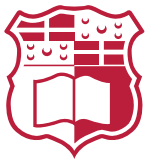
Analysing the performance of the Stack Neural Module Network architecture on the VCR dataset

Zachary Cauchi

Supervisor: Prof. Adrian Muscat

December 2023

Submitted in partial fulfilment of the requirements for the degree of MSc in Computer Science.



L-Università ta' Malta
Faculty of Information & Communication Technology

Abstract

In the field of Artificial Intelligence (AI) vision-language tasks, Visual Commonsense Reasoning (VCR) stands out as an interesting case of requiring an AI model to not only predict correct answers, but explain why those answers were chosen. The Stack Neural Module Network (SNMN) model, while not designed to target VCR tasks, also stands out for different reasons; it is a compositional model which tries to predict answers to Visual Question Answering (VQA) tasks via a memory stack used to store the intermediate steps taken to predict a final answer. These intermediate outputs can then be visualised to better understand how the model is trying to arrive at its conclusion. This study adapts the SNMN model to predict answers and rationales in the VCR tasks — attempting to obtain an accuracy better than random guessing and at most within 20% of more recent state-of-the-art models — while still retaining use of its memory stack to provide intermediate outputs. The results do not reach state-of-the-art accuracy and also showed signs of overfitting, but do suggest avenues for future work that may yet improve the model.

Acknowledgements

I would like to express my thanks to Professor Adrian Muscat for supervising me during this work, his feedback and guidance has been instrumental in this work and I wouldn't have achieved this without him. I would also like to thank my friends and family for supporting me throughout this endeavour, especially during those times where my confidence wavered.

Contents

List of Figures

List of Tables

List of Abbreviations

AI Artificial Intelligence.

BiLSTM Bidirectional LSTM.

BPE Byte-Pair Encoding.

CNN Convolutional Neural Network.

CorefNMN Coreference Neural Module Network.

DARTS Differentiable ARchiTecture Search.

DP-NMN Dual-Path Neural Module Network.

GPU Graphics Processing Unit.

GQA Graph Question Answsering.

imdb image database.

LNMN Learning Neural Module Network.

LSTM Long Short-Term Memory.

MAC Memory, Attention, and Compositional.

ML Machine Learning.

MLP Multi-Layer Perceptron.

MMN Meta Module Network.

N2NMN End-to-End Module Network.

NAS Neural Architecture Search.

NMN Neural Module Network.

NMN-VD Neural Module Network for Visual Dialog.

NN Neural Network.

PES Performance Estimation Strategy.

R2C Recognition to Cognition.

REF Referential Expression Grounding.

ResNet Residual Network.

RNN Recurrent Neural Network.

RPN Region Proposal Network.

RPP Reverse Polish Notation.

Seq2Seq Sequence-to-Sequence.

SNMN Stack Neural Module Network.

VCR Visual Commonsense Reasoning.

VD Visual Dialog.

VQA Visual Question Answering.

Glossary of Symbols

Bidirectional LSTM An RNN where two Long Short-Term Memory (LSTM) cells are used to propagate information both forward (through time) and backward (recall past information) for each time step..

Byte-Pair Encoding A compression algorithm which merges frequent character/byte sequences into a sequence. .

corpus A file containing every tokenised sentence in a dataset / series of sentences.

Differentiable ARchiTecture Search An architecture search technique for learning monolithic neural architectures which also supports gradient descent.

logit A function which represents a probability measured from 0 to 1. Mathematically, a logit is represented as $logit(p) = log(\frac{p}{1-p})$.

Neural Architecture Search A Machine Learning challenge where a model must learn the optimal architecture for a given task..

RNN A Recurrent Neural Network (RNN) is a class of neural network structures that can use its own output as input, in a cyclic manner. This allows it to process input data where order of sequence is important in deriving output, or process data of variable length.

Sequence-to-Sequence A learning strategy where an RNN learns to map an input sequence to a different-length output sequence..

Visual Dialog An extension of VQA which involves maintaining dialogue through multiple questions and answers instead of facing single question and answer. .

visual priming bias A phenomenon whereby questions asked about an image typically only mention subjects found in the image. This pattern of focusing on visible subjects is known as visual priming bias..

1 Introduction

The Visual Question Answering problem — which is a computer vision-language task whereby a system, given a question in the presence of an image, can predict an answer to the question [1] — has been leading up to a new problem: Visual Commonsense Reasoning (VCR). The VCR problem extends the VQA problem through the complexity of the questions being asked, which require more knowledge and insight to answer than is otherwise immediately apparent in a given image [2]. Datasets are available for both tasks, and there are numerous Machine Learning (ML) models which have been trained for both tasks.

A class of ML models targetting VQA tasks known as 'compositional models'[3] have proven to perform well on VQA datasets[4]. Such performance is attributed to the nature of their design whereby multiple smaller ML modules are used to divide and conquer the steps for solving a VQA task. To further explore the use of compositional models in such tasks, we will be looking towards taking an existing model and adapting it to solve tasks that require VCR.

While any compositional model could have been chosen for this work, the below characteristics were established to choose one model:

- The source code for the model and its distribution are available by the original authors along with steps for reproducing their results.
- The architecture of the model is such that each step taken to solve a VQA task is performed in a sequential manner which can be viewed at each individual step and should therefore be easier to interpret when compared to other non-compositional models. This same behaviour can also be ported to VCR tasks which should allow for better exploration of model performance on the task.
- The modular nature of the model architecture means future work can expand on its ability to solve VCR tasks without necessitating a complete redesign to the model architecture.
- The chosen model is fully differentiable, meaning it can be trained without reinforcement learning or supervision of any kind (such as expert layouts) and produce comparable performance to models trained with layout supervision.

With the above in mind, the below objectives were established:

- Obtain a working copy of the model.
- Confirm the model operates as intended by training it on VQA and produce accuracy results matching those published by the model authors (within a reasonable margin).

- Modify the model to be able to train and evaluate on the VCR dataset.
- Perform experiments on the model to test whether certain modifications will produce better results or not.
- Following an analysis of its performance, outline future work that may expand upon the findings.

This study will begin with a review of the literature available, covering the datasets available for these task types, the models which target these datasets, and a discussion of which model best meets the above criteria for use in this study. Following the literature review is the methodology of how this model is set up to run on the VCR dataset, including what experiments will be run. The results of these experiments will be explored along with a qualitative analysis of how the model performance compared to other VCR models covered in the literature review. To conclude, a retrospective and discussion of future work will be presented.

2 Literature review

For this work, we will be exploring a subset of AI models known as compositional neural network models. We will select a model of this type by exploring and discussing three such models, choosing one with which to proceed. These models, while not the state-of-the-art as of the time of planning this work, will be some of the more noteworthy models of this subset and will have had a notable influence on successive models. Additional models of this type will also be explored to see how they have been expanded and improved upon. We will also explore the datasets used by the models, determining the characteristics of the datasets, their strengths, and any observed drawbacks. Finally, we will round off with a comparison between the different tasks explored, the datasets that target these tasks, and the characteristics of the models covered (specifically those details attributed to their compositional nature).

2.1 Datasets

In this section, four VQA datasets will be explored and discussed to identify the reasons for which they had been developed and how they aim to improve upon the task of testing models for their targetted task. As the strengths and scopes of each one are explored, two final datasets will be discussed — which go a step beyond the VQA datasets — to explore the differences in task complexity and the vision/language challenges they aim to address.

2.1.1 The *SHAPES* dataset

The *SHAPES* dataset is a VQA dataset introduced by Andreas *et al.* [5] consisting of synthetic images designed to test the layout construction of compositional neural models. Each image-question pair consists of a simple image with 9 possible locations for objects and a number of visible shapes in each image. These shapes are simple uniform shapes (triangles, squares, or circles) with only a difference in color (red, green, or blue) to distinguish them (see Figure 2.1). The questions on the other hand are complex with each question containing up to 4 different object attributes, types, or relationships. The questions found in the dataset can be deliberately false (such as `Is a red shape blue?` or `Is the red square a triangle?`) or valid questions (such as `Is the red object left of a blue triangle a square?`).

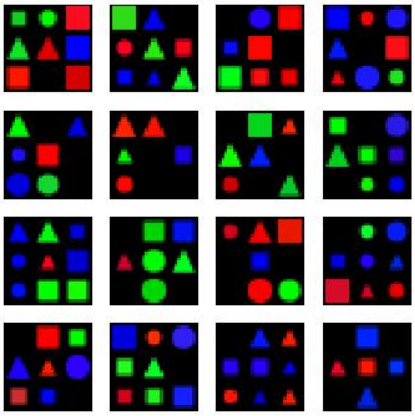


Figure 2.1 Example images from the SHAPES dataset.
Source: Andreas *et al.*[3]

2.1.2 The *VQA* dataset

The *VQA* dataset [1] is a natural image dataset composed of 204,721 images, 1,105,904 questions, and 10 acceptable ground truth answers per question. The images are taken from the COCO image dataset [6] real-life objects, scenarios, and entities, while the questions and answers are supplied by human annotators. All questions are open-ended, with an array of answers to select from and a subset of answers which are possible/correct (See Figure 2.2 for example image-question pairs with answers).

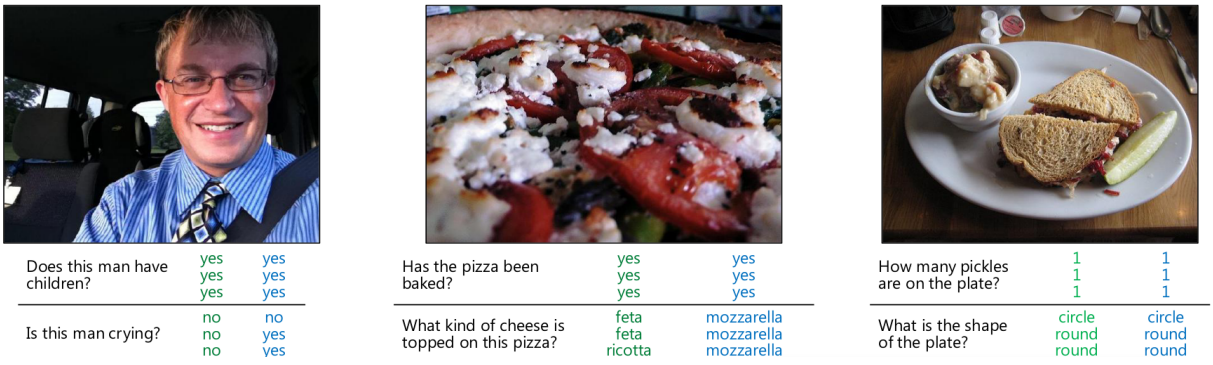


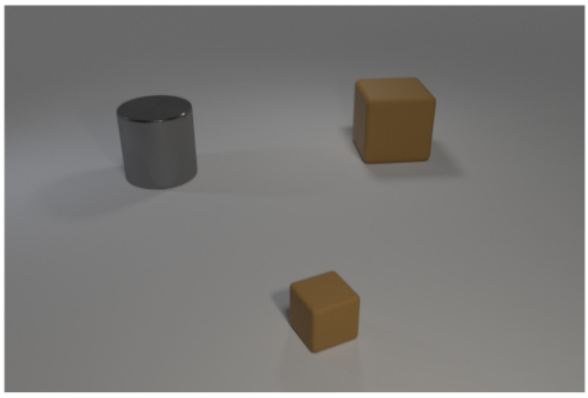
Figure 2.2 Example images from the VQA dataset with a question per image and answers. Green answers are valid answers for the given image while blue answers would be valid without the image. Only the green answers are considered correct answers by the dataset.
Source: Antol *et al.*[1]

2.1.3 The *CLEVR* dataset

The *CLEVR* dataset [7] is a VQA dataset designed to test and benchmark compositional VQA models. Similar to the *SHAPES* dataset, each image is a blank scene with any number of 3d shapes which can differ in shape, colour, size, and material (being either shiny

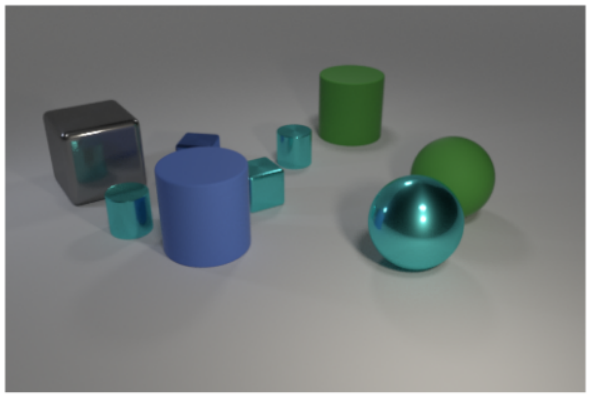
metal, or matte rubber).

Questions vary in the type of answer expected (such as counting, yes/no, object attributes), and are diverse in structure, length, query types used, and relationship queries (see Figure 2.3). In total, *CLEVR* contains 100,000 images and 864,968 questions, with a single correct answer being given per question.



Q: There is a rubber cube in front of the big cylinder in front of the big brown matte thing; what is its size?
A: small
Q-type: query_size
Size: 14

Q: What color is the object that is on the left side of the small rubber thing?
A: gray
Q-type: query_color
Size: 7



Q: Are there fewer metallic objects that are on the left side of the large cube than cylinders to the left of the cyan shiny block?
A: yes
Q-type: less_than
Size: 16

Q: There is a green rubber thing that is left of the rubber thing that is right of the rubber cylinder behind the gray shiny block; what is its size?
A: large
Q-type: query_size
Size: 17

Figure 2.3 Example images from the CLEVR dataset with a question per image and the correct answer. Additionally, there's also the type of question included (such as classifying the size or colour) and the size of the expert layout/program.
Source: Johnson *et al.*[7]

2.1.4 The *GQA* dataset

The *Graph Question Answsering (GQA)* dataset was introduced by Hudson and Manning[8] as a collection of highly compositional questions to better train compositional VQA models. The dataset contains over 110,000 images — sourced from various image datasets — and over 22,000,000 questions.

Alongside each image is a scene-graph which describes the objects in the image, object relations, and image location details. Each question in the training set describes a program in the form of semantic steps which — if executed by a training model — would lead to a greater probability of predicting the correct answer. These steps mimic how a person would apply reasoning to a question and provide an answer to it and should therefore train a model how to perform such reasoning.

