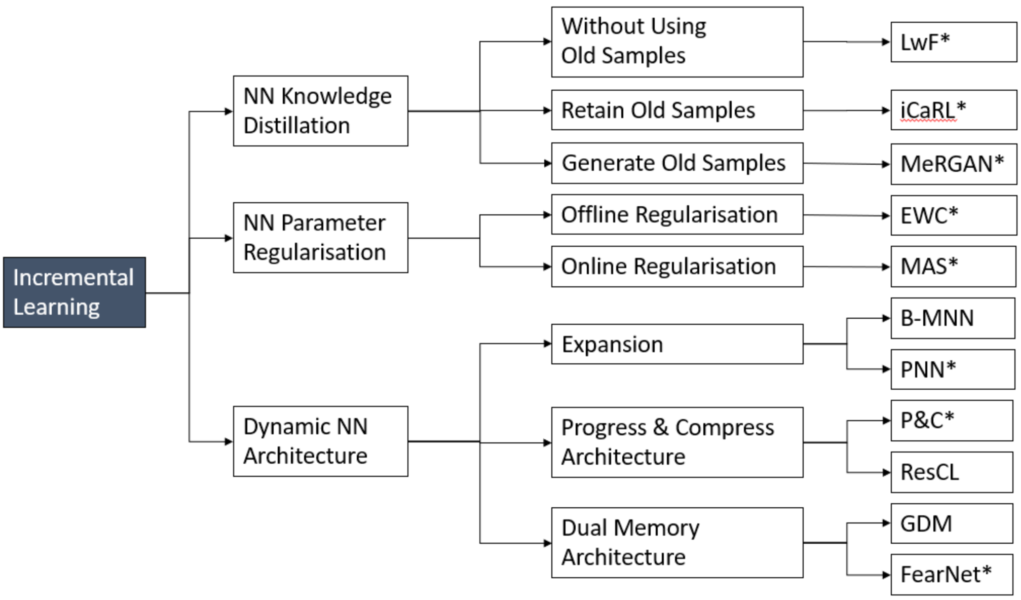


Figure 2.3: Ontology of Incremental Learners adapted from Liu et al [2]. The models shown are the most cited according to Liu et al (*) or I have mentioned it in the related work. Model's names in Table 2.1.

Description	Model Name	Cite
Learning Without Forgetting	LwF	[51]
Incremental classifier and representation learning	iCaRL	[52]
Memory Replay GANs	MeRGAN	[2]
Elastic Weight Consolidation	EWC	[53]
Memory Aware Synapses: Learning What (not) to forget.	MAS	[54]
Knowledge transfer in deep block-modular neural networks.	B-MNN	[55]
Progressive Neural Networks	PNN	[56]
Progress & Compress	P&C	[57]
Residual continual learning.	ResCL	[58]
Fearnet: Brain Inspired Model for Incremental Learning.	FearNet	[59]
Lifelong learning of spatiotemporal representations with dual-memory recurrent self-organisation.	GDM	[60]

Table 2.1: Model Names for Figure 2.3

Methods based on Parameter Regularisation employ techniques like constraining the modification of crucial parameters, implementing dropout, and applying early stopping. These methods share a common goal, to preserve knowledge from previous tasks. On the other hand, Knowledge Distillation techniques involve transferring knowledge from an old model to the current one. This transfer can be achieved through various approaches; (1) retaining old samples, as seen in the iCaRL model [52], (2) without relying on old samples, or (3) generating new samples. Additionally, dynamic architecture methods typically adapt the network structure flexibly to accommodate new tasks as they are introduced.

Dynamic Architecture strategies encompass three main approaches: (1) expansion, (2) progress and compress (P&C), and (3) dual memory (D-M) architectures. In Black-Modular neural networks (B-MNN) [55] and Progressive neural networks (PNN) [56], an additional neural network is incorporated alongside the existing one, akin to a 'piggy-back' neural net. This auxiliary network is trained for the new task. It's important to note that this differs from fine-tuning, which simply adds one more layer to a frozen memory. A limitation of this model is that the number of parameters grows exponentially for each new learned task.

Progress and Compress (P&C) architectures retain a constant number of parameters and are comprised of a knowledge base and an active column. During the compression phase, knowledge acquired in previous expansion phases is extracted to the knowledge base. The Elastic Weight Consolidation (EWC) strategy [53] is employed to safeguard the previously acquired knowledge. In the expansion phase, learning new tasks leverages the characteristics stored in the knowledge base through lateral connections. This training technique alternates to control the model's growth while preserving knowledge. However, these methods may face scalability limitations in complex multi-task incremental learning scenarios.

Dual memory architectures are rooted in the concept of complementary learning systems (CLS) theory [61], [62], drawing inspiration from the interplay between the hippocampus and neocortex systems, which balance fast and slow learning processes. This is a similar idea to Daniel Kahneman's book entitled 'Thinking Fast and Slow' where he defines two systems; system 1 operates fast also intuitively and system 2 operates with intentionality, allocating attention for tasks. Typically, a D-M architecture comprises of long and short-term memory components. The long term memory is dedicated to storing previous learning experiences, while the latter is focused on acquiring knowledge for current tasks. The 'Growing Dual-Memory' (GDM) [60] takes into account the consequences of incremental learning.

Replay methods excel by reinforcing memory through data rehearsal but struggle with high memory and computational costs. Regularisation techniques efficiently maintain knowledge without additional data storage, yet they can be rigid, hampering learning in complex scenarios. Parameter isolation effectively segregates knowledge, preventing task interference, but faces scalability challenges as the number of tasks increases. Self-supervised learning, while enhancing adaptability through unlabeled data, depends on the task relevance, which may not always align with learning objectives.

3
The Traditional VQA Model

3.1 Introduction

In the Human Visual System (HVS), there are two crucial components: the eye and our brain's interpretation of what the eyes transmit. Actual 'Seeing' occurs only after our brains process the signals from our eyes. These signals are quite basic, primarily consisting of information about edges, shapes, and motion rather than complete images. 'Seeing' is a complex function of the brain, and a significant part of our brain, actually the largest part, is focused on identifying what's in our field of vision. Our brains construct images through pattern recognition.

In many traditional machine learning tasks, significant effort is expended on feature engineering, the process of manually designing and selecting the appropriate input features for a model. *Representational learning* aims to automate this process. Instead of hand-crafting features, the model learns the most informative features from the data itself. Representational learning is central to DL. Deep Neural Nets (DNN), especially Convolution Neural Nets (CNN) and Autoencoders, are adept at learning layered and hierarchical representations from raw data. Each layer of the network learns increasingly abstracted features. For instance, in image recognition tasks, the early layers might recognise edges and textures, while deeper layers might recognise more complex structures like shapes or even entire objects.

Once a good representation is learned, it can be useful for a variety of tasks. For example, a Neural Net pre-trained on a large image dataset (e.g. ImageNet) can be fine-tuned for a more specific task, benefiting from the representations learned during pretraining. It reduces the need for labour-intensive feature engineering, making it easier to apply machine learning to new domains. One of the main challenges is ensuring that the learned representations are meaningful and interpretable.

While deep learning models are great at learning representations, visualising high-dimensional feature spaces directly is impractical, so methods like PCA, t-SNE, and UMAP are often employed to project these spaces into 2D or 3D for visualisation and analysis. This gives the model designer visual feedback as to how well data samples are been clustered in the feature space. Generally, the higher the number of encoded features, the better the images are clustered or organised, enabling improved classification accuracy.

3.2 The Deep Learning VQA Model

The main goal of the VQA Task is to train a function, which can predict an answer for a given question about a given image. DL models designed for the VQA task, integrate components for both *visual perception* to interpret the image, and, *textual understanding* to interpret the question, and then fuse this information to generate an answer. With reference to each of the elements shown in Figure 3.1 they carry out the following function:

- *Image Encoder* - This focuses on the process of turning the raw image into a fixed-sized feature vector that represents the image's contents through encoding.

- *Question Encoder* - The input question consists of a sequence of words, which is transformed into a semantic vector. The question is tokenised, and each word is represented using embeddings (like Word2Vec or GloVe) which is used as the semantic representation of the question. Alternatively, Transformer-based models like BERT or the ChatGPT family can be used to create embeddings.

- *Fusion of Image and Question Feature Vectors* - Once the vectors exist that represent both the image and the question, the next step is to fuse them.

- *Answer Prediction* - The fused vectors are then passed through one or more fully connected (dense) layers to predict the Answer.

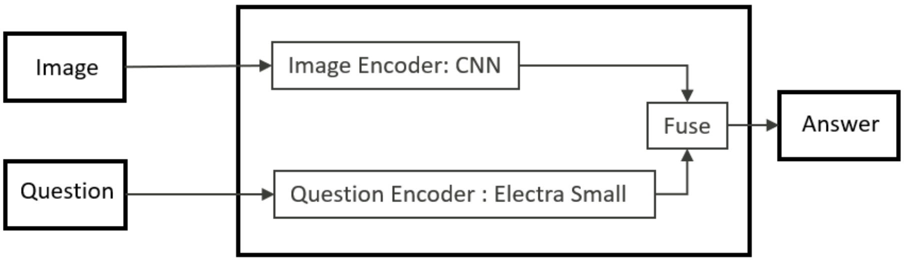


Figure 3.1: A Visual Question Answering Model.

A deep learning model is trained in a supervised, end-to-end fashion using datasets that accurately reflect the configured intention for a system. Take for example visual data collected and used, this could reflect what reality *is* like, or what reality *ought* to be like. It is very dependent on how the sample data has been curated. The dataset is of key importance because it determines what the model will learn and therefore determines what the model predicts the answer to be.

The created model is then validated and tested against a held-out dataset with performance measures based on accuracy scores. The confusion matrix shown in Figure 3.2 shows the number of data samples that were correctly categorised, as well as any errors and what they were actually categorised as. It can be seen here that Class 5 is predicted most accurately with a score of 988 on the diagonal, and Class 6 is the least accurate with a score of 485. When doing multi classifications each class's recall and precision can be calculated as shown for Class 1 in the Figure 3.3. The scores for True Positive Rate (TP), False Positive Rate (FP), True Negative Rate (TN) and False Negative Rate (FN) are used to calculate the individual metrics. The recall measures the proportion of positives that are correctly identified as such $TP/(TP + FN)$ and the precision measures the proportion of positives that correspond to the presence of the condition $TP/(TP + FP)$.

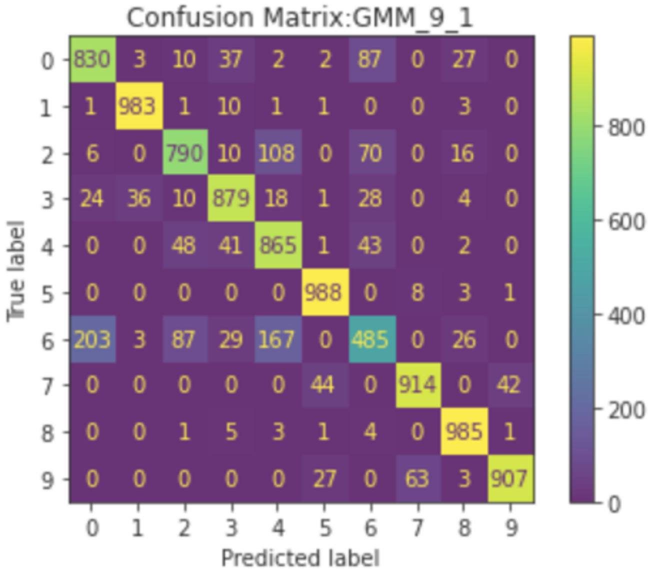


Figure 3.2: A Confusion Matrix for the Fashion-MNIST dataset.

		Predicted									
		0	1	2	3	4	5	6	7	8	9
Actual	0	TN	FP	TN	TN	TN	TN	TN	TN	TN	TN
	1	FN	TP	FN	FN	FN	FN	FN	FN	FN	FN
	2	TN	FP	TN	TN	TN	TN	TN	TN	TN	TN
	3	TN	FP	TN	TN	TN	TN	TN	TN	TN	TN
	4	TN	FP	TN	TN	TN	TN	TN	TN	TN	TN
	5	TN	FP	TN	TN	TN	TN	TN	TN	TN	TN
	6	TN	FP	TN	TN	TN	TN	TN	TN	TN	TN
	7	TN	FP	TN	TN	TN	TN	TN	TN	TN	TN
	8	TN	FP	TN	TN	TN	TN	TN	TN	TN	TN
	9	TN	FP	TN	TN	TN	TN	TN	TN	TN	TN

True Positive Rate	Recall = TP/(TP + FN)
Positive Predictiive Rate	Precision = TP/(TP+FP)

Figure 3.3: Metrics for Class 1 within a Classifier.

The key driver when creating this model is to optimally create a function that accurately predicts data samples at inference time that can be found within the data distribution of the training dataset. Whenever a new class needs to be added, the full network needs to be retrained. Reasoning is not possible over the structure and no external knowledge can be integrated. Hence the motivation to create a model that can incrementally learn.

3.2.1 *A DL VQA Example*

Structure

This example uses the following drop-in models which provide prior learning to the VQA. The model structure can be seen in Figure 3.4

- *Image Encoder* - A *CNN*.
- *Question Encoder* - A *transformer model* called *Electra small* [63]. Which has reduced resource demands and hence suited our processing capability.
- *Fusion of Image and Question Feature Vectors* -The encoded features were concatenated to form a longer vector.
- *Answer Prediction* - The concatenated vector is then passed through connected (dense) layers to predict the Answer.

Training Dataset

A publicly available dataset called *Easy-VQA* was used. It contains 2-Dimensional images of different shapes, each with a different colour and position. The dataset statistics can be seen in Table 3.1, example question and answer data can be seen in Table 3.2 and example images can be seen in Figure 3.5. The dataset can be downloaded at `https://github.com/vzhou842/easy-VQA`.

Input	**Train**	**Test**
Image	4,000	1,000
Question	38,575	9,673
Binary Questions (Y/N)	27,407	7,136

Table 3.1: Statistics for the Easy_VQA Dataset.

Data	**Details**
Answers	triangle, rectangle, circle, blue, teal, black, yellow, brown, red, grey, green, no, yes
Question 1 - Shape	what shape is in the image?
Question 2 - Shape	what shape is present?
Question 3 - Shape	what shape does the image contain?
Question 4 - Shape	what is the color_name shape?
Question 1 - Color	what color is the shape_name?
Question 2 - Color	what is the color of the shape_name?
Question 3 - Color	what color is the shape?
Question 4 - Color	what is the color of the shape?
Question 1 - Yes/No	is there a shape_name?
Question 2 - Yes/No	is there a shape_name in the image?
Question 3 - Yes/No	does the image contain a shape_name?
Question 4 - Yes/No	is a shape_name present?
Question 5 - Yes/No	is there not a shape_name?
Question 6 - Yes/No	is there not a shape_name in the image?
Question 7 - Yes/No	does the image not contain a shape_name?
Question 8 - Yes/No	is no shape_name present?
Question 9 - Yes/No	is there a color_name shape?
Question 10 - Yes/No	is there a color_name shape in the image?
Question 11 - Yes/No	does the image contain a color_name shape?
Question 12 - Yes/No	is there not a color_name shape?
Question 13 - Yes/No	is there not a color_name shape in the image?
Question 14 - Yes/No	does the image not contain a color_name shape?
Question 15 - Yes/No	is no color_name shape present?

Table 3.2: Example Data values from Easy_VQA Dataset.

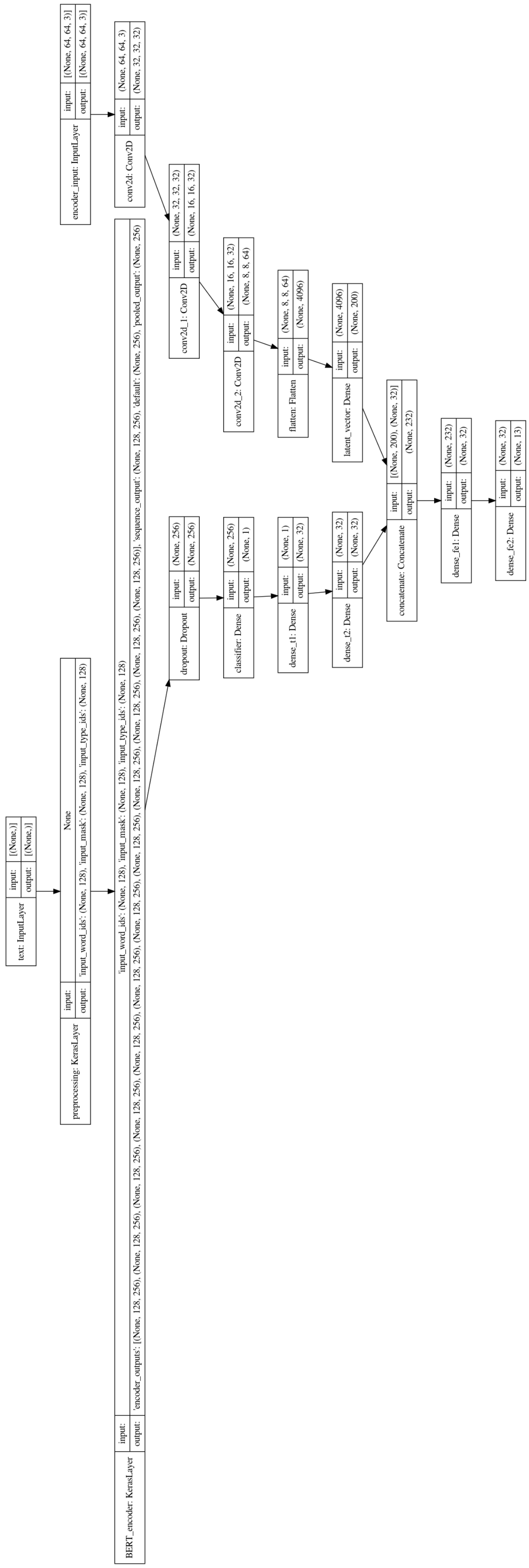


Figure 3.4: A Deep Learning VQA Model.

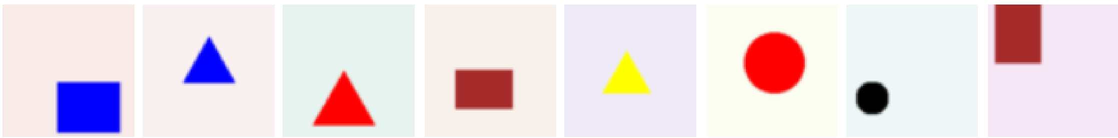


Figure 3.5: Example images from Easy_VQA Dataset.

Inference Time - Feature Embedded Layer Selection

During inference, the TensorBoard Embedding Projector uses Principal Component Analysis to graphically represent high-dimensional embeddings into 3D. This is useful to understand the clustering achieved in different embedding layers. Looking at Figure 3.6 which is a mid-feature embedding labeled by actual image, shows that the 3 Principal components are managing to separate data by shape size and colour. Figure 3.7 shows a later embedding layer from the neural net labeled by Answer, this shows the structure in the feature space can be separated by Answer Type; Yes/No, Shape, and colour. It is the structure of this feature space that enables a classifier to classify.

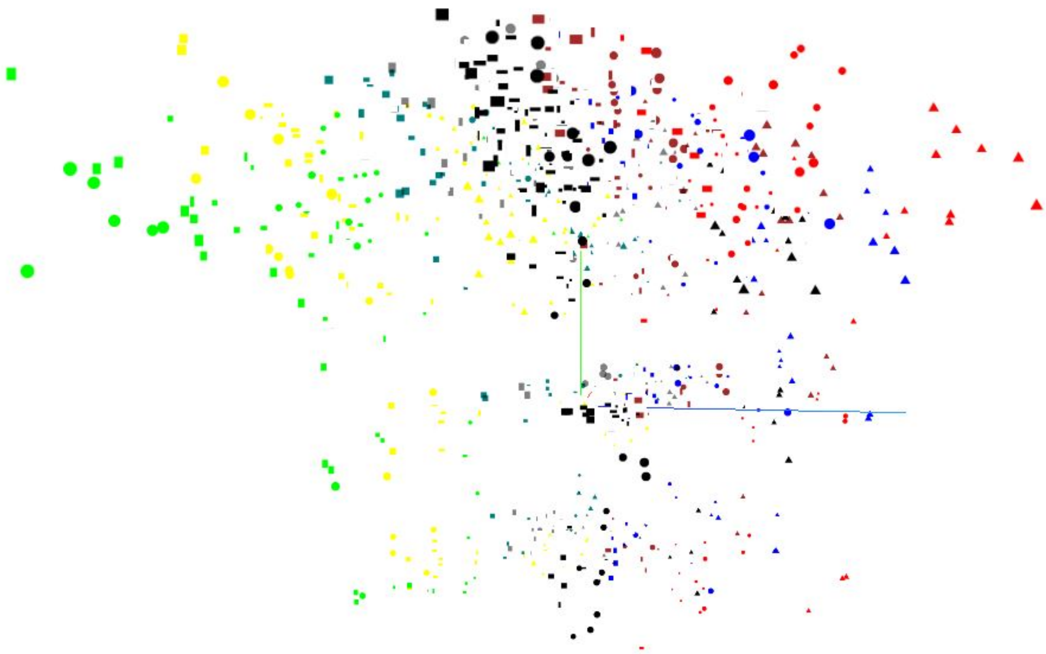


Figure 3.6: A Deep Learning VQA model with a 3-D Projection of a mid-feature embedding using the Easy_VQA dataset.

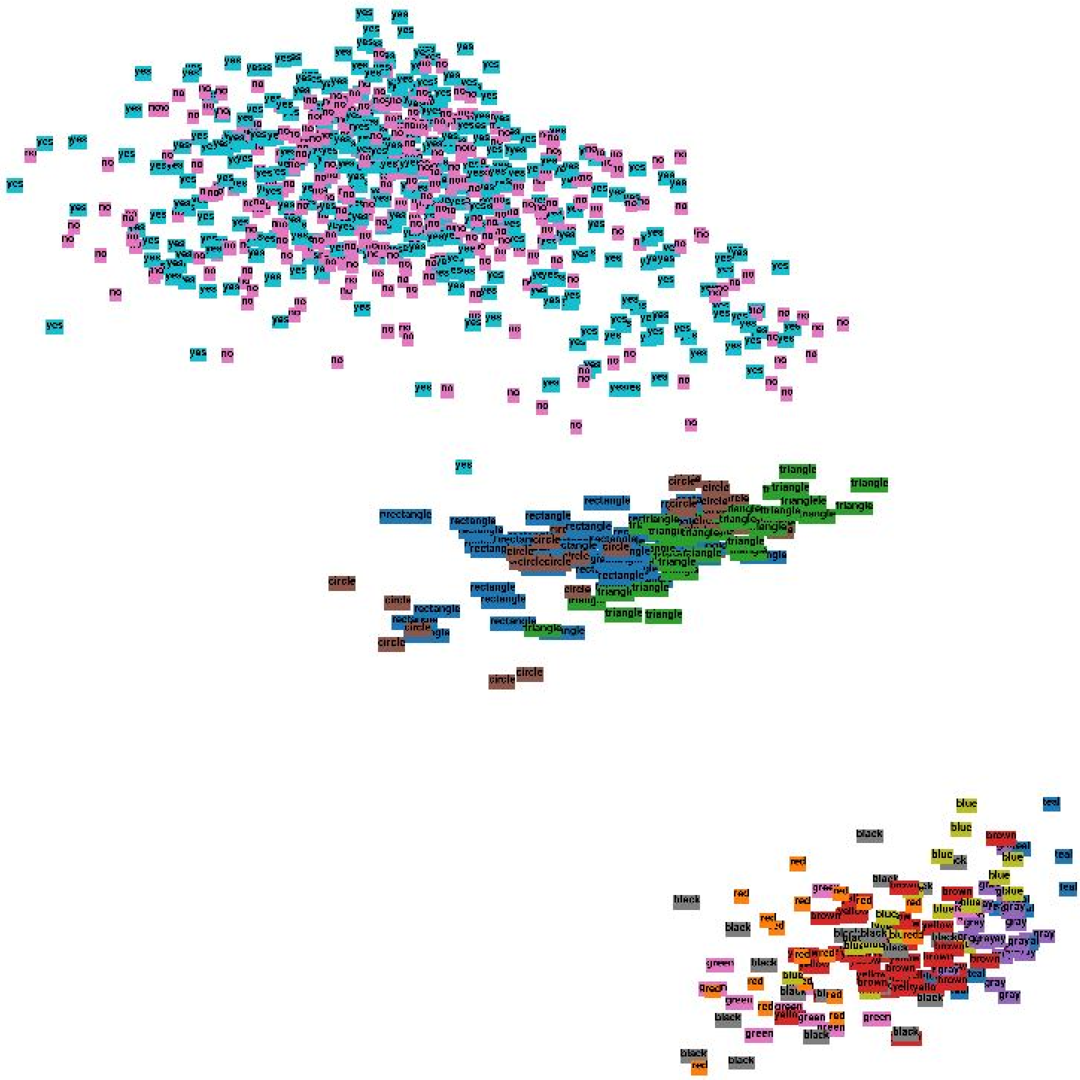


Figure 3.7: A Deep Learning VQA Task with a 3-D Projection of a late feature embedding using the Easy_VQA dataset.

Inference Time - Feature Number Selection

In these examples, **the image encoder in the traditional VQA model has been replaced by an Autoencoder** and the Easy_VQA dataset is still used. Again, using the Tensorflow projection and looking at the first three principal components of the bottleneck embedding layer, labeled with the actual input images, we visually assess the latent structure. Figure 3.8 shows when the bottleneck layer was set to 32 features and Figure 3.9 shows when the bottleneck layer was set to 200 features. It can be seen that the more features used, enables a more informed latent structure to be formed and hence improved separation of data samples.

This structural latent space is called the *Feature Space*. As data progresses through the CNN layers, it transforms, resulting in various representations at different levels of abstraction. Each of these representations can be thought of as occupying a specific point within a feature space. The key idea here is that the bottleneck layer acts as a reduced-dimensional representation that captures the essential features and variations in the data. This representation can be thought of as a manifold embedded within the high-dimensional feature space. The concept behind altering the data through multiple layers is to adapt the features for a specific task, like classification.

Ideally, when data reaches the layer to be extracted as a feature vector from a CNN, the multi-dimensional feature space (manifold) should be organised in a way that makes it easy to distinguish data points from different classes, either by separating them linearly or clustering them effectively. This is a form of *manifold learning*.

Figure 3.8: Visualisation of the bottleneck embedding layer of a Traditional VQA when the Image Encoding uses an Autoencoder with 32 features (Algo: t-SNE).

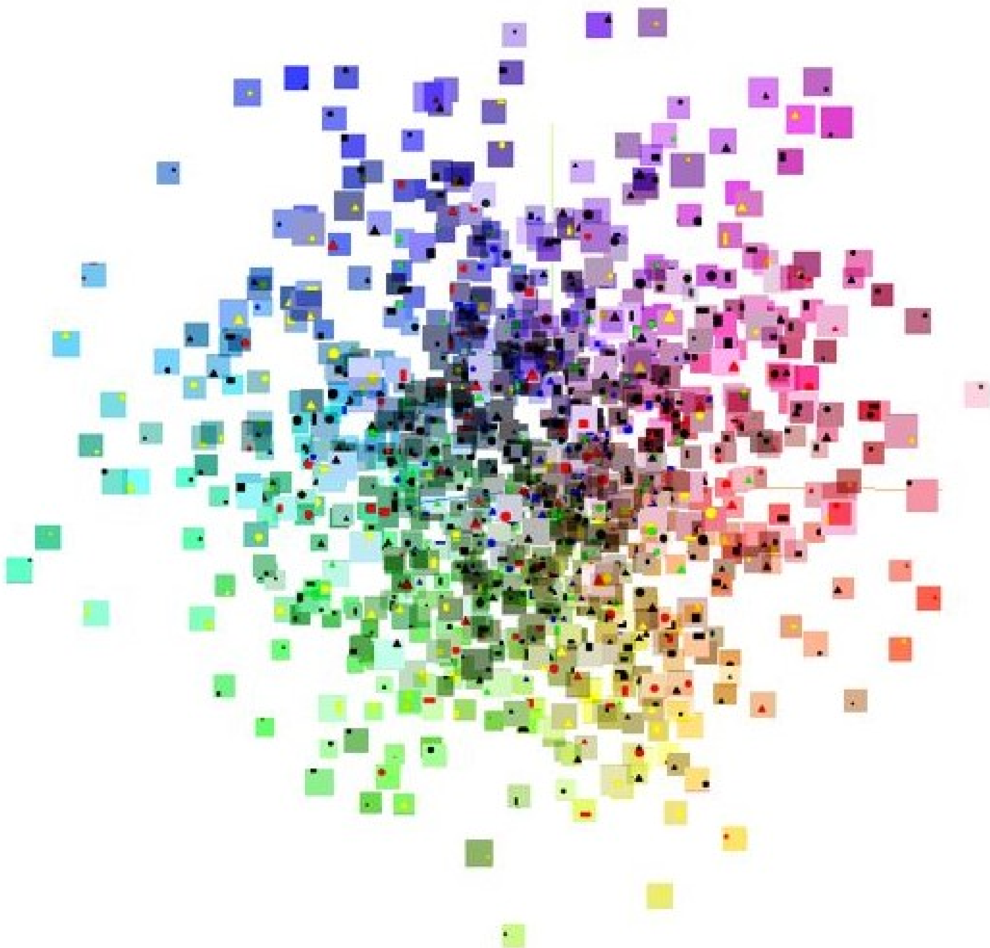


Figure 3.9: Visualisation of the bottleneck embedding layer of a Traditional VQA when the Image Encoding uses an Autoencoder with 200 features (Algo: t-SNE).

4

GMM-IL: The Gaussian Mixture Model Incremental Learner

4.1 Introduction

Looking at the structure in the traditional deep learning VQA as seen in Figure 4.1, the continuous unstructured image data is abstracted into discrete structured class data.

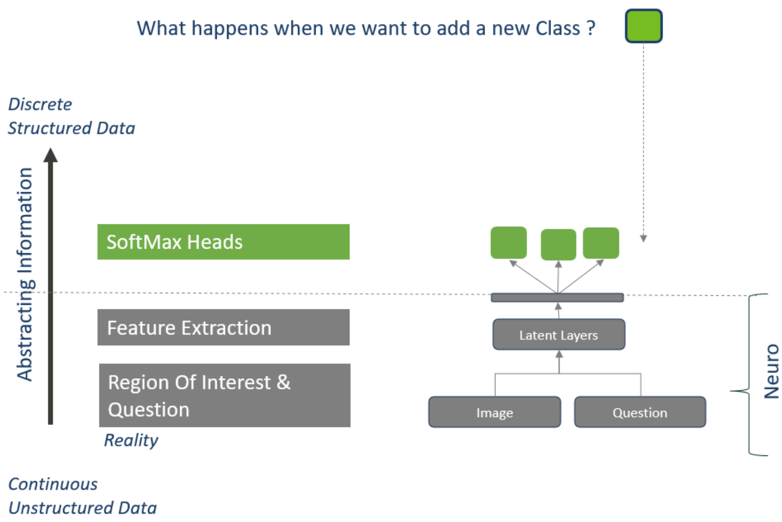


Figure 4.1: Data Flow for the Traditional VQA.

This is achieved by creating a function during the training of the neural net that can take the inputs and transform them into the correct outputs. At the heart of this thesis is the process of abstracting what we observe and translating it to a descriptive Class (Term), essentially how to *encode meaning*. This encoding is done through statistics.

The traditional VQA is a snapshot at a point in time when the current data in the current environment has been sampled and the current class requirements have been modeled. However, our reality is an evolving dynamic landscape and the need exists for the models to be incremental class learners. This naturally leads to the question, *What happens when we want to add a new class ?* As seen in Figure 4.2 we have a requirement over time to be able to grow an Ontology. That is, be able to add and delete classes as and when required.

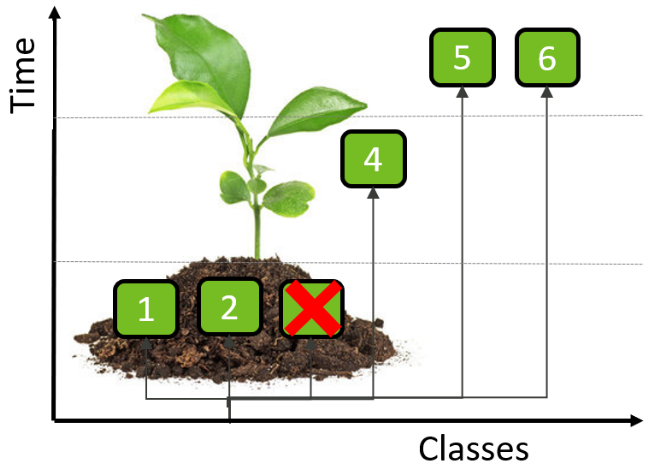


Figure 4.2: Growing an Ontology.

Incremental learning of new Classes without forgetting old Classes is essential for real-world problems but is extremely challenging for modern DL methods. Current incremental deep learners suffer from *catastrophic forgetting* when after learning Class A, they are then required to learn Class B. The issue occurs due to the sharing of a set number of weights in the neural network. These are optimised for Class A, however, to learn Class B the weights must be altered, resulting in new knowledge overwriting previous knowledge.

4.2 A Design Decision

In order to overcome this catastrophic forgetting, a universal function approximator is introduced, in the form of an independent *GMM*. This enables a separation of tasks between, learning the principal visual features and learning the class definition. The GMM also enables additional clustering and generative task functionality.

Figure 4.3 shows how during abstracting the data from the image, features are selected. Then, how each GMM is trained in terms of those features. Each GMM can then be grouped within a category that acts as an image classifier. **It is very important to note that here we replace the ability to accept multiple questions as inputs to having an implicit question which is, *'What is in this image ?'*** In this chapter we drill into how to incrementally learn classes. The limitation to the system that an implicit question introduces, will be addressed in chapter 6.

