Gabriel Santos Luz

Neural Network Architectures for CLEVRER

Belo Horizonte, Minas Gerais

2020

Gabriel Santos Luz

Neural Network Architectures for CLEVRER

Proposta de Pesquisa

Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Orientador: Douglas Guimarães Macharet

Belo Horizonte, Minas Gerais
2020

Sumário

1 Introduction

Current neural networks show poor generalization abilities compared to humans. In the paper [Greff, Steenkiste e Schmidhuber 2020], the authors argue that the cause for this problem is that existing neural networks are unable to represent, extract and relate symbol-like entities, which configures symbolic reasoning. In the machine learning literature there are several works that try to incorporate symbol manipulation to neural networks, among them, hybrid neuro-symbolic approaches and fully neural approaches have been studied. The authors argue that the latter is a promising approach and propose, throughout the paper, a conceptual framework to tackle the binding problem within connectionism.

The binding problem is, as stated in the paper: "The inability of existing neural networks to dynamically and flexibly bind information that is distributed throughout the network.". It is divided in three parts: Representation, Segregation and Composition. Segregation refers to how useful entities can be formed from sensory inputs, Representation refers to how these entities can be represented and their information separated, and Composition refers to how these entities can be manipulated to construct inferences, predictions and behaviors.

The paper presents a framework for understanding the binding problem in neural networks, alongside with a survey of machine learning approaches to solving each of the three parts. One important conclusion is that the three parts are inherently related and an effective solution to the binding problem should integrate representation, segmentation and composition, which is a positive point for purely connectionist approaches that can integrate these parts.

In the first part of this work, last semester POC 1, we explored a self-supervised learning algorithm, called Contrastively-trained Structured World Models (C-SWMs) [Kipf, Pol e Welling 2019]. It tackles the binding problem by generating and manipulating a graph structured representation of its input. We explored C-SWMs as a self-supervised algorithm for learning video representations that can be used by downstream video tasks. In terms of the binding problem, C-SWMs is a representative of the object slots approaches.

In this work, we hope to tackle the binding problem in the CLEVRER [Yi et al. 2019] dataset. It is composed of short videos of simple 3D objects colliding and causal reasoning natural language questions about the videos. CLEVRER is an interesting option as it is simpler than natural images datasets requiring less processing and storage, and it requires to effectively solve all the three parts of the binding problem.

1.1 General Goal

The general goal of this work is to propose a neural network algorithm to tackle the binding problem in computer vision video tasks. As a result, we hope to improve combinatorial generalization in neural networks, potentially improving learning efficiency and accuracy in many video tasks.

1.2 Specific Goals

In order to evaluate the effectiveness of the algorithm, we will use performance on the CLEVRER dataset as the main metric. We aim to develop a neural network algorithm that tackles all the three parts of the binding problem for the CLEVRER dataset.

More specifically, we aim to follow up on the paper [Ding et al. 2020], which is a fully connectionist approach to solving the binding problem, inline with [Greff, Steenkiste e Schmidhuber 2020]. In general lines, the approach consists of three major components: self-attention for temporal information integration (Transformer [Vaswani et al. 2017]), object-based discretization of the input frames and self-supervised learning inspired in the BERT model [Devlin et al. 2018]. The algorithm will be explained in more detail in the Theoretical Background chapter, and will be referred to as the base algorithm.

The original goals of the project were to re-implement the base algorithm and to modify it using the following guidelines:

- The base algorithm uses MONet [Burgess et al. 2019] on single frames to extract the objects representations (Segregation). However, this limits the frame rate that can be used, does not use temporal or motion information for object extraction and does not leverage the specialized neural networks for video data. Therefore, to solve these problems, we aim to develop a segregation module (object representation extractor) that uses a short sequence of frames instead of a single frame. A candidate network is the SlowFast [Feichtenhofer et al. 2019], which handles more frames with little overhead in comparison to a single frame.

- Another goal is to experiment with different contrastive pretext tasks for language models, such as the one used by CONPONO [Iter et al. 2020]. Contrastive learning has shown great results in computer vision and may benefit the self-supervised part, which is deemed essential for segregation.

- Lastly, we aim to modify the algorithm to be able to handle a continuous stream of input, as this may be necessary to apply it to other scenarios, such as reinforcement learning or longer video tasks.

However, due to problems in re-implementing the base algorithm, we had to change the goals. As a result, the project became focused on implementing and exploring neural network models for CLEVRER.

2 Theoretical Background

2.1 Fundamental Concepts

A survey on self-supervised visual feature learning [Jing e Tian 2020] defines important terms, from which we take some definitions that are fundamental to our work.