GQA
1. What is the **woman** to the right of the **boat** holding? umbrella
2. Are there **men** to the left of the **person** that is holding the **umbrella**? no
3. What color is the **umbrella** the **woman** is holding? purple

VQA
1. Why is the person using an umbrella?
2. Is the picture edited?
3. What's the color of the umbrella?

GQA
1. Is that a **giraffe** or an **elephant**? giraffe
2. Who is feeding the **giraffe** behind the **man**? lady
3. Is there any **fence** near the **animal** behind the **man**? yes
4. On which side of the image is the **man**? right
5. Is the **giraffe** is behind the **man**? yes

VQA
1. What animal is the lady feeding?
2. Is it raining?
3. Is the man wearing sunglasses?

Figure 2.4 Comparison of questions from both GQA (left) and VQA (right) datasets for the same image. The GQA questions feature greater emphasis on object relations and compositionality than the VQA questions are which are comparatively vague or ambiguous.
Source: Hudson and Manning[8]

2.1.5 The *VCR*

The *VCR* dataset [2] was introduced alongside the formalisation of the VCR task as the first dataset of its kind. The images in the dataset are largely frames from movies or clips, and are chosen because of the inherent context supplied by the movies that's required to understand the images. Because of this, each question in the dataset is about something present within context that cannot be immediately recognised by simple object detection and will thus require additional cognition to answer. The questions, answers, and rationales, also make use of bounding boxes to identify each person/object of interest, and uses their box names when referring to them (Figure 2.5 is an example of how these are used). Aside from answering each question, there is also the additional task of providing the rationale behind the given answer. In this subtask, the model would have to both produce a correct answer prediction for the given question, and then pair that answer with the correct rationale to justify the answer. There is only one correct answer and one correct rationale per-question. There are 3 modes of question-answering available by the dataset as broken down below:

- (Q $\rightarrow$A): Predict the correct answer for a given image and question.
- (QA $\rightarrow$R): Predict the correct rationale for the given answer to a given question and image.
- (Q $\rightarrow$AR): Using only the image and question, predict both the correct answer and correct rationale.

In total, there are 99,904 images, 264,720 questions, 1,058,880 answers, and 1,058,880 rationale. Each image in the dataset comes with many question files, each

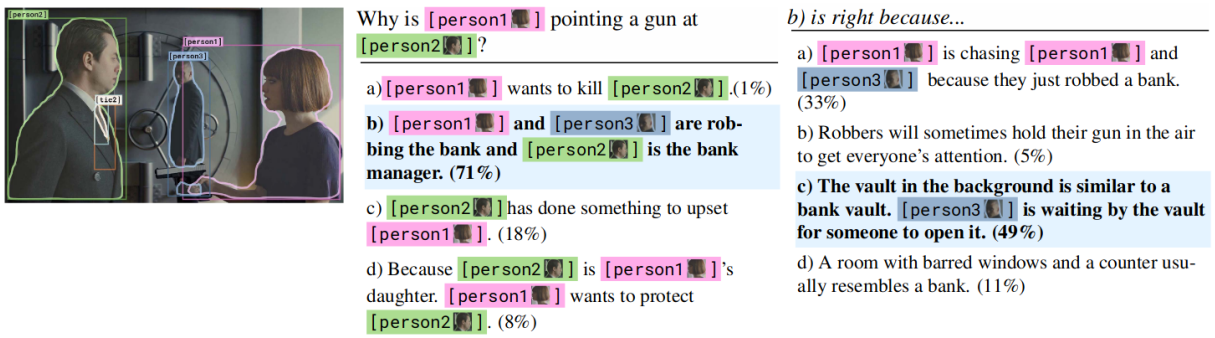


Figure 2.5 Example VCR task from the *VCR* dataset. The text shows how each object in the image is highlighted by the provided bounding box metadata.
Source: Zellers *et al.*[2]

containing a question about the image with one correct answer and one correct rationale per-question. A further 3 incorrect answers and 3 incorrect rationale are included with the correct ones, which are correct answers or rationale to one other question in the dataset (in other words, each answer/rationale is correct at least once across all questions in the dataset). Each question file (outside of the test fold) specifies the correct answers for the question, and the 'correctness' of each answer. Each image is accompanied with a metadata json file containing the dimensions of the image, the class names of the objects present (eg. person, car, dog, etc), and the bounding boxes and polygons identifying each object in the image. All bounding boxes and polygons were generated using the Detectron object detection system[9].

2.1.6 The *VisDial* dataset

The *VisDial* dataset — published by Das *et al.* [10] — formalises a new visual task known as Visual Dialog (VD). Where VQA and VCR both focus on an image with a single question and answer, Visual Dialog (VD) extends this by posing multiple questions per image to be answered sequentially, building a natural-flowing dialogue in the process. The dataset is composed of roughly 140,000 total VD entries where each entry contains one image sourced from the *COCO* dataset[6], one caption describing the image, and ten rounds of questions and answers. The questions in the image were created without access to the image, only the caption. Aside from this, subsequent questions also build upon the new context derived from answers to previous questions such as asking for more details about the previous answer. This causes the questions to flow like a natural dialogue between two persons, where the model is the only subject to know what's present in the picture. Another benefit to this approach is that it reduces the visual priming bias typically found in other visual datasets, where questions would focus only on visible subjects and therefore have easily-predictable answers. Evaluation on these VD tasks is performed using a custom evaluation protocol published alongside the dataset; given

the image, the image caption, the dialog history up until the question to be answered, the question to be answered, and the top candidate answers (where N = 100), the model must produce a sorted list of the candidate answers for the given question. Its performance is then evaluated by comparing the rank of the human response in the sorted list, checking if the response is in the top-k responses (*recall@k*), and the reciprocal rank of the human response among all other answers. Finally, the authors also publish a set of 3 encoders that show how to encode a VD task in a machine model. These encoders either embed the data in vector space and generate a joint embedding, use a sequence of Recurrent Neural Network (RNN)s with attention-over-history to the question-relevant history, or simply store the data in memory and perform lookup when generating a final context vector.

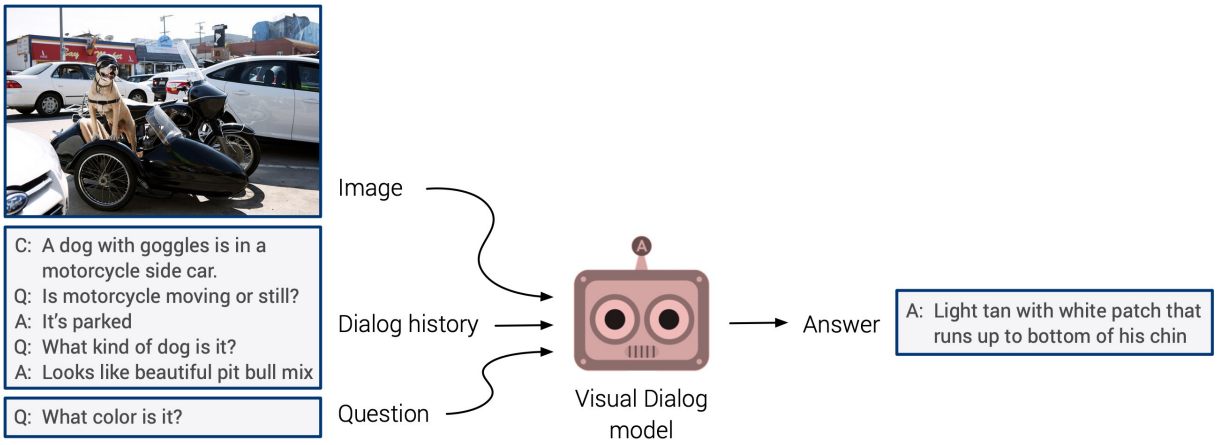


Figure 2.6 Example VD task from the *VisDial* dataset. The image is supplied with a caption, a model predicts answers to the questions for that image, and the model must maintain context for each following question using the dialogue history.
Source: `https://visualdialog.org`

2.2 Compositional Model Review

In this section, we will be exploring three compositional models that have been considered for this study.

Each subsequent model builds upon the works of the former, adopting a more modular and understandable approach for solving VQA tasks while also achieving better performance.

2.2.1 Neural Module Network

The Neural Module Network (NMN) model[3] is an attention-based compositional model which makes use of an array of Neural Network (NN) modules to solve VQA tasks. When

given an image-question pair, it predicts an answer to the question using the following procedure:

- The image and question are preprocessed, extracting their visual and textual features respectively.
- The image features, the question text and the question features are fed to the model as inputs.
- A new layout of NN modules is created (see Figure 2.7 for an example) by the question parser.
- The image features are inputted to each module, computing the output for each module.
- The text features are fed to an LSTM. Based on the input features, the outputs of only a specific set of the NN modules will also be fed into the LSTM.
- The LSTM and layout outputs are averaged together to produce a final classification prediction as the answer to the input question.

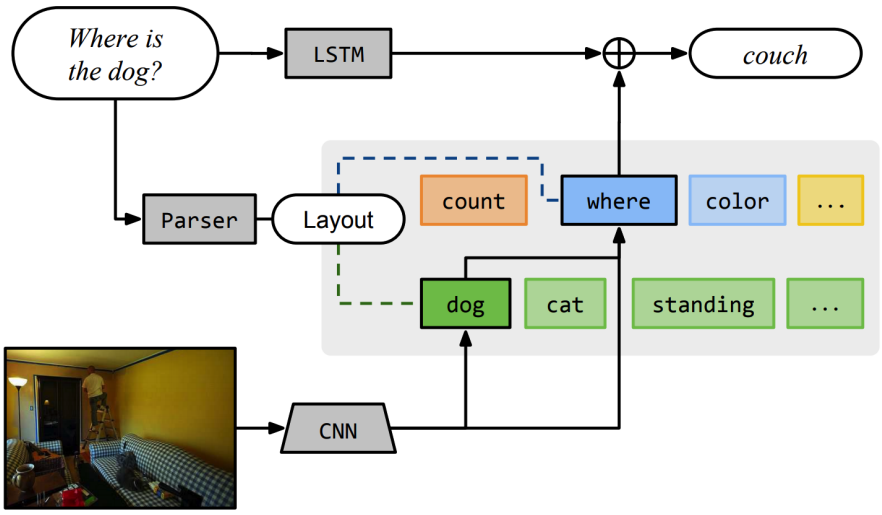


Figure 2.7 How an NMN predicts answers to a VQA task.
Source: Andreas *et al.*[5]

Each module in the NMN is described in the format `type[instance](arg1, ...)`, where `type` denotes the type of operation performed by the module (eg. `attend` will search for an object in the image) and `instance` identifies the module amongst other modules of the same type (eg. `attend[pillow]` identifies an `attend` module that looks for `pillow` objects in the image). Arguments such as instance or type weights, or other argument types, are shared by the modules. The paper introduced five module types for its NMN model which are given in Table 2.1.

Table 2.1 NMN module types and example uses.
Source: Andreas *et al.*[3]

Module Name	Module label	Inputs →Output	Example
Attention	`attend`	Img →Att	`attend[ladder]`
Re-Attention	`re-attend`	Att →Att	`re-attend[right]`
Combination	`combine`	Att, Att →Att	`combine[include]`
Classification	`classify`	Img, Att →Lbl	`classify[colour]`
Measurement	`measure`	Att →Lbl	`measure[count]`

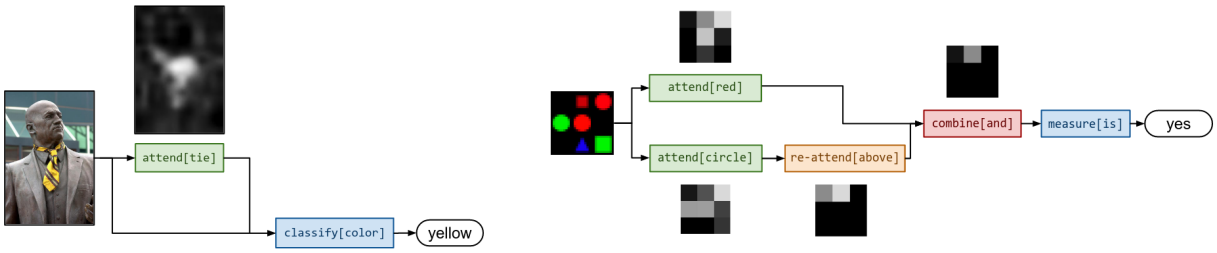


Figure 2.8 Example VQA tasks being broken down by the NMN. **Left:** Question: What colour is his tie? **Right:** Is there a red shape above a circle?
Source: Andreas *et al.*[5]

With the module instances prepared, the NMN model now needs to know which module instances are required for each question. To solve this, each question is converted into a layout, which identifies the modules required to answer the question. To obtain these layouts, each question is first parsed using the Stanford Parser [11], a tool which uses a pre-trained language model to output standardised representations of the questions using the Universal Representations v1 format [12]. These representations are then simplified and converted into tokens which represent the module types and instances supported by the NMN (for example: the question `"What is the colour of the cat left of the truck?"` could be converted into `"classify[colour](attend[cat](re-attend[left](attend(truck))))"`).

While the above provides the model with a solid approach to predicting the answer, it is still susceptible to errors due to overlooked grammatical cues in the question (for example: `"What is swimming?"` versus `"What are swimming?"`; both questions denote the answer is something that's swimming, but the second question indicates a plural answer which cannot be represented or conveyed by the layouts protocol established above). To solve this, the NMN uses a Long Short-Term Memory question encoder to detect such cues, and combines its output with the output of the modules. This effectively gives the final output of the model, the predicted answer to the image-question pair.