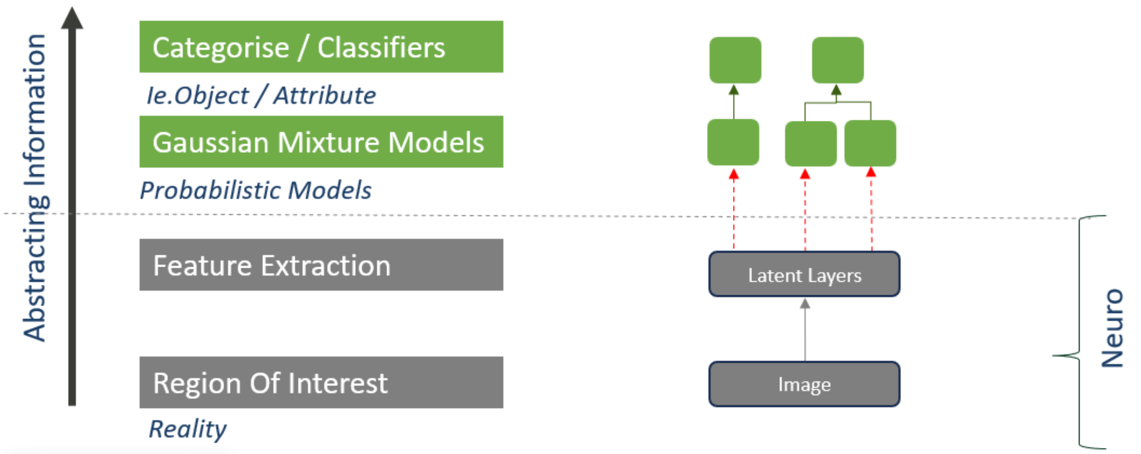


Figure 4.3: Class Incremental Learner.

4.3 The GMM-IL Model

4.3.1 *Introduction*

The GMM-IL Model was published under the title 'GMM-IL: Image Classification Using Incrementally Learnt, Independent Probabilistic Models for Small Sample Sizes'. Taking our inspiration from humans, the proposed model creates a specific structure at four levels and each level is aligned with a human concept and supports open-ended learning which combines the strengths of the symbolic approaches with insights from machine learning. The analogies between humans and machines are shown below for each of the four levels:

1. Human: Learns to see visual information.
   Machine: Learns a feature space of visual features during Autoencoder training.

2. Human: After seeing an object, are taught a name to give it meaning.
   Machine: After learning a prior feature space, train a probabilistic model to represent a class which gives that group of training images a symbolic meaning.

3. Human: Learn a new object without needing to see previous objects at the same time. Previously learnt objects can be imagined.
   Machine: Train a new class in the form of a Gaussian Mixture model without requiring access to previous training data. Previously learnt GMMs can be sampled and the resulting feature embedding decoded back to images when an Autoencoder is used.

4. Human: Identify objects they are paying attention to, in their field of view in real-time.
   Machine: Carry out Region Of Interest (ROI) classification for the contents of an image at inference time.

4.4 Methodology

Our aim is to classify each visual Class using incremental learning, trained on small sample sizes using a hybrid architecture as can be seen in Figure 4.4. The proposed architecture is modular in nature, enabling drop-in replacements for the Autoencoder and Gaussian Mixture Models. A description of the selection and training of these models can be found in Section 4.5.

The four levels and their interactions from the bottom up, are:

1. An *Autoencoder Model* trained once on a large corpus of unlabelled images, to learn a useful visual feature space from the image corpus. Detailed in Section 4.4.1.

2. Independent *Gaussian Mixture Models* which form the class definition, independently trained on a small number of visual features. Visual features are a result of encoding the defining class images. Detailed in Section 4.4.2.

3. A *Classifier* comprised of a set of learnt GMMs which can be added to as new classes become available. Detailed in Section 4.4.3.

4. *Classification* logic can be carried out across all the GMMs in a specific classifier to evaluate the likelihood that at inference time, a specific image belongs to a specific class. Detailed in Section 4.4.4.

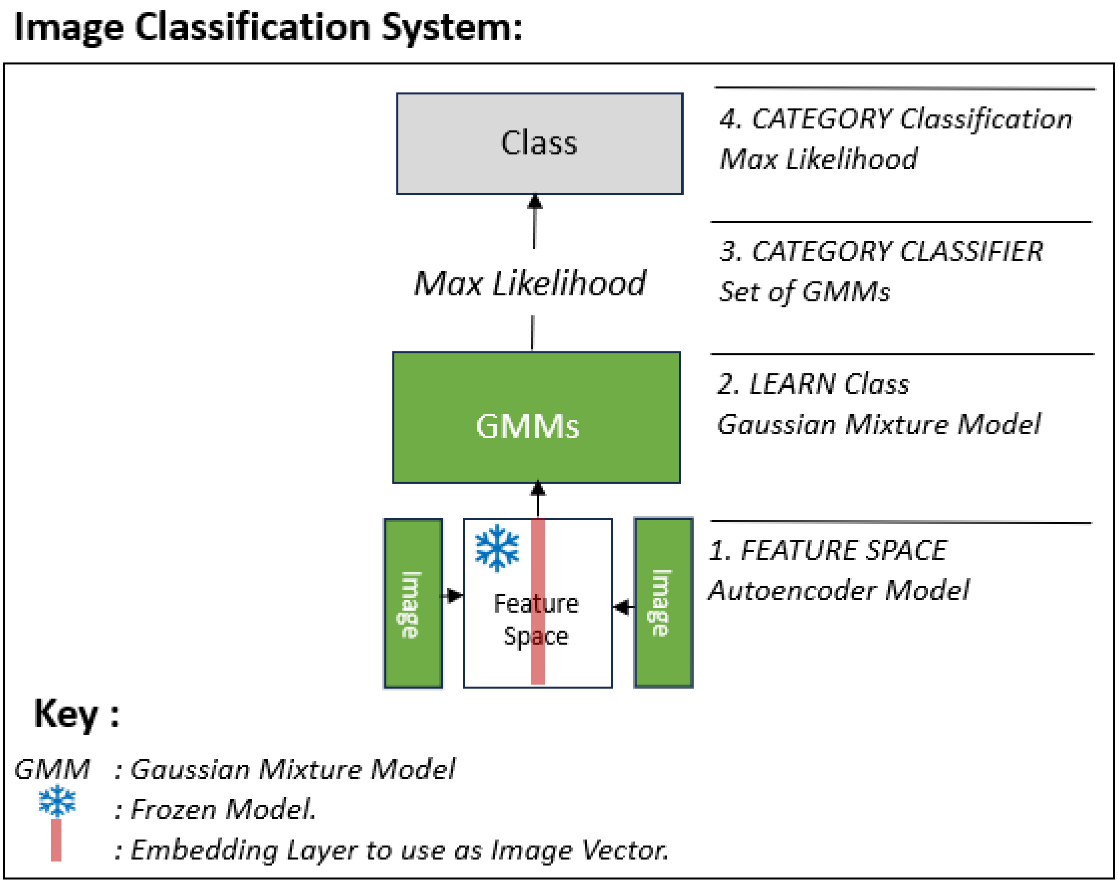


Figure 4.4: GMM-IL: (1) Autoencoder Model,(2) Gaussian Mixture Model, (3) Category Classifier, (4) Category Classification Logic.

4.4.1 *Autoencoder Model*

The Autoencoder transforms an image from a high dimensional space to a lower dimensional space for ease of manipulation. The main premise is that unsupervised training initialises this encoder on a vast number of image samples in a high compute environment. When encoding visual features, the interest is not only in the autoencoder's ability to reconstruct the input image, but also on encoding a useful representation. By useful it means the representation is not task specific, is spread throughout the feature space and contains visual isomorphisms at different scales. These attributes will enable it to generalise well to unseen symbolic classes. An Autoencoder was selected since Chadebec et al [64] carried out a case study benchmark, where they presented and compared 19 generative Autoencoder models. They found that the Autoencoder which did not try to manipulate the latent space in end-to-end training produced the highest classification accuracy.

4.4.2 *Gaussian Mixture Models*

When traditional DL networks are trained with backpropagation, the knowledge acquired is stored through the weights and connections between neurons. These weights control information storage. When training for a specific task, like image recognition, weights change to capture data patterns and relationships, thereby encoding the knowledge for the whole task. Since the knowledge is infused throughout the network, it is very hard to identify a discrete symbolic entity.

We chose a GMM to represent an entity in terms of the features within the neural net. This GMM or Class in our system, trained on feature encodings, were chosen for several reasons. GMMs offer flexibility, are capable of approximating a wide range of data distributions by combining multiple Gaussian distributions, which makes them suitable for complex data. They excel in clustering tasks, accommodating scenarios where data points (feature encodings) within a Class may follow distinct Gaussian distributions, which is particularly useful when Classes exhibit sub-populations or multiple modes. GMMs also provide a measure of uncertainty, essential when data points may belong to multiple clusters or when dealing with noisy data. As a generative model, they can simulate synthetic data resembling the observed distribution, useful for data augmentation and further analysis. Additionally, GMMs are computationally efficient and scalable in high-dimensional feature spaces. These GMMs can be grouped into specific categories to act as an image classifier.

Once a visual feature space has been established, specific classes can be built on top. Representative images are selected for a class to form a definition. First images are translated to feature encodings using the Autoencoder, these feature encodings are then used to train the GMMs [65]. The GMMs can act as a universal function approximator given enough components, it can accurately approximate any smooth function as shown by Maz'ya [66] enabling it to statistically capture the visual features for an independent class.

The following sections cover; the Maximum Likelihood Estimation (MLE) algorithm used to train the GMM, and the likelihood measure that quantifies the fit of a GMM to the current ROI image.

GMM - Maximum Likelihood Expectation

The GMM is comprised of Gaussian distributions, each with its own parameters for mean and covariance. The challenge is that for a given data point (a feature encoding), we don't know which Gaussian distribution or cluster it comes from. The Expectation Maximisation (EM) algorithm shown in Algorithm 1 [67] helps us solve this problem iteratively. In the E-step, using the current parameter estimates, it calculates the expected probabilities or assignments for each data point to belong to each Gaussian. Next, during the M-step, it adjusts the Gaussian parameters using these probabilities. This cycle repeats until the algorithm converges, leading to a maximum likelihood estimate of the parameters, even when the data contains missing values. Essentially, EM provides a structured method for iteratively fine-tuning the parameters to maximise the likelihood of observing the provided data.

The EM algorithm has strengths and weaknesses. It's reliable because it always converges and can deal with complex constraints in optimisation. On the downside, it can get stuck in local maximum points, not necessarily finding the best overall solution. This problem occurs because the choice of starting parameters greatly influences its convergence.

Algorithm 1: Expectation Maximisation for Gaussian Mixture Models.

Input:
Dataset $X = x_1, x_2, ..., x_N$ where x_i is a *feature vector* and N represents the total number of data points.
Parameters:
k: Number of Gaussian mixtures.
π_k : Prior probability of mixture k, initialised for each k.
μ_k : Mean of mixture k, initialised for each k.
Σ_k: Covariance matrix of mixture k, initialized for each k.
Output:
Updated parameters π_k, μ_k, Σ_k for each mixture k.

1. Initialisation Take initial guesses for the parameters π_k, μ_k, Σ_k for each Gaussian mixture.

2. Expectation Step (E-step): Calculate the probabilities $\gamma(z_n k)$
$\gamma(z_n k) = \frac{\pi_k \mathcal{N}(x|\mu_k,\Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x|\mu_j,\Sigma_j)}$

3. Maximisation Step (M-step): Using the estimated values of $\gamma(z_n k)$, we estimate the parameters π_k, μ_k, Σ_k to maximize the likelihood. To do this, we first recalculate the means using the following equations and then the updated values of μ_k to estimate the values of Σ_k's.

$\mu_k^{new} = \frac{1}{N_k} \sum_{i=1}^{N} \gamma(z_{nk}) x_n$

Updated covariances:
$\Sigma_k^{new} = \frac{1}{N_k} \sum_{i=1}^{N} \gamma(z_{nk})(x_n - \mu_k^{new})(x_n - \mu_k^{new})^T$

Updated mixing coefficients:
$\pi_k^{new} = \frac{N_k}{N}$

where $N_k = \sum_{i=1}^{N} \gamma(z_{nk})$
where $\gamma(z_{nk})$ is given by equation in E-Step.
4. Iterate steps 2 and 3 until convergence is reached.

GMM Likelihood

The system is designed to ground a KG by processing visual data and generating a *likelihood metric* for each given Class. This metric gauges how closely specific Classes match the observed data statistically. With a GMM, we can assess during inference how likely it is that a feature encoding belongs to a specific Class within a provided category. Specifically, the GMMs are designed to maximize the likelihood of accurately modeling the training data.

We will use the notation: π_i as the mixing coefficient, μ_i represents the mean, Σ_i denotes the covariance matrix, and, $\boldsymbol{\lambda}$ encompasses the entire collection of component parameters, i.e., $\lambda = (\pi_i, \mu_i, \Sigma_i)$. Here, the weights, means, and covariance matrices correspond to each Gaussian component.

Given observed feature encodings, as described in Equation (4.1), $p(\boldsymbol{x}|\lambda)$ provides the estimated likelihood of a feature encoding $\boldsymbol{x}$. This estimation is a combination of Gaussian distributions, each characterised by its unique mean, covariance, and mixing coefficient.

$$p(\boldsymbol{x}|\lambda) = \sum_{i=1}^{M} \pi_i g(\boldsymbol{x}|\boldsymbol{\mu_i}, \boldsymbol{\Sigma_i}) \tag{4.1}$$

In this context, $\boldsymbol{x}$ stands for a D-dimensional data encoding with continuous values (representing visual features). The $\pi_i, i = 1, ..., M$ represent the weights for the mixtures, and $g(\boldsymbol{x}|\boldsymbol{\mu_i}, \boldsymbol{\Sigma_i}), i = 1, ..., M$ are the densities of the Gaussian components. Each Gaussian component is a D-dimensional Gaussian function, defined as shown in Equation 4.2:

$$f(\boldsymbol{x}|\boldsymbol{\mu_i}, \boldsymbol{\Sigma_i}) = \frac{1}{(2\pi)^{\frac{D}{2}} |\boldsymbol{\Sigma_i}|^{\frac{1}{2}}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu_i})^T \boldsymbol{\Sigma}_i^{-1} (\boldsymbol{x}-\boldsymbol{\mu_i})} \tag{4.2}$$

This function relies on a mean vector $\boldsymbol{\mu_i}$ and covariance matrix $\boldsymbol{\Sigma_i}$. The mixture weights satisfy: $\sum_{i=1}^{M} \pi_i = 1$. The entire Gaussian mixture model is defined by the mean encoding, covariance matrices, and mixture weights associated with all its Gaussian components. We collectively represent these parameters using:

$$\lambda = \{\pi_i, \boldsymbol{\mu_i}, \boldsymbol{\Sigma_i}\} \quad i = 1, ..., M \tag{4.3}$$

Different variations of the GMM can be found, as demonstrated in equation 4.3. For example, the covariance matrices, $\boldsymbol{\Sigma_i}$, can either have full rank or be restricted to only diagonal elements. Parameters can also be shared or tied among the Gaussian components.

4.4.3 *Classifier*

Once the GMM for one class has been learnt, the next one can be incrementally learnt by simply training it and adding it to our set of GMMs in the classifier. This requires only the training data for the current class being learned and none of the training data for previous classes. Also, for the classifier to forget a class, it is as simple as removing a GMM from the classifier set.

4.4.4 *Classification Logic*

At inference time each GMM likelihood is calculated within a classifier and the class with the maximum likelihood score is selected as the classification.

4.5 Experiment Setup and Configuration

4.5.1 *Hardware and software*

All deep learning-based models were implemented using TensorFlow [68] Version 2.7.0. The code was written in Jupyter notebooks with Python Version 3.7.3. and CUDA version 11.2. All experiments conducted here were performed on a 64-bit Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz workstation with 64 CPU cores and 768GB RAM. NVIDIA® GeForce (driver version 495.44) with 4*GTX1080Ti each with 11GB RAM. Debian version 10.12 was used as the operating system. The scikit-learn and pycm [69] library were used for Metrics.

4.5.2 *Dataset*

The performance of our models and the benchmark model are evaluated on the public dataset Fashion-MNIST [70] which contains gray scale images of 10 clothing classes. The official training dataset for each class was split into, 80% creating a new training dataset for all classes of 48K and 20% creating a new validation dataset for all classes of 12K. 100% of the official test dataset (10K) was used for our test set. Within each dataset all classes contained the same number of images, where this changes in our experiments it is noted in that experiments section. The same dataset is used for 2 purposes:

1. Purpose 1 - Autoencoder Model Training : To train the Autoencoder model to create Visual Features. The premise is that the Autoencoder model will learn through unsupervised training on the largest image dataset possible, using vast computer power in a big data paradigm. That once carried out the resulting encoder/decoder will then be used on all vision tasks without alteration (frozen weights). However, for the experiments in this paper the dataset above is used to investigate if the classifier can learn unseen classes without that class having been used during the training of the Autoencoder.

2. Purpose 2 - GMM Training: To train the GMMs to create independent symbolic class representations. It is this dataset that is manipulated to investigate the impact on the classifier accuracy for; sample size, imbalanced classes and incrementally learnt class definitions.

To aid intuition in Figure 4.5 the feature space is visualised for the Fashion-MNIST dataset. Each image was encoded to a Feature Vector with 100 features and then using the t-SNE (t-distributed Stochastic Neighbour Embedding) method [71] has been projected to 2 visual features so that a 2D plot could be created. Note that no feature classes were involved in the training of our feature space. This results in a learnt representation of the most significant perceptual characteristics of images which are not biased by any symbolic feature classes. This enables the selected features to generalise well to future unseen classes. The plot also shows how well the training images have been split up in the feature space purely based on their visual features and the training dataset. Once an image has been encoded as a feature vector, the decoder can then be used on those feature vectors to reconstruct the original image.

4.5.3 *Evaluation Metrics*

Quantitative metrics of Accuracy score, weighted F1 score and Cohen Kappa are reported. The predictive accuracy metric measures the difference between the imputed values and their corresponding actual values. The weighted F1 score is used since class imbalance is investigated in an experiment. Since multi class classification is carried out, Cohen's kappa is used to measure the agreement between GMMs, a GMM classifies N images into C mutually exclusive classes.

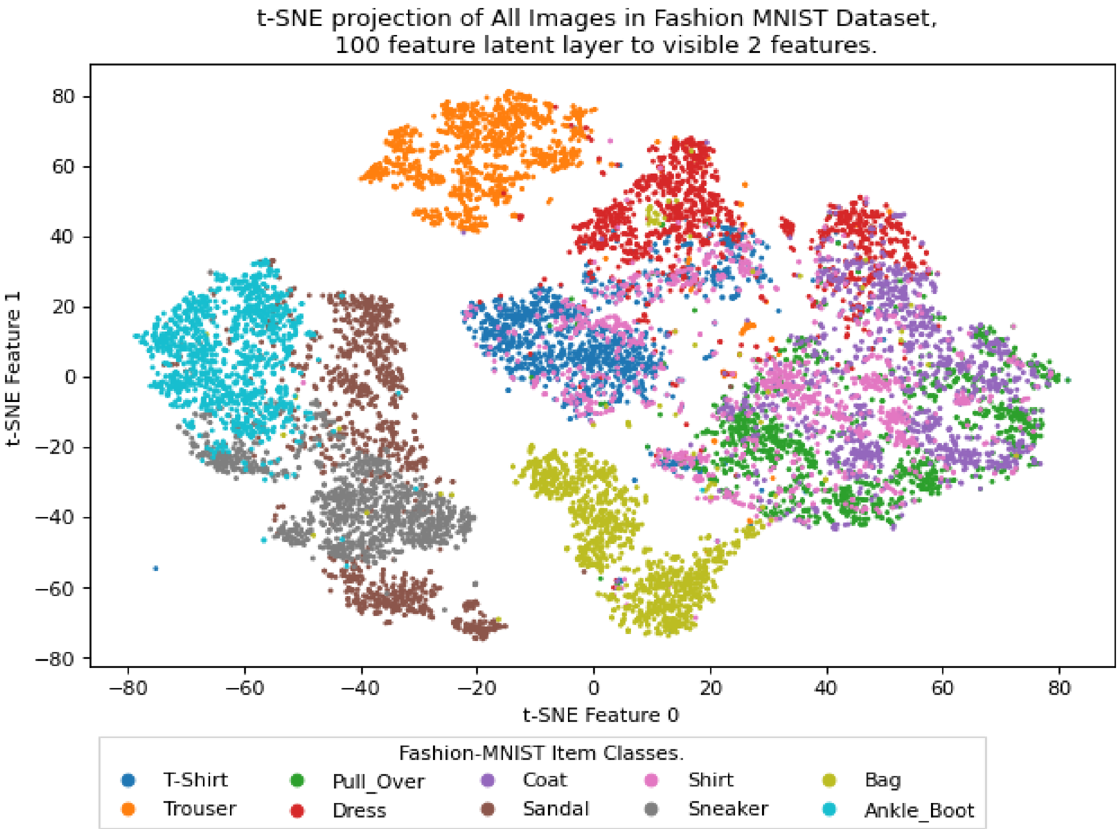


Figure 4.5: t-SNE Plot of Feature Space created through unsupervised Autoencoder training, on the full Fashion-MNIST dataset, coloured by ground truth class. Original feature dimensions = 100.

4.5.4 *Data Consistency*

When an experiment contains a suite of increasing or decreasing sample sizes, a dataset is managed to contain the same images as previously used to ensure experimental consistency. All reported test results are carried out using 100% of the held-out test dataset unless otherwise stated in an experiment.

4.5.5 *Model Setup*

Autoencoder

An Autoencoder is a type of artificial neural network used for unsupervised learning which is why it was initially selected. The Autoencoder compresses input data into a compact latent-space representation and then reconstructs the original data from this compressed form. It consists of two primary components: an encoder, which reduces the input into the latent space, and a decoder, which rebuilds the input from this compressed representation. During training, the network aims to minimise the difference between the input and its reconstruction, typically using a loss function like mean squared error. Autoencoders are handy for reducing dimensionality, detecting anomalies, and denoising data.

Experimenting using different filters and latent dimensions (bottleneck feature encoding) within the autoencoder structure as seen in Figure 4.6, the Autoencoder with the smallest final validation loss was selected. This was a convolutional neural net with 32 filters doubling to 64, with a latent dimension size of 100.

Trial	Model Number	Latent Dim	Final Val Loss
TR_01	CAE_32_32	10	0.276240587
TR_02	CAE_32_32	20	0.266255856
TR_03	CAE_32_32	30	0.261808693
TR_04	CAE_32_32	40	0.259342432
TR_05	CAE_32_32	50	0.257486165
TR_06	CAE_32_32	60	0.256188899
TR_07	CAE_32_32	70	0.255217791
TR_08	CAE_32_32	80	0.254425168
TR_09	CAE_32_32	90	0.253768295
TR_10	CAE_32_32	100	0.253221393
TR_11	CAE_32_4	10	0.282714337
TR_12	CAE_32_4	20	0.272032827
TR_13	CAE_32_4	30	0.267290175
TR_14	CAE_32_4	40	0.263914436
TR_15	CAE_32_4	50	0.262030035
TR_16	CAE_32_4	60	0.261395276
TR_17	CAE_32_4	70	0.25948891
TR_18	CAE_32_4	80	0.259308457
TR_19	CAE_32_4	90	0.258406371
TR_20	CAE_32_4	100	0.258716136

Trial	Model Number	Latent Dim	Final Val Loss
TR_21	CAE_32_64	10	0.274017751
TR_22	CAE_32_64	20	0.264336407
TR_23	CAE_32_64	30	0.260368437
TR_24	CAE_32_64	40	0.258074373
TR_25	CAE_32_64	50	0.256462365
TR_26	CAE_32_64	60	0.255080938
TR_27	CAE_32_64	70	0.25405404
TR_28	CAE_32_64	80	0.253075391
TR_29	CAE_32_64	90	0.252287656
TR_30	CAE_32_64	100	0.251611203
TR_31	CAE_32_64_128	10	0.273938924
TR_32	CAE_32_64_128	20	0.265063405
TR_33	CAE_32_64_128	30	0.261125028
TR_34	CAE_32_64_128	40	0.258871257
TR_35	CAE_32_64_128	50	0.257258713
TR_36	CAE_32_64_128	60	0.256006032
TR_37	CAE_32_64_128	70	0.254885674
TR_38	CAE_32_64_128	80	0.254182458
TR_39	CAE_32_64_128	90	0.253479779
TR_40	CAE_32_64_128	100	0.252678126

Figure 4.6: Autoencoder structure exploration.

The full structure for the used Autoencoder can be seen in Figure 4.7, it has two convolution layers with 3×3 filters (activated by ReLU activations [72]), and applied with a stride of 2 while maintaining the same size image through padding. The number of filters doubles from one layer to the next, starting with 32. The output of the last convolution layer just before the bottleneck, is flattened and then passed through a customisable dense layer, which creates our 100-dimensional feature space. A feature representation was required that can be decoded into a good image representation whilst still compressing the data to enable easier manipulation. The decoder mirrors the encoder, using convolutional transpose operators [73]. If you take a feature encoding and apply the matching decoder to the encoder used to create that feature encoding, it will transform it back into an approximation of the original image.

To evidence this we generate 400 Samples from each class GMM Model, then apply the decoder model from the Autoencoder. The first 10 reconstructed images for each class are shown in Figure 4.8.

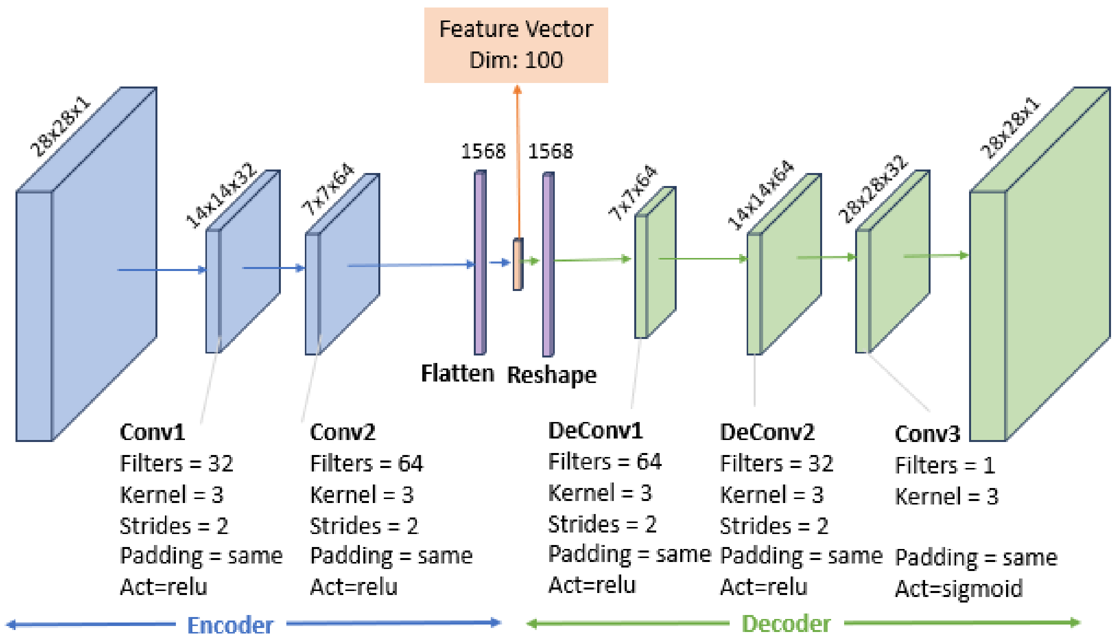


Figure 4.7: The Autoencoder, used to create a 100 Feature Vector, carried out with unsupervised training. Also, the decoder which transforms a Feature Vector back into an approximate image.

The autoencoder uses Adam [74] for optimisation. The learning rate is reduced according to a cosine function [75]. Using a search approach as seen in Figure 4.9 the following hyper-parameters were selected, a base learning rate of 0.003, and a final learning rate of 0.001, a maximum number of 20 updates, 5 warm up steps and trained with 40 epochs. The batch size was set to 50. Training was carried out using unannotated images using a binary cross entropy loss.

Figure 4.8: Reconstructing an approximate image from a feature encoding using the decoder from the Autoencoder.

Trial	Model Name	Loss	Base lr	Final lr	Val Loss
TR_01	CAE_32_64	MeanSquaredError	0.03	0.01	0.49477616
TR_02	CAE_32_64	MeanSquaredError	0.003	0.001	0.251524568
TR_03	CAE_32_64	MeanSquaredError	0.0003	0.0001	0.253289372
TR_04	CAE_32_64	BinaryCrossentropy	0.03	0.01	0.493631005
TR_05	CAE_32_64	BinaryCrossentropy	0.003	0.001	0.251493037
TR_06	CAE_32_64	BinaryCrossentropy	0.0003	0.0001	0.25348869

Figure 4.9: Autoencoder hyperparameter selection.

Gaussian Mixture Model

A GMM is a probabilistic model. Each GMM is trained on a dataset of images that represents a class. The Maximum Likelihood Estimation (MLE) for normal mixtures and Estimation Maximisation (EM) algorithm [76] (See Section 4.4.2) are used after setting the training hyperparameters. The GMM is initialised using K-Means centroids for the first Estimation step. Using a grid search candidate GMMs are created from which the model with the lowest Bayesian Information Criterion (BIC) validation score (to prevent over-fitting) is selected to represent that class.

A GMM was created for each symbolic class using encoded training images (100% dataset unless otherwise stated in each experiment). Each class GMM is comprised of a number of components each with a mean and co-variance. The diagonal of the co-variance gives the variance of the component and the rest of the matrix describes the relationship between each of the features dependent on the component type. To select a GMM for a class all the combinations of hyperparameters were evaluated. The hyperparameters evaluated were; i) Number of mixture components: 1 to 5 inclusive, ii) Covariance type: Tied, Diagonal, Spherical & Full, and, iii) Non-negative regularisation: 1.0e-2, 1.0e-3, 1.0e-4 and 1.0e-5. This resulted in 80 potential models per class. A total of 800 for the 10 class dataset. During GMM creation, occasionally when regularisation was low the *MLE* for normal mixtures did not converge due to singularities or degeneracy. Any model that did not converge was automatically eliminated from our potential selection. The selected GMM had the lowest validation *Bayesian Information Criterion (BIC)* score as seen in Table 4.1.

Figure 4.10 shows a two-component GMM fitting a bivariate distribution, complete with probability density distributions, for the *Ankle Boot* Class in our dataset. The main window presents the feature space comprised of Visual Features 1 & 2.

Every training image is represented in terms of these two features called a *Feature Encoding*, marked on the figure as crosses. The GMM is then trained to find the best number of components to use, by finding the lowest model BIC score. In Figure 4.10, each component is shown as an ellipse with a star located at its mean point. Two further windows, one for each dimensional feature, shows the probability density distributions belonging to each component.

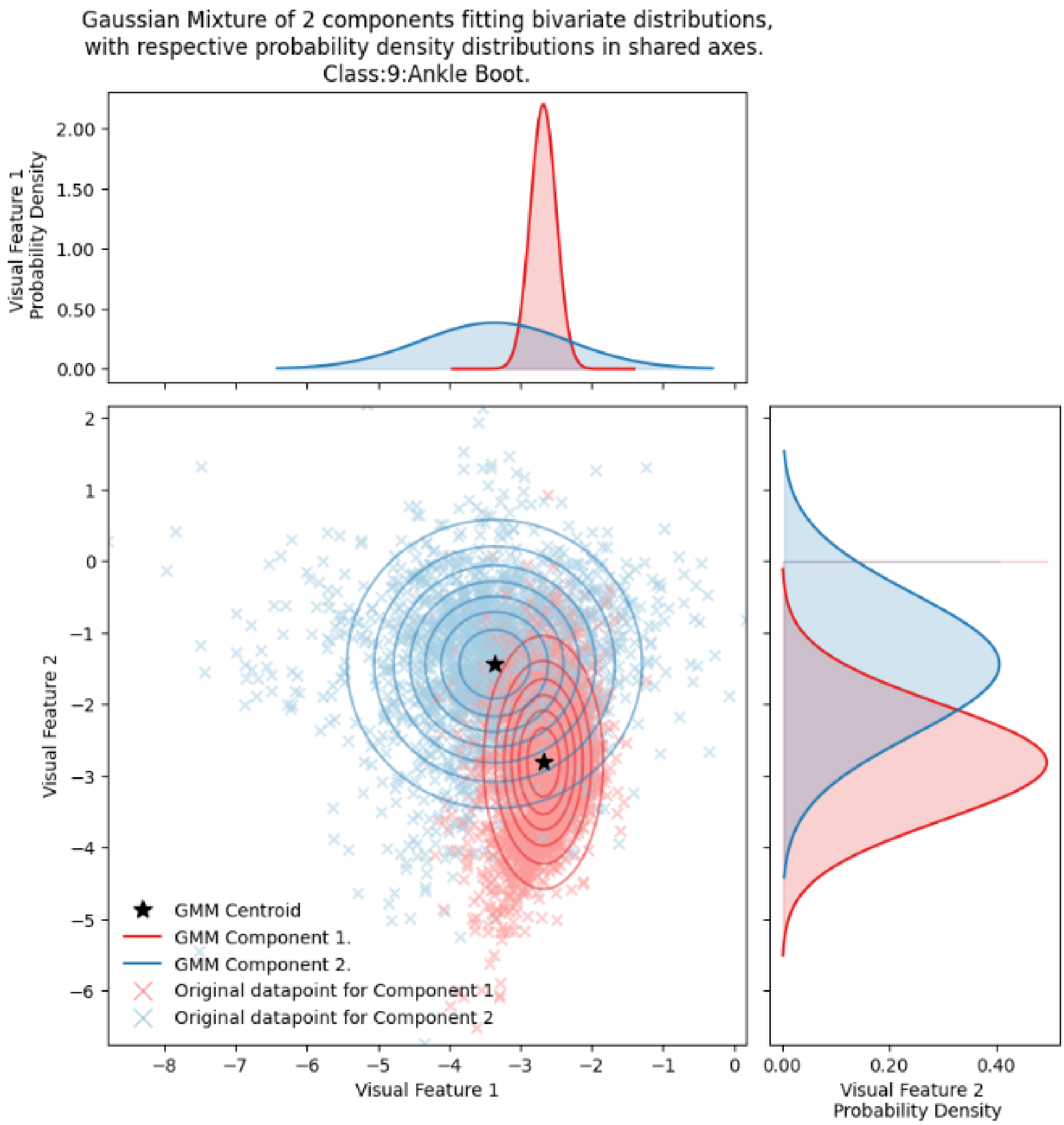


Figure 4.10: Gaussian Mixture Model with 2 components fitting bivariate distribution, with respective probability density distributions for the Ankle Boot Class.