- Label: A Deep Learning algorithm learns a function that maps from an input to an output. A label is a correct output for a certain data point in a dataset.
- Human-annotated label: A label that was annotated by a human.
- Supervised Learning: The setting in which learning methods use only human-annotated labels to learn. Here we will not differentiate between supervised and weakly-supervised learning.
- Unsupervised Learning: The setting in which learning methods do not use human-annotated labels.
- Self-supervised Learning: A subset of unsupervised learning, in which learning methods use datasets automatically generated from the original data.
- Pretext Task: Tasks that are designed to train self-supervised methods. They are used to automatically generate a dataset. An example is the task of given a grey scale image, predict the original colorized image. Note that pairs of input and output for this task can be automatically generated from a dataset of colorized images.
- Downstream Task: Tasks that are used to evaluate the representations learned through self-supervised learning. Usually, the downstream tasks are the ultimate goal the algorithm. An example is to train an algorithm to predict the next frame in a video (pretext task) and by transfer learning apply the learned features to image classification (downstream task).
- Representation Learning: According to [Bengio, Courville e Vincent 2013], Representation Learning is "learning representations of the data that make it easier to extract useful information when building classifiers or other predictors.". It is a powerful concept in Deep Learning, in fact, one can think of each layer in a neural network as learning a representation of its input. In our work it is a central concept as it allows to transfer knowledge learned from pretext tasks to downstream tasks. It is an important research area and there are many open questions.

- Transfer Learning: It aims to improve performance on a task A on a dataset D_A by using knowledge learned from a task B on a dataset D_B, where $D_A \neq D_B$ or $A \neq B$ [Tan et al. 2018]. One example is to train a neural network for image classification on a large dataset and then, using the learned parameters, train it on a small dataset for bird classification.

2.2 Self-Supervised Learning

Supervised learning using Neural Networks has achieved great success in many Computer Vision tasks. Most of this successes required large datasets with human-annotated labels, which are expensive to gather. As a result, generating annotated examples is a bottleneck that hinders supervised learning improvements, especially in applications that only have small annotated datasets. In this scenario, unsupervised learning presents a promising path for machine learning algorithms to achieve better results on many computer vision tasks, that would require extreme human effort and cost in the supervised learning scenario.

Among several unsupervised learning approaches, Self-Supervised Learning has been widely researched and is now being used to achieve great results in computer vision. The current state-of-the-art of self-supervised learning is dominated by contrastive approaches, which surpassed various types of pretext tasks, including generative ones. The current state-of-the-art, SimCLR [Chen et al. 2020], has achieved a top 1 accuracy on ImageNet [Deng et al. 2009] close to the supervised state-of-the-art in 2016, but using only 10% of the training labels.

In the context of video representation learning, self-supervised methods are being widely researched. The paper "Spatiotemporal contrastive video representation learning"[Qian et al. 2020] proposed the current state-of-the-art algorithm and shows a visualization of the historical results on the standard datasets used, Kinetics [Carreira e Zisserman 2017], UCF101 [Soomro, Zamir e Shah 2012] and HMDB51 [Kuehne et al. 2011].

The most popular approach to self-supervised learning consists in pretraining the model on a pretext task and then applying a transfer learning method to train in the downstream task. The two most common methods are fine-tuning the entire model on the downstream task [Chen et al. 2020] or by using a linear evaluation protocol [Zhang, Isola e Efros 2016] which consists in training, on a downstream task, a linear model on top of the self-supervised learned model with frozen weights. In other words, it trains a linear model that receives as input the representations outputted by the self-supervised model.

2.2.1 Language Models

Self-supervision has been widely explored in the Natural Language Processing (NLP) community, by using clever prediction and completion pretext tasks. Two of the main representatives are the language models BERT [Devlin et al. 2018] and GPT [Radford et al. 2019].

Compared to other models, such as GPT, one of the main advantages of BERT is to use bidirectional, which improves sentence level tasks performance and facilitates downstream task performance that will use it. As a result, the pre-trained architecture is very similar to the final downstream architecture. BERT uses a masked language model in which some tokens of the inputs are randomly masked and the objective is to predict them based on their context. It also uses next sentence prediction. More detailed, the pretext tasks used by BERT are:

- Masked Language Model: Randomly masks some tokens and predicts them. Differently from contrastive query and key pairs, it uses the entire context to predict the missing tokens. Sometimes it replaces the missing token with a random token and other times with a special mask token.
- Next Sentence Prediction: Some tasks require understanding the relationship between sentences, which is not captured by language modelling. Then it performs a binary next sentence prediction that predicts with a sentence is or is not the next one.

These language models have been use for video data [Sun et al. 2019], [Girdhar et al. 2019], and have even achieved state-of-the-art performance on standard datasets [Kalfaoglu, Kalkan e Alatan

There are also several approaches that integrate language modelling with contrastive learning, for example CONPONO [Iter et al. 2020]. This is intersting for videos as in the visual domain we cannot use a softmax function to predict a word token, as there usually are no discrete fixed tokens.

2.3 Video Architectures

In order to develop efficient and accurate algorithms for video tasks, it can be important to understand the current state-of-the-art in the most basic video tasks, specially action recognition. In a recent survey [Chen et al. 2020], it was shown that much progress has been made in making more efficient models but not as much in making them more accurate. Many architectures have been developed [Carreira e Zisserman 2017], for example 3D convolutions, decomposed 3D convolutions, Two Streams and recurrent models, but there is no clear winner. One of the most successful approaches is the SlowFast Network [Feichtenhofer et al. 2019].