Algorithm 1 A simplified pseudocode of how NMN solves a VQA task, from layout construction, to answer prediction.
Source: Original work written for this study

img_o = raw image data ▷ *Original image*
q_o = "What colour is his tie?" ▷ *Original question*
img_f = GetImgFeatures(img_o) ▷ *Convert to feature map*
q_{rep} = GetUDRepresentation(q_o) ▷ *Get a Universal Dependency representation*
q_{func} = MapToFunctions(q_{rep}) ▷ *"What colour is his tie" →"colour(tie)"*
q_l = "" ▷ *Final network layout.*
for all $w \in q_{func}$ **do** ▷ *First "colour", then "tie"*
    if IsRoot(w) **then**
        PushAnswerNode(q_l, w) ▷ *Either 'measure' or 'classify' node*
    else if IsLeaf(w) **then**
        PushAttendNode(q_l, w) ▷ *Always an 'Attend' node*
    else
        PushReAttentionNode(q_l, w) ▷ *Either 'reattend' or 'combine' node*
▷ *q_l is now "classify[colour](attend[tie])"*
a_{qe} = QEncoderPredictAnswer(q_o) ▷ *Predict answer using LSTM*
a_l = LayoutPredictAnswer(q_l) ▷ *Predict answer using layout and modules*
▷ *Get geometric mean of both predictions (layout-generated and LSTM-generated)*
$a_{final} = \sqrt[2]{a_{qe}a_l}$ ▷ *Final answer prediction.*

2.2.2 End-to-End Module Networks

Building on the NMN as an attention-compositional neural network, Hu *et al.* introduced End-to-End Module Network (N2NMN) as an NMN-based model with an improved layout policy and network assembly [13] (See Figure 2.9).

Similar to NMN, N2NMN uses neural modules which take one or two attention maps as input (depending on the module type) and outputs either another attention map or a probability distribution over the possible answers. Aside from the given input maps, a module-specific textual vector — obtained from the question being solved — is also made available at runtime. This textual vector is created by obtaining an attention map from word embeddings of each word in the question. With this, a layout expression is created from which the N2NMN is able to dynamically construct the modules needed using these textual vectors — as shown in Figure 2.9 — without relying on multiple separate, hard-coded module instances as is the case in the NMN model. Figure 2.11 illustrates how a question is parsed into a solvable layout.

The layout expression is then converted into a sequence of module tokens using Reverse Polish Notation as shown in Figure 2.10. This has the benefit of representing the solution to predicting the answer as a series of smaller VQA tasks. The sequence is then parsed through an attentional RNN [14]. First, all words in the question are embedded into word vectors which are then fed into a multi-layer LSTM, outputting the encoded

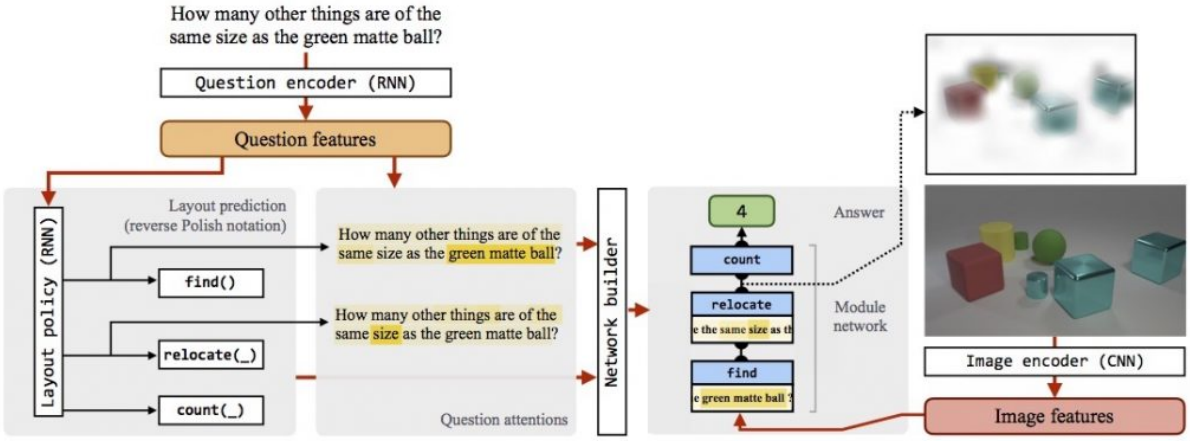


Figure 2.9 The topology of the N2NMN model, focusing on its approach to question representation and network layout assembly.
Source: `https://ronghanghu.com/n2nmn/`

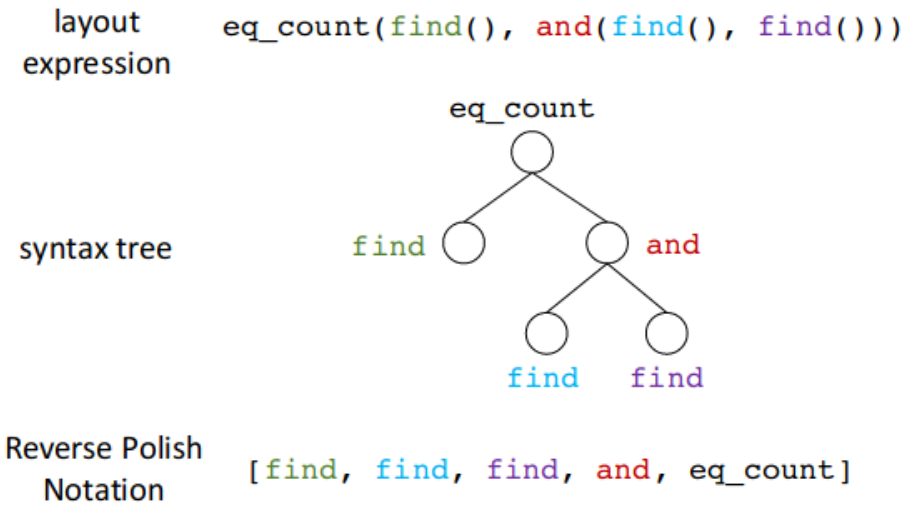


Figure 2.10 How N2NMN constructs its layout policies using an RPP sequence of module tokens.
Source: Hu *et al.*[13]

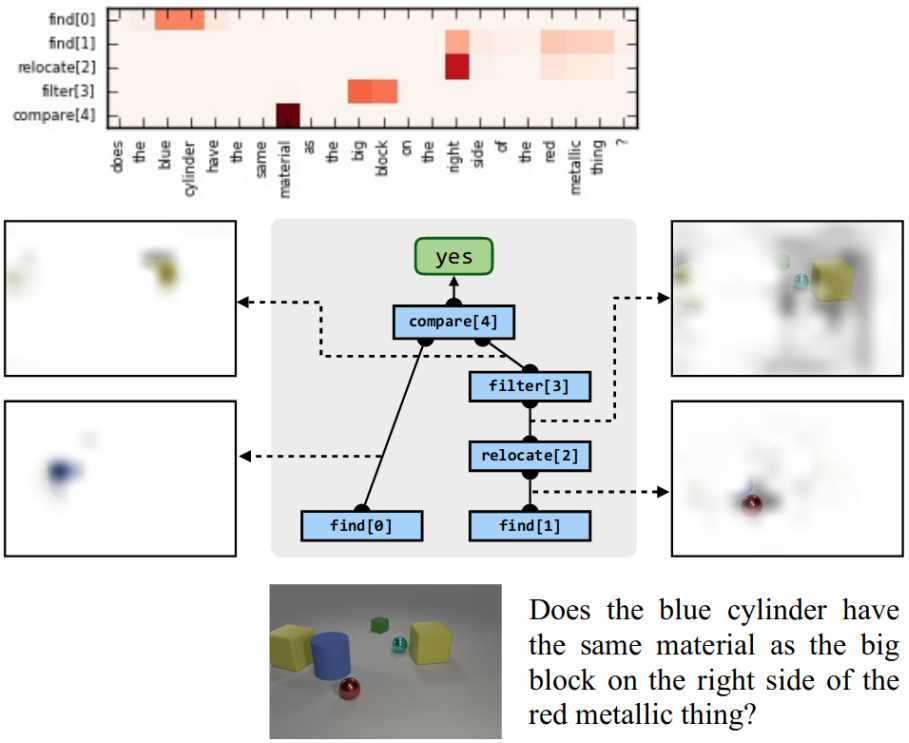


Figure 2.11 A sample breakdown of a VQA image-question pair, the textual attention for the question, the modules being called and their sequence, and the attentions being produced at each step.
Source: Hu *et al.*[13]

question as a vector of equal length. An LSTM decoder then generates an attention map for the given encoder output and input words. With this, a distribution of all possible layouts for the question can be predicted. To narrow this down to the final layout, the model uses a beam search to select the best layout available from the distribution. From this, the final network is assembled.

During training, the layout policy and module parameters are jointly trained, using Adam for parameter optimisation[15], and a loss function over the output answer scores to optimise these parameters. Due to the layout policy being a discrete training problem, the loss function is not fully differentiable and does not allow for training with full back-propagation. To circumvent this, those parts which are fully differentiable are trained with back-propagation, while those parts that aren't are trained using a policy gradient method optimised for reinforcement learning.

To optimise training of the layout policy, behavioural cloning is used to significantly reduce the starting loss of the model. This is done by pre-training the layout policy against a previously-trained layout policy that gives viable performance (referred to as the expert policy). Once trained, a suitable starting set of parameters are available to the sequence-to-sequence RNN and the neural modules. To avoid biasing of model performance on test sets, expert policies are only used when training on training sets.

2.2.3 Stack Neural Module Network

The End-to-End Module Network (N2NMN) model improved upon the original NMN, but can still be improved further in ways that leverage its sequence-based architecture. Succeeding the N2NMN in performance and readability is the SNMN model, published by Hu *et al.*[16]. The SNMN architecture is similar to that of N2NMN with the exception of how its layouts are selected; whereas the N2NMN layout policy selected a discrete set of modules in a layout, the SNMN layout controller uses a 'soft layout' where all modules are activated and their weighted outputs are averaged (See Figure 2.12). The difference in layouts means the SNMN is fully-differentiable and trainable with back-propagation.

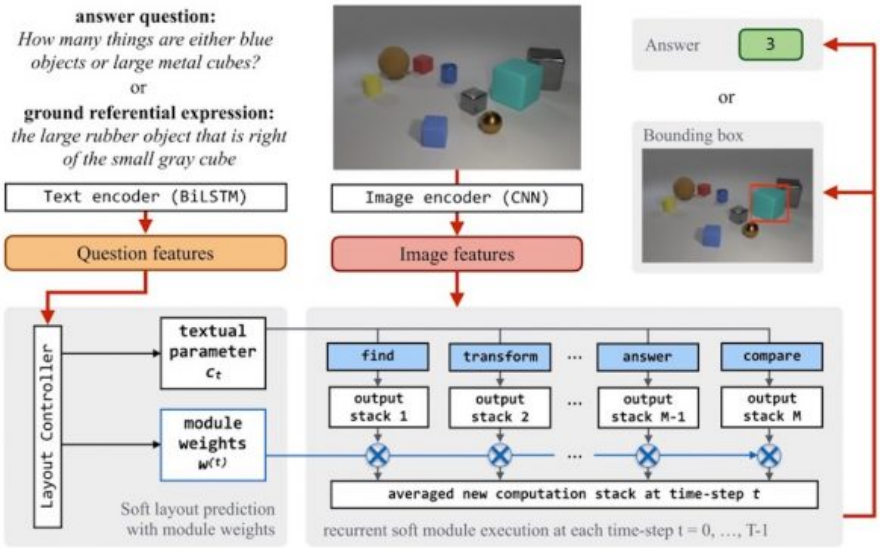


Figure 2.12 The topology of the SNMN model and how it solves VQA and REF tasks.
Source: `https://ronghanghu.com/snmn/`

The layout controller first encodes the input token sequence representing the question into a textual attention mask sequence representing the question by using a bi-directional LSTM. The controller then uses an MLP to predict a softmaxed attention vector containing a weight for each neural module in the model; this will be the soft layout. In addition to the soft layout, a textual attention vector is then predicted for each token in the question sequence and used to predict the textual parameter which will be inputted to each module in the network. The layout controller unrolls itself across all time-steps, repeating the above steps to produce a soft layout, textual parameter, and textual attention vector for each time-step.

Regarding modules, the SNMN uses the same module definitions as N2NMN but simplified in implementation in some cases (See Table 2.2 for a list of all implemented modules). The main differences between the two implementations is that SNMN uses a single `Compare` module for comparison operations and an `Answer` module for tasks such as measuring or describing. A `NoOp` module is also implemented which performs

Module Name	Inputs →Output	Example
`Find`	(None) →Att	`Find[`chair'](`)
`Transform`	Att →Att	`Transform[`left'](`)
`And`	Att, Att →Att	Used internally
`Or`	Att, Att →Att	Used internally
`Filter`	Att →Att	`Filter[`blue'](`)
`Scene`	(None) →Att	Used internally
`Answer`	Att →Ans	`Answer[`exist'](`)
`Compare`	Att, Att →Att	`Compare[`more'](`)
`NoOp`	(None) →(None)	Used internally

Table 2.2 SNMN module types and example uses. Some modules are only used internally, or are used as part of the implementation of other modules.
Source: Hu *et al.*[16]

no computation or contribution to the predicted answer, but serves to pad out layouts should they finish before reaching the expected layout size.

Due to the input data requirements of some of the modules, the model needs to be able to provide data from one time-step to a module in a future time-step. One example case being in `Compare(Find(), Transform(Find()))`, where the `Compare()` module needs to know the outputs of both the `Find()` module and `Transform()` module which were both executed in separate time-steps. To address this, a memory stack is used to store outputs from intermediate neural modules where each module can then pop and push data onto the stack as needed. The stack can store a pre-configured number of fixed-dimension vectors in its memory, while a one-hot vector - the same length as the stack - serves as the stack pointer. The stack is designed to store image attention maps which are equal in size to the image feature maps. Modules will then pop as many attention maps as needed and then push their output (if its an attention map) back onto the stack for other modules to use. The model implements differentiable push and pop operations for manipulating the stack and its stack pointer.

When the model prepares to execute a layout, it begins by initialising the stack and setting the stack pointer to the 1st stack element. From each time-step after initialisation, every module in the model is executed and popping any needed attention from the stack and pushing back onto the stack. The result of each module is averaged together to produce a new stack representing the final state of that time-step, and the same averaging is performed on the stack pointer to indicate the top-most element at the end of the time-step. The final output of the model is determined by averaging the weighted answer logits across all output modules (see Table 2.2) across all time-steps, and summing them to produce a final output logit.