GMM-IL Classifier

In creating a classifier with an ensemble of Gaussian Mixture Models (GMMs), the choice of hyperparameters directly influences the model's effectiveness in identifying specific patterns in the data. This approach combines several GMMs, each configured with specific hyperparameters. These hyperparameters are critical in customising each GMM within the ensemble, ensuring the classifier not only accurately categorises data but also generalises well to new, unseen data. The hyperparameters used in the experiments can be found in Table 4.1.

Class	**Reg CoVar**	**Comp. Num**	**Comp. Shape**	**BIC**
0: T-Shirts	0.001	2	full	218599.8304
1: Trousers	0.00001	2	full	92074.9725
2: Pull Over	0.001	5	tied	215139.5721
3: Dress	0.01	5	tied	218177.7049
4: Coat	0.001	3	tied	209474.0943
5: Sandal	0.0001	2	full	271939.6885
6: Shirt	0.01	5	tied	225819.9335
7: Sneaker	0.001	2	tied	166274.3903
8: Bag	0.01	5	tied	293328.9243
9: Ankle Boot	0.01	3	tied	226256.1887

Table 4.1: Baseline GMM hyperparameters for a set of GMMs. Reg CoVar- Regularising CoVariance, Comp Num-Component Number, Component Shape- (full=own shape, tied=same shape

In order to help with the intuition of a classifier comprised of a set of GMMs used for classification, using principal component analysis (t-SNE) 10 GMMs have been built with a 2 feature encoding. This then enables the creation of a 2D visualisation as shown in Figure 4.11. This map shows where individual GMM component distributions are, in the form of their GMM component mean values (stars) and co-variance contours (ellipses). Creating a grid across the map we generated 2500 feature encodings (Vector = [features 1,feature 2]) covering the map. Likelihood scores were then calculated for the 10 GMMs and the class with the highest probability of the image belonging to it shown. Only ellipses belonging to seven of the ten GMM classes are shown to aid visibility. Visually similar classes will have similar GMMs, leading to reduced discriminatory power.

In the context of Gaussian Mixture Models (GMMs), each component of the mixture can be seen as a Gaussian probability distribution when comparing two GMMs or assessing their overlap or the similarity between their components.

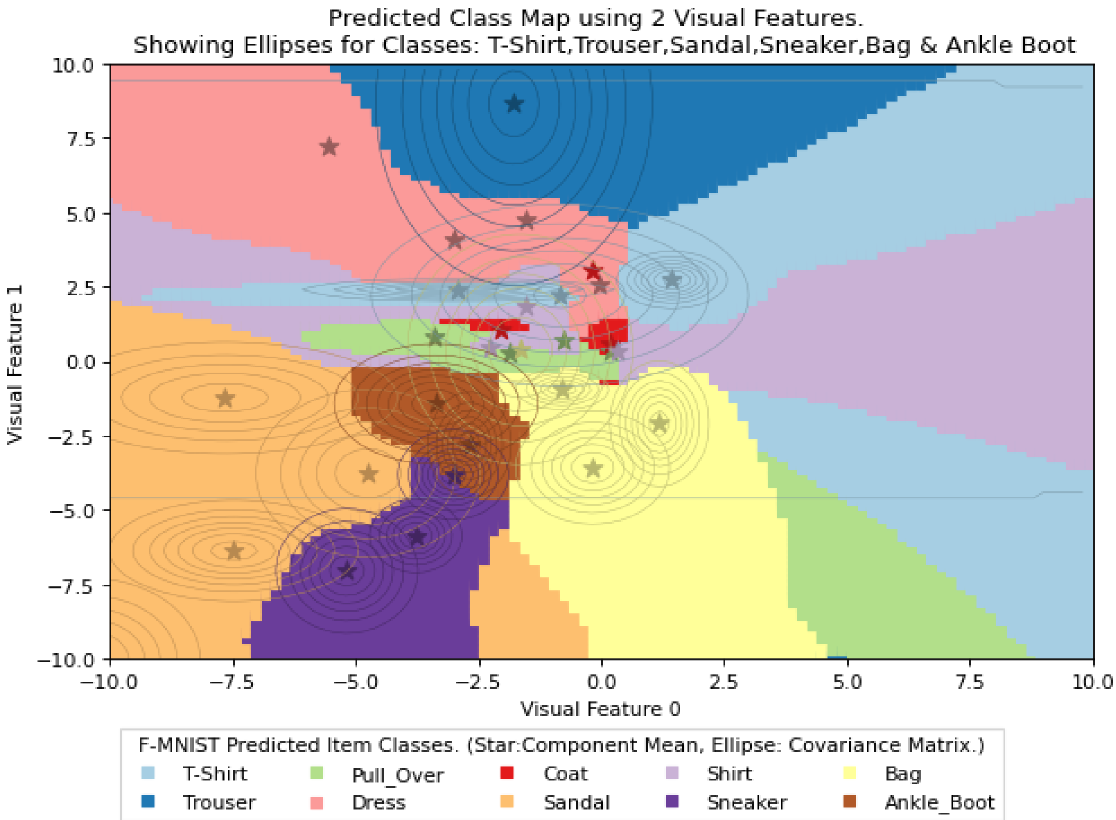


Figure 4.11: Map of the Predicted Classification using a 2 feature encoding, overlaid with GMM means (stars) and covariance contours (ellipses for just 7 classes to aid viewing) on the Fashion-MNIST dataset.

Figure 4.12 shows the calculated monte carlo distances between GMMs in the top matrix and the associated confusion matrix based on the training data for the same classifier below. It can be seen that where the GMMs are closest is where the most errors are made.

Monte-Carlo

		T-Shirt	Trousers	PullOver	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle Boot
T-Shirt	0	0.00	260.33	77.99	91.62	87.05	366.38	45.43	410.81	198.66	573.55
Trousers	1	436.30	0.00	209.51	148.07	178.55	1155.50	251.27	1764.67	1109.76	1519.70
PullOver	2	73.28	325.23	0.00	117.46	21.26	520.50	39.21	528.60	233.90	767.85
Dress	3	106.98	135.67	135.95	0.00	131.23	431.04	82.52	483.18	433.17	647.45
Coat	4	127.66	247.31	22.88	91.38	0.00	610.05	53.95	544.51	342.47	860.99
Sandal	5	598.44	386.93	925.65	327.86	660.36	0.00	585.59	65.73	569.33	61.81
Shirt	6	24.15	185.89	14.51	96.47	16.57	265.65	0.00	337.94	131.62	627.38
Sneaker	7	4339.03	3938.35	4685.83	5301.64	3531.20	472.04	4465.54	0.00	2262.14	473.99
Bag	8	107.97	301.83	128.65	133.96	116.08	133.48	78.68	117.07	0.00	305.00
Ankle Boot	9	1142.45	1159.02	1420.08	731.41	1151.21	147.54	951.88	122.38	901.61	0.00

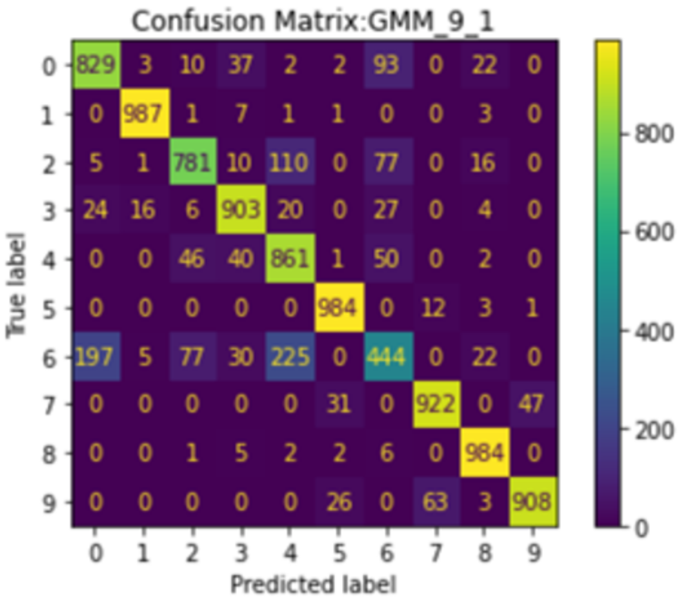


Figure 4.12: GMM Distances compared to a confusion matrix for GMM results.

There are many distribution distances for discrete distributions, such as Kullback-Leibler (KL) Divergence and Jensen-Shannon (JS) Divergence. However, KL divergence for GMMs has no analytically tractable formula and we need to resort to approximation methods. Cui et al[77] carry out a comparison of seven KL divergence approximation methods between two GMMs for satellite image retrieval. They conclude that, 'in principle, Monte Carlo method can achieve high accuracy provided a large number of samples are available'.

Jensen- Shannon Divergence (JSD) is a way to gauge how similar two probability distributions are. It's a smoothed and symmetric version of the Kullback-Leibler Divergence (KLD). JSD is calculated by taking the average of the KLD between each distribution and the average of the two distributions.

When you have two probability distributions P and Q, you can compute the Jensen-Shannon Divergence like this:

$$D_{JSD}(P||Q) = \frac{1}{2}D_{KL}(P||M) + \frac{1}{2}D_{KL}(Q||M) \tag{4.4}$$

For two discrete probability distributions, the KLD is given by: *Specifically, it measures the amount of information lost when M is used to approximate P or M is used to approximate Q.*

$$D_{KL}(P||Q) = \sum_{i} P(i). \log\left(\frac{P(i)}{Q(i)}\right) \tag{4.5}$$

where:

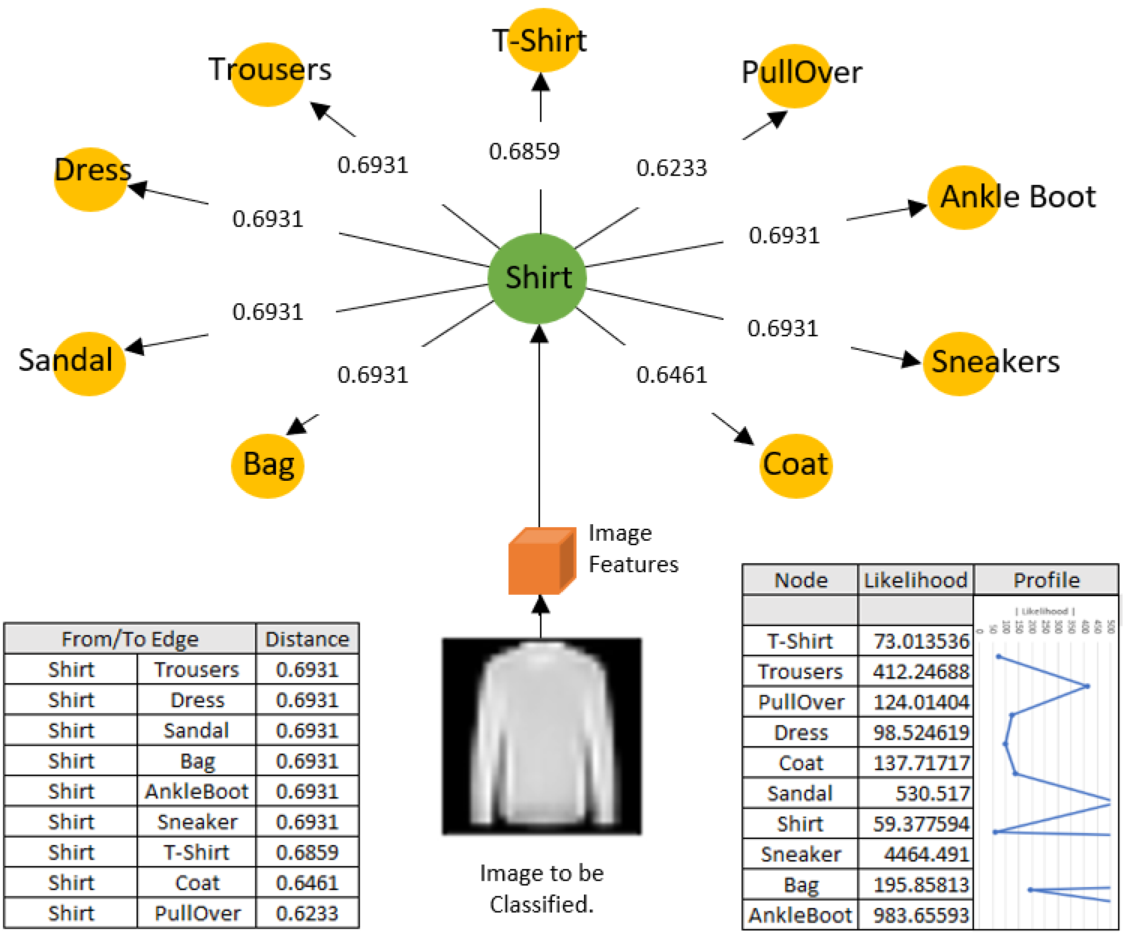
- $D_{KL}(P||Q)$ is the Kullback-Leibler Divergence from P to Q.
- $D_{KL}(P||M)$ is the Kullback-Leibler Divergence from P to M.
- $D_{KL}(Q||M)$ is the Kullback-Leibler Divergence from Q to M.
- $M is\ a\ mixed\ distribution = \frac{1}{2}(P+Q)$.

It's worth noting that $D_{KL}(P||Q)$ is not symmetric, meaning $D_{KL}(P||Q) \neq D_{KL}(Q||P)$.

A pairwise GMM distance can be calculated which gives an indication of how similar the GMM representations are, and hence the extent of the classifiers' discriminatory power. This was tested by calculating a pairwise distance matrix for the Fashion-MNIST dataset, using JSD.

The returned matrix was then correlated with the confusion matrix created using the training data at inference time. The resulting Spearman correlation coefficient was 0.78 which indicated that the similarity matrix could predict the level of classifier errors to some extent.

As shown in Figure 4.13, we can drill into why a specific image has been categorised into the wrong class. We can see that the image of the shirt is most likely to be categorised as a pullover since this GMM definition is the closest in distance. It is least likely to be categorised as a sneaker since the GMMs are furthest apart. In this way a visual encoding could be created to express an image in terms of other images. Similar to how Glove works for words.



From/To Edge		Distance
Shirt	Trousers	0.6931
Shirt	Dress	0.6931
Shirt	Sandal	0.6931
Shirt	Bag	0.6931
Shirt	AnkleBoot	0.6931
Shirt	Sneaker	0.6931
Shirt	T-Shirt	0.6859
Shirt	Coat	0.6461
Shirt	PullOver	0.6233

Node	Likelihood	Profile
T-Shirt	73.013536	
Trousers	412.24688	
PullOver	124.01404	
Dress	98.524619	
Coat	137.71717	
Sandal	530.517	
Shirt	59.377594	
Sneaker	4464.491	
Bag	195.85813	
AnkleBoot	983.65593	

Figure 4.13: Distances between GMMs and likelihood for particular image instance.

Benchmark Classifier (Softmax)

Our benchmark classifier (*Softmax*) is comprised of a deep learning network consisting of the same frozen encoder model, plus a dense layer with a Softmax activation. The hyperparameters were set to the same as described for the Autoencoder.

4.6 Results and Analysis

These experiments investigate the difference in performance between a multiple independent GMM heads (*GMMs*) and the benchmark method of a single Softmax head (*Softmax*). Both classifiers use the same encoder with frozen weights trained on ten classes for all experiments except Experiment 4.6.4 where it was trained on six classes. The first Experiment 4.6.1 establishes a reference baseline. The next experiment evaluates the classifiers accuracy when using; small sample sizes during training (Experiment 4.6.2) and when the sample size is imbalanced across classes (Experiment 4.6.3). Experiment 4.6.4 reports the classifiers results when incrementally learning pairwise unseen classes.

4.6.1 *Classifier Baseline*

See Table 4.2 for this classifier's accuracy results (Validation score was used to reduce overfitting).

The two classifiers were tested after building the models as described in Section 4.5.5. The results for training, validation and testing are shown in Table 4.2. *Softmax* outperforms *GMMs* when 100% of each dataset is used and all classes are balanced.

Data	**Classifier**	**Acc %**	**F1**	**CK**
Train	GMM	86.82	0.86	0.89
Valid	GMM	84.71	0.84	0.87
Test	GMM	85.57	0.85	0.87
Train	Softmax	97.97	0.98	0.98
Valid	Softmax	90.39	0.90	0.92
Test	Softmax	90.37	0.90	0.91

Table 4.2: Classifier Accuracy for balanced classes using 100% Training, Validation and Testing datasets. Acc: Accuracy, F1: Weighted F1 Score, CK: Cohen Kappa.

4.6.2 *Small Sample Sizes*

Focusing on small sample sizes a range of 5 to 20 (inclusive) samples are used. Stepping through each sample size all GMM models for both classifiers are retrained using the initial hyperparameter settings. As can be seen in Figure 4.14 the *GMMs* perform with a higher accuracy than the corresponding *Softmax* for sample sizes smaller than 12.

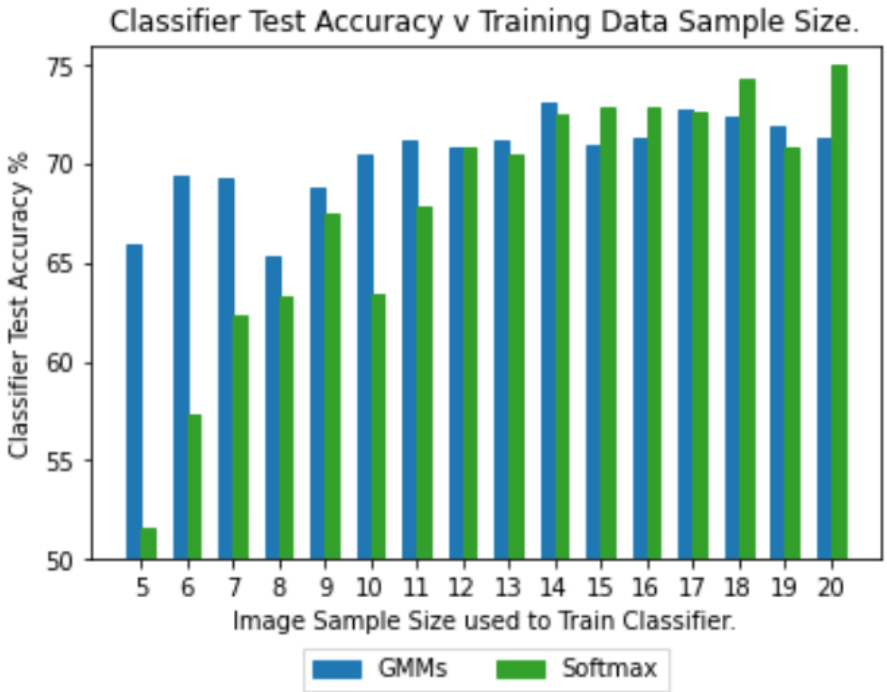


Figure 4.14: Classifier test accuracy for a training sample size of 5 to 20 inclusive.

4.6.3 *Imbalanced Classes*

As Krawczyk discusses [78], in classification problems, class imbalance occurs when there is a significant disparity in the number of samples for different classes. Typically, the minority classes are the ones of primary interest, either because they represent rare occurrences or because collecting data for these classes is expensive.

In this work three imbalanced ratio profiles are created and the classifiers weighted F1 Scores are reported. The range 5 to 15 is selected, as in the 'Small Sample Sizes' experiment it was shown to be a range of interest. Using 5 as low and 15 as high

3 imbalanced class datasets were created. Imbalances that were covered are; (1) Extreme ratio difference of 1 class high and 9 classes low. (2) A 50:50 ratio difference of 5 classes high, 5 classes low and, (3) A stepped profile, classes start at 5 samples and increment 1 sample until 14 samples.

Taking Exp_1 as an example in Figure 4.15 when training the classes, Trial 1 has 15 samples for C1 and 5 samples for all the others. In the next trial the samples are rotated once so now C2 has the 15 samples and the rest 5 samples. This continues 10 times, then the mean accuracy is taken from across all 10 trials for a class.

Classes	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
Trial 1	15	5	5	5	5	5	5	5	5	5
Trail 2	5	15	5	5	5	5	5	5	5	5
Trail 3	5	5	15	5	5	5	5	5	5	5
Trail 4	5	5	5	15	5	5	5	5	5	5
Trail 5	5	5	5	5	15	5	5	5	5	5
Trial 6	5	5	5	5	5	15	5	5	5	5
Trail 7	5	5	5	5	5	5	15	5	5	5
Trail 8	5	5	5	5	5	5	5	15	5	5
Trail 9	5	5	5	5	5	5	5	5	15	5
Trail 10	5	5	5	5	5	5	5	5	5	15

Figure 4.15: An example of class trials in Exp_1, where Sample sizes rotate through each class and then the mean for each class is calculated.

Both classifiers were retrained using the initial hyperparameters, GMM models where then fitted to new sample sizes based on the imbalanced experiment profile. Each Experiment was repeated 10 times with the class numbers rotating through the experiment profile. Figure 4.16 shows the mean accuracy and 95% confidence intervals per experiment and classifier type. Experiment 1, 2 and 3 had p values of 0.000, 0.001 and 0.018 respectively. In all three experiments the *GMM* outperformed the *Softmax* when trained on sample sizes under 15.

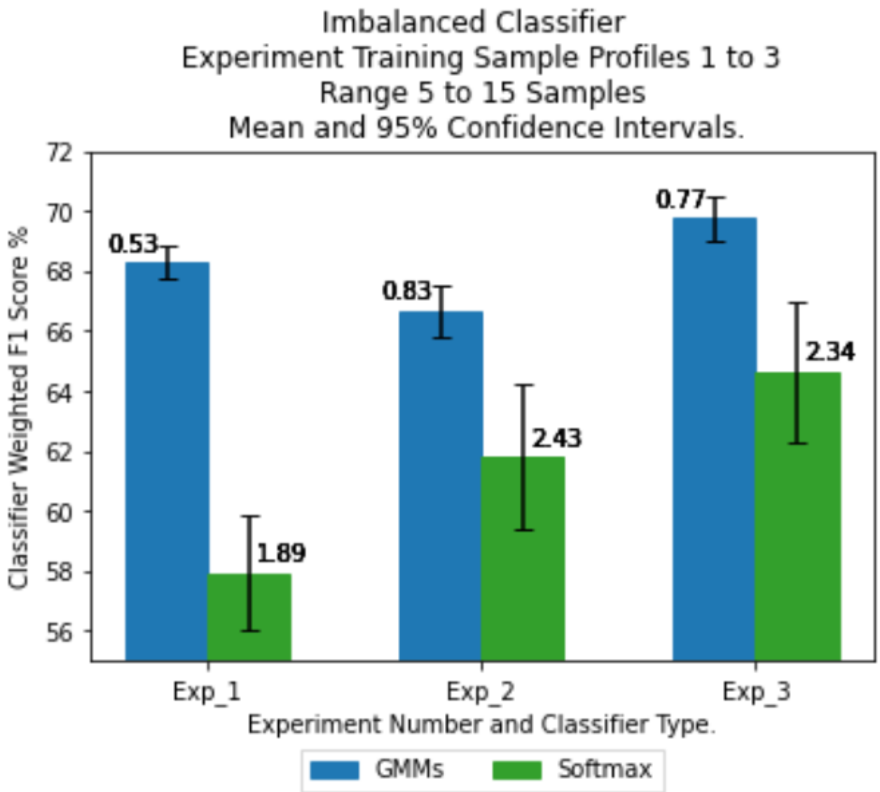


Figure 4.16: Classifier Weighted F1 Score for 3 Imbalanced Training Dataset Profiles. (Exp_1 : 1 class n15 & 9 classes n5), (Exp_2 : 5 classes n15 & 5 classes n15), (Exp_3 : Classes start at 5 samples and increment to 14 samples.), Classes rotated 10 times. Mean and 95% confidence intervals shown.

4.6.4 *Class Incremental Learning*

Softmax classifiers learn all classes at once using all the training data. They do not perform as accurately when they are required to learn classes over time and have no access to previous training data.

Following the benchmark method taken from Krichmar et al [79], and Kolouri et al [80] who used the Split MNIST dataset to learn consecutive pairs. For our dataset this is pairs of clothes e.g., Pair 1: T-Shirt, Trousers, Pair 2: PullOver, Dress, Pair 3: Coat, Sandal, Pair 4: Shirt, Sneaker, Pair 5: Bag, Ankle Boot. The following adjustments are then made, combining the first 3 pairs, which makes 6 classes trained in Task 1. The Autoencoder model is trained first using these 6 classes and the resulting encoder is frozen. Then, the 6 GMM Models are trained using their encoded images. This frozen encoder is then used for further tasks with just the GMM Models been trained. Pair 4 are used for Task 2 and Pair 5 used for Task 3. The reason Tasks contain 2 classes is to enable the *Softmax* to classify without having access to prior training data. For clarification the 3 Tasks were configured as follows:

1. Task_1 established the accuracy when the encoder was trained on 6 classes, the classifier heads (*GMM* and *Softmax*) were tested on 6 classes using 100% datasets. The classification was assigned to the class with the greatest probability/likelihood.

2. Task_2 established the accuracy when the classifiers were trained as per Task_1 with 2 further classes, the classifier heads (*GMM* and *Softmax*) were tested on 8 classes using 100% datasets. The classification was assigned to the class with the greatest probability/likelihood.

3. Task_3 established the accuracy when the classifiers were trained as per Task_2 with 2 further classes, the classifier heads (*GMM* and *Softmax*) were tested on all 10 classes using 100% datasets. The classification was assigned to the class with the greatest probability/likelihood.

Task_1, Task_2 and Task_3 were repeated 10 times as the classes were rotated, and the mean and 95% confidence values were calculated across all 10 combinations per classifier type.

From the results shown in Figure 4.17 it can be seen that initially the *Softmax* is more accurate than the *GMMs*. However, after each Incremental Task, the *Softmax* accuracy decreases significantly more than the *GMM*. This shows the *GMMs* have a greater ability to retain class definitions than the *Softmax*.

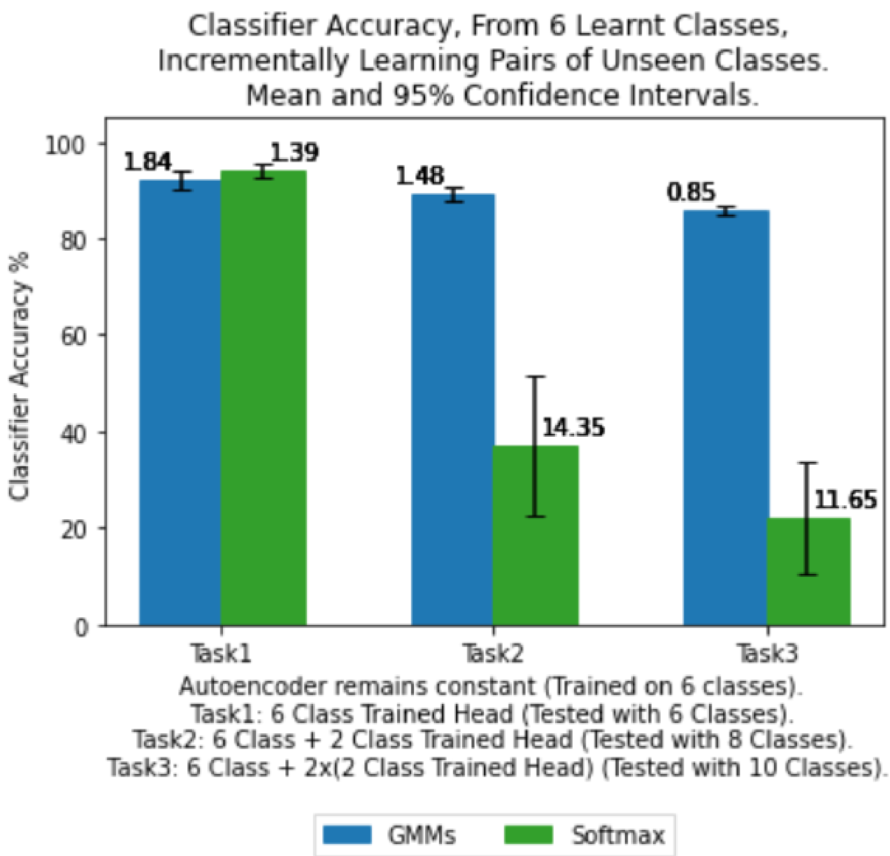


Figure 4.17: Classifiers Incrementally Learning Three Tasks.

4.7 Discussion

An architecture was created that enables transferred visual learning and the incremental addition of class definitions in the form of GMM models. The visual learning carried out used a smaller sample size than would ultimately be used to enable us to control the content of the visual features and verify that unseen classes could be learnt.
This classifier's accuracy could be improved by making the following amendments.

- Autoencoder - Our main premise is that this model's accuracy is dependent on the quality of the feature space created by the autoencoder, by using a state of the art autoencoder the granularity and quality of the visual features will be improved and hence the discriminatory power of the classifier increased.

- Gaussian Mixture Models - Amendola et al [81] state that there is the possibility of more modes than means when Gaussians are combined. Further investigation needs to be carried out to optimise the accuracy of the GMM likelihood landscape for a set of GMMs.

5

Planning a Neuro-Symbolic Incremental Learner

5.1 Introduction

The GMM-IL model simplifies class learning by focusing solely on the implicit question, 'What's in the image?' and removing the ability to ask varied questions. This chapter presents the intermediate research that informed the decisions taken for NS-IL model, detailed in Chapter 6. Unlike the GMM-IL, the NS-IL model reintroduces the capacity to handle multiple questions through the integration of the Image Classifier, KGs, and LLMs to allow users to pose diverse, complex questions and receive precise, contextually rich answers.

Figure 5.1 shows the different properties of the Neural Nets (NN) and Symbolic paradigms. NN use *induction* in the form of statistical methods to infer general rules. Symbolic AI uses *deduction* to understand and reason about a situation. Neuro-Symbolic Combiners integrate *abduction* and *induction* enabling two important functions, inductive learning to acquire accurate abductive theories and, abductive reasoning to inductively revise the existing KG to improve its accuracy. Architectures that contain Neuro-Symbolic Combiners are termed as *Neuro-Symbolic* Hybrid Systems.

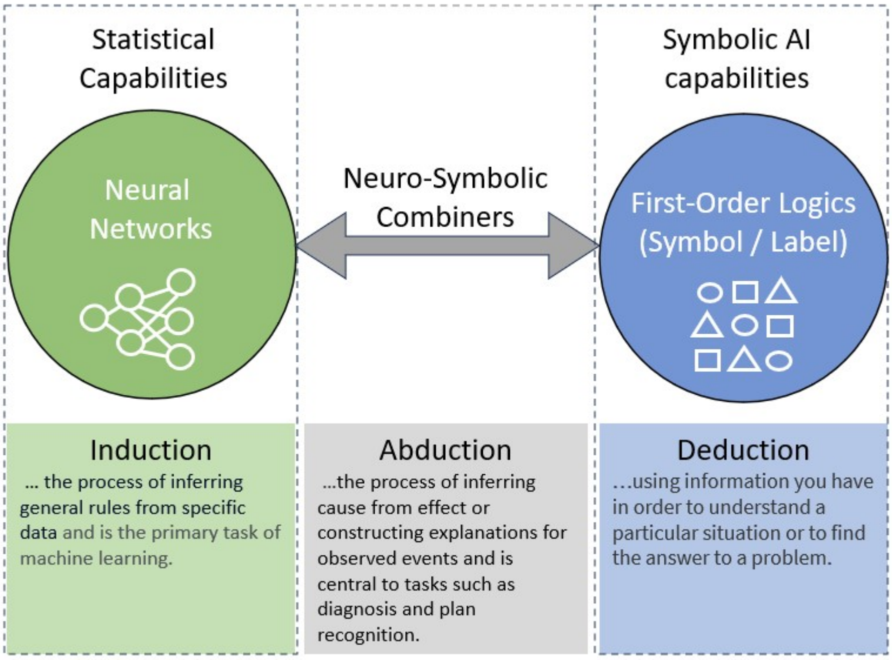


Figure 5.1: Combining Neural Networks and Symbolic Paradigms.

5.2 Addition of a Knowledge Graph

The Gaussian Mixture Model (GMM) excels in clustering and classifying visual features, while the knowledge graph excels in mapping out relationships and structured data. The GMM offers a probabilistic analysis of visual elements, and the knowledge graph embeds this analysis in a network of interrelated concepts. To effectively store and query image classification data for Regions of Interest (ROIs), we've adopted a graph database known for handling complex, interconnected relationships between nodes. This approach not only allows for intricate and nuanced queries but also supports flexible schema evolution, a key component for incremental learning.

In the GMM-IL framework, we created a classifier for each category, each classifier being a composition of various Gaussian Mixture Models (GMMs). Now, to enrich the model's capability to understand and process questions about the stored images. KGs add a layer of structured knowledge and reasoning. The data flow and interaction mechanism for this enhanced model are depicted in Figure 5.2.

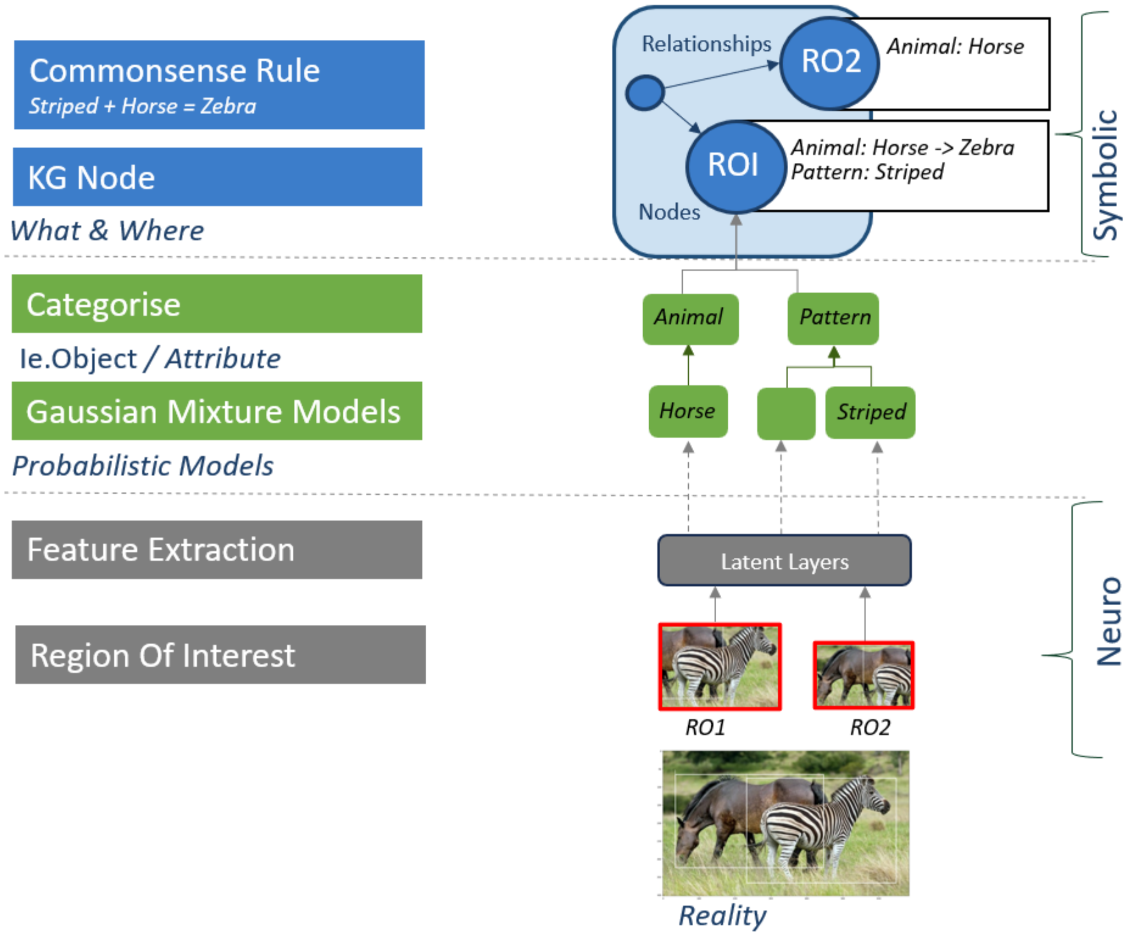


Figure 5.2: GMM-IL integrated to a KG.

To illustrate the capabilities of the system, consider a scenario with two categories: Animal and Pattern. Each category is defined by numerous Gaussian Mixture Models (GMMs). Suppose an ROI Node is assigned both categories. We establish a commonsense rule: If the Animal category identifies a horse and the Pattern category recognises stripes, the system updates the identified Animal to 'Zebra'. In this way the system infers a Zebra even though it has never been trained on Zebra data, demonstrating the application of Neuro-Symbolic data, this data is shown within Figure 5.2.

5.2.1 *The ROI Node*

All models are simplifications; they're useful but not fully representative of reality. This concept is applied in our research, where we draw boundaries known as boundary boxes around objects, assign them classes, and treat them as stable 'Regions of

Interest' nodes. However, it's crucial to remember that these models, while helpful for analysis and understanding, are simplifications and don't capture the full complexity and dynamic nature of the real world. The *binding problem* describes how we select and integrate distinct features into the *right combinations*, referring to the cognitive process of identifying and combining various visual features from the environment, such as shapes, colours, and textures, to recognise meaningful concepts. Figure 5.3, illustrates how reality has been abstracted into the system entities to *represent* human cognition.

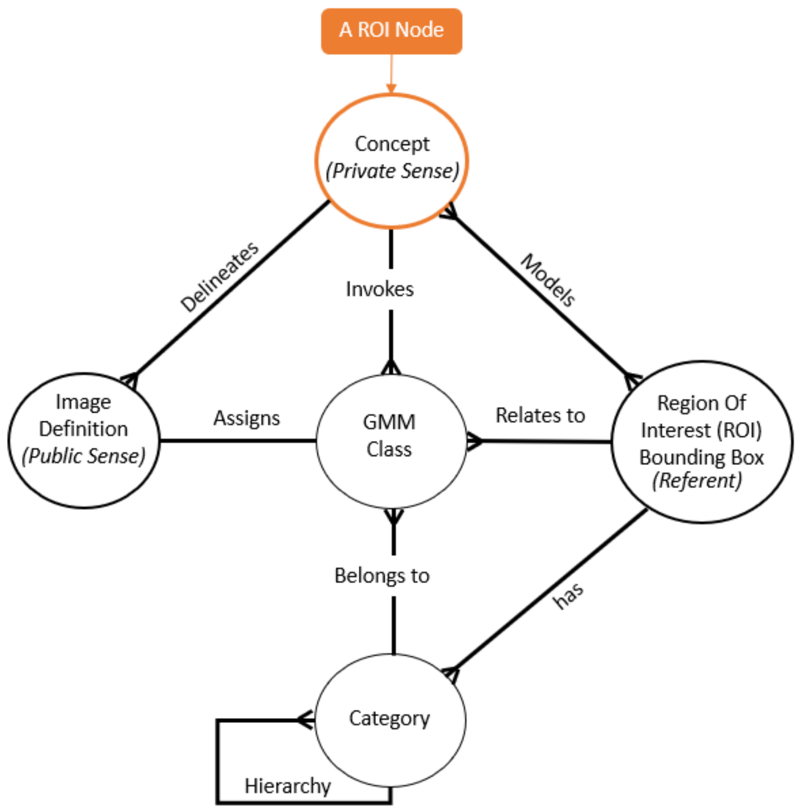


Figure 5.3: The ROI Node.

The aim of the thesis is to encode *visually grounded* concepts. Figure 5.3 shows the schema for how this is achieved for a concept modeled as a *Region Of Interest KG Node*. A concept is an internal mental construct created by a human. It is formed through many observations, here we show the visual aspects as bounded Regions Of Interest. Also, a bounded Region Of Interest can be associated with many concepts. The concept has many GMM classes associated with it, selected at inference time through the use of categories. The GMM Classes have a number of training images associated with them and hence a concept is delineated by those images. The Concept remains private to an individual, whilst the definition held in images, reflects a common consensus of what is meant by the GMM Class. The GMM-IL classes were each trained using 20 representative images. An ROI has many associated categories and a category has many constitute GMM classes.

Implementing our schema in the NS-IL system is a strategic decision that offers numerous benefits. It simplifies the processing of complex information by following an Ontological structure of *Categories* and *Terms*, breaking it down for easier interpretation and quicker decision-making. This structure enhances the accuracy and efficient processing of visual information. The configuration of the mechanism means it can evolve, accommodating new types of sensory data such as, 'sound' to expand the utility of the system. The data that is held, provides traceable and explainable results through clear visual definitions which is crucial for transparency and accountability, especially in sectors where these factors are mandatory. Using our schema for a KG Node ensures that responses are grounded consistently, which improves user interaction and satisfaction. The schema enables an enhancement of the 'Term' to define the next level of associated sub-symbolic data.

5.3 Integrating GMMs with a KG

With a view to benchmarking our NS-IL against the NS-VQA paper [38] these steps are carried out to process a scene image:

1. Train the following GMM classes using the GMM-IL Autoencoder and group them into categories to form a classifier, as shown in Figure 5.4.

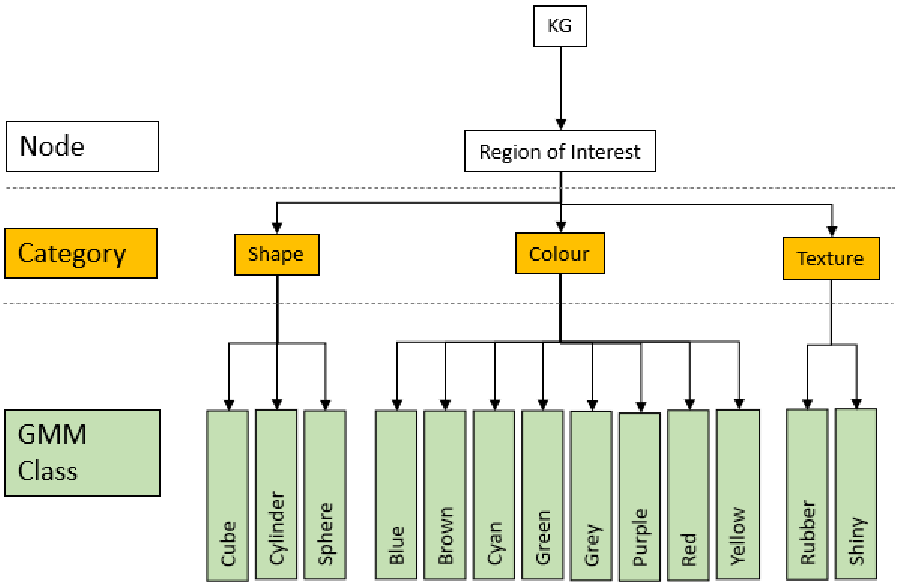


Figure 5.4: The Ontology for a ROI.

2. Initially a neo4j database is created on the server after which the following code is run to connect to it, a graph object is created which is used in future interactions. Initialisation is carried out by first deleting all previous data, this could be changed if we wanted to process more than one image at a time.

```
from py2neo import Graph
graph = Graph("neo4j://<ServerPath>:7687", auth=("<username>", "<
    password>"))
graph.delete_all()
```

3. The nodes are created for the generic concepts. This shows an example of a shiny class.

```
Attr_Shiny= Node("AttributePrototype", name='shiny')
graph.create(Attr_Shiny)
```

4. A scene node is created.

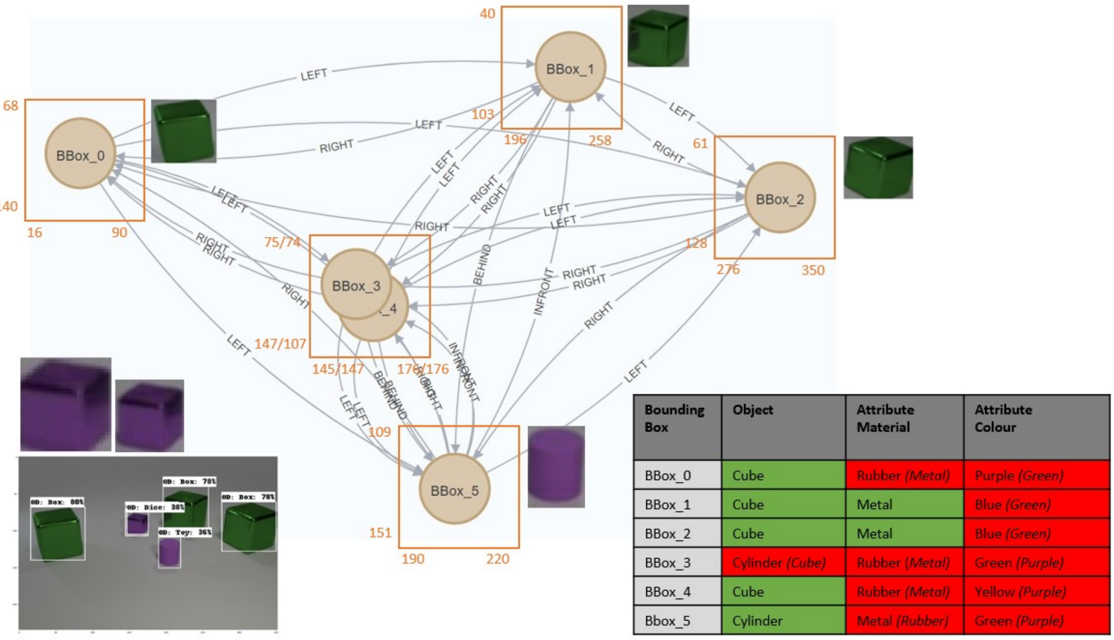
```
New_Scene_Name = "Scene"
#New_Scene_Name = "Scene_" + str(New_Number) #for more than 1 scene
New_Scene_Name_Str = "'" + New_Scene_Name + "'"
Scene_Image_Filename = "'" + str(Filename) + "'"
New_Scene_Name_Hand = Node("SceneFile", name=New_Scene_Name, File=
    Scene_Image_Filename)
graph.create(New_Scene_Name_Hand)
```

5. Using an object detector (Inception_Resnet) bounding boxes are identified and listed. A new ROI node is created for each bounding box on the list, logging the time and location within the scene to which it is linked.

```
New_BBox_Name = "ROI_" + str(New_Number)
New_BBox_Name_Str = "'" + New_BBox_Name + "'"
Timestamp = datetime.datetime.now()
Time_Now_Str = "'" + str(Timestamp) + "'"
New_BBox_Name_Hand = Node("ROI", name=New_BBox_Name, Object_Name
    = "Unknown",xmin= str(xmin),xmax= str(xmax),ymin= str(ymin),
    ymax= str(ymax))
graph.create(New_BBox_Name_Hand)
```

6. The likelihood for each GMM is calculated for each ROI node. The GMM offering the highest likelihood within a category assigns its classification to that category. This classification gets recorded within the ROI Node. Additionally, the likelihood value is kept in the edge between the ROI node and the generic class. This setup allows every class instance to access its related metadata through the corresponding generic class, so data only has to be stored once.

Figure 5.5 shows the results when the KG is queried. It shows two ROI Nodes; BBox_3 & BBox_4 for one object. this is a problem with the object detector which is not classed as a problem for our research. As long as our results match whatever is found in the list of bounding boxes we treat the results as correct. Therefore, it was checked against the received list of bounding boxes and the positioning information at the ROI nodes and relative information between the ROI nodes was found to be correct. Checking the ROI nodes it was found that the classification information was only 38.89% correct.



Bounding Box	Object	Attribute Material	Attribute Colour
BBox_0	Cube	Rubber *(Metal)*	Purple *(Green)*
BBox_1	Cube	Metal	Blue *(Green)*
BBox_2	Cube	Metal	Blue *(Green)*
BBox_3	Cylinder *(Cube)*	Rubber *(Metal)*	Green *(Purple)*
BBox_4	Cube	Rubber *(Metal)*	Yellow *(Purple)*
Bbox_5	Cylinder	Metal *(Rubber)*	Green *(Purple)*

Figure 5.5: Creating a Knowledge Graph for the CLEVR database.

The Autoencoder trained in an unsupervised way, failed to extract meaningful features for our specific image recognition task. Consequently, we switched to an Autoencoder trained on the ImageNet dataset, aligning it with scene images containing ImageNet classes. Despite this adjustment, the issue persisted. After several trial and error iterations, a functional system was found that used an image encoder (VGG19 model), trained on the ImageNet dataset using supervised learning. Section 5.3.1 delves deeper into the efforts to capture useful features for the feature encodings on a task. These challenges explain the selection and creation of the NS-IL dataset and what prompted **the shift from an Autoencoder used in the GMM-IL model to the VGG19 used in the NS-IL model**.

5.3.1 *Feature Encoding Discussion*

To visualise high-dimensional data, during the creation of the GMM-IL model we used t-SNE for dimensionality reduction, during this period the TensorBoard projector was also used, which supports both t-SNE and UMAP. Initially, t-SNE was chosen for its ability to reveal clusters within the data. However, t-SNE was computationally demanding and its results could vary due to its sensitivity to hyperparameters, complicating consistent analysis. For NS-IL it was decide to use UMAP which processed data faster, had reproducible outcomes and maintained both the local and global data structures, offering a fuller understanding of the data relationships.

The use of an image encoder ensures that raw visual data is transformed into a compact and useful feature space. This encoding becomes foundational as these features are then processed by the GMM. Various image encoders were experimented with, which seemed to indicate the following generalisations:

- Encoders such as VGG19, utilising ImageNet weights produced a multimodal space with useful structure to which GMM classes could be modeled. The feature space and category were aligned.
- VGG16 Autoencoder - unsupervised training produced a feature space which didn't capture enough 'useful' structure to differentiate between different GMM classes.
- Variational Autoencoder (VAE) - These are not useful since they create a unimodal space $\mathcal{N}(0,1)$ space, which means you can't use GMMs because the entire data manifold is one huge normal distribution and doesn't have the modes to support more than one gaussian for the GMM to fit to.
- Encoder from Segment anything [82], which employs a minimally adapted Vision Transformer (ViT). The features of a ViT consist of a grid of 16×16 with each grid "slot" having a 4096 dimensional vector, which when flattened gives you 1048576 features and would need billions of data points to fit. Used as an encoder it doesn't inherently structure the feature space in a way that would be conducive to distinguishing between different categories or classes when using GMM. There's nothing regularising the embedding space towards needing similar cluster structures.

Hence, we selected the VGG19 encoder trained on ImageNet as a category appropriate encoder to use in Chapter 6. The resulting feature space enabled the GMMs to have enough spread to differentiate between classes. The GMMs, being probabilistic, handles the inherent uncertainties and variations in visual data, ensuring robustness

in the system's visual understanding.
Every class has a definition shaped by a distribution of 'useful' features from a learned feature space. The essence of this system lies in identifying these 'useful' features in the feature space specific to each category of classes to be trained. For instance, in the context of the Object Category, 'useful' might refer to distinguishing characteristics like 'pointy,' 'corner,' or 'circle.' The challenge is to determine the optimal method for training the image encoder, whether through targeted images/classes or employing an Autoencoder.

5.4 Integrating KG with a Large Language Model

The addition of a Large Language Model (LLM) into our system puts the explicit questioning ability back into the model by using advanced natural language processing capabilities, enabling it to understand and generate human-like responses. This bridges the gap between the system's structured, symbolic knowledge base and the complexities of human language. LLMs, a recent innovation in AI, efficiently translate user queries into forms compatible with the Knowledge Graph (KG) and then convert the KG's outputs into intelligible answers. This provides our system with the last component to provide the functionality required for a Neuro-Symbolic Incremental Learner (NS-IL) Visual Question Answering (VQA) system, as shown in Figure 5.6.

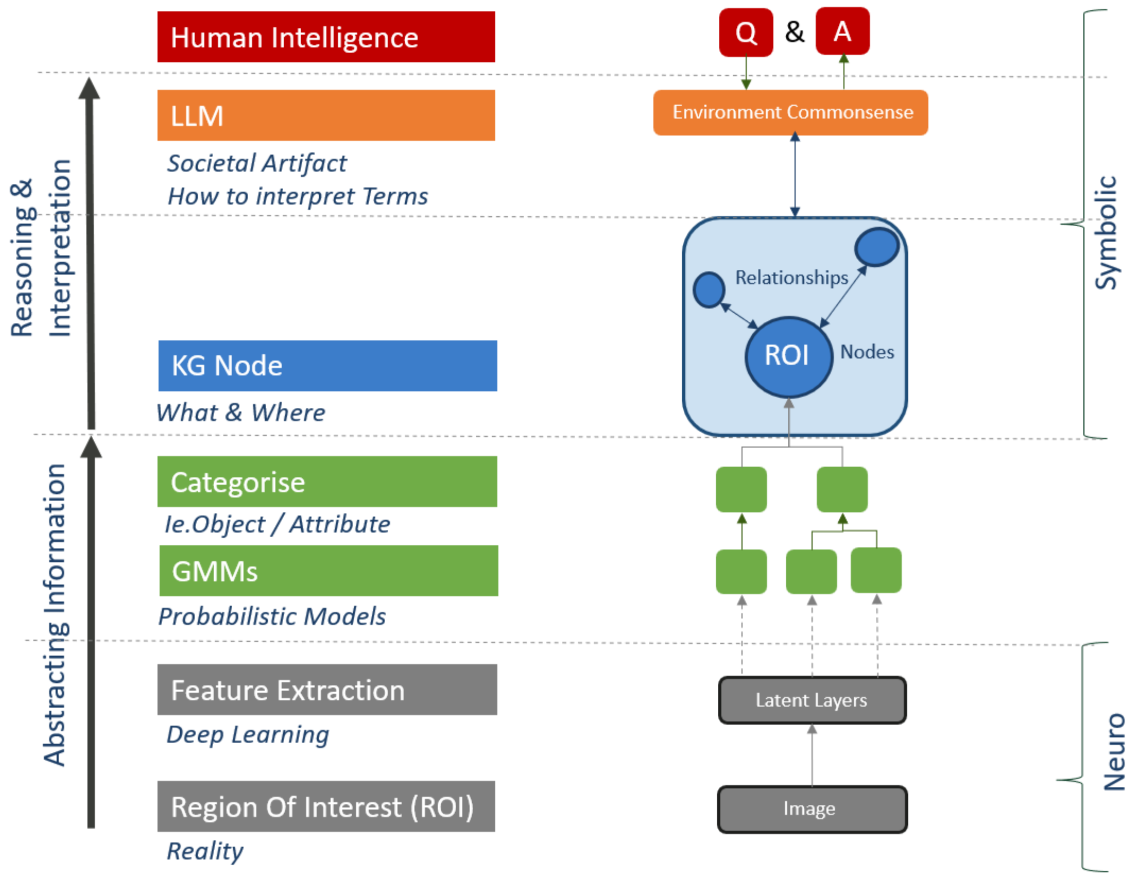


Figure 5.6: Proposed architecture for the NS-IL.

6

NS-IL: The Neuro-Symbolic Incremental Learner Model

6.1 Introduction

In the main introduction the VQA Task was outlined. In Figure 6.1 the three components are identified by the green, blue and orange areas. Each area is now expanded :

1. Image Classification System (*Green*): In Section 6.2, Steps 1, 2, 3 and 4 describe a system that can infer the class from a set of objects in an image.

2. Knowledge Graph (*Blue*): In Section 6.3, Step 5 describes the stored classes and relationships. Step 6 creates ROIs nodes from identified Classes and Step 7 adjusts those Nodes by applying Commonsense rules.

3. Human Curiosity & Large Language Models (*Orange*): In Section 6.4, translating human natural language questions into machine executable queries is explained in Steps 8 and 9. Step 10 translates the returned results back into a natural language answer.

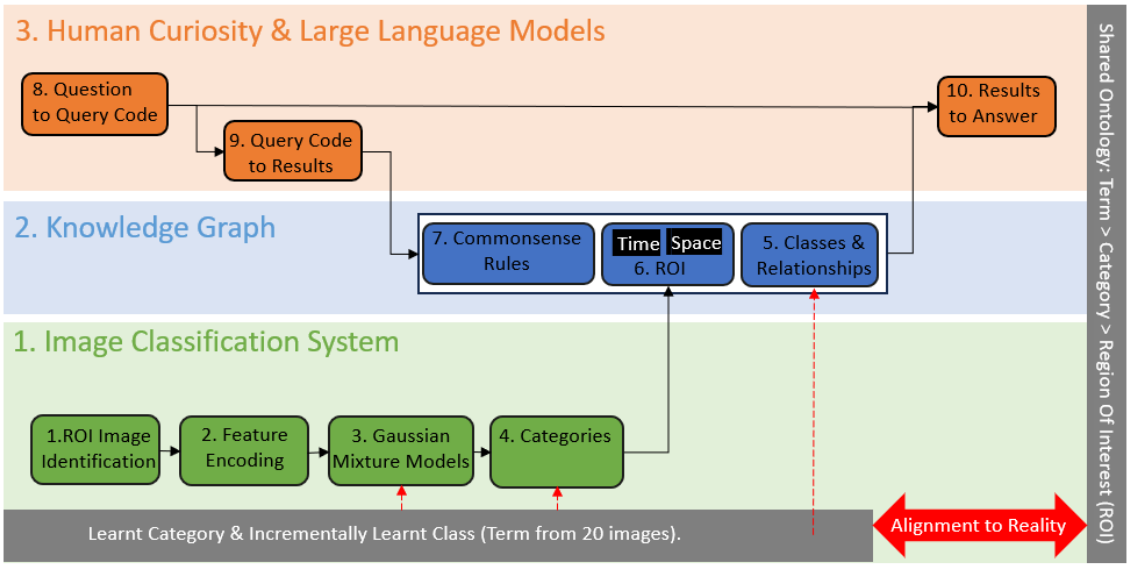


Figure 6.1: The 3 Components of the NS-VQA Task: 1 - Classification System, 2 - Knowledge Graph, & 3 - Question to Answer.

Each component and its associated steps will be covered in further detail in the next sections. Section 6.5 shows the individual model selections for the steps, together with their setups and configurations.

6.2 Methodology: Image Classification

Each step in the Image Classification system shown in Figure 6.1 is assigned a specific task. Step 1 identifies the ROI (bounding boxes) in the scene, Step 2 encodes each ROI image into a feature space prior to the addition of meaning in the form of a class (format category:class) through learnt categorical GMM Classes. Step 4 associates the category:class to the appropriate ROI node in the KG.

6.2.1 *Step 1: ROI Image Identification*

The scene image is what represents truth and enables us to ground the inferred *ROIs* with visual reality within the VQA task. The scene is split into several single-concept ROI images. Each bounding box marks a region from which category:class(es) will be classified. Additional context information is also retained for later use: the timestamp when the bounding box entry was created (useful for tracking objects in temporal activities if more than one image is shown); the position of the bounding box within the context image (helpful for generalisations over known object proximity's to enhance inference activities); and the file location of the scene image, providing the source for all the bounding box regions in the image. These additional details are used in Step 6 when the ROI Node is generated to optimise inference capabilities later.

6.2.2 *Step 2: Feature Encoding*

Each identified ROI bounding box is handled individually. The procedure involves capturing the pixel contents of the bounding box and encoding them for Step 3 of the pipeline. Converting the ROI image into a feature encoding offers benefits such as data compression, feature extraction, and noise reduction.

6.2.3 *Step 3: Gaussian Mixture Models*

A Gaussian Mixture Model was selected to define a class in terms of a set of features. A *GMM* approach was taken since features can have complex, multi-modal distributions rather than simple, single-mode distributions. This architectural choice mitigates three major weaknesses of neural nets: their susceptibility to 'catastrophic forgetting', the extended retraining duration required, and the substantial sample sizes needed to train a new class. With independent GMMs, our architecture permits the flexible addition or removal of classes, even with limited sample size.

6.2.4 *Step 4: Categories*

The feature encoding of the ROI image is fed into each of our pre-trained GMMs to generate a class specific likelihood. This likelihood signifies the class conditional probability given the feature encoding. The likelihoods are used in Step 6.

6.3 Methodology: Knowledge Graph

Within a *Context*, the *ROI Nodes* are more intricate and evolving than fixed *Classes*. An ROI Node embodies one or many Categories, each category containing many possible classes. For example for a ROI node representing a kettle, the visual information would return *Object:kettle* and *Attribute:Shiny*, taken together these categories of information build a richer definition for the 'kettle' Concept. This kettle concept can then be linked to other concepts such as; afforded an *Action* like 'pour water' and be located in a particular *Location* such as 'kitchen'. Helping individuals to understand how that Concept operates in reality.

6.3.1 *Step 5: Classes & Relationships*

In Step 5 the knowledge graph was initialised with classes and relationships. These represent the classes and relationships a human has learnt during their interaction with the environment and, whilst they may be adapted over time, generally they are stable and not subject to change.

6.3.2 *Step 6: ROIs*

The use of ROI nodes enables many visual classes of information to be held. The ROI node contains many categorical classifications by associating specific GMM classes with each category. This process allows us to calculate for a given classifier, the most probably 'seen' class. In Step 6, when the system identifies a new ROI in the given image, an individual node is generated for each ROI listed. The process flow for node and edge generation in the KG is shown in Figure 6.2. For each generated ROI node the ROI's coordinates are saved and the timestamp of when it was created. The object's class is ascertained, and the relevant information associated for that class, such as its usual room location and associated actions generate edges. Additionally, any visual attribute classes are also listed against that ROI.

6.3.3 *Step 7: Commonsense Rules*

After the ROI class information has been synthesised into an ROI Node in the knowledge graph, commonsense rules can be applied to the node information. For instance, consider a scenario where you're unfamiliar with a zebra but someone tells you that a zebra resembles a striped horse. In such a case, upon encountering a ROI with an object class of 'horse' together with an attribute class 'striped', you could amend the object class to 'Zebra' without ever seeing an actual zebra in reality.

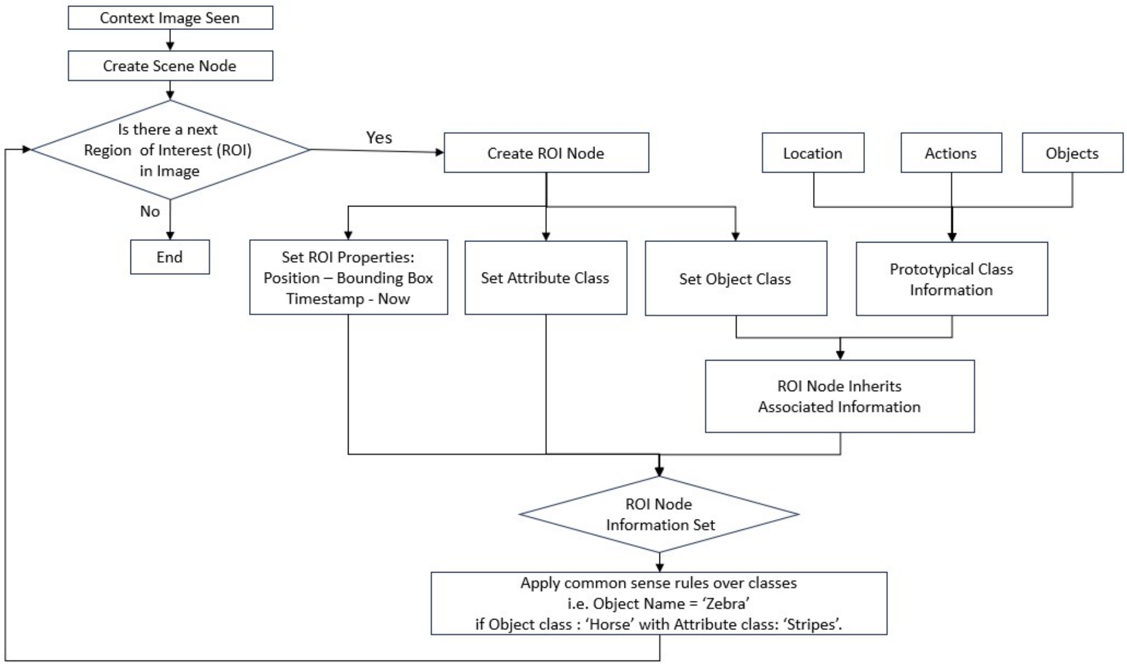


Figure 6.2: Knowledge Graph: Synthesising Grounding Information & Commonsense.

6.4 Methodology: Human Curiosity & Large Language Models

Large language models, such as the ChatGPT family, are advanced neural networks, specifically transformers, designed to understand and generate human-like text. They're trained on massive amounts of text data, enabling them to perform various natural language processing tasks, from basic grammar correction to complex tasks like answering questions and creative writing. The 'large' in their name signifies the immense number of parameters they possess, often in the billions. The LLM can produce coherent, contextually relevant, and almost human-like text.

Knowledge graphs (KG) and large language models (LLM), can complement each other effectively. KGs provide structured facts, domain-specific knowledge, and symbolic reasoning, while LLMs offer general language processing and broad knowledge.

Together, they create a capacity to comprehend and interact with the world.

The contents of the given image have been transformed into structured semantic data within the KG. Our focus now shifts from the vision domain to the language domain.

6.4.1 *Step 8: Question to Query Code*

The KG is now made useful through the direction of a human. The human forms a question with an associated intent. This is best shown by the difference between these two questions, *What are you doing ?* and *What are you trying to do ?* The first question will give an answer about the actions that are being carried out, the second will give an answer about why you are trying to do those actions. To a certain extent, it is the responsibility of the human to provide a clear question using terms and provide a context.

6.4.2 *Step 9: Query Code to Results*

This process turns complex language into machine-readable queries, the ideal being to solely use the terms held in the KG. Using the given question a prompt is automatically engineered to direct the LLM effectively. The prompt includes the question, the KG schema, and some programming language examples. We send this prompt to the LLM, and it replies with a graph query statement using terms from our KG. A LLM is used since they can consider context in their responses. Given a well-phrased context or preamble, they can produce answers that are relevant to the specific situation described, making interactions feel more intuitive for the user.

6.4.3 *Step 10: Results to Answer*

The results received from the KG are included in a new engineered prompt, which also incorporates the initial question together with instructions. These elements are sent to the LLM which returns a contextualised Answer that the user can understand.

6.5 Experiment Setup and Configuration

This section describes the selection, configuration and training processes for each model employed throughout the different steps of our pipeline. To provide clarity at the boundaries of the steps, the step inputs and outputs are documented in each subsection, which also underscores the responsibilities of each model.

6.5.1 *Hardware and software*

All deep learning-based models were implemented using TensorFlow [68] Version 2.13.0. The code was written in Python Version 3.9.2, Neo4j Version 5.9.0 and CUDA Version 11.8. All experiments conducted here were performed on a 64GB Intel i9-9900K workstation with 16 CPU cores at 3.6GHz and an NVIDIA GeForce RTX 4090 with 24.5GB RAM using NVIDIA driver version 535.54.03. Debian version 11.7 was used as the operating system.