Model	SHAPES	CLEVR	VQAv1	VQAv2
NMN[5]	90.6	72.1[13]	55.1	-
N2NMN[13]	**100.0**	83.7	64.9 (test-dev)	63.3
SNMN[16]	-	**96.5**	**66.0**	64.0

Table 2.3 Comparison of models across the 3 VQA datasets discussed. Results are measured in percentage accuracy (%) and obtained from the highest-scoring run with all performance optimisations (such as expert layout) enabled.
Source: Aggregated from their respective publications[3, 13, 16].

2.2.4 Model comparison

To discuss and compare the models discussed in Sections 2.2.1 to 2.2.3, we will be focusing on their performance across the *SHAPES*, *VQA*, and *CLEVR* datasets, discussing and elaborating on the performance metrics presented in Table 2.3. We will then follow up by highlighting the limitations of each model, and conclude by looking into how easy it is to understand the steps taken by each model to produce the results.

Both NMN and N2NMN performed well on the *SHAPES* dataset, scoring 90.6% and 100% respectively. While the images themselves are not complex at all - being only low-resolution images of 3-coloured 2d shapes - it served as a benchmark for testing the dynamics of a model's layout-construction. The questions are also yes/no questions meaning most of the performance may be carried by the models final stage which could be performing guesswork on the text of the question. SNMN was not tested on this dataset so its performance is not known, but could be assumed to be on-par with N2NMN given the similarity in module implementation.

While the N2NMN and SNMN both achieve scores of 83.7% and 96.5% on the *CLEVR* dataset respectively, the NMN model was not tested on the dataset at the time of its publishing. Despite this, it was modified to use the expert layout of the N2NMN model so it would be able to train and test on the *CLEVR* dataset, with which it was able to score 72.1%[13].

On the *VQAv1* dataset, NMN, N2NMN, and SNMN achieved test scores of 55.1%, 64.9%, and 66%, respectively. Aside from those results, N2NMN and SNMN were also tested on VQAv2 and scored 63.3% and 64.0% respectively. Even when tested on natural images, the SNMN model remains effective, narrowly surpassing the N2NMN.

Despite the performance of the NMN, there are limitations concerning its ability to answer yes/no questions; as was suggested by Andreas *et al.*[5], the model seems to suffer from overfitting when training with yes/no questions. Aside from this, the model uses hard-coded textual parameters which lead to measurably worse performance in the CLEVR dataset. This problem does not occur in N2NMN since the modules use soft-attention module parameters instead of hard-coded textual parameters. Additionally,

the N2NMN seems to learn additional optimisations regarding how it attends to the image (for example: in Figure 2.11, the layout policy seems to 'coordinate' two `find` modules so that one looks to the 'right side of the red metallic thing' while the other looks to the left to try and find the 'red metallic thing').

The main drawback to the N2NMN is its inability to use back propagation during training, instead relying on end-to-end training using reinforcement learning. This limitation is addressed in the SNMN model, which is able to use back propagation during training thanks to its fully-differentiable layout controller and stack memory structure. The SNMN model also shares the same optimisations seen in N2NMN of composing a future time-step parameter into the current attention to optimise future time-steps. Additionally, the SNMN model produces more human-interpretable results than the other models, and is supported by an experiment which compares the SNMN model to a more closely-integrated model known as MAC[17] using human evaluators[16]. The model itself shared a very similar approach of using sequential time-steps with textual and visual parameters at each step, but does not use modular networks like the discussed models (see Chapter 2.3.3 for further details on its implementation). While the SNMN model was found to be more understandable and logically-predictable by human evaluators, its performance in the VQA dataset was worse (by about 2%) compared to the MAC model which was the state-of-the-art at the time[16].

When examining each of these models for ease of inclusion in this project, all models are open-source, with their code-bases being released by the original authors. The original NMN model however, is discouraged for use by the original author who favours the newer N2NMN model for being easier to set up and better performing[1]. Building upon this, the N2NMN and SNMN both share strong similarities in implementation, general architecture, and aims, with the main differences being that the SNMN uses a different memory structure and modified layout controller. Aside from those, it also uses more recent versions of their dependencies which will help with the development given the expected GPU driver requirements for the specified dependency versions are no longer publicly available. Both these models use this legacy version so ultimately both would require the same rewrite if done. As a result of the reasons above, the benefits outlined in the discussion of results, and the ease of predictability of the model for human users who do not know the implementations of the model, the SNMN model was chosen for this study.

2.2.5 Recognition to Cognition

Having discussed and explored the above VQA models, we will now explore two VCR models to explore the differences between the VQA models reviewed in Sections 2.2.1

[1] See `https://github.com/jacobandreas/nmn2/blob/master/README.md`

to 2.2.3 and these VCR ones. The first VCR model chosen is the Recognition to Cognition (R2C) model[2], introduced alongside the formal VCR task declaration and the VCR dataset[2]. The model a different architecture to the compositional models discussed, instead using the following steps to produce its predictions: **ground**, **contextualise**, then **reason**.

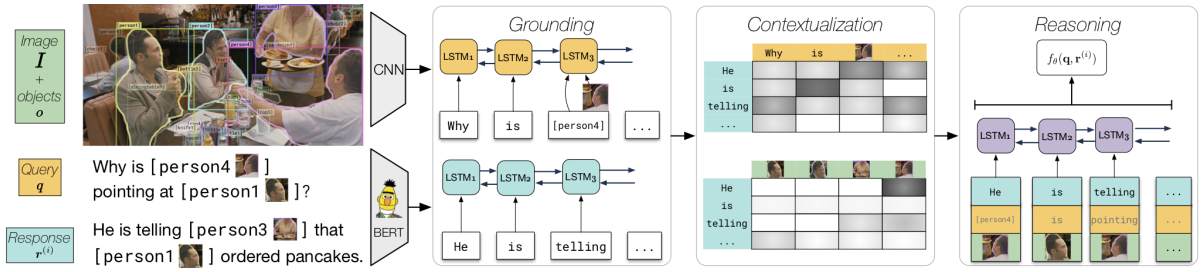


Figure 2.13 Basic overview of the R2C model and how it solves a VCR task.
Source: `https://github.com/rowanz/r2c/`

All image features are extracted using ResNet-50[18] while language representations of the questions and responses are obtained using BERT[19]. The model is trained to reduce the multi-class cross entropy between the prediction of each response for the question, and the correct-most response.

When given a question-response pair, the model begins by **grounding** the question and response, by locating the objects in the image which are referenced. By doing so, it derives the meaning of the question and the intention of each answer and rationale. The grounding module begins by learning an image-language representation for the given tokens, which it shares for the question and responses (since they share the same vocabulary and tags). Using this representation, the textual features of the question and responses are obtained. The Residual Network (ResNet)-generated object features are then combined with each object label's embedding to produce a shared hidden representation. This shared hidden representation, combining both image and textual features, is fed into a Bidirectional LSTM (BiLSTM) to produce the final outputs of the grounding module.

The outputs of the grounding module are then **contextualised** using the contextualiser module. This performs a cross-multiplied softmax between the question, response, and object attentions to obtain the final contextualised representation of the question and the responses.

Finally, **reasoning** is performed by the model inside its reasoning module. This is composed of a bidirectional LSTM which uses the contextualised representation of the question, object attentions, and responses, as input. The output is then concatenated with the question and answer representations at each timestep for better gradient control by the loss function. The final concatenated sequence is max-pooled, where an Multi-Layer Perceptron (MLP) predicts a logit representing confidence in the question-

response pair.

2.2.6 Merlot-Reserve

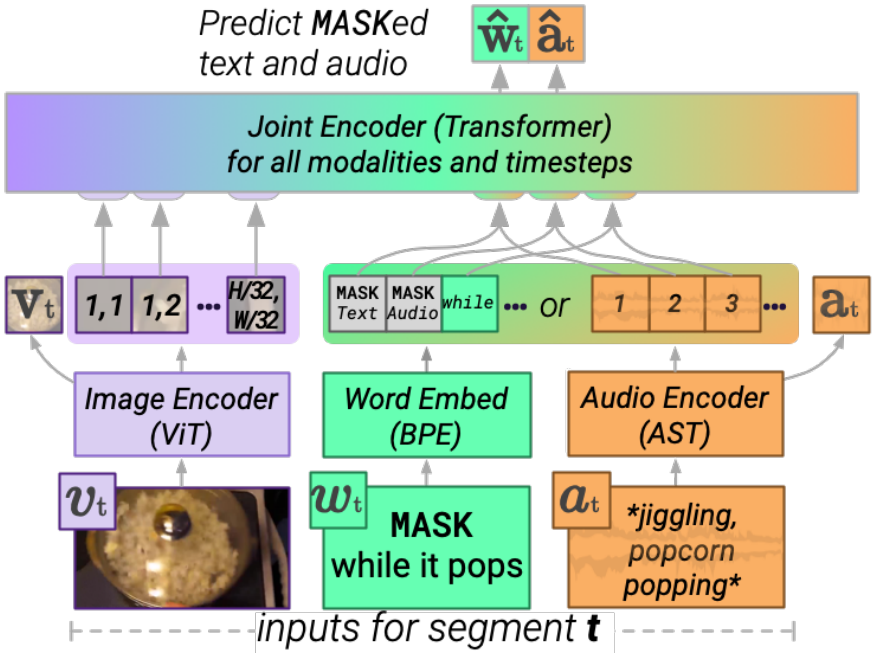


Figure 2.14 An overview of the MERLOT-RESERVE model. Of particular note is it's ability to train using audio data unlike any previously-covered model.
Source: Zellers *et al.*[20]

To conclude, one final model which can perform VCR tasks will be explored. The model in question is MERLOT-RESERVE, another VCR model by Zellers that can train on images, and either text or audio [20]. The model is transformer-based, unlike previously-explored models which are RNN-based, and uses a joint encoder to combine both encoded image data and word embeddings into a final prediction. The word embeddings are composed using a Byte-Pair Encoding (BPE) embedding table which creates embedding from a sequence of subword embeddings unlike others such as BERT or GLOVE which generate whole-token embeddings[21]. It also introduces a novel method for training called Contrastive Span Training, where the model is trained on a dataset of video-audio-subtitle entries to produce a generalised model that can be finetuned onto VCR and other tasks[20]. To train, each video is paired with a text/audio sequence that has a region masked out and the model must predict the correct audio or text that matches the masked as closely as possible. The result is a model that achieved state-of-the-art results at its time[20] and still ranks among the top 15 scoring VCR models at the time of writing this study[1].

[1] https://visualcommonsense.com/leaderboard/

2.3 Other Compositional Models

At the time of this study, the above models weren't the only compositional models based on the NMN. Therefore, this section will explore NMN-based models that come after the previously explored models but are still noteworthy due to improvements they make upon the original models. Such models may either expand upon the neural module inventory with new module types or introduce structural changes to the model architecture. In all cases, they will demonstrate unique advancements over those models discussed previously which propose new research interests within the NMN domain.

2.3.1 NMNs±

Chen *et al.*[22] introduced NMNs± as an augmented NMN which can perform basic arithmetic operations on questions. The model supports addition and subtraction modules — which take up to three number arguments — and a date comparison function. Using these additional modules, the model was able to outperform the original NMN on an expanded text-only question-answer dataset known as DROP. While the model is not trained on VQA tasks, it would be safe to assume a similar performance improvement may be expected for those VQA tasks that require arithmetic operations (such as comparing the counts of two object types in an image).

2.3.2 Dual-Path Neural Module Network

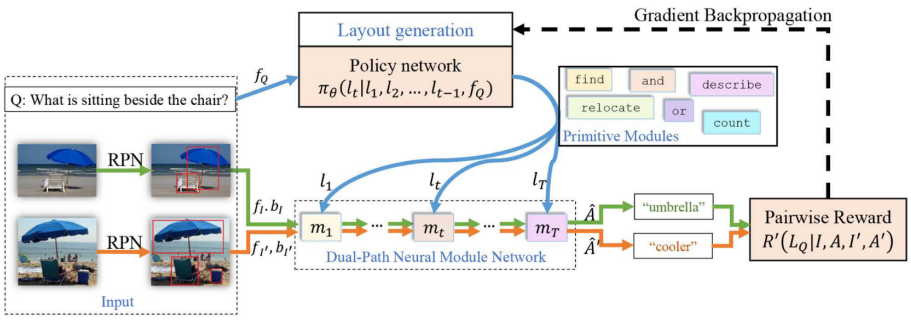


Figure 2.15 Overview of the DP-NMN model. Each reasoning step attends to two images instead of one and processes each image in parallel, generating a pair of answers instead of a single answer.
Source: Su *et al.*[23]

The Dual-Path Neural Module Network (DP-NMN) model was presented by Su *et al.* [23] as an N2NMN-based model capable of pairwise learning. The model is capable of solving two VQA tasks in parallel, sharing the same question but using a different image and different expected answer. The model uses a pre-trained RPN to extract visual and

spatial features from both images and feeds them into a module layout generated using a similar approach as was used by Hu *et al.* in their N2NMN. Instead of training using back-propagation, the model generates a reward for each pair of predicted answers and uses that to optimise the network layouts. To reduce the likelihood of overfitting on the text semantics, a second pairwise reward is then generated if both answer predictions in a pair are correct, requiring the model to correctly discern each correct answer in the pair.

2.3.3 MAC Network

The MAC model [17] is another such model that uses general-purpose neural cells. It makes use of a single Memory, Attention, and Compositional (MAC) cell at each timestep to represent each reasoning step used by the model, the structure of which can be seen in Figure 2.16. Each cell reads the control state and updates it according to the question and reasoning step its performing. It then applies this controlled reasoning to the current memory state using the image features as additional input and outputs the new intermediate result to the memory state.

2.3.4 Learnable Neural Module Network

Proposed by Pahuja *et al.* [24], Learning Neural Module Network (LNMN) is based on the SNMN model however, similar to the Meta Module Network (MMN) model, uses general-purpose neural modules instead of hard-coded ones. The aim of this architecture was to explore the use of general-purpose neural modules as a robust and generalisable alternative to hard-coded modules and as such, does not achieve better outright performance compared to the SNMN. Each neural module (or 'cell') is classified as either an Answer Module (outputs a memory features to be stored in the stack) or an Attention Module (outputs an attention map) and can either take 3 inputs or 4 inputs as seen in Figure 2.17. Each cell contains nodes which perform internal processing of cell inputs or prior node outputs and provide To perform training on the model weights, the model performs gradient descent on a batch from the training set. To perform training on the model cells and architecture, the model performs an alternative gradient descent step on a random sample batch from the validation set. This approach to training both model cells and model weights is based on Differentiable ARchiTecture Search (DARTS) which is an architecture search algorithm used for training Convolutional Neural Networks (CNNs) and RNNs[25]. When evaluated on the CLEVR dataset, the model achieved an accuracy 1% less than the SNMN the model was based on, despite using general-purpose modules instead of hand-tailored ones.