6.5.2 *Image Classification System*

Step 1: Regions of Interest Identification

Input: Context Image.
Output: A list of ROI (bounding boxes) in the image.
Our model takes single concept bounding boxes as input. We generate this input list using an object detector model, specifically a pre-trained Inception-ResNet-v2, which is a variant of Faster R-CNN with ResNet V2. This model is available at: `https://tfhub.dev/google/faster_rcnn/openimages_v4/inception_resnet_v2/1`. Trained on the Open Images V7 dataset, which is a dataset comprising approximately of 9M images, annotated with image-level classes, object bounding boxes, object segmentation masks, visual relationships, and localised narratives. It produces a list of bounding boxes, which we then filter by requiring a class probability thresholds greater than that found in Table 6.1. We also regulate the number of instances using a hyperparameter, capping it at 40. Generally if it's a class of interest the threshold is set to a low value, if it's a class the object detector has found but not of interest to us the threshold is set high.

Class	Value	Class	Value	Class	Value
Kitchen appliance	0.9	Carnivore	0.9	Animal	0.9
Sword	0.9	Antelope	0.9	Plant	0.9
Sculpture	0.9	Office supplies	0.9	Drink	0.9
Knife	0.8	Pear	0.9	House	0.9
Coffee cup	0.8	Countertop	0.9	Pitcher	0.9
Wine	0.6	Tap	0.9	Cutting board	0.9
Horse	0.6	Kitchen utensil	0.9	Tree	0.9
Banana	0.53	Cattle	0.9	Mule	0.9
washbasin	0.5	Building	0.9	Bathroom accessory	0.9
stove	0.5	Food	0.9	Desk	0.9
Fork	0.5	Spatula	0.9	Table	0.9
Mug	0.4	Beer	0.9	Plumbing fixture	0.9
Teapot	0.4	Jug	0.9	Giraffe	0.9
Bottle	0.1	Rhinoceros	0.9	Bull	0.9
fryingpan	0.1	Mango	0.9	Fruit	0.9
Spoon	0.1	Vegetable	0.9	Zebra	0.9
Wine glass	0.1	table	0.9	Dagger	0.9
Screwdriver	0.0	Coffee Tool	0.9	Kettle	0.9
teaspoon	0.0	Tableware	0.9	Mammal	0.9
Kitchen knife	0.0	Door handle	0.9	-	-

Table 6.1: Class Filter for object detector.

Step 2: Feature Encoding

Input: List of ROI images given by bounding boxes.
Output: Feature Encoding for ROI image(s).

The VGG-19 Image Encoder is utilised, as detailed in Simonyan and Zisserman [3], sourced from `tensorflow.keras.applications.vgg19`. This encoder, pre-trained on the ImageNet dataset's 1000 classes, utilised the corresponding weights. By omitting the top layer, we generated a 25088-dimensional visual feature encoding for each ROI image. These encodings were then processed by the GMMs for analysis.

VGG-19 is a well-known CNN model, it has been effectively used in image classification, pattern recognition, and speech recognition. It is characterised by its depth and simplicity, the model comprises 19 weight-bearing layers, including 16 convolutional layers, which are followed by three fully connected layers. Utilising consistent 3x3 filters and max-pooling operations, VGG-19 excels in extracting hierarchical spatial features from images, ranging from basic edges to intricate patterns. Pre-trained on a massive dataset such as ImageNet, VGG-19 has not only achieved competitive performance in classification tasks but has also become a foundational encoder in various computer vision applications, from object detection to content-based image retrieval, owing to its robust feature extraction capabilities.

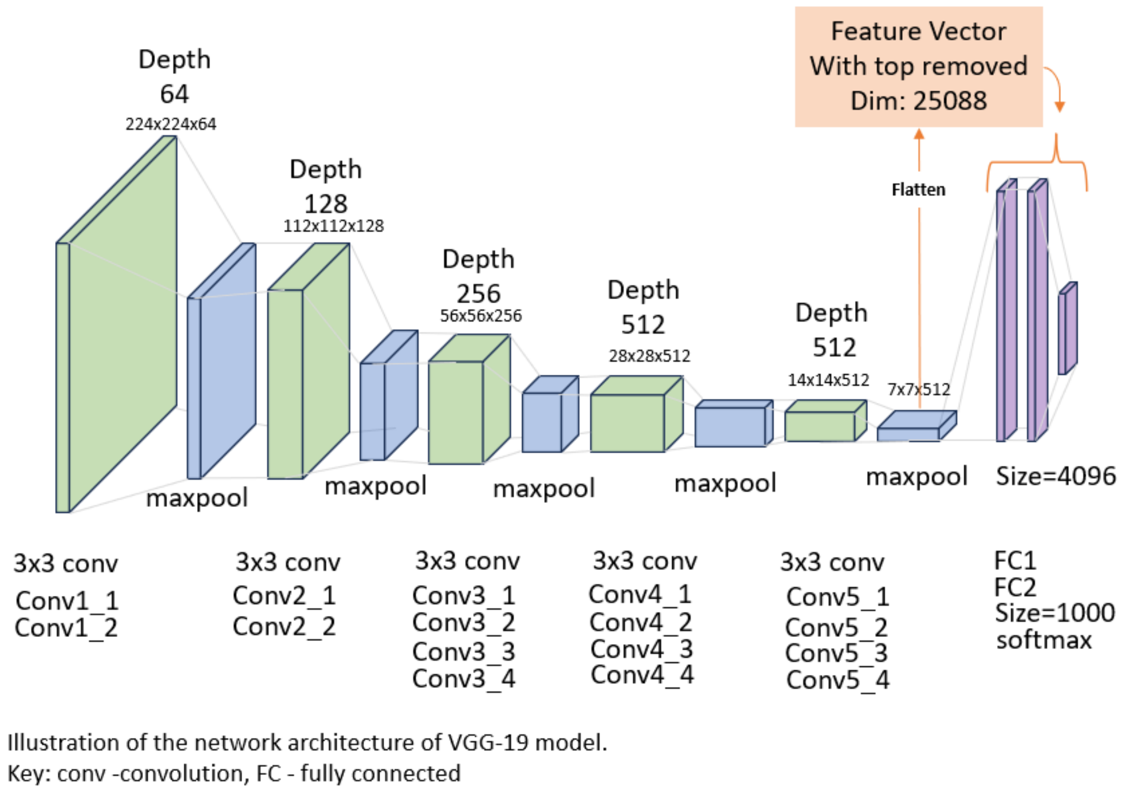


Figure 6.3: The VGG-19 Neural Net [3], used to create a 25088 Feature Vector, trained with supervised training on Dataset: ImageNet.

Step 3: Gaussian Mixture Models - Training

Fourteen GMMs were trained. Each GMM model was trained on 20 representative images. The training images were obtained by using the class name as a search term in Google, 20 of the returned images were selected and centrally cropped and encoded. Figure 6.4 & Figure 6.5 show example sets of images used to train the Object category class *Banana* and the Attribute category class *Striped*. This means you can train an object class for "spoon" and train multiple attribute classes for materials like wood, plastic, or metal. Importantly, to encode the category there must be an image encoder capable of identifying the desired features in all the example class images. For instance, visually distinguishing between china and plastic can be challenging.

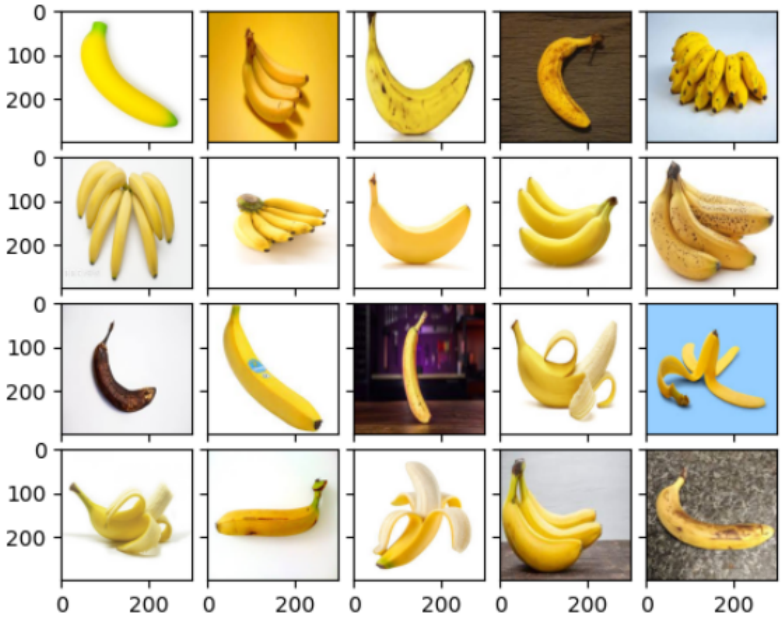


Figure 6.4: Banana Class Training Images.

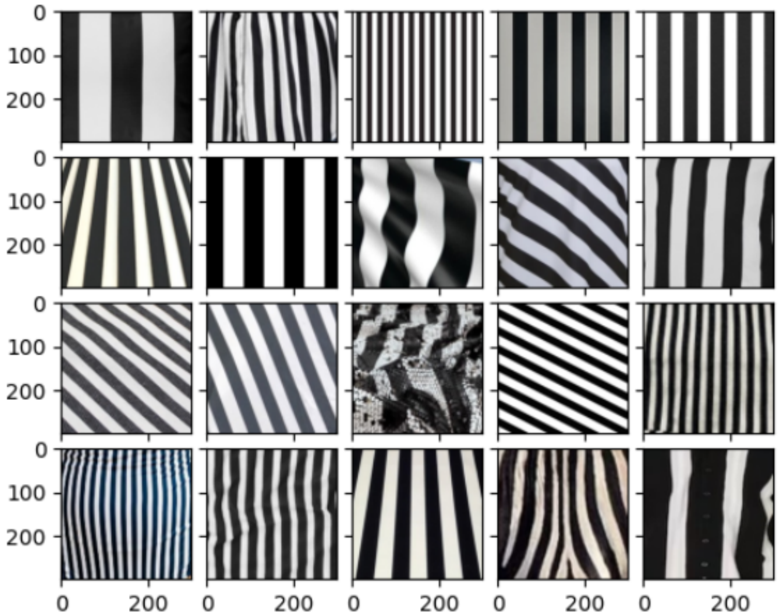


Figure 6.5: Striped Class Training Images.

The 20 images were encoded by the image encoder producing 20 x 25088 dimensional feature encodings, which a GMM was then fitted to. The GMM hyperparameters were investigated by testing the accuracy for all classes using a component number of 1 through to 5 and also a trial with each class being picked by its lowest bic score which we term *Automatic* and index as 0 in the Table. Figure 6.6 shows the results for 100% of the training and validation dataset. Taking the validation values to avoid overfitting the model, we select component number 3, since this is a good balance between complexity and accuracy. Therefore the hyperparameters to train the GMMs are : i) Number of mixture components: 3, ii) Covariance type: Diagonal, and iii) non-negative regularisation: 1.0e-2. 10 GMM models with these hyperparameters were then created. The model that achieved the lowest BIC score was retained for subsequent use. Figure 6.7 illustrates the range of BIC scores achieved per GMM Model (*Creating the Object classifier*) when trained, Table 6.2 shows a list of the retained minimum BIC scores after 10 trials per GMM class. In subsequent testing, GMMs are tested on a test dataset consisting of five images per class. The 'Objects' classifier, reported an accuracy of 92.31%, an F1 score of 92.04%, and a Cohen's Kappa score of 93.63% .

Training Data				Validation Data			
Trial	BIC_Comp_Num	Model_Acc	Model_F1	Trial	BIC_Comp_Num	Model_Acc	Model_F1
#5.1	1	0.8556	0.8496	#5.1	1	0.8449	0.8381
#5.2	2	0.8741	0.8736	#5.2	2	0.8560	0.8553
#5.3	3	0.8911	0.8904	#5.3	3	0.8625	0.8618
#5.4	4	0.9019	0.9015	#5.4	4	0.8682	0.8682
#5.5	5	0.9154	0.9152	#5.5	5	0.8715	0.8716
#5.6	0	0.9025	0.9022	#5.6	0	0.8667	0.8660

Figure 6.6: GMM Component Number Investigation.

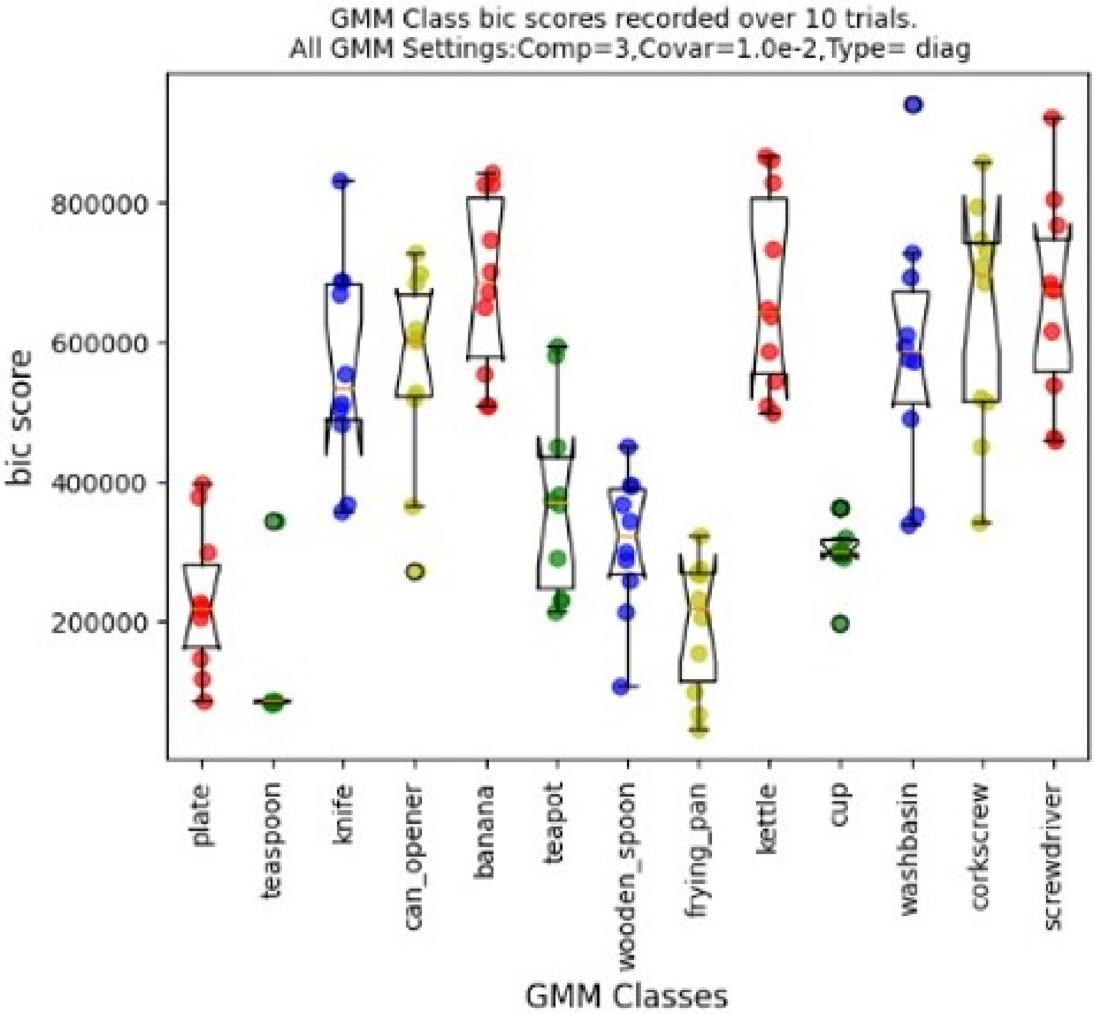


Figure 6.7: GMM Class Models: Distribution of Bayesian Information Criterion score.

Class	Reg CoVar	Comp. Num	Comp. Shape	BIC
0: plate	$1e^2$	3	diag	32048.9749858462
1: teaspoon	$1e^2$	3	diag	84015.02517372137
2: knife	$1e^2$	3	diag	369096.505543082
3: can_opener	$1e^2$	3	diag	437297.0493718943
4: banana	$1e^2$	3	diag	468663.6017532455
5: teapot	$1e^2$	3	diag	211324.83007161587
6: wooden_spoon	$1e^2$	3	diag	39630.36155217671
7: frying_pan	$1e^2$	3	diag	34081.78180753725
8: kettle	$1e^2$	3	diag	489804.4885691839
9: cup	$1e^2$	3	diag	223413.239309614
10: washbasin	$1e^2$	3	diag	547889.2854903314
11: corkscrew	$1e^2$	3	diag	383366.8048321256
12: screwdriver	$1e^2$	3	diag	286082.7113044384

Table 6.2: GMM Class Models: Configuration of best models.

In order to help the readers' intuition the 25088 dimensions are reduced to 2 dimensions through the use of UMAP [83] so they can be visually plotted as shown in Figure 6.8, where the data points (encoded image) can be seen as crosses and then coloured according to the three GMM Components fitted to them for the banana class. Two additional panels show the probability density distributions for each component according to the visual feature dimension.

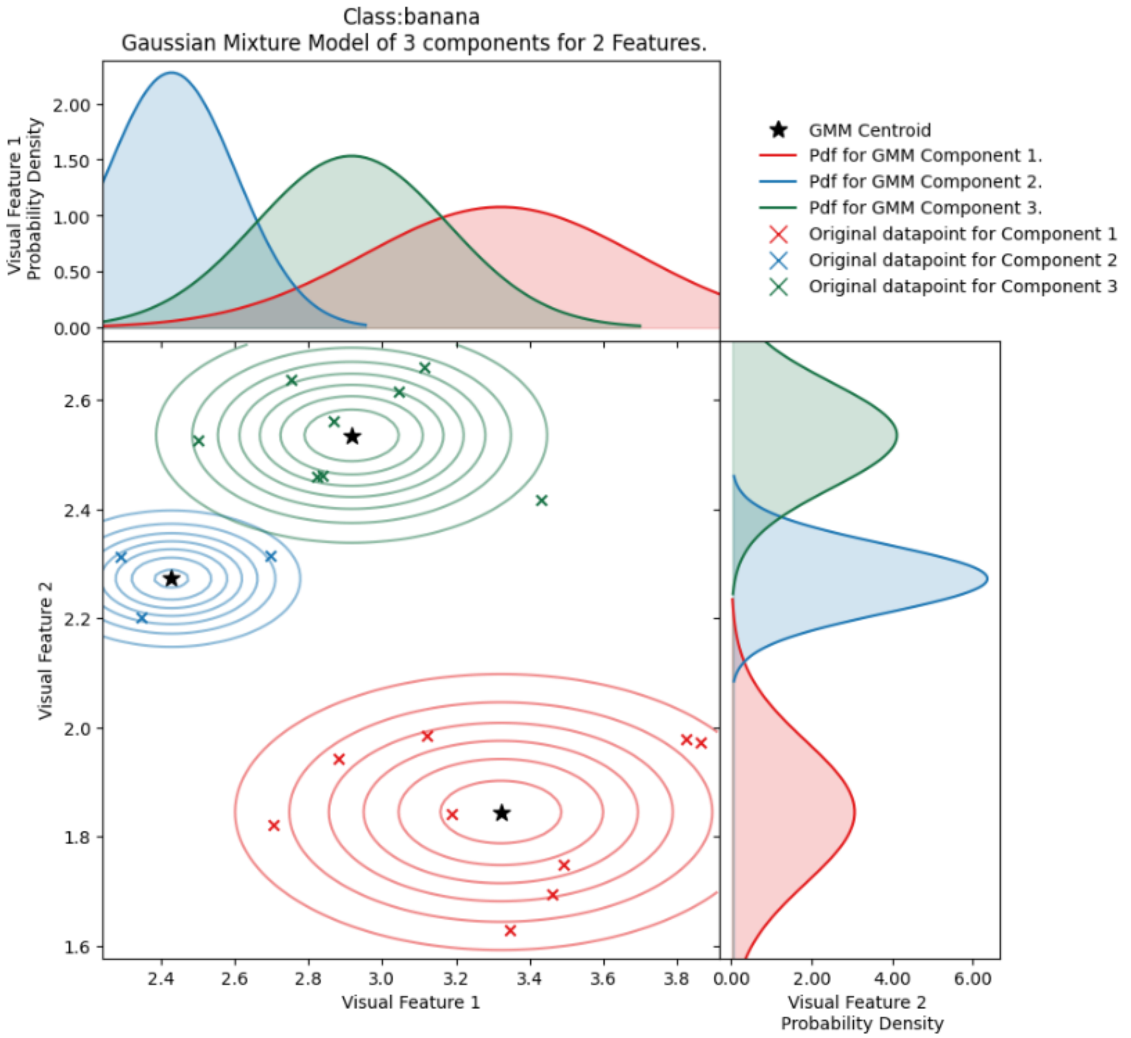


Figure 6.8: Class feature encodings fitted to a 3 component Gaussian Mixture Model. Class shown: Banana

Step 3: Gaussian Mixture Models - Inference

Input: Feature Encoding for an ROI Image(s).
Output: A list of class likelihoods.

This is the stage in the pipeline where information is abstracted from a neural net to a symbolic representation. Listing 6.1 finds the GMM likelihood and creates a dictionary for use during the KG generation.

```
Likelihood_Array = []
GMM_Array = []
GMM_Array = os.listdir(GMM_Images_Object_Dir)
GMM_Array.remove('.ipynb_checkpoints')
GMM_Number = len(GMM_Array)
print(GMM_Array)

for counter in range(0,GMM_Number):
    GMM_Filename = GMM_Models_Object_Dir + 'GMM_'+ GMM_Array[counter] +'.pkl'
    with open(GMM_Filename, 'rb') as f:             # Load GMM Class Model
        Model = pickle.load(f)
        Label_Likelihood = Model.score_samples(image_latent_embedding)
        Likelihood_Array.append(Label_Likelihood[0])
GMM_Dict = defaultdict(list)
for counter in range (0, GMM_Number):
    GMM_Dict[GMM_Array[counter]].append(float(Likelihood_Array[counter]))
print(GMM_Dict)
```

Listing 6.1: Creation of GMM Likelihood Dictionary.

Step 4: Categories.

In Figure 6.9, we specify the ROI nodes, categories and classes used in the NS-IL model. The Ontology for the NS-IL was designed to create a very small closed world. This was intentional so that the focus is kept on the model mechanism.

Two categories were selected of *Object* and *Attribute*. These categories determine the classifier boundaries i.e. which GMMs are compared together. Object GMM Classes were selected based on those that existed both in the ImageNet dataset and in kitchen reality. Additional classes of horse, wine_glass, and wine_bottle were added for specific experiments to be carried out. Attribute Classes were added to trial the generation and use of multiple categories. The image at inference time is encoded and then presented to each GMM within a classifier, the GMM with the

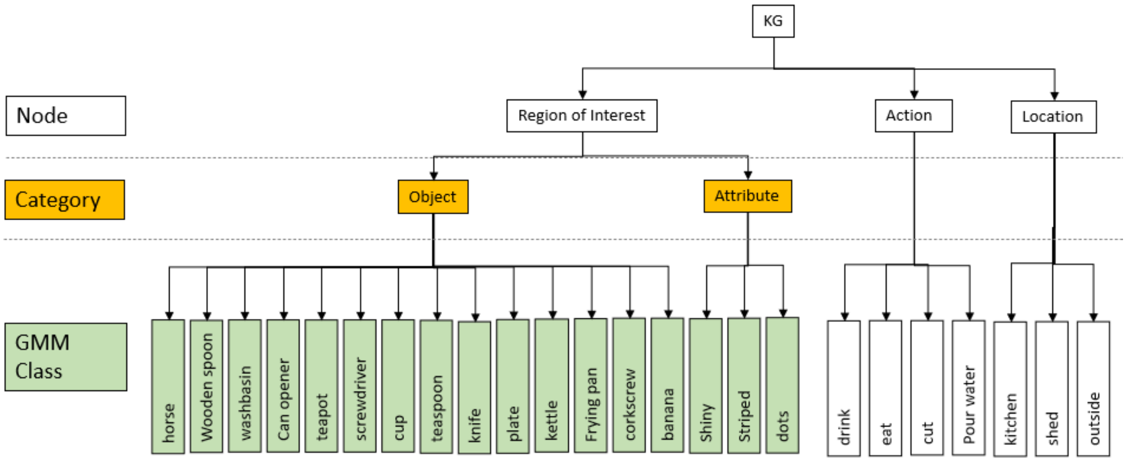


Figure 6.9: Ontology: Showing ROI Node, Category & GMM Classes.

highest likelihood is noted. It is this class that determines the classification for a category.

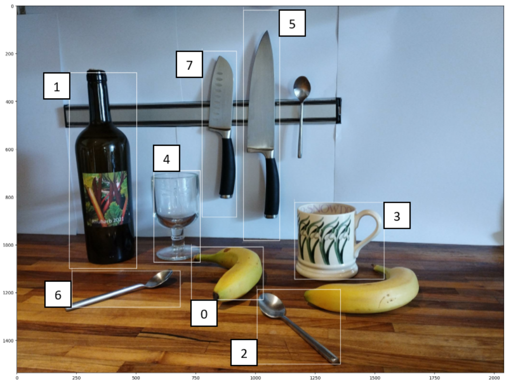
6.5.3 *Knowledge Graph*

KGs encapsulate structured representations of information through nodes and relationships. In our system, they enhance computer vision tasks by offering a deeper semantic understanding, surpassing mere object recognition to contextualise relationships within captured scenes. During reasoning, these graphs aid in relationship detection and scene comprehension, providing contextual understanding, relationship analysis, and additional information vital for tasks like zero-shot learning, image retrieval, and visual question answering.

The Knowledge Graph (KG) leverages the Ontology as a Schema to structure its nodes. The configuration determines how rich the semantic space is, based on what subsymbolic information is specified. This forms the basis of the KG's nodes generation. The presence of the Ontology simplifies detailed queries by the LLM, facilitates knowledge inference, and ensures consistency and accuracy in information representation.

The Knowledge Graph is implemented in Neo4J, a high performance NoSQL database designed for handling nodes and relationships [84]. The Cypher language is used to operate and manage this graph database. Utilising Py2neo, a client library and toolkit, to interface with Neo4j from within our Python application. This allows the execution of Cypher code via a Python wrapper.

The NS-IL system initialises and then generates a KG for an image. Using the Image in Figure 6.10 to process. Steps 5 to 7 break this process down into it's constitute parts



ROI	Object Name	Correct?
ROI_0	Banana	Y
ROI_1	Wine Bottle	Y
ROI_2	Teaspoon	Y
ROI_3	Kettle	N (cup)
ROI_4	Wine Glass	Y
ROI_5	Knife	Y
ROI_6	Teaspoon	Y
ROI_7	Knife	Y

Figure 6.10: An Image and its Object Classifications at inference time.

Step 5: Classes & Relationships

Table 6.3 shows the items treated as metadata and used to initialise the KG. Figure 6.11 is a screenshot of the resulting KG when the image in Figure 6.10 is processed.

Item	Description
Scene Node (Red)	The scene node holds the directory location of the image that has been processed. The node holds the directory information of where the image is located.
Object Node (Orange)	horse, plate, teaspoon, knife, can_opener, banana, teapot, wine_glass, wooden spoon, frying pan, kettle, cup, washbasin, wine_bottle, corkscrew, screwdriver
Attribute Node (Green)	striped Use to test multiple categories for an ROI Node.
Location (Light Blue)	outside, kitchen, shed
Action (Purple)	cut, eat, drink from
Relationship	Object_Class AFFORDS Action. Assigned when a specific object class has an affordance. e.g. a banana affords 'eat', a cup affords 'drink from', and a knife affords 'cut'.
Relationship	Object_Class FOUND_AT Location. Assigned when an Object is normally located somewhere.

Table 6.3: The Items generated in the KG as Metadata.

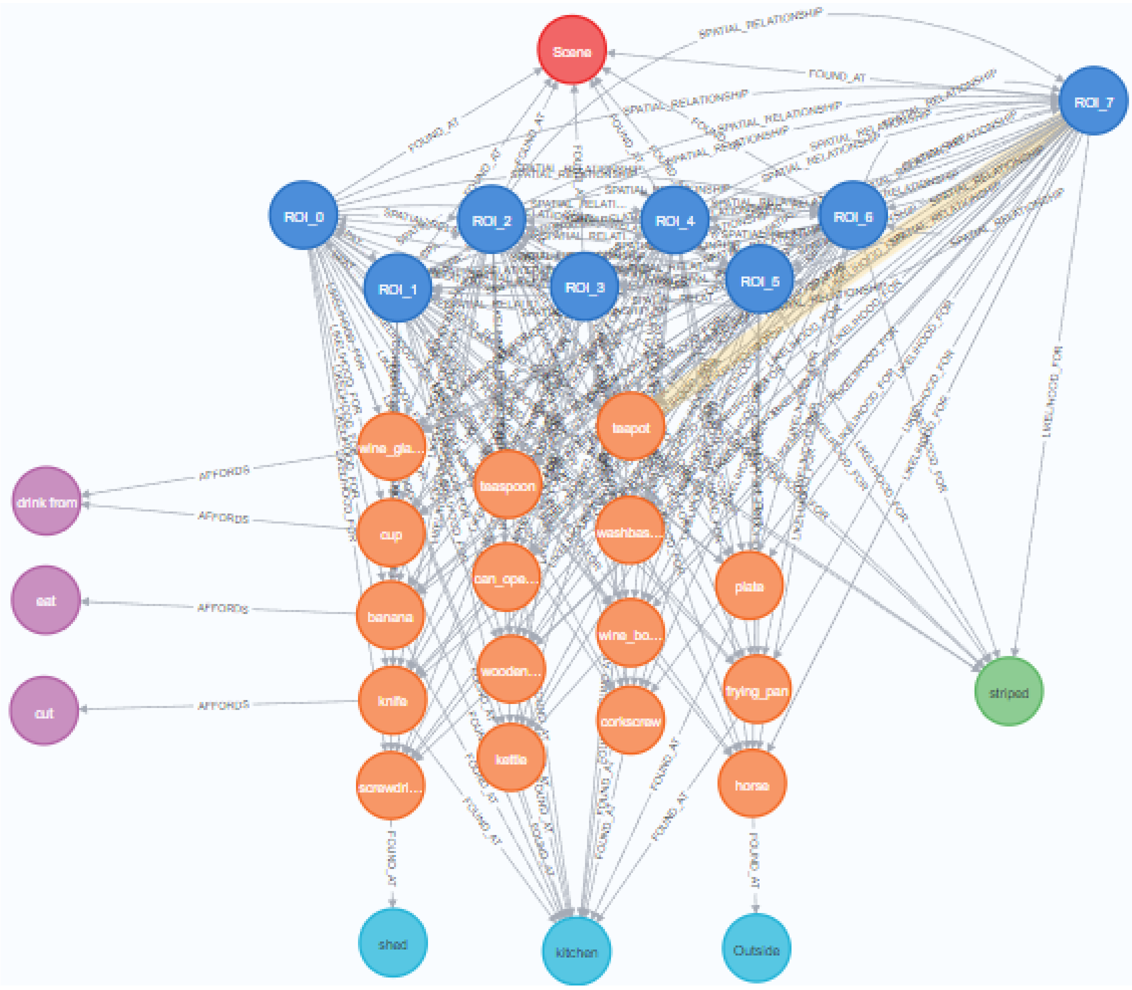


Figure 6.11: Populating the KG - Graphic.

Step 6: ROIs

In Step 4, the Region of Interest (ROI) nodes received assignments of Object and Attribute categories. This assignment enables the identification of the maximum likelihood class for each category upon node generation. There are no errors identified, only the most likely class seen, the inference made is dependent on the available classes in the system. The knowledge graph generation process calculates these class likelihoods at creation time as depicted in Figure 6.2. Each edge between an ROI Node and its associated class holds the grounding likelihood or certainty that that class was 'seen'.

Table 6.4 shows the items that are generated at Image inference time.

Item	**Description**
ROI	A Region of Interest identifier to provide individual segments of an image to be held with associated information.
Relationship	ROI_X FOUND_AT Scene The ROIs within the given image.
Relationship	ROI_X LIKELIHOOD_FOR Attribute_X. The likelihood that ROI_X has this attribute.
Relationship	ROI_X LIKELIHOOD_FOR Object_Class. The likelihood that ROI_X belongs to this Class .
Relationship	Source Relativity to Target Relativity

Table 6.4: The Items generated in the KG at inference.

Step 7: Commonsense Rules

After the new ROIs are generated in the knowledge graph, Commonsense rules can be applied to classes and relationships in order to hold the best-perceived world model. The Commonsense rules that are applied are outlined in Table 6.5.

Given the lack of depth information, certain assumptions are made about the bounding boxes in the 2D image. Although these simplifications might lead to inaccurate inferences about an objects' relative size and position, they are instrumental in demonstrating possibilities for our proof of concept NS-IL model. Note the origin is the top left of an image.

Item	**Description**
Relationship	Source Object Relativity to Target Relativity LEFT: A relationship is established between a source object and a target object when the x_{max} of the source object is less than the x_{min} of the target object. RIGHT: A relationship exists between a source object and a target object when the x_{min} of the source object exceeds the x_{max} of the target object. INFRONT: A relationship is defined between a source object and a target object if the y_{min} of the source object is greater than the y_{max} of the target object. BEHIND: A relationship forms between a source object and a target object when the y_{max} of the source object is less than the y_{min} of the target object.
Object_Name Update	For a category object labeled as 'horse' with a category Attribute labeled as 'Striped' amend object name to 'Zebra'.

Table 6.5: The Commonsense Rules applied to the KG after inference.

6.5.4 *Large Language Model*

The LLM is utilised through OpenAI's Application Programming Interface (API) of 'gpt-4'. The KG node and edge semantics are aligned based on suggestions from the LLM, gained by prompting it with our schema and asking it, "Suggest improvements in the naming of nodes and relationships in the given schema." The Schema is shown Listing 6.2.

```
Node properties: [['output'], [{'labels': 'ObjectPrototype', 'properties': ['name']}], [{'labels': '
    ↪ActionPrototype', 'properties': ['name']}], [{'labels': 'AttributePrototype', 'properties':
    ↪['name']}], [{'labels': 'RoomPrototype', 'properties': ['name']}], [{'labels': 'SceneFile', '
    ↪properties': ['File', 'name']}],
 [{'labels': 'ROI', 'properties': ['Action', 'Attribute', 'Time_Seen', 'ymax', 'xmax', 'name', 'xmin
     ↪', 'ymin', 'Object_Name']}]]

Relationship Properties:[['output'], [{'properties': ['Time_Seen', 'ymin', 'xmin', 'ymax', 'xmax'],
    ↪'type': 'FOUND_AT'}], [{'properties': ['Value'], 'type': 'SPATIAL_RELATIONSHIP'}], [{'
    ↪properties': ['Likelihood_Value'], 'type': 'LIKELIHOOD_FOR'}]]

Relationships: [['output'], [{'relationship': 'AFFORDS', 'source': 'ObjectPrototype', 'target': ['
    ↪ActionPrototype']}], [{'relationship': 'FOUND_AT', 'source': 'ObjectPrototype', 'target': ['
    ↪RoomPrototype']}], [{'relationship': 'SPATIAL_RELATIONSHIP', 'source': 'ROI', 'target': ['ROI
    ↪']}], [{'relationship': 'FOUND_AT', 'source': 'ROI', 'target': ['SceneFile']}], [{'
    ↪relationship': 'LIKELIHOOD_FOR', 'source': 'ROI', 'target': ['AttributePrototype', '
    ↪ObjectPrototype']}]]
```

Listing 6.2: The Neo4j Schema.

Listing 6.3 is an example of what ChatGPT replies to this request. Slight amendments were made however since the suggestions change each time the actual Terms used were selected by a human.

```
The schema you've shared outlines a structure for a knowledge database focusing on objects, actions,
    ↪ attributes, rooms, scene files, and regions of interest (ROIs), along with their
    ↪relationships. The naming conventions used in the schema are crucial for clarity, consistency,
    ↪ and ease of understanding. Here are some suggestions for improvements in the naming of nodes
    ↪ and relationships:

Improvements in Node Naming
ObjectPrototype could be renamed to Object if it's meant to represent instances of objects in scenes
    ↪ rather than a prototype design. If it truly is a template for objects, then ObjectTemplate
    ↪might be clearer.

ActionPrototype similarly could be Action or ActionType if it represents types of actions that can
    ↪be performed, clarifying that it's a category rather than an instance of an action.

AttributePrototype might be better named as Attribute or AttributeType, depending on whether it's
    ↪meant to categorize attributes or represent individual instances.
```

```
RoomPrototype could be renamed to RoomType if it represents categories of rooms, or simply Room if
    ↪it's meant to represent individual rooms.

SceneFile is quite clear, but to ensure it's understood as representing digital files, you might
    ↪consider SceneDataFile or SceneAsset.

ROI (Region of Interest) is a technical term that might not be immediately clear to all users.
    ↪Consider renaming it to InterestRegion or SceneRegion for clarity.

Improvements in Relationship Naming
AFFORDS relationship between ObjectPrototype and ActionPrototype is well named, indicating what
    ↪actions an object enables. No change suggested here unless clarity is needed for non-
    ↪technical users, in which case ENABLES_ACTION could be more descriptive.

FOUND_AT relationship is clear but could be more descriptive. For the relationship between
    ↪ObjectPrototype and RoomPrototype, consider LOCATED_IN. For the relationship between ROI and
    ↪SceneFile, DETECTED_IN_SCENE might provide more context.

SPATIAL_RELATIONSHIP between ROIs could be more specific depending on the nature of the spatial
    ↪relationship. If it's about proximity, consider NEARBY or ADJACENT_TO. If it's hierarchical,
    ↪WITHIN or CONTAINS might be better.

LIKELIHOOD_FOR relationship indicating the probability of an ROI being associated with an
    ↪AttributePrototype or ObjectPrototype could be renamed to PREDICTED_AS or
    ↪HAS_ATTRIBUTE_LIKELIHOOD to clarify that it's about prediction or classification confidence.

General Suggestions
Consistency in Naming: Ensure that all node and relationship names follow a consistent naming
    ↪convention, either using CamelCase, snake_case, or another standard, and stick to it
    ↪throughout the schema.
Verbosity vs. Clarity: While shorter names are neater, sometimes a bit more verbosity can add
    ↪clarity, especially for users who might not be familiar with the domain or technical terms.
Domain-specific Terms: If the database is intended for users deeply familiar with the domain, using
    ↪domain-specific terminology might be more appropriate. However, if it's meant for a broader
    ↪audience, simpler terms could be more accessible.
Adapting these suggestions can enhance the understandability and usability of your knowledge
    ↪database schema by making it more intuitive and aligned with the expectations of its intended
    ↪ users.
```

Listing 6.3: ChatGPTs reply on been asked for suggestions.

Step 8: Question to Code

The integration between ChatGPT and the KG follows the guidelines from the Neo4j blog by Tomaz Bratanic's, detailed at https://neo4j.com/developer-blog/generating-cypher-queries-with-chatgpt-4-on-any-graph-schema/. The prompt is created by concatenating three pieces of text:

1. The Task: We state that as a Natural Language Query Generator (NLQG)

of Cypher queries, its task is to generate a Cypher query for a Neo4j graph database based on the provided schema definition.

2. The Instructions: The KG schema is generated and the LLM told to only use this. It is specified that if the generated code says Object Name this should be translated to Object_Name since this is hard coded in our Neo4j structure. We tell the LLM to remove any empty results fields.

3. The Translation Examples:
   Listing 6.4 shows the Cypher statements used to create a meaning function by pairing natural language with corresponding code examples. This approach leverages the capacity of Large Language Models (LLMs) to quickly learn from a limited set of examples. By presenting the model with Cypher statement examples, it acquires the ability to generate similar Cypher statements through imitation.

```
Examples:
# What can I drink from ?
MATCH (q:ROI{Action:'drink from'}) RETURN q.Object_Name
 # What can I cut ?
MATCH (q:ROI{Action:'cut'}) RETURN q.Object_Name
# What affords drink from ?
MATCH (q:ROI{Action:'drink from'})return q.Object_Name
# What can I eat ?
MATCH (q:ROI{Action:'eat'})return q.Object_Name
# Where is a cup ?
MATCH (q:ROI{Object_Name:'cup'})-[r]->(o:ROI) return q.Object_Name,r.Value,o.Object_Name
# What can be found in the kitchen ?
MATCH (o:RoomPrototype{name:"kitchen"})-[r:FOUND_AT]->(p) RETURN p.name
# Where is the 1st Object in relation to the 2nd object ?
MATCH (k:ROI{Object_Name:'knife'})-[r]->(b:ROI{Object_Name:'banana'}) RETURN k.
    ↪Object_Name,k.name, r.Value, b.Object_Name,b.name
"""
```

Listing 6.4: LLM Prompt Engineering - Training through Examples.

Step 9: Code to Results

The Cypher prompt is then executed on the KG and a response is received containing a list of results. An example is shown in Listing 6.5. Each results array is comprised of a [source, value] relation, and [target, value] tuple.

Step 10: Results to Answer

The *Results string* from Step 9 is taken and merged with a new *task* and *instructions* before being submitted to the Large Language Model (LLM) for the *Answer*.

- The Task: This uses the original Question and the Results, a sentence is constructed that communicates the Answer to the user who posed the Question.
- The Instructions: to focus on the provided Question and Results details. Also, that the response shouldn't include any explanations or apologies.

The prompt is sent via the API to the LLM and the user receives their Answer.

Listing 6.5 shows a worked example, the Question the user asks, the question the LLM translates into Cypher Code, the Cypher Code executed on the KG with the Results, and then the Results translated into a natural language Answer by the LLM.

```
Question:
Where is the knife in relation to banana?

Cypher:
MATCH (k:ROI{Object_Name:'knife'})-[r]->(b:ROI{Object_Name:'banana'}) RETURN k.Object_Name,k.name, r.
    ↪Value, b.Object_Name,b.name

Results:
[['k.Object_Name', 'k.name', 'r.Value', 'b.Object_Name', 'b.name'], ['knife', 'ROI_7', 'BEHIND', '
    ↪banana', 'ROI_0'], ['knife', 'ROI_5', 'BEHIND', 'banana', 'ROI_0']]

Answer:
The knife is located behind the banana.
```

Listing 6.5: A worked example of Question to Answer

6.6 Results and Analysis

This section introduces our experimental setup, focusing on how we examine and validate the capabilities of the NS-IL system. Initially, the feature space is explored to understand the classification mechanisms in place. The system's adaptability is then demonstrated by showing that introducing new terms does not compromise the integrity of existing ones. Our experiments also highlight the process of transforming visual information from images into structured knowledge within the knowledge graph (KG), emphasising the synthesis of meaning. A key part of our investigation validates the theoretical framework that combines symbolic and neural information to deduce new concepts, illustrated through the example of identifying a Zebra. Finally, the system's question-answering performance is evaluated by analyses of its effectiveness in navigating and extracting information from the knowledge graph.

6.6.1 *Created Dataset*

A dataset of five images was created, each image depicting a photo of everyday objects that matched the selected ontological terms of items found in a kitchen. To reduce image noise, which was causing issues, we covered the textured tiles with paper. The GMM classes, trained on web images, had not 'seen' these specific item instances before.

6.6.2 *Step 2 & 3 - Feature Encoding & Classification*

This method allows us to visually interpret the classification boundaries and the effectiveness of the feature space in distinguishing between classes. To improve understanding and enable a visualisation, UMAP [83] is applied to compress the 25088 dimensional feature encoding to 2 features, which can be seen in Figure 6.12. The figure shows Gaussian Means by 'stars' and the Gaussian Covariance Contours by 'ellipses' for each GMM. This highlights the predicting feature space regions for each class. In figure 6.13 the original 260 training images (20 images/class) are displayed and the respective regions with the highest prediction likelihood.

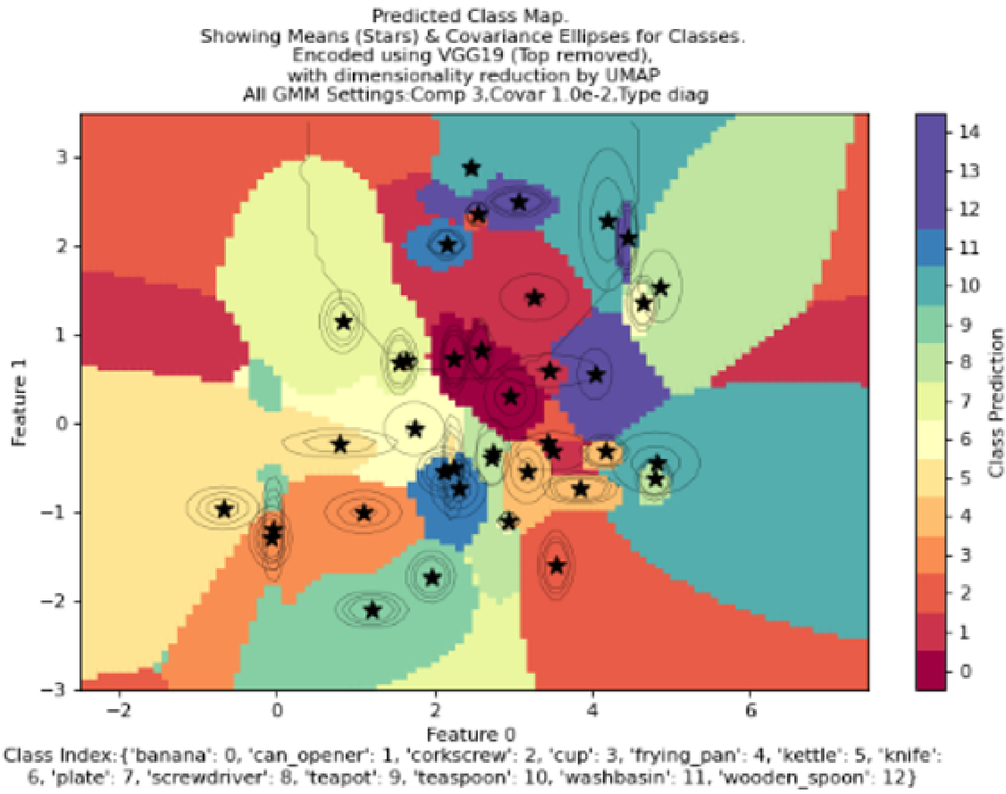


Figure 6.12: Classification boundaries for with GMMs.

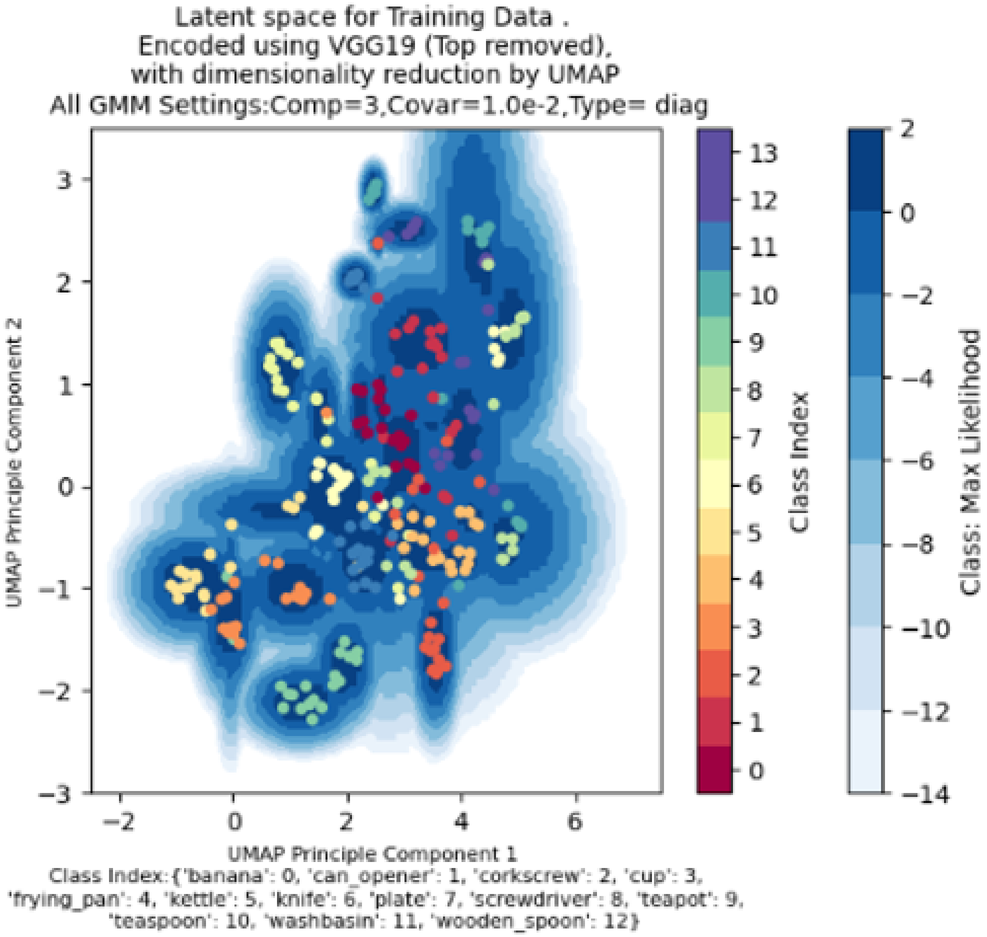


Figure 6.13: Likelihood boundaries with training data overlaid.

6.6.3 *Step 5 - Addition of GMM Classes*

Using the Image seen in Figure 6.14 with the initial set of GMMs we noted NS-ILs classification together with each GMMs likelihood. Then we add two new classes for *Wine_Bottle* and again note the NS-ILs classification together with each GMMs likelihood. The outcome of *Before* and *After* the new class addition can be seen in Table 6.6. The Likelihoods for ROI_1 the wine bottle can be seen in Figure 6.15 and 6.16, the likelihoods for ROI_4 the Wine Glass can be seen in Figure 6.17 and 6.18.

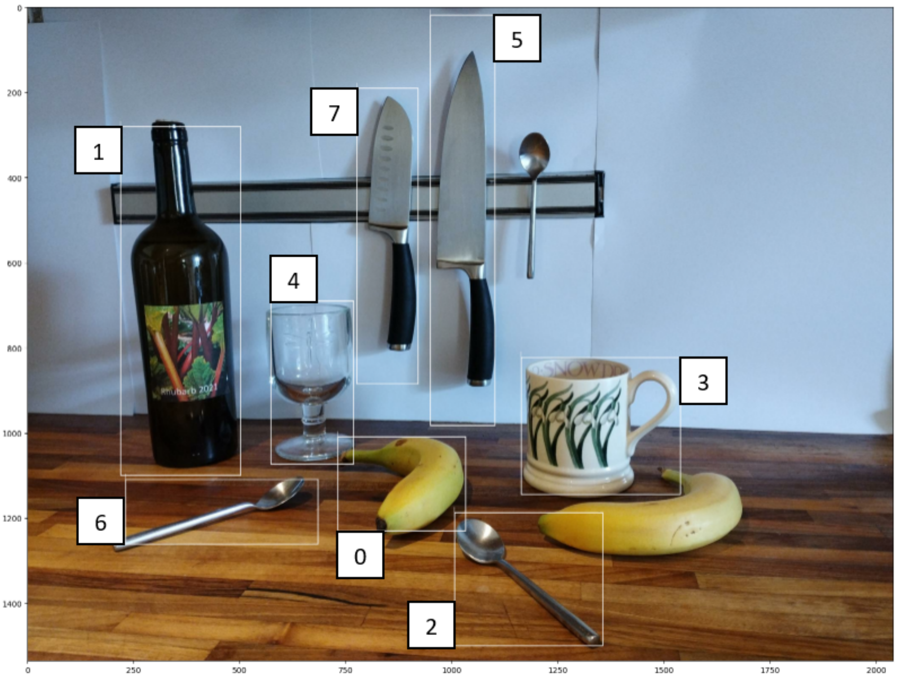


Figure 6.14: The Image used for Before and After New Classes added.

ROI Name	Class Before	Outcome Before		Class After	Outcome After
ROI_0	banana	Y		banana	Y
ROI_1	washbasin	N (Bottle)		Wine_bottle	Y
ROI_2	teaspoon	Y		teaspoon	Y
ROI_3	Kettle	N(cup)		Kettle	N (cup)
ROI_4	knife	N(Glass)		Wine_glass	Y
ROI_5	knife	Y		knife	Y
ROI_6	screwdriver	N(Teaspoon)		screwdriver	N(Teaspoon)
ROI_7	knife	Y		knife	Y

Table 6.6: ROI Classifications for Image: Before and After Addition of Wine Glass and Wine Bottle.

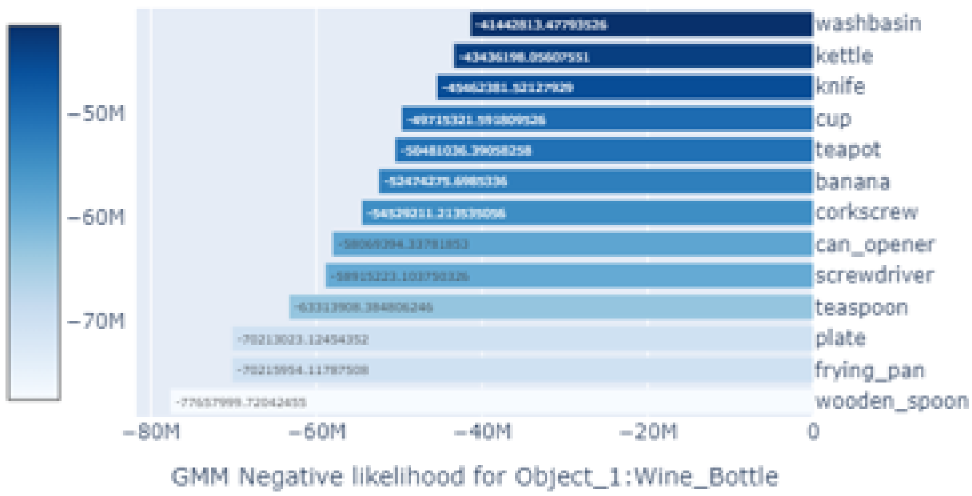


Figure 6.15: GMM Negative likelihood for ROI_1:Wine_Bottle, Before Addition of New Classes

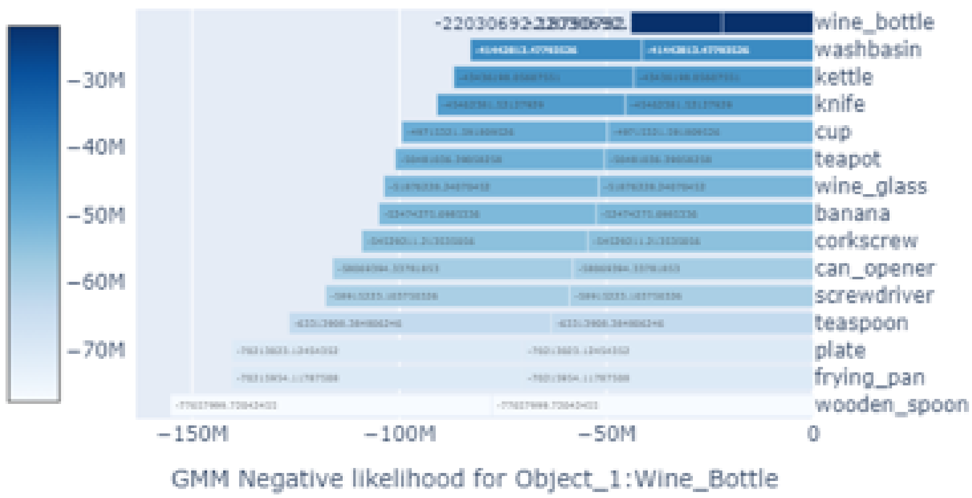


Figure 6.16: GMM Negative likelihood for ROI_1:Wine_Bottle, After Addition of New Classes

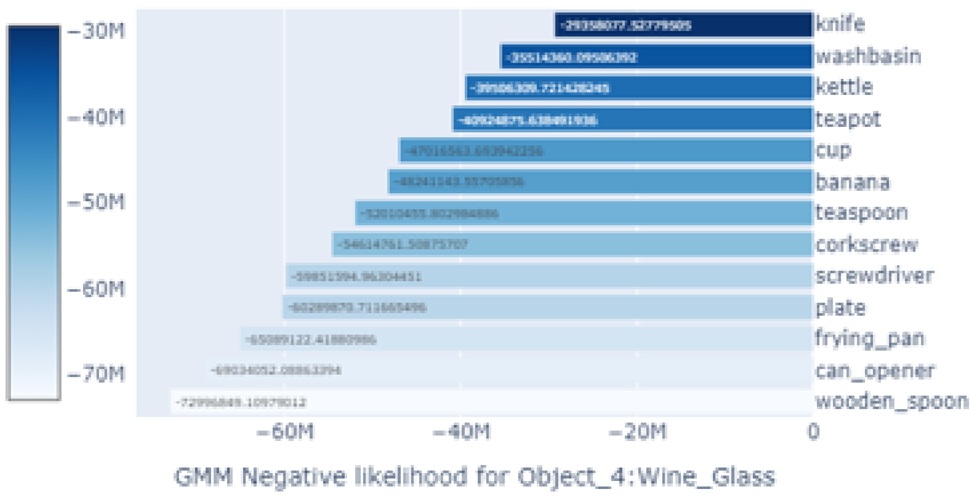


Figure 6.17: GMM Negative likelihood for ROI_4:Wine_Glass, Before Addition of New Classes

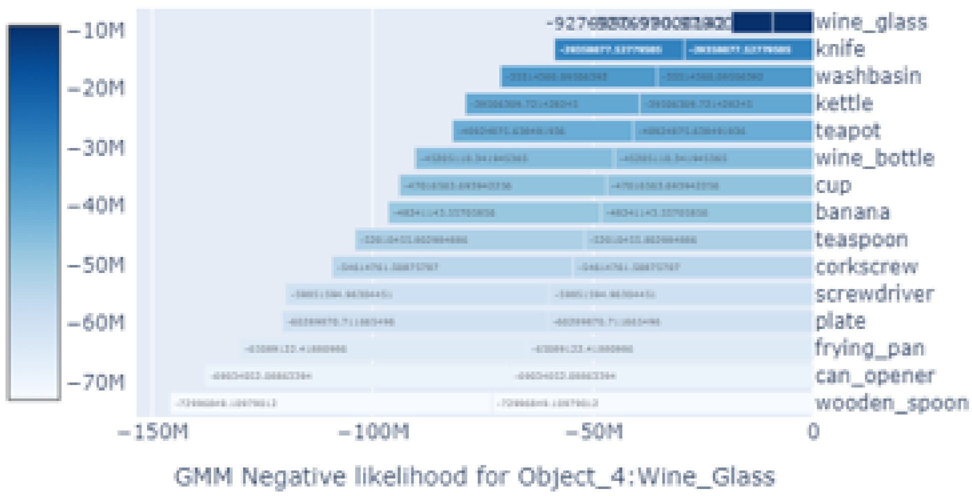


Figure 6.18: GMM Negative likelihood for ROI_4:Wine_Glass, After Addition of New Classes

It can be seen that the new class for Wine Bottle which the Image Encoder has never seen before successfully identifies the Wine Bottle. In a similar manner the Wine Glass Class has successfully enabled the wine glass to be classified. The order of Likelihoods of all the other Classes has not altered. For ROI_3 which is classified as a kettle incorrectly even though the kettle and cup classes are not close to each other. We hypothesis that the sweeping line of the snowdrop design is similar to a teapots handle. For ROI_6 it can be seen that screwdriver and teaspoon are next to each other in the projected space seen in Figure 6.19.

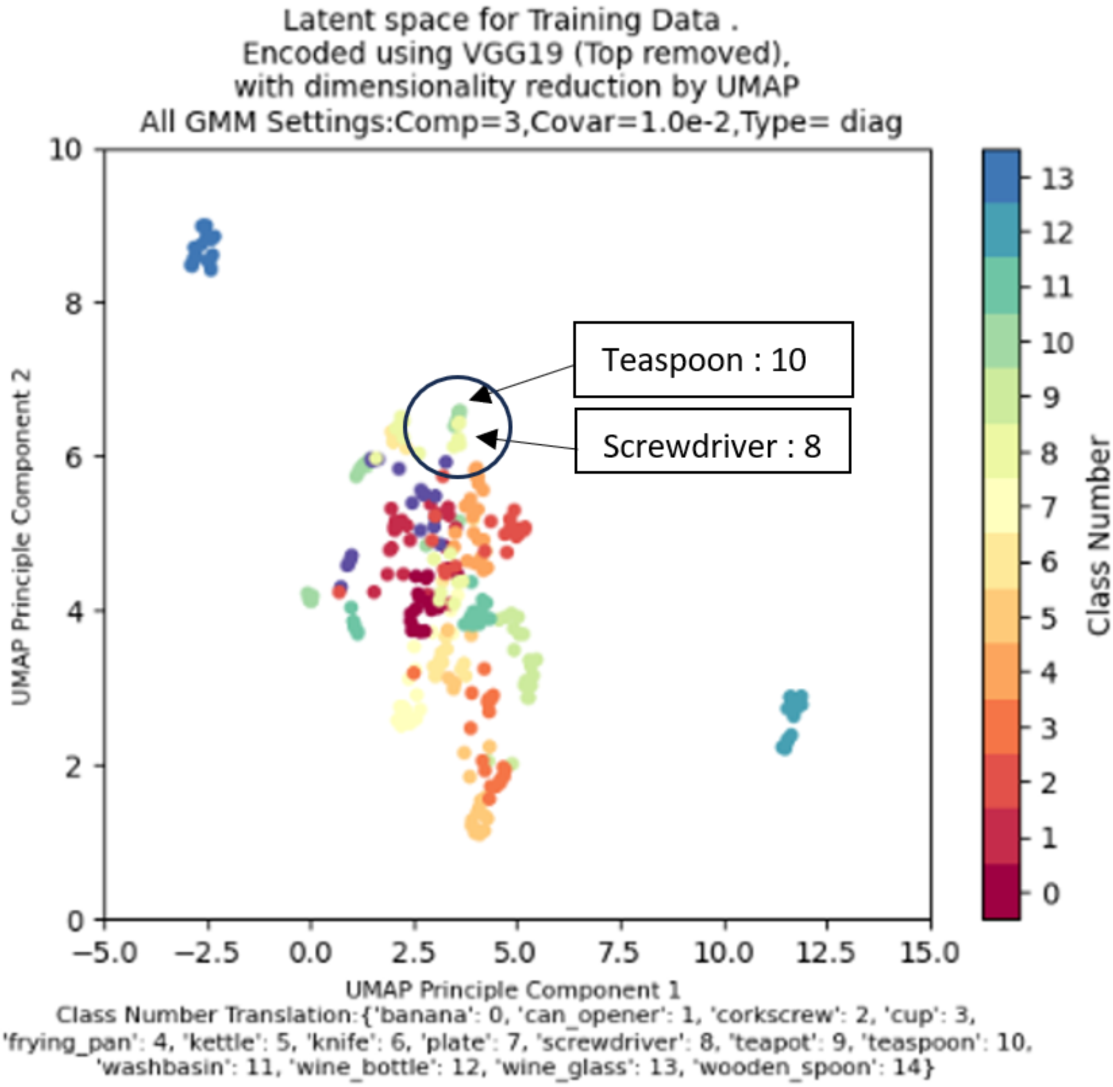


Figure 6.19: The UMAP Projection of the Classification Space.

6.6.4 *Step 6 - Creating ROIs*

The vision pipeline is evaluated using GMMs detailed in Table 6.2. To verify the accuracy of image interpretations, we queried the knowledge graph (KG). Figure 6.20 displays the tested image, and Table 6.7 provides details on each ROI, including object classes with their associated affordances and bounding box coordinates. Figures 6.21 to 6.28 illustrate the likelihood of each GMM class for each ROI. The results successfully identified all details, though the distribution of likelihood values indicated that certain classes were more easily recognised than others.

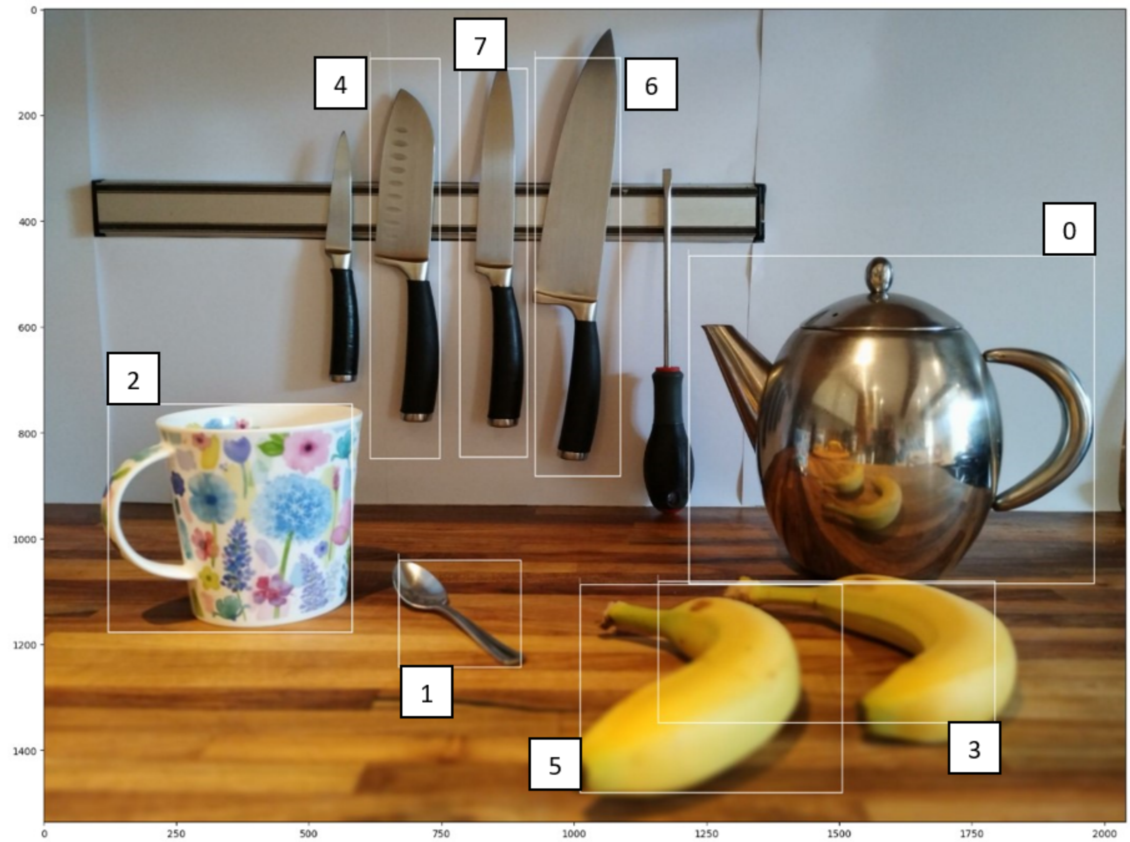


Figure 6.20: KG Synthesis of Information Image.