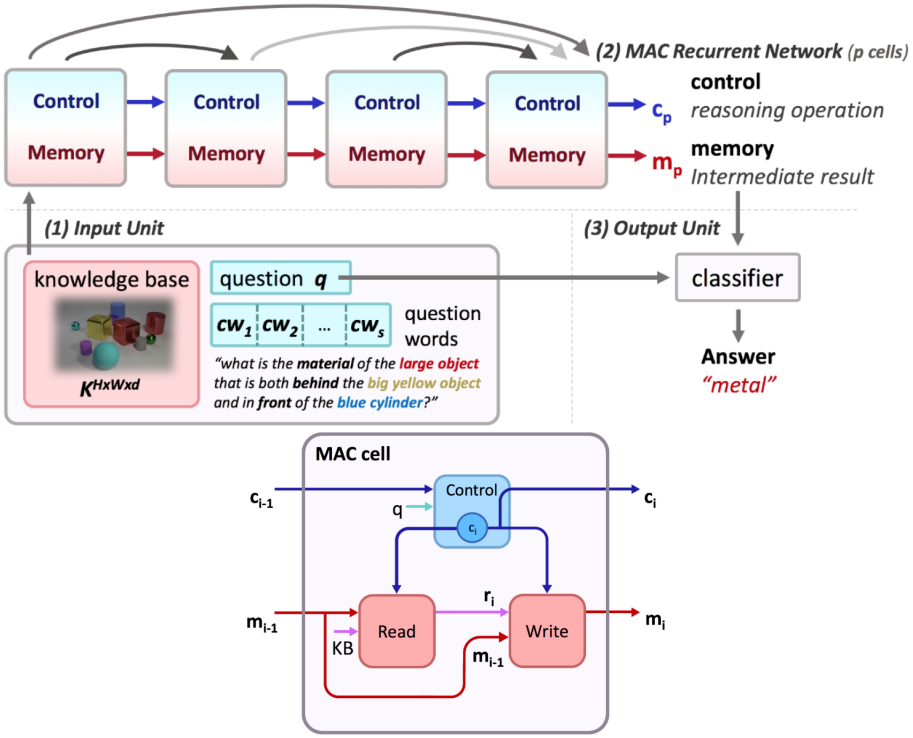


Figure 2.16 (Top) Architecture overview of the MAC network model. Similar to the SNMN model, the image and question are passed to a sequence of MAC cells which share the intermediate results in a memory structure and a classifier uses that memory to predict the answer. (Bottom) Overview of a MAC cell, showing how it processes memory information and control state to perform reasoning and produce a new intermediate result and control state.
Source: Hudson and Manning[17]

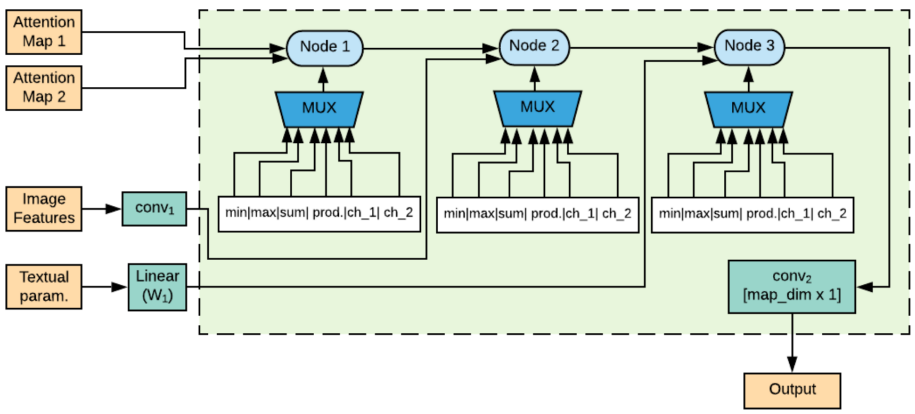


Figure 2.17 Architecture of a 4-parameter LNMN cell which accepts image features, two attention maps, and textual parameter as input
Source: Pahuja *et al.*[24]

2.3.5 Meta Module Network

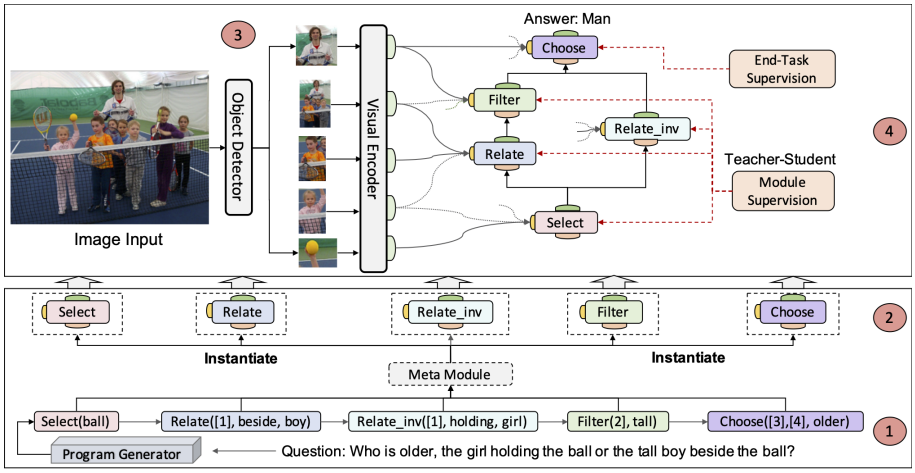


Figure 2.18 Architecture overview of the MMN model, starting with the question-parsing (bottom) until it predicts the final answer (top)
Source: Chen *et al.*[26]

NMN relies on its neural modules to understand and predict answers to VQA questions. However, as the complexity of the questions scales up, so would the module set need to be augmented accordingly which would lead to greater complexity. This also means the model cannot be applied to new questions which use newly-seen tasks is introduced (such as training primarily for object relationships but then being tasked with counting and object-based arithmetic). To address this, Chen *et al.* introduced MMN [26] which uses general-purpose network modules instantiated on-the-fly according to the key-value pairs provided by the model. Given a function recipe — denoting what task types and parameters are required for an instance of a module — a new module is instantiated according to this spec. Using this approach, when an unseen recipe is encountered, a new module can still be instanced using pretrained parameters and similar trained recipes. To train these modules, the module-generating function itself is trained using a learning strategy based on the Teacher-Student framework[26]. Given an entry from the GQA dataset, the model generates a series of goals using the scene-graph that the instantiated modules must learn. By instantiating modules which get closer to reaching these intermediate goals, the module generator learns to understand the scene-graph of each question-image pair, similar to how a 'Student' would learn the reasoning needed to arrive at the same answer that their 'Teacher' expects.

2.3.6 Coreference Neural Module Network

One of the main challenges with visual tasks discussed so far (VQA, VCR, and VD) is visual coreference resolution. Given a sentence or sentences which, when referrencing

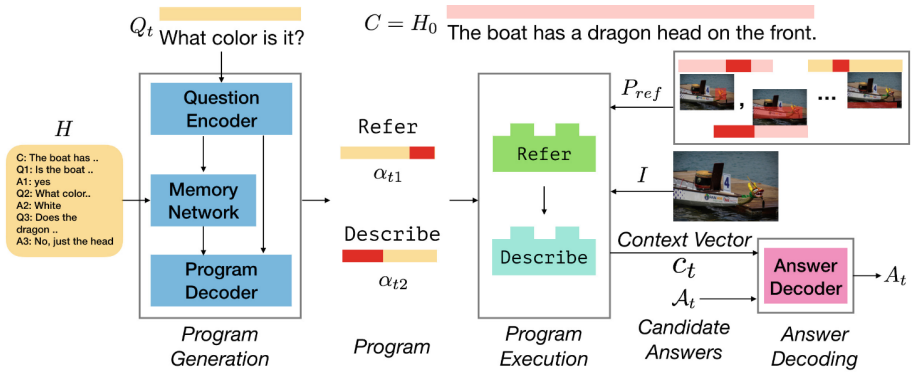


Figure 2.19 Architecture overview of the CorefNMN model, with emphasis on the newly-introduced 'Refer' module and its internal function.
Source: Kottur *et al.*[27]

an entity, may refer to an entity more than once with more than one noun, phrase, or pronoun (eg. 'Is the lady handing her phone to her sister, the other lady' where 'lady' and 'her' both refer to one lady despite another lady being present in context). To improve the NMN model accuracy in detecting and handling this problem, Kottur *et al.* introduce CorefNMN, an N2NMN-based model augmented with new modules to handle VD tasks [27]. One of the main changes to the model architecture is the introduction of a reference pool; a dictionary containing the output attention value of the 'find' modules of the program using the text parameter input as the key. Each value in the reference pool is therefore an entity which is paired with the first word/phrase to identify it. Using this dictionary as input, alongside a textual parameter input, a new 'Refer' module attends to the entity referred to inside the dictionary, producing a soft attention over the dictionary to select the most likely attention map. Internally, the module measures two things; the likelihood of each key against the target text parameter, and how long its been since the target entity was last mentioned. The softmaxed value of each key is then applied to the attention map values to obtain the final module output. Aside from the 'Refer' module two other modules are also introduced; a 'Not' module which effectively inverts the input attention map to focus on other regions of the image, and an 'Exclude' module which looks for instances of the specified entity in the text parameter which are not found in the input attention which is also provided.

2.3.7 Neural Module Network for Visual Dialog

Building upon the Coreference Neural Module Network (CorefNMN) model, Cho and Kim introduced NMN-VD[28] which improves upon the neural module inventory and program generation of its CorefNMN predecessor. The model builds upon the CorefNMN model by improving how the model generates programs based on impersonal pronouns. Since impersonal pronouns refer to objects not previously encountered in dialog and

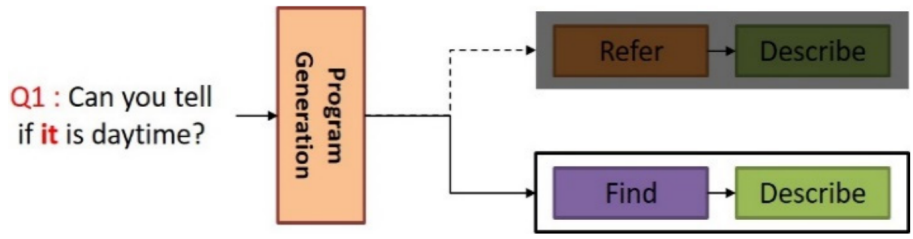


Figure 2.20 An overview of how the NMN-VD handles impersonal pronouns by avoiding the *Refer* module at program generation and using *Find* modules instead.
Source: Cho and Kim[28]

would thus lead to incorrect association between the pronoun and a completely unrelated object. To address this, the model checks for impersonal pronouns at the question stage and does not use the *Refer* module while still using *Refer* modules for personal pronouns. Aside from this, a new output module — known as the Compare module — is also introduced to handle comparison tasks between two objects. This is done by comparing the visual region containing both objects, the image features, and the textual feature parameter, to compute a final result. Finally, the model also improves upon the *Find* module by using three attention iterations to produce more accurate attention regions.

2.4 Discussion

This section gives a retrospective of all that was explored in the reviewed literature, starting from the computer vision-language tasks tackled, the problems that arose, and how they approached them with possible solutions.

2.4.1 Vision-Language Tasks

VQA is the first task discussed and the simplest in structure and challenge; one image, one question, and one required answer which can be multi-label (open-ended with one or more tokens) or multi-class (only one accepted answer such as counting, true/false questions, etc). Of the reviewed datasets, it has the largest coverage [1, 5, 7, 8] with the CLEVR dataset having the greatest testing coverage among NMN-based models[4]. This would make sense as the dataset, while using synthetic images, prioritises highly-compositional questions which require multiple reasoning steps to predict an answer, similar to GQA.

VCR is the next discussed task type to be formalised[2]. This task extends VQA by using realistic images taken from still video frames, omitting knowledge mostly found in the moments leading up to the taken image. Through this method, models will need to focus on inferring knowledge either from commonsense knowledge or from finer details in the image. Unlike VQA, the VCR dataset[2] is one of the only few datasets present

	SHAPES	VQA	CLEVR	GQA	VCR	VisDial
NMN[3]	Yes	Yes	No	No	No	No
N2NMN[13]	Yes	Yes	Yes	No	No	No
SNMN[16]	No	Yes	Yes	No	No[1]	No
NMNs±[22][2]	No	No	No	No	No	No
DPNMN[23]	No	No	Yes	No	No	No
MAC[17]	No	Yes[3]	Yes	Yes	No	No
LNMN[24]	No	No	Yes	No	No	No
MMN[26]	No	No	Yes	Yes	No	No
R2C[2]	No	No	No	No	Yes	No
MERLOT[20]	No	No	No	No	Yes	No
CorefNMN[27]	No	No	No	No	No	Yes
NMNVD[28]	No	No	No	No	No	Yes

[1] This will be implemented and tested in this study.
[2] Was only developed and tested on a bespoke dataset[22].
[3] Tested and evaluated on v1.0 of the dataset[17].

Table 2.4 Summary of reviewed models which were trained and tested against which datasets.
Source: See model names for references.

		SHAPES[3]	**VQA (2.0)**[1]	**CLEVR**[7]	**GQA**[8]	**VCR**[2]	**VisDial**[10]
	Task Type	VQA	VQA	VQA	VQA	VCR	VD
Details	# Images	244	204K	100K	113K	110K	140K
	# Questions	15K	1.1M	853K	22M	290K	1.4M
	# Answers	244	11M	853K	22M	290K	1.4M
	Image Type	Synthetic	Realistic	Synthetic	Realistic	Realistic	Realistic

[1] Sourced from `https://visualqa.org/`.

Table 2.5 Breakdown of which visual tasks each dataset targets including the statistics of each dataset.
Source: See citations near dataset names.

for this task type due to its recent introduction. The R2C and MERLOT-RESERVE models that were reviewed in Section 2.2.5 and 2.2.6 were both trained and tested on this dataset.

VD is the last task type of these three to be introduced and formalised[10]. It follows a more human dialogue-like flow where each image is paired with a caption and question-answers pairs provided as data to the model. The questions and answers build context around the image which offer a stricter benchmark on visual comprehension for compositional models such as NMN. Similar to VCR, the VisDial dataset is one of the only datasets which present this task type [10].

Despite being task types with different data layouts and amounts of input data, they all share the same goal of providing answers to questions which are grounded in images. The main difference between the tasks is in how each of their datasets tackle the various pitfalls and challenges of answering these questions. CLEVR, SHAPES, and GQA, all primarily focus on the compositionality of their questions with a focus on how compositional models perform reasoning steps to get to the correct answer[3, 7, 8]. Table 2.5 shows that VQA, VCR, GQA, and VisDial all use natural or realistic images instead of synthetic computer-generated images, arguing that since these better mimic real-life scenarios, they would allow a model to adopt more robust reasoning[1, 2, 8, 10]. One experiment by [29] — where an N2NMN-based model was trained on CLEVR and then tested on both the CLEVR test set and a custom dataset of CLEVR-like realistic images — found the model had no consistency in both accuracy between both test sets and between multiple questions around the same image [29]. They also concluded that further experiments and analysis on the pretraining of models on virtual datasets would need to be carried out before determining if virtual datasets would be viable for real-world applications or not, and just how much of an impact real-world variables such as lighting and noise affect model predictions [29].

2.4.2 Modules of the NMN Models

Now that the vision/language tasks have been discussed, the approaches that each model takes to solving the tasks in a compositional manner will be explored.

Marked as the first major step for solving a task, the model must first determine the layout of neural modules, parameters to be passed between these modules, and which module instances to activate. This challenge is known as Neural Architecture Search (NAS), and is the task of finding a suitable model architecture across a search space of possible architectures (as seen in Figure 2.21)[30]. The architecture chosen is the one most likely to provide the highest prediction/performance for the given model input when executed. This plays a core part in NMN models and is explored in Table 2.6 since having high-performant modules would not make for good predictions without

the proper layout and input parameters. Based on the models reviewed, most appear to employ discrete (D) architecture selection either based on rule-based parsing [3, 26] or with a Sequence-to-Sequence (Seq2Seq) RNN to generate the layout [13, 22, 23, 27, 28]. On the other end, some models employ a non-discrete (ND) soft layout selection, all of which use a BiLSTM cell to generate module weights and text attention using the given input text embeddings [16, 17, 24].

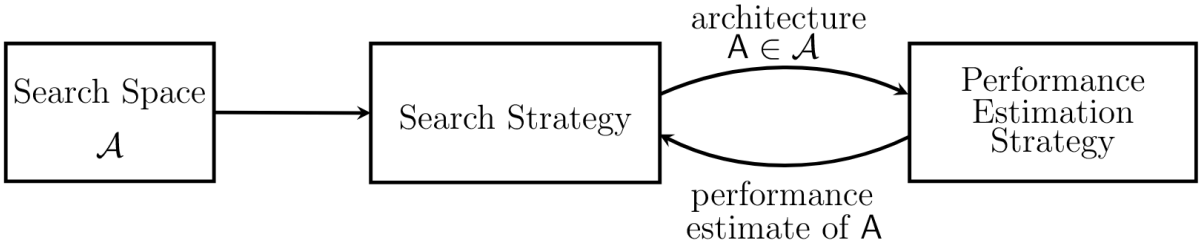


Figure 2.21 An abstract overview of the task showing how a search strategy chooses the architecture for a search space and how to preemptively filter architectures that obviously wouldn't be suitable using PES.
Source: Elsken *et al.*[30]

Following the layout selection are the actual modules. Each model has an inventory defined by several neural module types which are dedicated to specific tasks such as finding objects or comparing, etc. Most of the models discussed have specific, hand-designed module types (S) that are designed to learn best at specific tasks [5, 13, 16, 22, 23, 27, 28]. Most models with specific types share a very similar inventory with modules such as finding objects, relocate attention, combine attentions, etc. Some models implement additional module types for even more specific use-cases such as arithmetic tasks [22] or for performing visual coreference resolution [27, 28] (see chapters 2.3.6 and 2.3.7). There are some models however, which employ an inventory of generic module types referred to as cells (G) that the model must learn itself what modules it should instance and use for solving the tasks for which it's being trained [17, 24, 26]. While the implementations vary across models, they generally posess inputs for at least one or more visual/textual attentions, a general-purpose unit which applies some function/s on the inputs to produce an output, and (optionally) an answering unit for producing the final output that the model will convert to an answer prediction.

Aside from the modules, some models also support a memory structure for storing intermediate mdoule outputs. In the case of both the SNMN and LNMN, these are in the form of a shared differentiable memory stack. In both models, the modules/cells are both able to push/pull attention maps to the stack using a soft-selected stack pointer based on the module/cell weights. The MAC model also implements memory, but in a different implementation and for a different purpose. The model is trained with a fixed-length sequence of MAC cells. However, each cell has a second output — besides the attention outputs — which contains a hidden state created by the previous cell. The cell

can compute a new hidden state for sharing with the next cell but it can also interpolate between this new state and the previous state, effectively deciding whether or not to skip its own reasoning step. This enables the model to soft-reduce the number of reasoning steps it can use in a differentiable manner.

2.4.3 Learning strategies

Creating a layout and module arrangement is half the problem, the other half is training the model to recognise what works best. To train the modules and layout generator of a module, a training strategy would need to be employed, either as a single strategy training the model in an end-to-end manner, or multiple strategies working in tandem but training the layout generator and modules separately.

One major factor deciding how the discussed models are trained is their layout differentiability. If a model constructs its layout using a non-discrete or soft selection (such as the SNMN model using module weights and text attention), it is considered fully differentiable since the layout selection generated will be unique to the given input and no other input/s. If the model uses a discrete selection (such as the N2NMN model which produces discrete layouts), it is not fully differentiable since different inputs can produce the same non-unique layouts.

As seen in table 2.6, most models are in fact not fully differentiable, using either rule-based parsers[3, 26] or Seq2Seq RNN-based layout generators[13, 22, 23, 27, 28]. These models largely use reinforcement learning — with a layout policy being used to optimise the layout generator — and using backpropagation on those areas of the model (such as the neural modules) that are differentiable. On the other hand, the fully-differentiable models all feature a BiLSTM for layout generation and produce variable-length layouts[17]. These use backpropagation across both the neural modules and layout generator.

Additionally, some models explore different learning beyond just reinforcement or backpropagation. For instance, LNMN uses DARTS[25] which is a differentiable learning strategy based on backpropagation. MMN — in contrast to DARTS — proposed a learning strategy based on the Teacher-Student framework[31] where a reward/guideline metric is generated by a 'Symbolic Teacher' that the meta modules learn to imitate as 'Students'[26].

2.5 Conclusion

In this literature review, we have explored a selection of models and datasets designed to solve ML visual/language tasks. We have also touched on the VCR task which will be tackled by this work. In the following chapters, we will discuss how the model was

VQA/VD Models				
Model	NAS approach	Selection	Modules	Type
NMN	Rule-based parsing (see Chapter 2.2.1).	D	find, transform, combine, describe, measure	S
N2NMN	Seq2Seq RNN (See Chapter 2.2.2).	D	find, relocate, and, or, filter, count, exist, describe, less, more, equal, compare	S
SNMN	BiLSTM-generated module weights with txt attention (See Chapter 2.2.3).	ND	find, transform, and, or, filter, scene, answer, compare, noOp	S
NMNs±[22]	Similar to N2NMN with type-constrained grammar[32].	D	find, filter, relocate, find-num, find-date, count, compare-num-lt, time-diff, find-max-num, span, compare-date, add, sub	S
DPNMN	Similar to N2NMN with an RPN for spatial information.	D	find, relocate, and, or, describe, compare	S
MAC	BiLSTM and fixed-length cell array where each cell can 'skip' itself and relay input to the next cell.	ND	MAC cells	G
LNMN	Similar to SNMN	ND	General modules (Expand further).	G
MMN[26]	Rule-based parser feeding into a coarse-to-fine program generator.	D	Meta modules	G
CorefNMN[27]	Similar to N2NMN but augmented with a memory network for attention-over-text.	D	find, relocate, and, or, filter, count, exist, describe, less, more, equal, compare, not, refer, exclude	S
NMNVD[28]	Similar to CorefNMN.	D	find, relocate, and, refer, describe, compare	S

Table 2.6 A full breakdown of the model inventory and layout construction architecture of each model discussed. 'Selection' denotes whether the module selection of a model is a soft selection (ND) — and thus trainable with back-propagation — or fully discrete (D) and instead requires an alternate learning strategy such as reinforcement learning. The 'Type' specifies whether the modules listed are 'Specified' (S) in that they have a fixed behaviour and only their weights and biases are learned, or they're 'Generic' (G) and are able to learn and apply different behaviours without prior implementation or knowledge. **Source:** Adapted from Fashandi[4] with additional model information from models covered in this literature review

adapted to work on the VCR task and dataset, the experiments and setups for evaluating on VCR, the ablations done, and conclude with a discussion of future work.

3 Methodology

Now that the compositional models and the VCR task were explored, this chapter will follow by going into detail about how the SNMN model was adapted to the VCR dataset. The details of how the dataset is prepared will be discussed first, going over any the extracting of image and text features, how these are processed, and any other data to be extracted. The model adaptations performed will then follow, covering how the model will operate on the newly-generated data. Finally, the experiments and how they were set up will be discussed.

3.1 Data preparation

Before the model can begin to perform VQA tasks, the required data must first be prepared into a format that will be understood by the model. The procedure below is followed across all datasets trained and tested on, with variations being made depending on the structure of the data.

The images are first pre-processed into a feature-set using a ResNet-152 model [18] — pre-trained on the ImageNet[1] dataset [33] — which outputs a feature map of each image. The question-answer pairs found in the dataset are processed after the images. For each question and answer sentence, the sentence is first collected into a single corpus for later processing. The sentence is tokenised into a series of words, numbers, and/or symbols representing the sentence. Each occurrence of a token in the sequence is recorded into a vocabulary file which keeps track of every token encountered in the dataset. Each entry in the vocabulary file contains both the token and the number of occurrences of the token in the corpus.

The image features, questions, and answers, are then converted into an image database (imdb) file. This file contains a record for every VQA task, for each split of the dataset (training set, validation set if present, and test set). Each record identifies the image by its feature file. A question and all relevant answers are saved as tokenised variants in the record, along with the correct answer for that question. If the split does not mark the correct answers (such as with the test set), then the correct answer fields are simply omitted.

With the imdb files prepared, next would be to prepare the text embeddings for the model. This is done by converting each token in the vocabulary file into a 300-dimensional word vector following the same procedure as was used by Hu *et al.* in their N2NMN model [13]. For this, a GloVe model [34] was trained on the prepared dataset corpus and vocabulary — obtained when preparing the imdb files — to produce a word

[1] `https://www.image-net.org/`

embeddings file, where each entry belongs to the token on the same line number in the vocabulary file.

3.1.1 Preparing for VCR data

There are a number of properties about the dataset that need to be handled when preparing the dataset for processing. To begin with, each VCR task in the dataset is referred to as an 'annotation' which links one unique question and several answers and rationales to an image. Each question, answer, rationale, and image, have a unique annotation index based on the fold and split they're found. These indices are important as each annotation entry inside the dataset uses these indices to refer to which answer/rationale are correct and which image to use. There is only one correct answer/rationale per-annotation, which is the one unique to that annotation alone - all other wrong answers/rationales in that annotation are copies from other annotations and referenced as such by their indices. Aside from these, an 'interestingness score' is provided by the annotation authors (not the dataset authors themselves, but the ones to whom the annotation task was outsourced) for each annotation, as a subjective ranking of how interesting the annotation would be. There's also a 'likelihood score' provided by the annotation authors whereby they assess how likely it is that the question, answer, and rationale given by them actually fit the context of the source movie the annotated image was taken from. Finally, there's a ranking of each answer and rationale by correctness in descending order, where the correct choice is rank 0, rank 1 would be the first wrong choice, and so on. For the purpose of this work, both the interestingness score and the likelihood score are ignored and the only correctness considered per-annotation is whether the choice is correct or not.

Aside from the annotations entries, each image in the dataset contains a metadata file, describing the image. Each file contains the names of object classes found in the image (such as person, car, food, etc). Aside from the above classes, each object is also identified by a region which can be used to locate the object in the image, and a segmented polygon which highlights the object in the image. The model will not use the object regions in the metadata file because it would fall outside the scope of this work.

Like in the previous datasets, the VCR dataset is compiled into imdb files. These files contain the same image name, feature path, the question, all answers, and all rationales, for each annotation. Besides the above, additional preprocessing is done to make the data compatible with the model and also obtain the word embeddings. The sentences also make reference to the objects described in the image metadata file by pointing to an index. This is replaced by the object class described in the metadata file to avoid troubles encountered in inferring what object is being referenced by the image (for eg. a sentence like 'What is [1] pointing to?' becomes 'What is *person* pointing

to?'). Each token encountered, along with the total number of occurrences of that token, is extracted into a vocabulary file. Each sentence (question, answer, or rationale) is added to a corpus file, which will be used to by GLOVE to prepare the word embeddings. Currently, there is no filtering made when preparing the corpus, so duplicate sentences, whether correct or wrong, are also added. Additionally, BERT will also be used for generating word embeddings since Zellers *et al.* found that their model performed best for VCR when using BERT embeddings[2].

3.2 Model adaptation

For the purposes of this study — to perform VCR tasks using the SNMN architecture — several adaptations and modifications were needed to the model. The VQA implementation of the model is used as the base since it most closely matches the VCR dataset since both the VQA and VCR datasets represent plausible real-life settings (the CLEVR implementation, like the dataset, is mainly focused on benchmarking the performance of VQA on synthetic images).