Name	Object	Affordance	xmin	xmax	ymin	ymax
ROI_0	teapot	None	1216	1981	467	1085
ROI_1	teaspoon	None	669	900	1042	1243
ROI_2	cup	drink from	121	582	747	1177
ROI_3	banana	eat	1011	1506	1088	1480
ROI_4	knife	cut	615	747	94	849
ROI_5	banana	eat	1158	1794	1082	1348
ROI_6	knife	cut	926	1088	93	882
ROI_7	knife	cut	784	912	113	846

Table 6.7: ROI Classification for Figure 6.20

Figure 6.21: GMM Negative likelihood for ROI_0: teapot

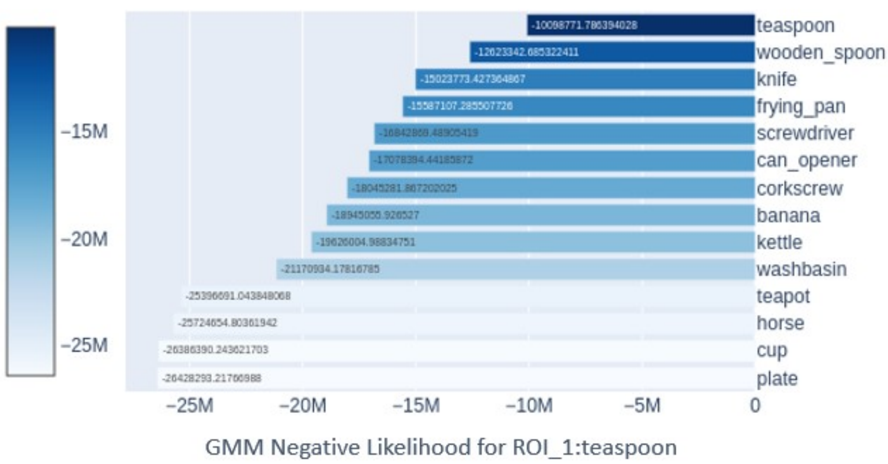


Figure 6.22: GMM Negative likelihood for ROI_1: teaspoon

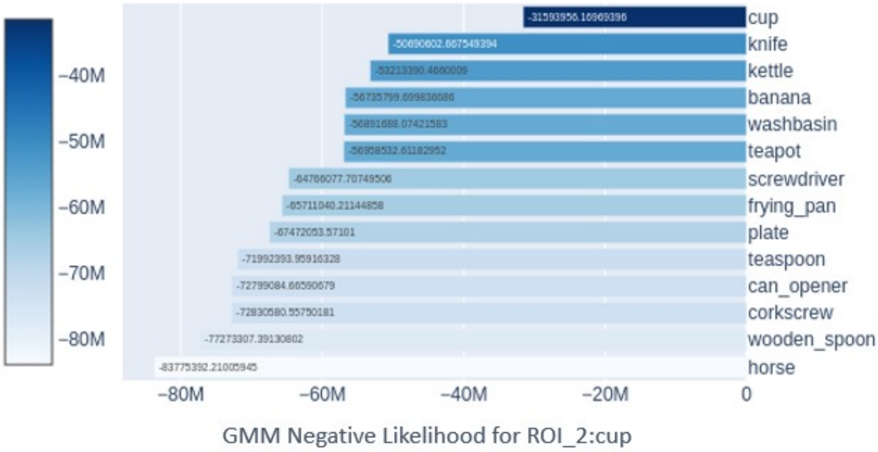


Figure 6.23: GMM Negative likelihood for ROI_2:cup

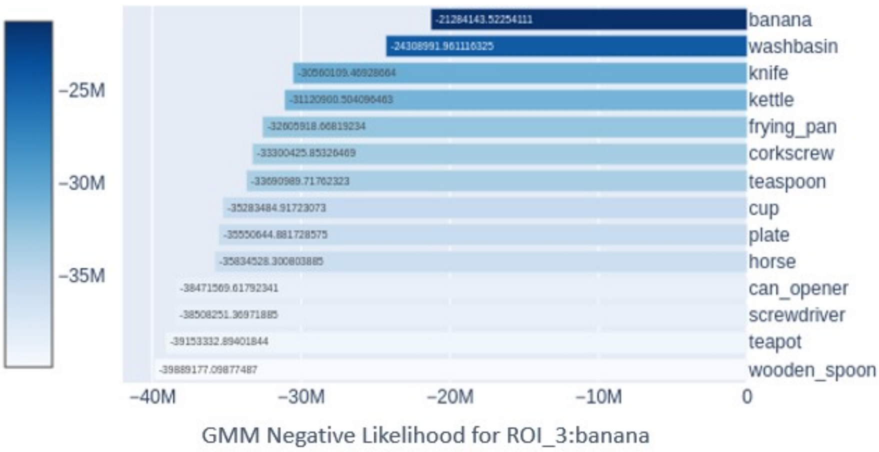


Figure 6.24: GMM Negative likelihood for ROI_3:banana

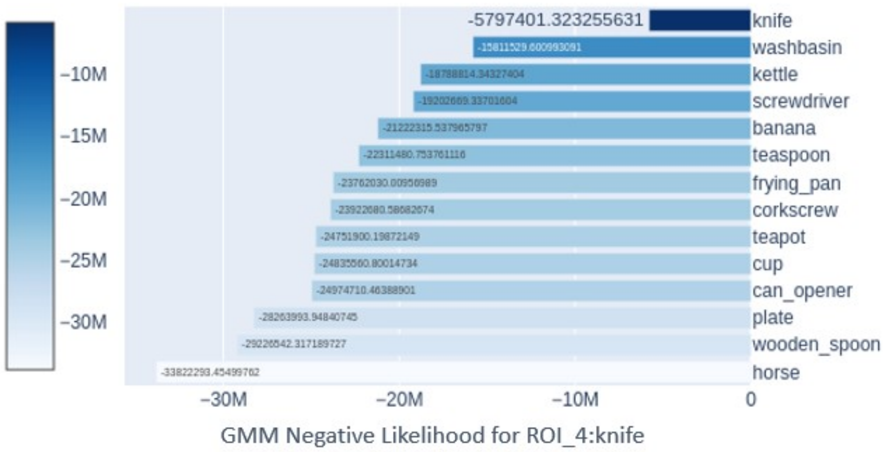


Figure 6.25: GMM Negative likelihood for ROI_4:knife

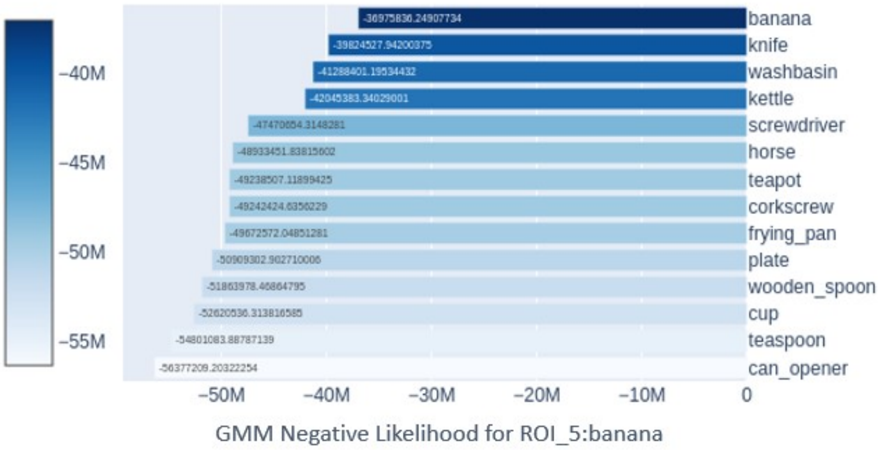


Figure 6.26: GMM Negative likelihood for ROI_5:banana

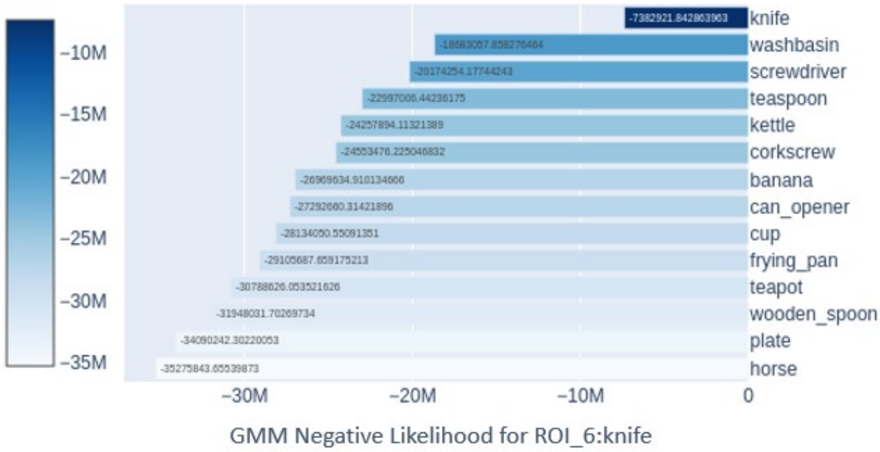


Figure 6.27: GMM Negative likelihood for ROI_6:knife

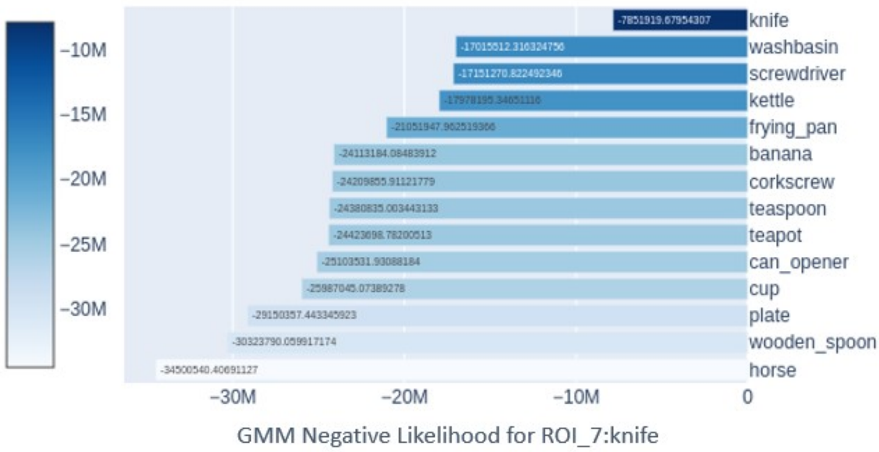


Figure 6.28: GMM Negative likelihood for ROI_7:knife

6.6.5 *Step 7 - Applying Commonsense Rules*

We introduced two new classes, *horse* and *striped*, by training on 20 random images for each. A threshold was set for the *striped* attribute classifier to determine when something was indeed striped. We established a rule, "If an object is classified as *horse* and simultaneously identified with the *striped* attribute, it is to be reclassified as *Zebra*. This methodology enables the creation of new Terms through the combination of existing object and attribute classes. Initial tests identified two horses, as depicted in Figures 6.30 & 6.31. However, one bounding box labeled with the 'striped' attribute was correctly relabeled as 'Zebra', as detailed in Table 6.8. This experiment demonstrates our system's capability to conceptualise a 'Zebra' without directly training on the visual data of the animal.

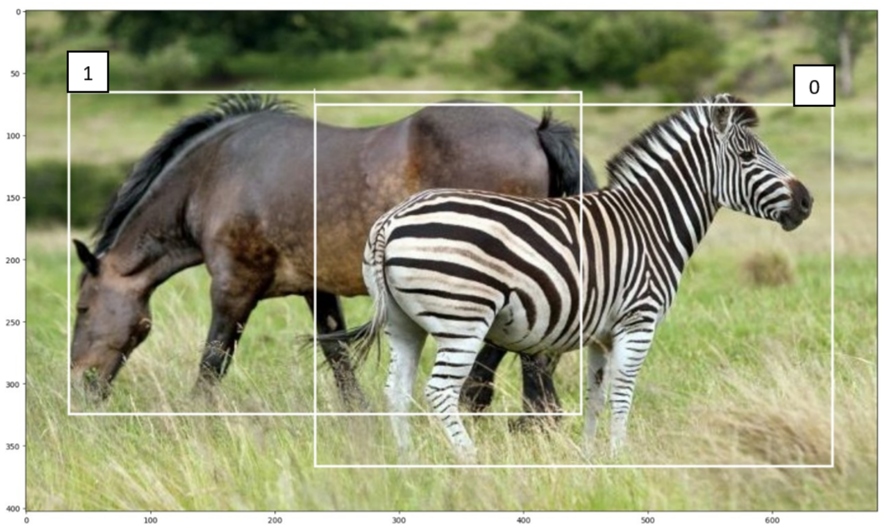


Figure 6.29: Term creation using Commonsense rules image.

Name	Object	Attr.	xmin	xmax	ymin	ymax
ROI_0	zebra	striped	232	649	76	366
ROI_1	horse	None	34	447	66	324

Table 6.8: Labeling Information for Figure 6.29.

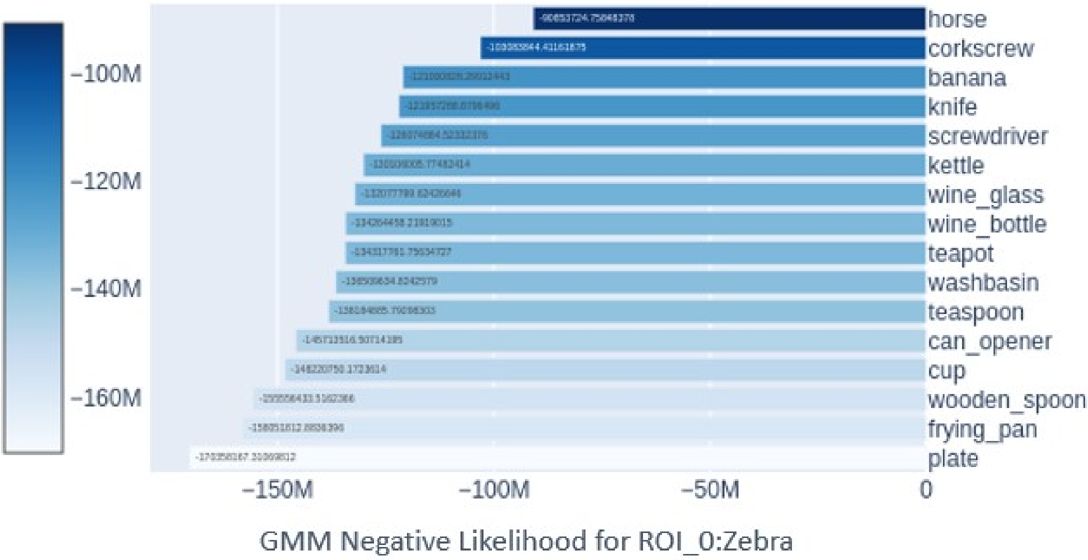


Figure 6.30: Ground truth for Figure 6.29, ROI_0

Figure 6.31: Ground truth for Figure 6.29, ROI_1

6.6.6 *Steps 8 to 10 - Question to Answer*

An experiment was conducted to evaluate the full NS-IL pipeline's efficiency using a test set of five context images. These images included objects in different numbers, positions, and frequencies. Five questions were posed for each image. The created prompt, coding statement and results from executing the statement are recorded. Figure 6.32 summarises the accurate classifications (76.9%), while Figures 6.33 to 6.37 offer detailed results for each test image.

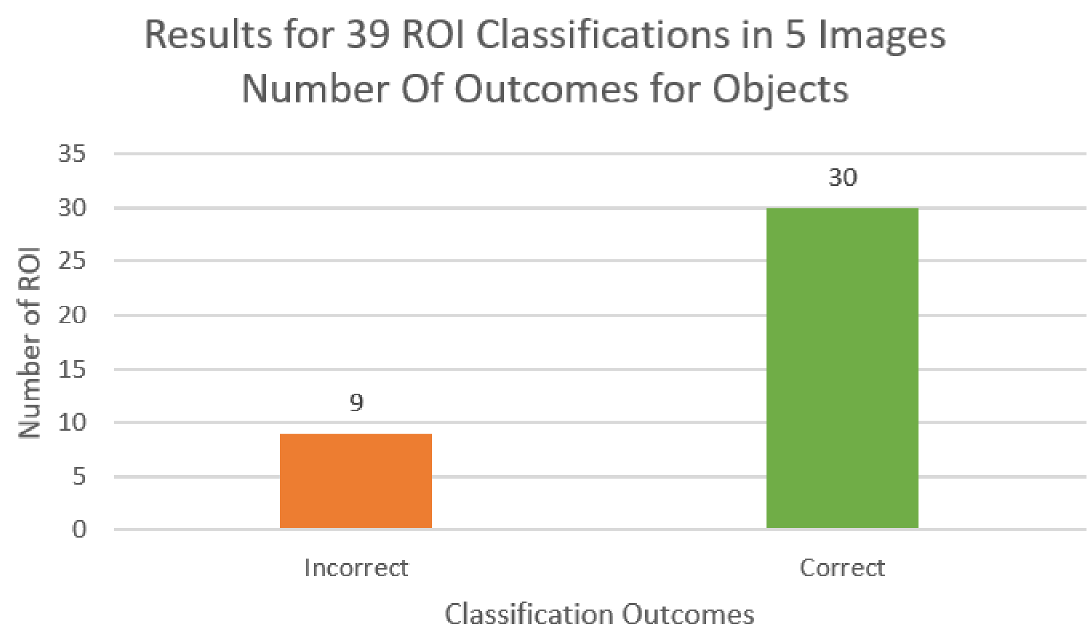
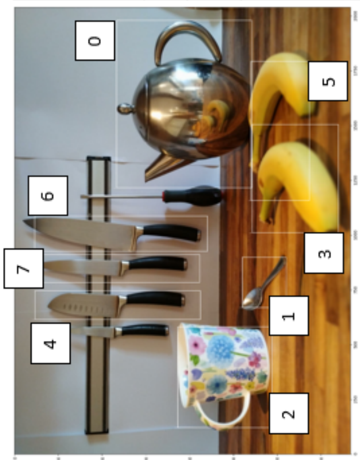


Figure 6.32: VQA Task - Summarised Results for 5 images.

	a. Question / d. Answer		b. Cypher Code / c. Results
1a	What affordance does knife have ?	1b	MATCH (q:ROI{Object_Name:'knife'}) RETURN q.Action
1d	The affordance that a knife has is to cut.	1c	[['q.Action'], ['cut'], ['cut'], ['cut']]
2a	What can I eat ?	2b	MATCH (q:ROI{Action:'eat'}) RETURN q.Object_Name
2d	You can eat a banana	2c	[['q.Object_Name'], ['banana'], ['banana']]
3a	What can I cut ?	3b	MATCH (q:ROI{Action:'cut'}) RETURN q.Object_Name
3d	You can cut with a knife.	3c	[['q.Object_Name'], ['knife'], ['knife'], ['knife']]
4a	What is the relationship between cup and teapot ?	4b	MATCH (c:ROI{Object_Name:'cup'})-[r]->(t:ROI{Object_Name:'teapot'}) RETURN c.Object_Name, r.Value, t.Object_Name
4d	The cup is to the left of the teapot.	4c	[['c.Object_Name', 'r.Value', 't.Object_Name'], ['cup', 'LEFT', 'teapot']]
4a	How many knives are there?	5b	MATCH (k:ROI{Object_Name:'knife'}) RETURN count(k)
5d	There are 3 knives.	5c	[['count(k)'], [3]]

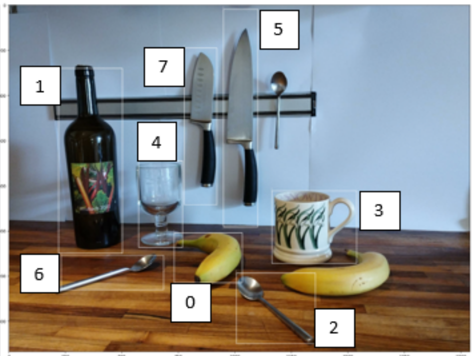


ROI	Object Class	Correct
ROI_0	teapot	Y
ROI_1	teaspoon	Y
ROI_2	cup	Y
ROI_3	banana	Y
ROI_4	knife	Y
ROI_5	banana	Y
ROI_6	knife	Y
ROI_7	Knife	Y

Image A

Figure 6.33: VQA Task for Image_A. Identified Object, Generated Query and Answer.

Image B

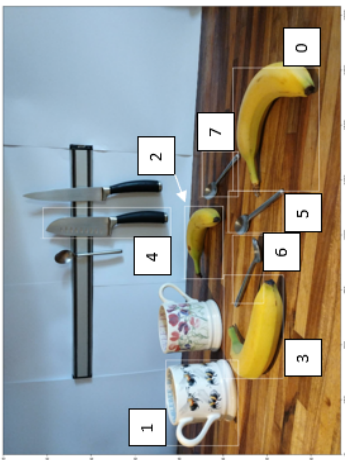


ROI	Object Class	Correct
ROI_0	banana	Y
ROI_1	wine_bottle	Y
ROI_2	teaspoon	Y
ROI_3	kettle	N (Cup)
ROI_4	Wine_glass	Y
ROI_5	knife	**Y**
ROI_6	teaspoon	Y
ROI_7	knife	Y

	a. Question / d. Answer		b. Cypher Code / c. Results
1a	What affordance does a banana have ?	1b	MATCH (q:ROI{Object_Name:'banana'}) RETURN q.Action
1d	The affordance that a banana has is that it can be eaten.	1c	[['q.Action'], ['eat']]
2a	What can I drink from ?	2b	MATCH (q:ROI{Action:'drink from'}) RETURN q.Object_Name
2d	You can drink from a wine glass.	2c	[['q.Object_Name'], ['wine_glass']]
3a	What can I cut ?	3b	MATCH (q:ROI{Action:'cut'}) RETURN q.Object_Name
3d	You can cut with a knife.	3c	[['q.Object_Name'], ['knife'], ['knife']]
4a	Where is the knife in relation to the banana ?	4b	MATCH (k:ROI{Object_Name:'knife'})-[r]->(b:ROI{Object_Name:'banana'}) RETURN k.Object_Name,k.name, r.Value, b.Object_Name,b.name
4d	The knife is positioned behind the banana.	4d	[['k.Object_Name', 'k.name', 'r.Value', 'b.Object_Name', 'b.name'], ['knife', 'ROI_5', 'BEHIND', 'banana', 'ROI_0'], ['knife', 'ROI_7', 'BEHIND', 'banana', 'ROI_0']]
5a	How many bananas are there ?	5b	MATCH (b:ROI{Object_Name:'banana'}) RETURN count(b)
5d	There is 1 banana.	5c	[['count(b)'], [1]]

Figure 6.34: VQA Task for Image_B. Identified Object, Generated Query and Answer.

Image C

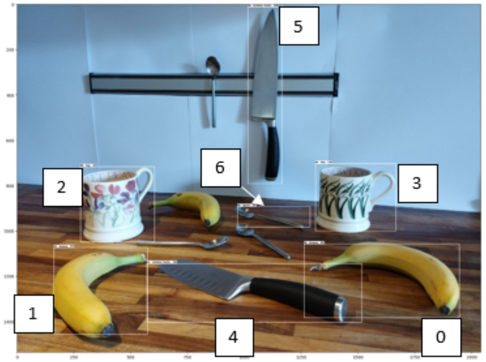


ROI	Object Class	Correct
ROI_0	banana	Y
ROI_1	cup	Y
ROI_2	banana	Y
ROI_3	banana	Y
ROI_4	knife	Y
ROI_5	teaspoon	**Y**
ROI_6	knife	**N (teaspoon)**
ROI_7	banana	Y

	a. Question / d. Answer		b. Cypher Code / c. Results
1a	1. What affordance does a knife have ?	1b	MATCH (q:ROI{Object_Name:'knife'}) RETURN q.Action
1d	The affordance that a knife has is the ability to cut.	1c	[['q.Action'], ['cut'], ['cut']]
2a	2. What can I eat ?	2b	MATCH (q:ROI{Action:'eat'}) RETURN q.Object_Name
2d	You can eat a banana.	2c	[['q.Object_Name'], ['banana'], ['banana'], ['banana']]
3a	3. What can I cut ?	3c	MATCH (q:ROI{Action:'cut'}) RETURN q.Object_Name
3d	You can cut using a knife.	3c	[['q.Object_Name'], ['knife']]
4a	4. What is the relationship between cup and banana ?	4b	MATCH (c:ROI{Object_Name:'cup'})-[r]->(b:ROI{Object_Name:'banana'}) RETURN c.Object_Name, r.Value, b.Object_Name
4d	The relationship between a cup and a banana is that the cup is to the left of the banana.	4c	[['c.Object_Name', 'r.Value', 'b.Object_Name'], ['cup', 'LEFT', 'banana'], ['cup', 'LEFT', 'banana']]
5a	5. How many knives are there ?	5b	MATCH (k:ROI{Object_Name:'knife'}) RETURN count(k)
5d	There is one knife.	5c	[['count(k)'], [1]]

Figure 6.35: VQA Task for Image_C. Identified Object, Generated Query and Answer.

Image D

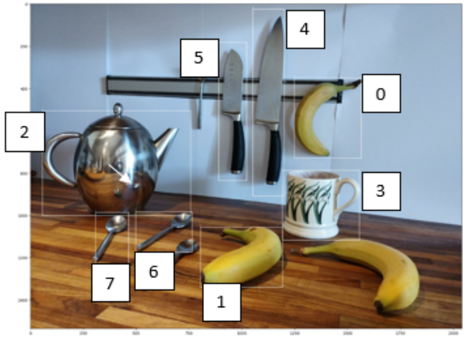


ROI	Object Class	Correct
ROI_0	banana	Y
ROI_1	banana	Y
ROI_2	Kettle	N (Cup)
ROI_3	Kettle	N (Cup)
ROI_4	washbasin	N (Knife)
ROI_5	knife	**Y**
ROI_6	washbasin	N (TeaSpoon)

	a. Question / d. Answer		b. Cypher Code / c. Results
1a	What affordance does a knife have ?	1b	MATCH (q:ROI{Object_Name:'knife'}) RETURN q.Action
1d	The affordance that a knife has is the ability to cut.	1c	[['q.Action'], ['cut']]
2a	What can I eat ?	2b	MATCH (q:ROI{Action:'eat'}) RETURN q.Object_Name
2d	You can eat a banana.	2c	[['q.Object_Name'], ['banana'], ['banana']]
3a	What can I cut ?	3b	MATCH (q:ROI{Action:'cut'}) RETURN q.Object_Name
3d	You can cut using a knife.	3c	[['q.Object_Name'], ['knife']]
4a	What is the relationship between the knife and banana ?	4b	MATCH (k:ROI{Object_Name:'knife'})-[r]->(b:ROI{Object_Name:'banana'}) RETURN k.Object_Name, r.Value, b.Object_Name
4d	The knife is positioned behind and to the left of the banana, and then it is moved to the right of the banana.	4c	[['k.Object_Name', 'r.Value', 'b.Object_Name'], ['knife', 'BEHIND', 'banana'], ['knife', 'LEFT', 'banana'], ['knife', 'BEHIND', 'banana'], ['knife', 'RIGHT', 'banana']]
5a	How many knives are there ?	5b	MATCH (k:ROI{Object_Name:'knife'}) RETURN count(k)
5d	There is one knife	5c	[['count(k)'], [1]]

Figure 6.36: VQA Task for Image_D. Identified Object, Generated Query and Answer.

Image E



ROI	Object Class	Correct
ROI_0	banana	Y
ROI_1	banana	Y
ROI_2	teapot	Y
ROI_3	kettle	**N (Cup)**
ROI_4	knife	Y
ROI_5	Knife	**Y**
ROI_6	washbasin	**N (Teaspoon)**
ROI_7	washbasin	**N (Teaspoon)**

	a. Question / d. Answer		b. Cypher Code / c. Results
1a	What affordance do bananas have ?	1b	MATCH (q:ROI{Object_Name:'banana'}) RETURN q.Action
1d	The affordance that bananas have is that they can be eaten.	1c	[['q.Action'], ['eat'], ['eat']]
2a	What can I drink from ?	2b	MATCH (q:ROI{Action:'drink from'}) RETURN q.Object_Name
2d	You can drink from a cup, glass, bottle, or mug	2c	[['q.Object_Name']]
3a	What can I cut ?	3b	MATCH (q:ROI{Action:'cut'}) RETURN q.Object_Name
3d	You can cut using a knife.	3c	[['q.Object_Name'], ['knife'], ['knife']]
4a	Where is the knife in relation to the banana ?	4b	MATCH (k:ROI{Object_Name:'knife'})-[r]->(b:ROI{Object_Name:'banana'}) RETURN k.Object_Name,k.name, r.Value, b.Object_Name,b.name
4d	The knife is to the left of the banana in ROI_5 and ROI_4, and it is also behind the banana in ROI_5 and ROI_4.	4c	[['k.Object_Name', 'k.name', 'r.Value', 'b.Object_Name', 'b.name'], ['knife', 'ROI_5', 'LEFT', 'banana', 'ROI_0'], ['knife', 'ROI_4', 'LEFT', 'banana', 'ROI_0'], ['knife', 'ROI_5', 'BEHIND', 'banana', 'ROI_1'], ['knife', 'ROI_4', 'BEHIND', 'banana', 'ROI_1']]
5a	5. How many bananas are there ?	5b	MATCH (b:ROI{Object_Name:'banana'}) RETURN count(b)
5d	There are 2 bananas.	5c	[['count(b)'], [2]]

Figure 6.37: VQA Task for Image_E. Identified Object, Generated Query and Answer.

6.6.7 *NS-IL: Disambiguation Proficiency*

Focussing specifically on the NS-ILs proficiency in translating varied phrasing of similar questions into consistent and accurate graph traversal paths. The dataset is formed from:

- The image seen in Figure 6.38.
- The five questions taken from Figure 6.33, are repeated in Table 6.9 to 6.13. For each Main Question shown, ChatGPT-4 creates five linguistic variations.

This creates a test dataset of 25 questions which probe the linguistic ambiguities between the KG and LLM. Each set of five variations is presented to NS-IL in a distinct session, ensuring the model starts afresh, without any residual influence from prior interactions. The answers generated can be seen in Tables 6.9 to 6.13.

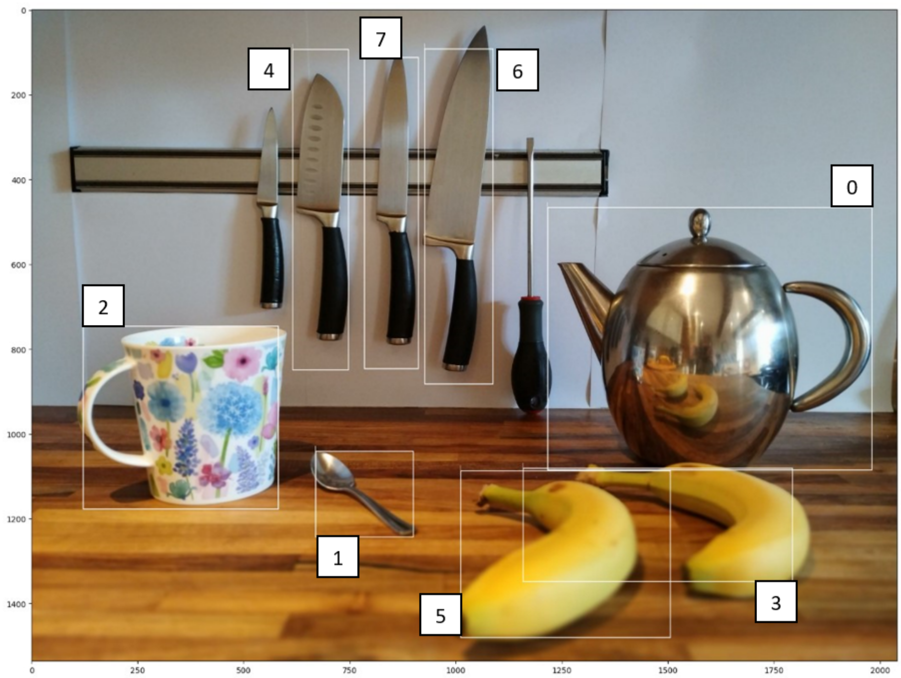


Figure 6.38: NS-IL Disambiguation for Image.

No.	Question & Answer
	Main_Question_A: Based on just the image, what affordance does the knife have?
A1	Q: What's the knife's main affordance in the image? A: The affordance that a knife has is to cut.
A2	Q: In the image, what affordance is evident for the knife? A: The affordance evident for the knife in the image is to cut.
A3	Q: From the image, what's the primary affordance of the knife? A: The primary affordance of the knife from the image is not provided.
A4	Q: Observing the image, what affordance can you associate with the knife? A: The affordance you can associate with the knife, as observed in the image, is to 'cut'.
A5	Q: What affordance can be deduced for the knife in the image? A: The affordance that can be deduced for the knife in the image is that it can be used to cut.

Table 6.9: Questions A: Comparison of the Answer generated by NS-IL of 5 randomly generated questions with the same sense.

No.	Question & Answer
	Main_Question_B: Based on just the image, what can I eat?
A1	Q: From the image, what's edible? A: From the image, the edible item is a banana.
A2	Q: What does the image suggest I can eat? A: The image suggests that you can eat a banana.
A3	Q: Observing the image, what appears ready to eat? A: The image shows that what appears ready to eat.
A4	Q: In the image, what can be consumed? A: The image shows items that can be consumed.
A5	Q: Based on the image, what seems suitable to eat? A: It seems that a banana would be suitable to eat.

Table 6.10: Questions B: Comparison of the Answer generated by NS-IL of 5 randomly generated questions with the same sense.

No.	Question & Answer
	Main_Question_C: Based on just the image, what can I cut?
A1	Q: Observing the image, what's cuttable? A: Based on the image you observed, the item that is cuttable is a knife.
A2	Q: What does the image indicate can be cut? A: The image indicates that a knife can be cut.
A3	Q: In the provided image, what seems ready for cutting? A: The image indicates that it's ready for cutting.
A4	Q: What in the image appears to be sliceable with a knife? A: The bread.
A5	Q: From the image, what looks like it can be cut? A: From the image, it appears that a knife can be cut.

Table 6.11: Questions C: Comparison of the Answer generated by NS-IL of 5 randomly generated questions with the same sense.

No.	Question & Answer
	Main_Question_D: Based on just the image, what is the relationship between the cup and teapot?
A1	Q: How does the image portray the cup in relation to the teapot? A: The image portrays the cup, identified as ROI_2, to the left of the teapot, identified as ROI_0.
A2	Q: In the image, how is the cup associated with the teapot? A: In the image, the cup is positioned to the left of the teapot.
A3	Q: Observing the image, what connection is evident between the cup and teapot? A: The connection evident between the cup and the teapot in the image is that the cup is to the left of the teapot.
A4	Q: What's the depicted relationship of the cup and teapot in the image? A: In the image, the cup is depicted to the left of the teapot.
A5	Q: Based on the image, how do the cup and teapot interact? A: The cup is positioned to the left of the teapot based on the image.

Table 6.12: Questions D: Comparison of the Answer generated by NS-IL of 5 randomly generated questions with the same sense.