3.2.1 Layout generator

The biggest difference between the original implementation and the current implementation comes from the VCR dataset using single-choice (multiclass) questions instead of open-ended questions with one-word answers. The original model as a result would only encode the question text and ignore the answer/rationale text completely (see Figure 3.1). Since the questions are single-choice, the model would need to look at each answer (and rationale) individually, as if it were part of the question text. To support this, the input unit of model was modified to encode the question, answer, and rationale as input by concatenating their embeddings together into a single encoded vector (see Figure3.2). The input unit would also produce an attention mask that would properly identify the lengths of each input sentence without relying on padding. Once both the encoding and attention mask are produced, the layout generator proceeds with the same flow as the original model.

3.2.2 Output and loss function

Another problem arising from this question type is that it's incompatible with the original loss function of the program. The original loss function used a softmax cross-entropy over the whole vocabulary, which is good for multilabel classification (selecting one or more correct choices) but not for multiclass classification (only one correct choice). The

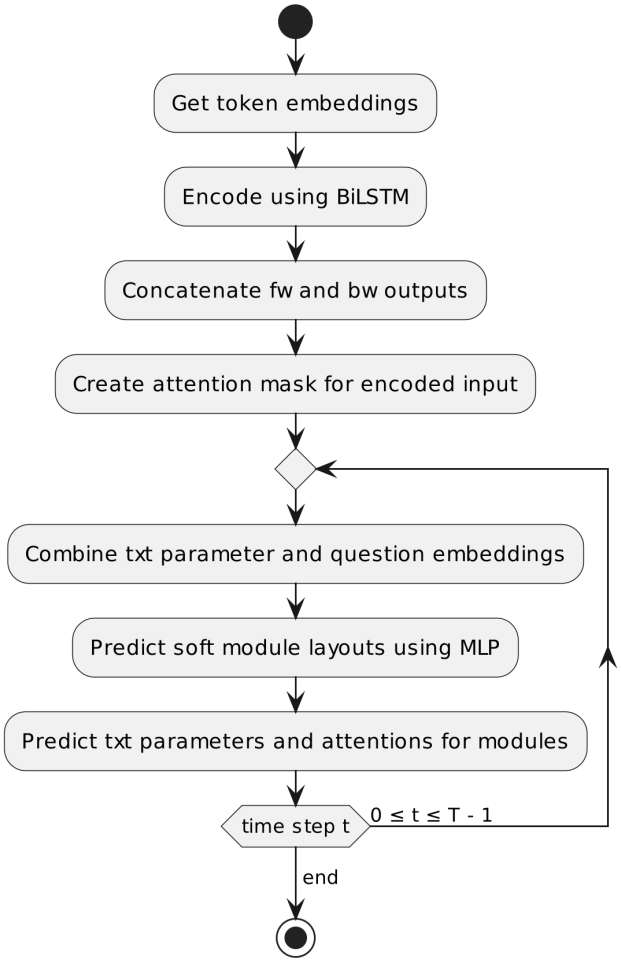


Figure 3.1 Flow diagram of how the SNMN converts the input question to a layout.
Source: Original diagram prepared for this study

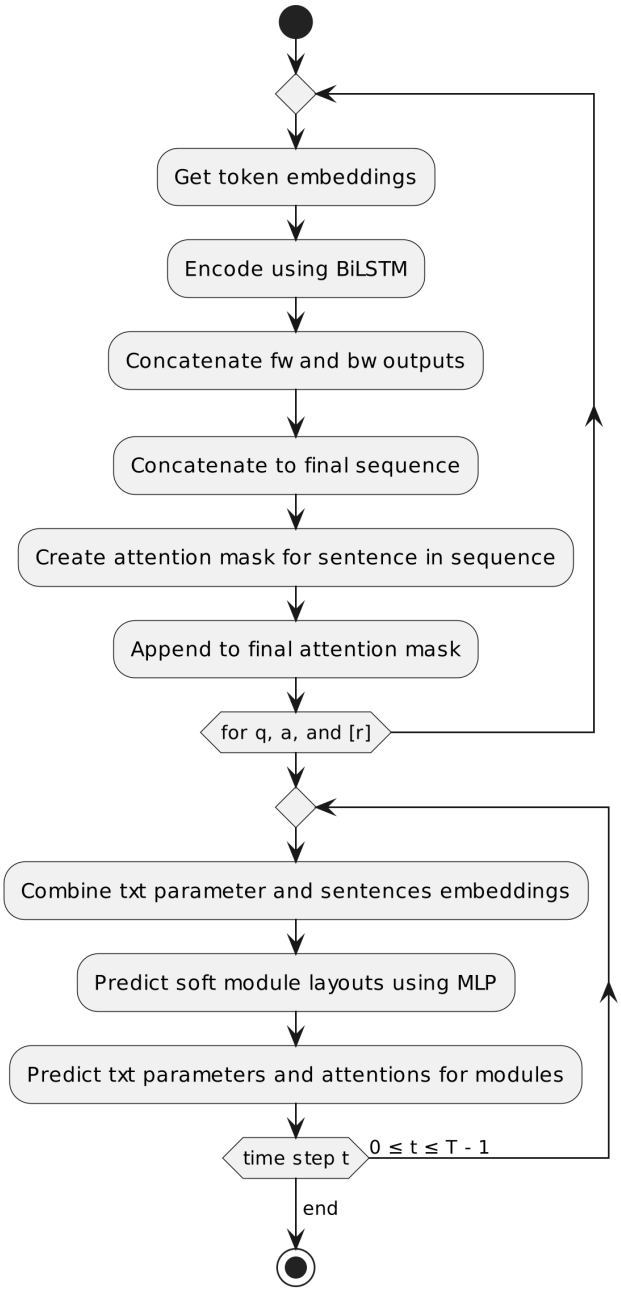


Figure 3.2 Flow diagram of how the VCR-adapted SNMN converts the input question, answer, and rationale to a layout. Note that the rationale is only used when answering VCR questions in QAR mode.
Source: Original diagram prepared for this study

new loss function uses a sigmoid cross-entropy over the prediction logits for each combination of question, answers, and rationales. In other words, for each VCR task with one question, four possible answers, and four possible rationales, the loss function will expect sixteen probability scores, with the score closest to 1 being given to the correct answer and rationale. To form the input for the loss function, the model is run once for each input combination for the one VCR task, a softmax vector is created from the combination of outputs, and used as input for the loss function.

3.3 Experiments

Several experiments were conducted on the model to evaluate its performance on VCR. The experiments were designed to test the models accuracy over the three main task types (Q→A, QA→R, Q→AR) discussed in Section 2.1.5 using different word embedding approaches and input encoder configurations. Combinations of task type, BiLSTM configuration in the model input unit and input token embedding types (contextual and non-contextual) were chosen as experiments to target what factors improve performance and what task types see the biggest improvement in prediction accuracy.

An additional experiment was also conducted on QA→R to determine how big an influence the input answers had on the predicted rationale, and whether using previously-predicted answers as input would result in a significant drop in accuracy. To test this, the model was first trained to predict the correct answers in Q→A mode, exporting the predicted answers to a prediction file. Then a new model was trained in QA→R mode, using the predicted correct answers instead of the true correct answers. This first model would be referred to as the Q→Ap model with the latter being the QAp→R model.

3.3.1 Setup

The model was trained on a single machine node with up to two GPU nodes to use, depending on the configuration used for each experiment. Each model was trained using a configuration file that selected the hyper-parameters used and task type to be tested. To keep track of the model training history, model checkpoints were taken during training after every predefined number of steps. Training was performed up to the specified step count, after which the model was evaluated using each of the recorded checkpoints. To avoid recording the performance of an overfit model, the model performance chosen was taken from the step with the highest accuracy and not the most recent step.

3.3.2 Ablations

A set of ablation experiments were developed as part of the main experiments to determine the accuracy contribution of various components of the model. Namely, those experiments deciding embedding types and BiLSTM configuration.

The ablation tested is the use of context-aware token embeddings over context-free token embeddings, and whether a higher embedding dimensionality contributes to improved performance or not. For this, BERT was chosen as the sentence-level context-aware embedding, retaining the same embeddings published by Zellers *et al.* alongside the VCR dataset and their R2C model[2]. To test context-free embeddings, GLOVE is used since it is the same embedding scheme used by SNMN and N2NMN in their training and evaluation. As an additional measure in testing context-free embeddings, Word2Vec embeddings are used which are generated using a Continuous Bag-Of-Words model[35]. Two sets of Word2Vec embeddings are generated, one with 300-dimensional vectors to match the GLOVE dimensionality and one with 768-dimensional vectors to match the BERT vectors. This will both determine how big an effect both vector dimensionality and sentence-level context have on accuracy.

Another ablation is the testing of how the model input unit encodes the input sentences. Seeing as the original model was designed with only a single token sequence in mind, one BiLSTM encodes the sequence, but since we have more than one sequence as input, multiple BiLSTMs are needed to encode each sequence. As an ablation, another experiment is conducted where the model uses a single BiLSTM to encode all three sequences. This evaluates whether a single BiLSTM would bottleneck the layout generation or improve it.

4 Results

Now that the methodology has been covered, the results of the experiments described in Section 3.3 will be presented. A discussion of the results and the challenges encountered will also be given.

4.1 Evaluation results

The accuracy results for the experiments can be found in Table 4.1. All experiments were carried out by training on the vcr-train set first, then testing against the vcr-eval set. Each experiment model is trained for up to 75k steps with checkpoints taken at 5k step intervals. The batch size used is 64 for Q→A tasks, 16 for QA→R and Q→AR using BERT, and 24/32 for QA→R and Q→AR using Word2Vec-768 and Word2Vec-300 respectively.

4.2 Challenges

Several problems were encountered during training. The most common problem was unstable learning which resulted in NaN loss errors or slow learning. NaN losses were frequent when training the model using word2vec embeddings and didn't occur at all when using GLOVE or BERT. Slow learning rates were observed during glove and word2vec training, especially with Q→AR training. Another factor in getting good predictions was batch size, which was strained by the task size and amount of data involved. To work around the limited batch size available, training on QA→R and Q→AR modes was performed on a multi-GPU setup to allow for increased batch size. This setup appeared to produce better learning rates and prediction performance compared to single-GPU training in some models, at the cost of increased runtime due to the overhead of keeping the training parameters synchronised between the two GPU models. To examine these findings, an experiment was conducted by comparing the performance of a model trained on multiple GPUs with increased total batch size vs single-GPU training, the results for which can be found in Table 4.2. While it appears that the model does indeed improve on the results obtained, they do not appear to be stable, with the 2-GPU-batch24 result performing 0.1% worse than the 1-GPU-batch24 model.

4.2.1 Discussion of results

In almost every task type, the BERT embeddings model outperformed the other models with GLOVE or Word2Vec embeddings and eaching as high as 63% in Q→A tasks.

Experiment results (vcr-eval)							
Wrd Embed.	Q→A Shr.	Q→A Sep.	QA→R Shr.	QA→R Sep.	QAp→R Shr.	Q→AR Shr	Q→AR Sep.
Rand Guess	25%	25%	25%	25%	25%	6.25%	6.25%
Glove-300	52.1%	51.4%	**60.8%**	47.2%	25.7%[3]	6.4%[3]	6.4%[3]
BERT-768	**63.2%**	**63.8%**	59.5%[3]	**60.8%**[3]	**60.7%**[3]	**24.7%**[3]	**22.2%**[3]
w2v-300	32.5%[1]	35.1%	25.2%	32.8%[23]	-	-	-
w2v-768	32.8%[1]	33.9%	25.1%	32.3%	-	-	-

[1] Problems with loss function resulting in partial training or lack of training.
[2] Training program crashed at least once and had to be resumed from prior checkpoints.
[3] Trained on multi-Graphics Processing Unit (GPU) configuration.

Table 4.1 Evaluation results from running the model across combinations of different token embeddings, VCR task types, and layout generator configurations. *Shr.* refers to models with a shared BiLSTM while *Sep.* refers to one BiLSTM per input sentence.
Source: Original performance results obtained for this study.

QA→R Shr performance		
GPU Count	Unit Batch Size	Performance
2	32	**25.9%**[1]
2	24	25.4%
1	48	25.6%
1	24	25.5%

[1] A batch size of 48 was attempted, but resulted in Out-of-Memory errors on the training environment.

Table 4.2 Experiment results comparing differences in training performance between single-GPU and multi-GPU setups on QA→R tasks with shared BiLSTM and GLOVE embeddings. The batch size in multi-GPU experiments is per unit GPU and must be multiplied by the number of GPUs to obtain the true batch size.
Source: Original performance results obtained for this study.

Metric	Count	Percentage
Correct answers (Q→Ap)	14,483	**54.58%**
Correct rationales (QAp→R)	16,108	**60.68%**
Correct answers, correct rationales	8,927	33.64%
Incorrect answers, incorrect rationales	4,876	18.38%
Correct answers, incorrect rationales	5,556	20.94%
Incorrect answers, correct rationales	7,175	27.04%
Total records	26,534	100.00%

Table 4.3 Breakdown of results for the QAp→R BERT-model experiments from Table 4.1. Note that the 'correct rationales' metric reflects the score obtained in the results while the 'correct answers' metric uses vcr-val answer predictions by the original Q→Ap BERT-model which seeded the data for the QAp→R model.
Source: Original performance results obtained for this study.

GLOVE achieved the second-best performance overall with up to 52% in Q→A, but showing no learning signs in Q→AR tasks with an avg. accuracy only 0.15% higher than random guessing. Word2Vec performend the worst across all tasks and failed to complete the full training course on QAp→R and Q→AR tasks. The results suggest that the contextual embeddings generated by BERT contribute significantly to the model performance (aligning with the results found by Zellers *et al.* where using GLOVE also resulted in worse accuracy[2]).

Interestingly, most models seem likely to overfit, producing peak accuracy at evaluation checkpoints between 15k-30k iterations for Q→A and QA→R tasks. This behaviour is most apparent in the BERT models where prediction accuracy peaks at around 20k iterations before regressing. The Q→AR task was less likely overfit, with the model peaking at the end of training, suggesting more training iterations are needed. It might be suitable to explore different training strategies in the future which would help prevent overfitting, such as the training strategy used by DP-NMN as discussed in Chapter 2.3.2.

When performing the QAp→R tasks, BERT achieved 60% while GLOVE failed to achieve a meaningfully higher score than random guessing (26% compared to 25%) and Word2Vec failed outright to complete its training. It appears the BERT embeddings might allow for the model to compensate for possibly-incorrect answers, although this would need to be explored further. If true, this may explain why the GLOVE and Word2Vec models fail to produce meaningful accuracy as it does not rely on sentence-level context between sentences. To better analyse this, the exact results of evaluating both this model and the seeding model (the Q→Ap BERT-model) on vcr-val are provided in Table 4.3. The results obtained are then merged together into a single prediction set and compared to the true answers from the dataset. While the Q→Ap model scored an answer prediction accuracy of 54.58% and the QAp→R model scored a rationale pre-

diction accuracy of 60.68%, only 33.64% of all answer-rationale pairs are both correct. Additionally, 5k of the 14k answer predictions did not lead to a correct rationale prediction (38.36%). Despite this, 7k of the 16k correct rationale predictions obtained these results with incorrect answer predictions (44.54%). The results suggest that while the models perform well individually, they do not serve well as intermediate output. That said, it seems that when both models predict individual scores, the combined predictions have better accuracy than the highest-scoring Q→AR model's score of 24.7%. This may be another avenue for future work whereby two separate models predict each answer and rationale separately and the aggregated predictions serve to solve Q→AR tasks.

4.2.2 Qualitative analysis against other VCR models

The top results from Table 4.1 are compared against the other VCR models in Table 4.4. As expected, the model does not outperform the VCR models, being almost 40% less accurate in Q→AR tasks when compared to MERLOT-RESERVE. Q→A and QA→R however produced comparable results to the R2C model, only being 6.5% worse at most. Given the large difference in accuracy between BERT and GLOVE, a large factor in the performance similarity might be attributed to BERT. It seems as though MERLOT-RESERVE might be making a large improvement thanks to the increased generalisability of both the model owing to its training, and for its subword-based embeddings using BPE tables. This would align with the growing number of generalised models such as MMN, LNMN, and now MERLOT-RESERVE. Another possible contributing factor is the training approach; whereas this SNMN model trained solely on the VCR-train set, MERLOT-RESERVE pretrained on a much larger dataset combining different data sources (image, text, and audio) in various combinations, and then fine-tuned onto VCR for testing.

Besides the embeddings themselves, there may also be the problem of subject inferrence. Currently, the model preprocesses the dataset before training on it such that unique instances of an object are replaced by the generic object name and so sentences can often become saturated with subjects (eg: a sentence like 'Why did [1] and [2] steal [3]'s bike' would become 'Why did person and person steal person's bike'). If there were a way for the model to better distinguish each object reference (such as the visual coreference resolution approach used by Neural Module Network for Visual Dialog (NMN-VD)[28]), the model might perform better in the QA→R and Q→AR tasks.

Results comparison			
Model	Q→A	QA→R	Q→AR
VCR (val)	63.8%	67.2%	43.1%
VCR (test)	65.1%	67.3%	44.0%
MERLOT-RESERVE (L)	**84.0%**	**84.9%**	**72.0%**
SNMN	63.8%	60.8%	24.7%

Table 4.4 Experiment results of the SNMN model compared to the other SNMN models. The results chosen were the highest-accuracy models from the previously-discussed experiments.
Source: R2C results: Zellers *et al.*[2], MERLOT-RESERVE results: Zellers *et al.*[20], SNMN results: Original performance results obtained for this dissertion.

5 Conclusion

This study has explored how a subset of computer language-vision models known as the NMN models could be adapted to not only answer questions about an image, but also provide the reasoning behind its answer; whereas the original model could express its reasoning through the image with which it was prompted, this new model can provide reasoning grounded in commonsense knowledge that is not immediately available in the image. Several computer language-vision tasks were explored — namely VQA, VCR, and VisDial — and the model-training challenges that they target. A selection of NMN models were explored to identify what makes these models desirable among other language-vision model types, namely their compositional nature and their explainability. This study presented how one such NMN known as the SNMN performs on the VCR dataset, including the modifications needed to the make the model support the dataset. The results are not state-of-the-art — but comparable in accuracy to the reference VCR model in the Q→A and QA→R tasks — and struggles when longer input text sequences are supplied. The model memory stack was not examined in this study to determine how the model arrives at its answers, which may serve as a good starting point for future work to explore how the model attends to the image and text features at each timestep, especially where rationale is concerned. Such an analysis may help to understand how the model would need to be modified to improve upon its accuracy in VCR tasks. One such opportunity may be to explore the emergence of generalised models (such as LNMN, MMN) and see if generalised reasoning would allow the model to learn better from commonsense knowledge than specified reasoning. Additional work may explore new training strategies, especially to minimise training errors such as those encountered during this study (Zellers *et al.*'s MERLOT-RESERVE demonstrated that training on a generalised dataset and fine-tuning onto VCR produces a model with very good accuracy). The qualitative analysis of the QAp→R model results suggested that two models may perform better at Q→AR than one, due to how the accuracy of the QAp→R model and seed model produced a combined accuracy higher than the Q→AR model. Further experimentation with QAp→R tasks may complement the above-mentioned analysis on the models reasoning. Finally, one other experiment worth considering would be to change the order information is presented to the model; in all cases, the model was presented with the text inputs in the order a person would naturally expect them, starting with a question, followed by the answer, and (optionally) the rationale to that answer. What if the model were presented with the rationale first — or the answer first — and the question last, in a Reverse Polish Notation-inspired format?

References

[1] S. Antol *et al.*, "VQA: Visual question answering," eng, in *2015 IEEE International Conference on Computer Vision (ICCV)*, IEEE, 2015, pp. 2425–2433, ISBN: 1467383910.

[2] R. Zellers, Y. Bisk, A. Farhadi, and Y. Choi, "From recognition to cognition: Visual commonsense reasoning," no. arXiv:1811.10830, Mar. 26, 2019, ISSN: 2331-8422. DOI: 10.48550/arXiv.1811.10830. arXiv: 1811.10830[cs]. [Online]. Available: http://arxiv.org/abs/1811.10830.

[3] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein, "Neural module networks," eng, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2016, pp. 39–48, ISBN: 9781467388511.

[4] H. Fashandi, "Neural module networks: A review," eng, *Neurocomputing (Amsterdam)*, vol. 552, p. 126 518, 2023, ISSN: 0925-2312.

[5] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein, "Deep compositional question answering with neural module networks," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, ISSN: 10636919, vol. 2016-Decem, 2016, pp. 39–48, ISBN: 978-1-4673-8850-4. DOI: 10.1109/CVPR.2016.12.

[6] T.-Y. Lin *et al.*, "Microsoft COCO: Common objects in context," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., Cham: Springer International Publishing, Feb. 20, 2015, pp. 740–755, ISBN: 978-3-319-10602-1. [Online]. Available: http://arxiv.org/abs/1405.0312.

[7] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick, *CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning*, Dec. 20, 2016. DOI: 10.48550/arXiv.1612.06890. arXiv: 1612.06890[cs]. [Online]. Available: http://arxiv.org/abs/1612.06890.

[8] D. A. Hudson and C. D. Manning, *GQA: A new dataset for real-world visual reasoning and compositional question answering*, May 10, 2019. arXiv: 1902.09506[cs]. [Online]. Available: http://arxiv.org/abs/1902.09506.

[9] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He, *Detectron*, https://github.com/facebookresearch/detectron, 2018.

[10] A. Das *et al.*, "Visual Dialog," eng, *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 5, pp. 1242–1256, 2019, ISSN: 0162-8828.

[11] D. Klein and C. D. Manning, "Accurate unlexicalized parsing," in *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - ACL '03*, vol. 1, Sapporo, Japan: Association for Computational Linguistics, 2003, pp. 423–430. DOI: 10.3115/1075096.1075150. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1075096.1075150.

[12] J. Nivre *et al.*, "Universal dependencies v1: A multilingual treebank collection," in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, Portorož, Slovenia: European Language Resources Association (ELRA), May 2016, pp. 1659–1666. [Online]. Available: https://aclanthology.org/L16-1262.

[13] R. Hu, J. Andreas, M. Rohrbach, T. Darrell, and K. Saenko, *Learning to reason: End-to-end module networks for visual question answering*, Sep. 11, 2017. arXiv: 1704.05526[cs]. [Online]. Available: http://arxiv.org/abs/1704.05526.

[14] D. Bahdanau, K. Cho, and Y. Bengio, *Neural machine translation by jointly learning to align and translate*, May 19, 2016. DOI: 10.48550/arXiv.1409.0473. arXiv: 1409.0473[cs,stat]. [Online]. Available: http://arxiv.org/abs/1409.0473.

[15] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, Jan. 29, 2017. DOI: 10.48550/arXiv.1412.6980. arXiv: 1412.6980[cs]. [Online]. Available: http://arxiv.org/abs/1412.6980.

[16] R. Hu, J. Andreas, T. Darrell, and K. Saenko, "Explainable neural computation via stack neural module networks," eng, in *Computer Vision – ECCV 2018*, ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, pp. 55–71, ISBN: 9783030012335.

[17] D. A. Hudson and C. D. Manning, "Compositional attention networks for machine reasoning," eng, *arXiv.org*, no. arXiv:1803.03067, Apr. 24, 2018, ISSN: 2331-8422. DOI: 10.48550/arXiv.1803.03067. arXiv: 1803.03067[cs]. [Online]. Available: http://arxiv.org/abs/1803.03067.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," arXiv:1512.03385, arXiv, Dec. 10, 2015. arXiv: 1512.03385[cs]. [Online]. Available: http://arxiv.org/abs/1512.03385.

[19] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," no. arXiv:1810.04805, May 24, 2019. DOI: 10.48550/arXiv.1810.04805. arXiv: 1810.04805[cs]. [Online]. Available: http://arxiv.org/abs/1810.04805.

[20] R. Zellers *et al.*, "Merlot reserve: Multimodal neural script knowledge through vision and language and sound," *arXiv.org*, 2022, ISSN: 2331-8422.

[21] B. Heinzerling and M. Strube, "BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, N. C. (chair) *et al.*, Eds., Miyazaki, Japan: European Language Resources Association (ELRA), May 2018, ISBN: 979-10-95546-00-9.

[22] J. Chen, X.-Y. Guo, Y.-F. Li, and G. Haffari, "Teaching neural module networks to do arithmetic," no. arXiv:2210.02703, Oct. 6, 2022, ISSN: 2331-8422. arXiv: 2210.02703[cs]. [Online]. Available: http://arxiv.org/abs/2210.02703.

[23] K. Su, H. Su, J. Li, and J. Zhu, "Toward accurate visual reasoning with dual-path neural module networks," eng, *Frontiers in robotics and AI*, vol. 7, pp. 109–109, 2020, ISSN: 2296-9144.

[24] V. Pahuja, J. Fu, S. Chandar, and C. J. Pal, "Structure learning for neural module networks," eng, *arXiv.org*, 2019, ISSN: 2331-8422.

[25] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," eng, *arXiv.org*, 2019, ISSN: 2331-8422.

[26] W. Chen, Z. Gan, L. Li, Y. Cheng, W. Wang, and J. Liu, "Meta module network for compositional visual reasoning," eng, *arXiv.org*, 2020, ISSN: 2331-8422.

[27] S. Kottur, J. M. F. Moura, D. Parikh, D. Batra, and M. Rohrbach, "Visual coreference resolution in visual dialog using neural module networks," eng, in *Computer Vision – ECCV 2018*, vol. 11219, Cham: Springer International Publishing, 2018, pp. 160–178, ISBN: 9783030012663.

[28] Y. Cho and I. Kim, "Nmn-vd: A neural module network for visual dialog," eng, *Sensors (Basel, Switzerland)*, vol. 21, no. 3, pp. 1–18, 2021, ISSN: 1424-8220.

[29] G. Sejnova, M. Tesar, and M. Vavrecka, "Compositional models for VQA: Can neural module networks really count?" *Procedia Computer Science*, vol. 145, pp. 481–487, 2018, ISSN: 18770509. DOI: 10.1016/j.procs.2018.11.110. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1877050918323986.

[30] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," eng, *Journal of machine learning research*, vol. 20, 2019, ISSN: 1532-4435.

[31] M. Zimmer, P. Viappiani, and P. Weng, "Teacher-Student Framework: a Reinforcement Learning Approach," in *AAMAS Workshop Autonomous Robots and Multirobot Systems*, Paris, France, May 2014. [Online]. Available: https://hal.science/hal-01215273.

[32] N. Gupta, K. Lin, D. Roth, S. Singh, and M. Gardner, "Neural module networks for reasoning over text," eng, *arXiv.org*, 2020, ISSN: 2331-8422.

[33] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, ISSN: 1063-6919, Jun. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[34] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. [Online]. Available: http://aclweb.org/anthology/D14-1162.

[35] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," eng, *arXiv.org*, 2013, ISSN: 2331-8422.

Appendix A Generating the input files used by the model

Before running the model, the dataset is first prepared into a set of binary tfrecords files. These files allow for streaming data to the model in a more optimised manner than json-based files. To prepare these files, the image features are first generated using a ResNet152 model, with each feature file saved as a tfrecords file. A script then generates the textual data through the following steps:

- Extracting the individual VCR entries and sorting them by set.
- Record the vocabulary found in all entries and output it to a file.
- Compile a corpus file from the entries.
- Save the entries into imdb files according to set.

The GLOVE word embeddings for the imdb files are generated using a script to perform the below steps:

- Generate a co-occurrence matrix on the corpus and vocabulary files that were previously compiled.
- Convert the co-occurrence matrix to a final 300-dimensional embeddings file.
- Convert the embeddings into a binary file for loading and parsing by the model at startup.

To generate the BERT embeddings files, the existing R2C author-provided VCR embeddings are downloaded. They are then extracted into a set tfrecords files according to the set they belong to (train, val, and test).

A script generates the Word2Vec embeddings from the previously-generated vocabulary file. The script is configurable to determine the model type used for generating the embeddings and the output vector size.

The final dataloader constructs an optimised data pipeline for the model to consume which concurrently loads and prefetches these separate file sources and maps them into the final expected format for the model to train, evaluate, and test itself.