No.	Question & Answer
	Main_Question_E: Based on just the image, how many knives are there?
A1	Q: How many knives does the image show? A: The image shows 3 knives.
A2	Q: Counting in the image, how many knives can you see? A: In the image, you can see three knives.
A3	Q: Observing the image, how many knives are depicted? A: The image depicts three knives.
A4	Q: From the image, how many knives are visible? A: From the image, you can see that there are 3 knives visible.
A5	Q: In the provided image, how many instances of knives can be found? A: In the provided image, there are 3 instances of knives.

Table 6.13: Questions E: Comparison of the Answer generated by NS-IL of 5 randomly generated questions with the same sense.

For most variations, the answers are consistent across the five different phrasings. For instance, in Set B, the consistent answer is that a banana is edible. Such consistency indicates that the model understands the core intent of the question, irrespective of its phrasing. In Set C, there's an inconsistency. The main question and its variations are asking what in the image can be cut, but several answers indicate that a knife itself can be cut which is a misinterpretation. In Set A, the model returned non-specific answers like "not specified" or "not provided" for some variations. This could indicate that certain phrasings caused ambiguity or that the image did not provide a clear context for the knife's main affordance. Looking at the cypher code it shows that instead of using the term 'Action', the use of 'Main_Affordance' and 'Primary_Affordance' caused no results to be returned from the KG. In Set D, all five variations aim to understand the spatial relationship between the cup and teapot. The model consistently identified the cup as being to the "left" of the teapot, indicating a strong semantic understanding of spatial relationships in the image. For Set E, the model consistently identified "3 knives," indicating reliable count-based object detection across varied phrasings. In some cases, like in Set C, the change in phrasing led the model to provide syntactically correct but semantically incorrect answers. For instance, saying "a knife can be cut" is a syntactically correct sentence but doesn't make semantic sense in most contexts.

Whilst A4 in Set C showed that ChatGPT-4 hallucinations have not been eliminated. As when asked, "What appears sliceable with a knife?" it answers "The bread" which is not an object in the image, however sliceable was also not in the ontology, showing how important the terminology used is. The answer showing the most potential was A1 in Set D which provided specific details like "identified as ROI_2" and "identified as ROI_0," which suggests the model can identify and tag different regions of interest in the image. To investigate if we could reuse this prompt, we submitted Image_C with the question "1. How does the image portray the cup in relation to the teaspoon?" to which the Answer was, "The image portrays the cup in the 'ROI_1' position to the left of the teaspoon in the 'ROI_7' and 'ROI_5' positions". Checking the KG created these, they were indeed the tags allocated to these objects.

6.6.8 *NS-IL: Benchmarked against ChatGPT4-V*

In these experiments, we aim to assess the efficacy of two distinct models in an image driven, question and answer context. First, we have our Neuro-Symbolic model, denoted as NS-IL which employs ChatGPT-4 as a Q&A interface for users and derives its answers from a knowledge graph. In contrast, we have ChatGPT4-V, an iteration of ChatGPT-4 specifically trained to interpret an image alongside a query and generate an answer independently, without the assistance of any external knowledge graphs or databases. NS-IL and ChatGTP4-V are discussed qualitatively and also quantitatively. The dataset we use is Figure 6.38 again, together with 5 questions shown in Table 6.14.

Qualitative comparison

The Image alongside the questions are presented to both models. To maintain consistency, ChatGPT4-V was initialised in a separate chat session for each question set, ensuring that no prior chat history influenced the outcome. The directive given to ChatGPT4-V was to solely utilise the associated image for each query. We then evaluated the accuracy of the responses based on human interpretation. Through a qualitative assessment, we analysed the quality and relevance of the answers to gain a deeper understanding of the pros and cons of each model.

No.	Question & Answer
1	Q1: Based on just the image, what affordance does the knife have? ChatGPT_V_A1: The knife has the affordance of cutting or slicing. NS_IL_A1: The affordance that a knife has is to cut.
2	Q2: Based on just the image, what can I eat? ChatGPT_V_A2: You can eat the bananas. NS_IL_A2: You can eat a banana.
3	Q3: Based on just the image, what can I cut? ChatGPT_V_A3: You can cut the bananas with the knife. NS_IL_A3: You can cut with a knife.
4	Q4. Based on just the image, what is the relationship between the cup and teapot? ChatGPT_V_A4: The relationship between the cup and teapot is that the teapot can be used to pour a beverage, likely tea, into the cup. NS_IL_A4: The relationship between a cup and a teapot is that the cup is to the left of the teapot.
5	Q5: Based on just the image, how many knives are there? ChatGPT_V_A5: There are five knives in the image. NS_IL_A5: There are 3 knives.

Table 6.14: Using the Image in Figure 6.38, Comparing Question & Answers for two models NS-IL (with KG) & ChatGPT4-V (without KG).

Table 6.14 shows the Answers from both models. Note that the NS-IL system will only know about objects in the identified bounding box which can be seen in Figure 6.38. ChatGPTs prior of common sense can be seen to be advantages in the information used to know in Q3, that knives cut bananas, and in Q4, that teapots are used to pour beverages. It can also be said that NS-IL has only used the information held in the knowledge graph and therefore in Q4 has used the relationship based on location. It should also be noted that NS-IL also knows the exact pixel location where an object is placed in the image and what time it was seen, together with the ability to identify each object individually. All answers were acceptable.

Quantitative Comparison

In this experiment, we conduct a comparative analysis between NS-IL and ChatGPT4-V. The experiment involves assessing the answers generated by these models for the set of five images and associated questions found in Section 6.6.6. We quantitatively evaluated the generated answers in terms of the following categories:

- Factual Correctness, as determined by human judgment,
- Factual Incorrectness, as determined by human judgment.
- The correct incorporation of commonsense knowledge in addition to the visual content of the image.
- The inappropriate use of commonsense knowledge about the image.

The results are presented in Figure 6.39, showing answer accuracy and their associated 95% confidence interval. The results indicate NS-IL outperformed ChatGPT4-V in terms of factual correctness, indicating that NS-IL generated responses that were more factually accurate as deemed by a human assessor. Conversely, NS-IL also exhibited a higher rate of factual incorrectness compared to ChatGPT4-V. ChatGPT4-V consistently demonstrated the application of commonsense knowledge alongside the image content, while neither model displayed instances of incorrect commonsense usage. It is worth noting that NS-IL exhibited lower variance in confidence levels across its responses when compared to ChatGPT4-V.

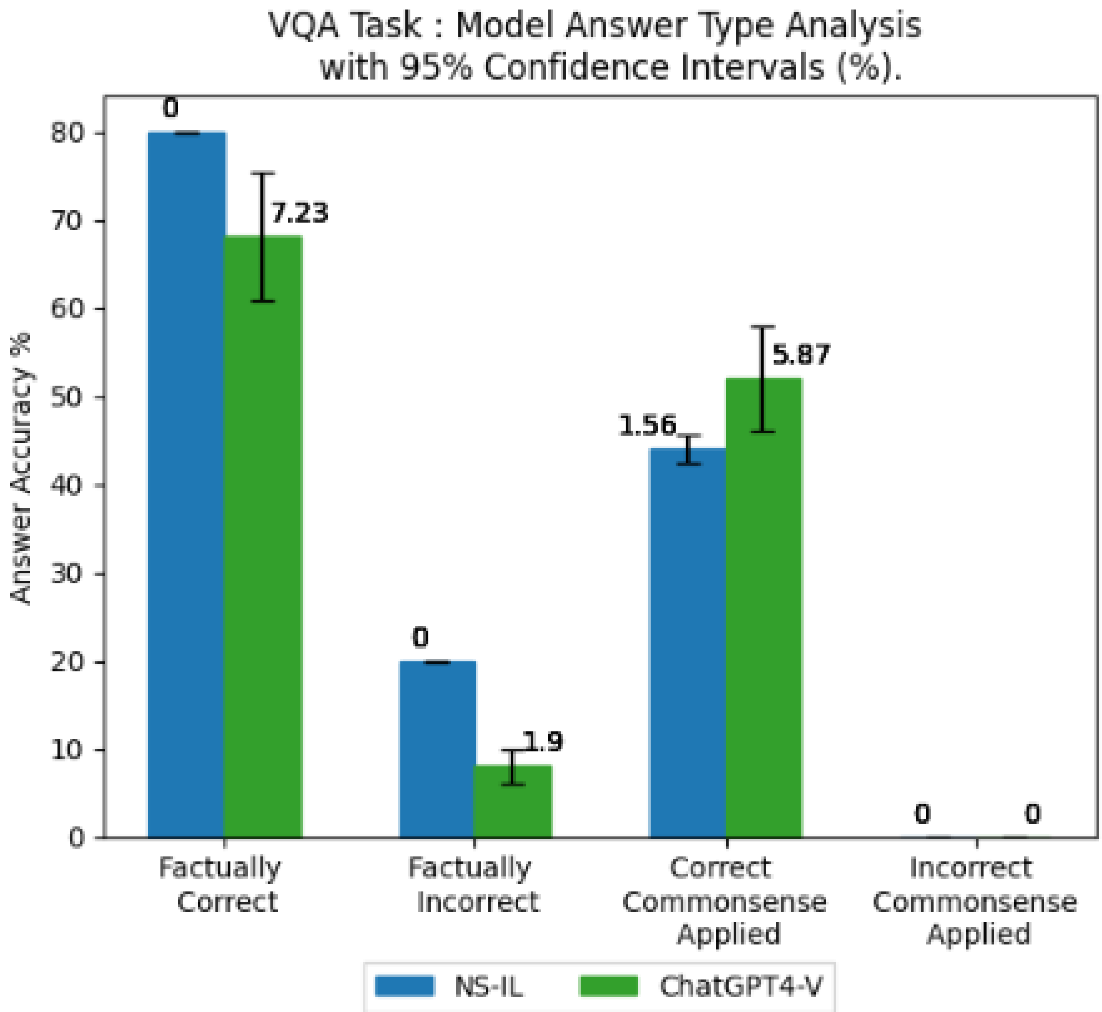


Figure 6.39: Answer Analysis for NS-IL & ChatGPT4-V using a dataset of 25 questions and 5 images.

6.7 Limitations

The system was created using a small dataset of classes, further research needs to be carried out to test its capacity to scale and handle a more comprehensive and dynamically changing knowledge space. Automated Object and Attribute class creation, could be achieved by utilising an existing VQA dataset such as Visual Genome (101,174 images from MSCoco with 1.7 million QA pairs).

6.8 Discussion

We create a structure of 'Things' that humans internally think about, they are bounded as regions of interest, identified and classified within a world model that holds the *What* and *Where* information. Through *Questions* and *Answers* about their immediate environment, humans are helped to decide their next actions in the world. The AI representations are strategically aligned with human concepts to enable a useful communication tool. By externalising the independent training of new classes using limited sample sizes from the NN, the proposed system effectively addresses the issue of catastrophic forgetting inherent in neural networks. While unsupervised learning in the creation of the feature space is possible using Autoencoders, our study acknowledges that achieving a conceptually valuable feature space necessitates supervised learning. One potential improvement is to investigate the formation of the feature space, training the image encoder on a dataset that contains classes such as 'pointy,' 'corner, 'straight,' and 'circle' to enable an improved feature set.
Setting specific likelihood thresholds for each GMM class could enhance classification accuracy, with classifications only made if the likelihood surpasses its designated threshold. Such tweaks could bolster our architecture's performance and adaptability across various tasks.

In question-answering systems, foundational models such as large language models(LLM) present distinct advantages, such as no training needed. However, when it comes to precision, consistency, and structured data management, KGs undoubtedly take the lead. These graphs offer a structured data format that ensures a high level of data integrity and reliability, with explicitly mapped-out relationships between entities. This structure aids the precise delivery of answers to complex relational queries. On the other hand, while LLMs exhibit adaptability to diverse and unstructured queries and can process multimodal inputs, they lack the inherent data organisation and domain-specific tailoring that KGs can offer. KGs' scalability and auditability set them apart, making them an optimal choice for contexts requiring systematic data retrieval and transparency of decision-making.

7

Summary and Conclusions

7.1 Summary

This research investigated the development of a Neuro-Symbolic Incremental Learner (NS-IL) tailored for Visual Question Answering (VQA), with the aim of aiding individuals with visual impairments. The NS-IL model, devised as a solution, utilised Gaussian Mixture Models (GMMs) and neural networks to incrementally acquire visual definitions for terms. These GMMs are integrated into a Knowledge Graph (KG), establishing grounded memory and enabling natural language querying through a Large Language Model (LLM).

An ontology is employed to align terms used across the system, enhancing organisation and retrieval processes through structured key:value storage. This alignment ensures precise granularity of each term throughout the system.

The effectiveness of the system relies on understanding the context of posed questions, a challenge currently being tackled through various LLM prompt engineering and fine-tuning methods. It's crucial to ensure that retrieved values are stored within their original context.

Central to system accuracy is the quality and utility of the visual Feature Space, which must effectively categorise images into associated terms. Achieving this involves careful collection of training images implicitly implying the class, or explicit annotation of images with the class. Further research is needed to understand the visual features and categories humans use to describe the world, facilitating alignment in language use and enabling meaningful communication with Digital Assistants.

7.2 Conclusion

Our initial hypothesis was that, 'the use of independent Gaussian Mixture Models as a mechanism to link Neural Nets to a Symbolic memory, will lead to a more flexible Visual Question Answering Task'. Reflecting upon our research questions, these are our findings:

- To support independent GMM incremental learning specific to visual classes in traditional neural networks, several architectural modifications are required. As demonstrated in Chapter 4, our approach is to separate the task into two distinct parts: learning the principal visual features and learning the class definition. This separation of tasks allows the system to effectively learn new classes **without any catastrophic forgetting issues**, which is a notable advantage over standard neural network architectures. Once the principal visual features are learned, they can be reused for new classes **without retraining the image encoder**. The experiments highlighted that a GMM consistently **outperformed the Softmax function for sample sizes smaller than 12**. In fact, based on three class imbalance experiments, the GMM demonstrated superior performance compared to the Softmax when trained on sample sizes under 15.

- Through the addition of a KG (Memory), a **flexible question-and-answer functionality** was achieved. The GMMs were essential to the **grounding of the KG** through there linkage to the Neural Nets. The LLM enabled natural language Q&A which was found to be **more consistent that ChatGPT-V**. The system **learns visual representations neurally and makes them available symbolically**, thereby facilitating logical reasoning capabilities. A schema **defines how Meaning is attributed to Terms from visual regions of interest**.

The research intent is to abstract visual meaning from training images and assign a Term. This Term should be one that the wider population has reached a consensus for. This then affords us the ability to use language to discuss and query a reality we don't physically have to be present in. Our compositional ML model, called NS-IL combines an Image Encoder, Gaussian Mixture Models, a Knowledge Graph, and a Large Language Model, creating a novel end-to-end VQA system.

In conclusion, the proposed NS-IL architecture represents a holistic approach to cognitive AI tasks, combining the strengths of neural networks and symbolic AI. With the ability to incrementally learn classes, the system can evolve with the environment. While challenges remain, the potential benefits in terms of adaptability and versatility make it a compelling direction for future research and application.

8

Future Possibilities

8.1 A Passive Agent for the Visual Impaired

The NS-IL architecture was a first step towards creating a passive AI Assistant to help those that need it. Taking inspiration from the blog available at https://blog.dileeplearning.com/p/ingredients-of-understanding, Figure 8.1 illustrates how the Neuro-Symbolic Incremental Learner model can be conceptualised as a passive Agent designed for interaction with humans or other Agents. The question, answering ability is adapted to enable dialogue / message passing and the image functionality extends into video. The current system is based purely on semantic knowledge and images, this could be extended with additional multi-modal signals to build a Sensorimotor world model as an engine to simulations. Which enables counterfactual reasoning through hyperparameter searches attuned to identified human goals. The Visual Dialogue Task, allows the Assistant to pose questions and on receiving answers adjust the contents of its knowledge graph (configuration state) to match both the environment and the user's context. It is worth noting that whilst the human has a fixed capability the AI Assistant has a scalable capability, but could also potentially message with other Agents to access 'others' knowledge and teach each other through continual learning mechanisms. Ethics should be key to any implementation such as this, together with how the patient's questions and answers are stored. As AI shifts towards personalisation, the context of terms/-words will play a pivotal role. The capability to tailor systems to individual needs

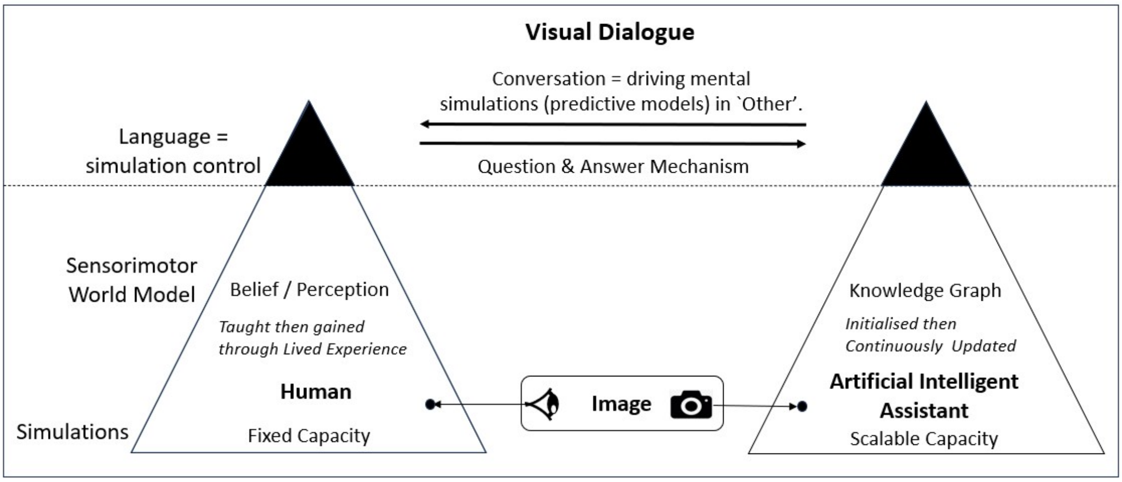


Figure 8.1: Visual dialogue between an AI Assistant and Human.

will highlight what structure they contain and the depth of the concepts held. As LLM and Language Vision Model (LVM) research advances, clearer strategies for connecting sentence structures, categories, terms, as well as prompt engineering, will mature and become clearer. This will also guide efforts to mitigate model generated misconceptions or "hallucinations". Leveraging a question-answer framework can be beneficial for both enhancing machine learning and providing valuable information to a system. A natural progression will be to include additional datasets and allow seeing users to augment their vision with location and topic specific knowledge that they currently do not have.

8.2 Next Steps

With the benefit of hindsight, the adjustments that could be made and the identified improvements are :

- To research the internal structure of the base feature space and how to align it with the dependent downstream categorical classes. An example would be a dataset containing images and associated feature classes such as 'pointy,' 'corner,' 'straight,' and 'circle' and evaluate if visually similar images cluster together in the feature space, then test to see if GMM classification accuracy is higher.
- Setting specific likelihood thresholds for each GMM class could enhance classification accuracy, with classifications only made if the likelihood surpasses its

designated threshold. Such tweaks could help our architecture's performance and adaptability across various tasks.

- The inferred concepts within the KG hold a seen likelihood, further research is needed to investigate how to incorporate these likelihoods into logical inferences to aid logical reasoning.

- This thesis describes the creation of symbolic definitions for regions of interest. The process through which the world is abstracted into classes and categories should be investigated to improve the principled creation of Ontologies and hence alignment with humans to reduce misunderstandings.

Bibliography

[1] Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. Unifying Large Language Models and Knowledge Graphs: A Roadmap. *arXiv*, 2306.08302, 2023.

[2] Hao Liu, Yong Zhou, Bing Liu, Jiaqi Zhao, Rui Yao, and Zhiwen Shao. Incremental learning with neural networks for computer vision : a survey. *Artificial Intelligence Review*, 56(May):4557 –4589, 2023.

[3] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pages 1–14, 2015.

[4] Rupert R.A. Bourne, Seth R. Flaxman, Tasanee Braithwaite, Maria V. Cicinelli, Aditi Das, Jost B. Jonas, and Et Al. Magnitude, temporal trends, and projections of the global prevalence of blindness and distance and near vision impairment: a systematic review and meta-analysis. *The Lancet Global Health*, 5(9):e888–e897, 2017.

[5] Tom B Brown, Jared Kaplan, Nick Ryder, Tom Henighan, Mark Chen, Ariel Herbert-voss, Daniel M Ziegler, Gretchen Krueger, Amanda Askell, Christopher Hesse, Sam Mccandlish, and Et Al. Language Models are Few-Shot Learners. *Advances in neural information processing systems*, pages 1877–1901, 2020.

[6] Liunian Harold Li, Mark Yatskar, Cho-jui Hsieh, and Kai-wei Chang. VisualBERT: A Simple and Performant Baseline for Vision and Language. *arXiv*, 1908.03557(2):1–14, 2019.

[7] Text-to-text Transformer, Colin Raffel, and Peter J Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *arVix*, 1, 2023.

[8] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ili, and Danielet al Hesslow. BLOOM: A 176B-Parameter Open-Access Multilingual Language Model. *arXiv preprint arXiv*, 2211.05100(Nov), 2022.

[9] Long Ouyang, Pamela Mishkin, Jeff Wu, Jacob Hilton, Amanda Askell, Paul Christiano, and Ryan Lowe. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

[10] Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. Visual ChatGPT: Talking, Drawing and Editing with Visual Foundation Models. *arXiv*, 2303.04671, 2023.

[11] Jacob Devlin, Ming Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *Association for Computational Linguistics*, 1:4171–4186, 2019.

[12] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv*, 1907.11692, 2019.

[13] Hugo Touvron, Thibaut Lavril, Xavier Martinet, Marie-anne Lachaux, Timothee Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, and Guillaume Lample. LLaMA : Open and Efficient Foundation Language Models. *arXiv preprint arXiv*, 2302.13971, 2023.

[14] Emily M Bender and Alexander Koller. Climbing towards NLU : On Meaning , Form , and Understanding in the Age of Data. *Proceedings of the 58th annual meeting of the association for computational linguistics*, 1(2):5185–5198, 2020.

[15] Chaoning Zhang, Chenshuang Zhang, Sheng Zheng, Yu Qiao, Chenghao Li, Mengchun Zhang, Sumit Kumar Dam, Chu Myaet Thwal, Ye Lin Tun, Le Luang Huy, Sung-ho Bae, Lik-hang Lee, Yang Yang, Heng Tao Shen, In So Kweon, and Choong Seon Hong. A Complete Survey on Generative AI (AIGC): Is ChatGPT from GPT-4 to GPT-5 All You Need? *arXiv preprint arXiv*, 2303.11717, 2023.

[16] Tianyi Li, Mohammad Hosseini, Sabine Weber, and Mark Steedman. Language Models Are Poor Learners of Directional Inference. *arXiv*, 2210.04695, 2022.

[17] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, and Et Al. Chain-of-Thought Prompting Elicits Reasoning in Large Lan-

guage Models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

[18] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Emily Reif, Paul Barham, and Jameset al Bradbury. PaLM : Scaling Language Modeling with Pathways. *Journal of Machine Learning Research*, 24.240:1–113, 2023.

[19] Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyan Xu, and Yongfeng Zhang. Natural Language is All a Graph Needs. *arXiv*, 2308.07134, 2023.

[20] Liunian Harold Li, Pengchuan Zhang, Haotian Zhang, Jianwei Yang, Chunyuan Li, Yiwu Zhong, and Et Al. Grounded Language-Image Pre-training. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10965–10975, 2022.

[21] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models. 2022.

[22] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual Instruction Tuning. (NeurIPS):1–25, 2023.

[23] Xi Chen, Xiao Wang, Soravit Changpinyo, AJ Piergiovanni, Piotr Padlewski, Daniel Salz, Sebastian Goodman, Adam Grycner, Basil Mustafa, Lucas Beyer, Alexander Kolesnikov, Joan Puigcerver, Nan Ding, Keran Rong, Hassan Akbari, Gaurav Mishra, Linting Xue, Ashish Thapliyal, James Bradbury, Weicheng Kuo, Mojtaba Seyedhosseini, Chao Jia, Burcu Karagol Ayan, Carlos Riquelme, Andreas Steiner, Anelia Angelova, Xiaohua Zhai, Neil Houlsby, and Radu Soricut. PaLI: A Jointly-Scaled Multilingual Language-Image Model. *arXiv:2209.06794v4*, pages 1–33, 2023.

[24] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling Vision Transformers. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2022-June:12094–12103, 2022.

[25] Alec Radford, Jong Wook Kim, Hallacy Chris, Ramesh Aditya, Goh Gabriel, Agarwal Sandhini, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models

From Natural Language Supervision. *In International conference on machine learning PMLR*, pages 8748–8763, 2021.

[26] Jacob Menick and Felix Hill. Multimodal Few-Shot Learning with Frozen Language Models. *Advances in Neural Information Processing Systems*, 34:200–212, 2021.

[27] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, and Yanaet al Hasson. Flamingo: a Visual Language Model for Few-Shot Learning. *Advances in Neural Information Processing Systems*, 35:23716–23736, 2022.

[28] Anthony Meng Huat Tiong, Junnan Li, Boyang Li, Silvio Savarese, and Steven C.H. Hoi. Plug-and-Play VQA: Zero-shot VQA by Conjoining Large Pretrained Models with Zero Training. *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 951–967, 2022.

[29] Woojeong Jin, Yu Cheng, Yelong Shen, Weizhu Chen, and Xiang Ren. Low-resource Prompt-based Learning for Vision-Language Models. *Association for computational Linguistics*, 1:2763–2775, 2022.

[30] Tsung Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8693 LNCS(PART 5):740–755, 2014.

[31] Aishwarya Agrawal, Jiasen Lu, Stanislaw Antol, Margaret Mitchell, C Lawrence Zitnick, Dhruv Batra, and Devi Parikh. VQA : Visual Question Answering. pages 1–25.

[32] Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. Visual7W: Grounded question answering in images. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-Decem:4995–5004, 2016.

[33] Justin Johnson, Li Fei-Fei, Bharath Hariharan, C. Lawrence Zitnick, Laurens Van Der Maaten, and Ross Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. *Proceedings - 30th IEEE*

Conference on Computer Vision and Pattern Recognition, CVPR 2017, 2017-Janua:1988–1997, 2017.

[34] Drew A Hudson and Christopher D Manning. GQA: A New Dataset for Real-World Visual Reasoning and Compositional Question Answering.

[35] Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. OK-VQA : A Visual Question Answering Benchmark Requiring External Knowledge. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3190–3199, 2019.

[36] Danna Gurari, Qing Li, Abigale J. Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P. Bigham. VizWiz Grand Challenge: Answering Visual Questions from Blind People. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3608–3617, 2018.

[37] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as Agents. pages 1–55, 2023.

[38] Kexin Yi, Antonio Torralba, Jiajun Wu, Pushmeet Kohli, Chuang Gan, and Joshua B. Tenenbaum. Neural-symbolic VQA: Disentangling reasoning from vision and language understanding. *Advances in Neural Information Processing Systems*, 31:1031–1042, 2018.

[39] Jaemin Cho, Jie Lei, Hao Tan, and Mohit Bansal. Unifying Vision-and-Language Tasks via Text Generation. *International Conference on Machine Learning*, 139:1931–1942, 2021.

[40] Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B. Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. *7th International Conference on Learning Representations, ICLR 2019*, pages 1–28, 2019.

[41] Ramakrishna Vedantam, Karan Desai, Stefan Lee, Marcus Rohrbach, Dhruv Batra, and Devi Parikh. Probabilistic Neural-symbolic Models for Interpretable

Visual Question Answering. *Internation conference on machine learning*, 1:6428–6437, 2019.

[42] Michael McCloskey and Neal J. Cohen. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. *Psychology of learning and motivation*, 24(C):109–165, 1989.

[43] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences of the United States of America*, 114(13):3521–3526, 2017.

[44] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A Continual Learning Survey: Defying Forgetting in Classification Tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3366–3385, 2021.

[45] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in Neural Information Processing Systems*, 30:6468–6477, 2017.

[46] Arslan Chaudhry, Puneet K. Dokania, Thalaiyasingam Ajanthan, and Philip H.S. Torr. *Riemannian Walk for Incremental Learning: Understanding Forgetting and Intransigence.* Springer International Publishing, 2018.

[47] Marc Masana, Xialei Liu, Bartlomiej Twardowski, Mikel Menta, Andrew D. Bagdanov, and Joost van de Weijer. Class-Incremental Learning: Survey and Performance Evaluation on Image Classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5513–5533, 2022.

[48] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks : A review. *Neural Networks*, 113:54–71, 2019.

[49] Yong Luo, Liancheng Yin, Wenchao Bai, and Keming Mao. An appraisal of incremental learning methods. *Entropy*, 22(11), 2020.

[50] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, and Et Al. A continual learning survey : Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–85, 2021.

[51] Zhizhong Li and Derek Hoiem. Learning without Forgetting. *European Conference on Computer Vision*, 9908:614–629, 2016.

[52] Sylvestre Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental classifier and representation learning. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Jan:5533–5542, 2017.

[53] Abhishek Aich. Elastic Weight Consolidation (EWC): Nuts and Bolts. *arXiv*, 2105.04093:1–7, 2021.

[54] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory Aware Synapses: Learning What (not) to Forget. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11207 LNCS:144–161, 2018.

[55] Alexander V. Terekhov, Guglielmo Montone, and J. Kevin O'Regan. Knowledge transfer in deep block-modular neural networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9222(July):268–279, 2015.

[56] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive Neural Networks. *Computer Science*, 1(June), 2016.

[57] Jonathan Schwarz, Jelena Luketina, Wojciech M. Czarnecki, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. *35th International Conference on Machine Learning, ICML 2018*, 10:7199–7208, 2018.

[58] Janghyeon Lee, Donggyu Joo, Hyeong Gwon Hong, and Junmo Kim. Residual continual learning. *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence*, 34(4):4553–4560, 2020.

[59] Ronald Kemker and Christopher Kanan. Kemker Kanan - 2018 - FearNet Brain-Inspired Model For Incremental Learning.pdf. In *ICLR*, pages 1–16, 2018.

[60] German I. Parisi, Jun Tani, Cornelius Weber, and Stefan Wermter. Lifelong learning of spatiotemporal representations with dual-memory recurrent self-organization. *Frontiers in Neurorobotics*, 12(November):1–15, 2018.

[61] James L Mcclelland, Bruce L Mcnaughton, and Randall C O Reilly. Why there are Complementary Learning Systems in the Hippocampus and Neocortex : Insights from the Successes and Failures of Connectionist Models of Learning and Memory. *Psychological review*, 102(3):419, 1995.

[62] Dharshan Kumaran, Demis Hassabis, and James L Mcclelland. What Learning Systems do Intelligent Agents Need ? Complementary Learning Systems Theory Updated. *Trends in Cognitive Sciences*, 20(7):512–534, 2016.

[63] Kevin Clark, Minh-Thang Luong, Quoc V.Le, and Christopher Manning. Electra : Pre - Training Text E ncoders As Discriminators R ather Than Generators(ICLR2020). *ICLR 2020*, pages 1–18, 2020.

[64] A Benchmarking Use Case, Clément Chadebec, and Louis J Vincent. Pythae : Unifying Generative Autoencoders in Python. *Advances in Neural Information Processing Systems*, pages 21575–21589, 2022.

[65] Douglas Reynolds. Gaussian Mixture Models. *Encyclopedia of Biometrics*, 741:659–663, 2009.

[66] Vladimir Maz'ya and Gunther Schmidt. On approximate approximations using Gaussian kernels. *IMA Journal of Numerical Analysis*, 16(1):13–29, 1996.

[67] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data Via the EM Algorithm . *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[68] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, and Et Al. TensorFlow: A system for large-scale machine learning. *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016*, 1:265–283, 2016.

[69] Sepand Haghighi, Masoomeh Jasemi, and Shaahin Hessabi. PyCM : Multiclass confusion matrix library in Python. *Journal of Open Source Software*, 3(25):729, 2018.

[70] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST : a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv preprint arXiv*, 1708.07747, 2017.

[71] Geoffrey Hinton Laurens van der Maaten. Visualising Data using t-SNE. *Journal of Machine Learning Research 9*, 9(1):2579–2605, 2008.

[72] Nair Vinod and Geoffrey E Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. *Proceedings of the 27th International Conference on Machine Learning*, 1(3):807–814, 2010.

[73] Matthew D. Zeiler, Dilip Krishnan, Graham W. Taylor, and Rob Fergus. Deconvolutional networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2528–2535, 2010.

[74] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pages 1–15, 2015.

[75] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic Gradient Descent With Warm Restarts. *Conference paper at ICLR*, pages 1–16, 2017.

[76] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data Via the EM Algorithm . *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[77] Shiyong Cui and Mihai Datcu. Comparison of Kullback-Leibler divergence approximation methods between Gaussian mixture models for satellite image retrieval. *International Geoscience and Remote Sensing Symposium (IGARSS)*, 2015-Novem:3719–3722, 2015.

[78] Bartosz Krawczyk. Learning from imbalanced data : open challenges and future directions. *Progress in Artificial Intelligence*, 5(4):221–232, 2016.

[79] Soheil Kolouri, Nicholas Ketz, Xinyun Zou, Jeffrey Krichmar, and Praveen Pilly. Attention-Based Selective Plasticity. *arXiv preprint arXiv*, 1903.06070, 2019.

[80] Polina Kirichenko, Mehrdad Farajtabar, Dushyant Rao, Balaji Lakshminarayanan, Nir Levine, Ang Li, Huiyi Hu, Andrew Gordon Wilson, and Razvan Pascanu. Task-agnostic Continual Learning with Hybrid Probabilistic Models. *ICML 2021 Workshop INNF*, pages 1–23, 2021.

[81] Carlos Améndola, Alexander Engström, and Christian Haase. Maximum number of modes of Gaussian mixtures. *Information and Inference*, 9(3):587–600, 2020.

[82] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment Anything. *arXiv preprint arXiv*, 2304.02643, 2023.

[83] Leland McInnes, John Healy, and James Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv preprint arXiv*, 1802.03426, 2018.

[84] Zhihong Nan and Xue Bai. The study on data migration from relational database to graph database. *Journal of Physics: Conference Series*, 1345(2), 2019.