2.4 CLEVRER

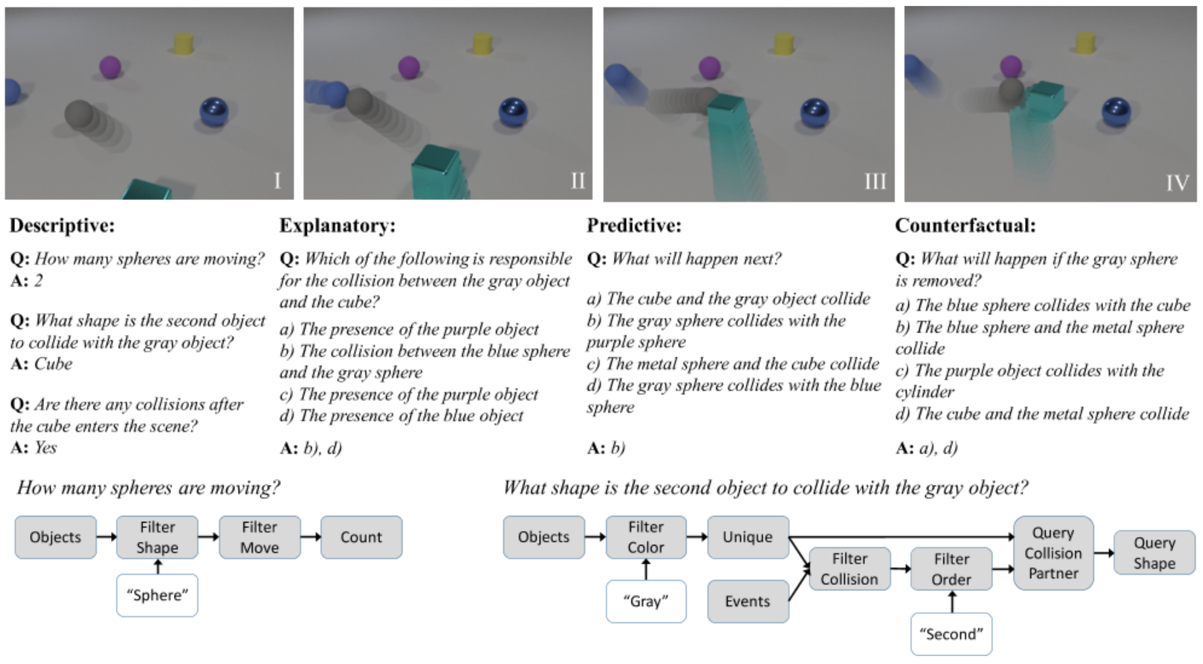
CoLision Events for Video REpresentation and Reasoning (CLEVRER) is a visual reasoning dataset, focusing on temporal and causal reasoning. It is similar to the CLEVR

[Johnson et al. 2017] dataset, but it uses videos of simple 3D shapes colliding, instead of static images.

The authors designed the dataset in a manner that the proposed tasks focused on reasoning in the temporal and causal domains, while maintaining simplicity in the visual domain. As a result, the dataset uses simple 3D objects that can be of one of three shapes (sphere, cylinder or cube), eight colors (gray, red, blue, green, brown, cyan, purple or yellow) and two materials (rubber or metal), composing a simple visual domain.

The dataset is composed of 5 second videos of different objects moving and possibly colliding, generated by a physics engine, and the task is to answer a question about the video. The questions are posed in natural language (English) and are divided into 4 categories: descriptive, explanatory, predictive and counterfactual.

The dataset is composed by 219,918 descriptive questions, 33,811 explanatory questions, 14,298 predictive questions and 37,253 counterfactual questions. All the descriptive questions can be answered by a single word, which makes it possible to model the problem as a multi-class classification problem over the possible answers. All the other types of questions are answered in a multiple choice true or false fashion, in which there is a maximum of 4 choices per question and each choice is classified as true or false. For each video there are several questions. An example can be seen in figure 1.



Descriptive:

Q: *How many spheres are moving?*
A: *2*

Q: *What shape is the second object to collide with the gray object?*
A: *Cube*

Q: *Are there any collisions after the cube enters the scene?*
A: *Yes*

Explanatory:

Q: *Which of the following is responsible for the collision between the gray object and the cube?*
a) The presence of the purple object
b) The collision between the blue sphere and the gray sphere
c) The presence of the purple object
d) The presence of the blue object
A: *b), d)*

Predictive:

Q: *What will happen next?*
a) The cube and the gray object collide
b) The gray sphere collides with the purple sphere
c) The metal sphere and the cube collide
d) The gray sphere collides with the blue sphere
A: *b)*

Counterfactual:

Q: *What will happen if the gray sphere is removed?*
a) The blue sphere collides with the cube
b) The blue sphere and the metal sphere collide
c) The purple object collides with the cylinder
d) The cube and the metal sphere collide
A: *a), d)*

Figura 1 – Visualization of an example of the CLEVRER dataset.

2.5 Base Algorithm

The paper "Object-based attention for spatio-temporal reasoning: Outperforming neuro-symbolic models with flexible distributed architectures"[Ding et al. 2020], is a strong approach for the CLEVRER dataset and will be used as the base approach for the algorithm developed in this work.

As stated in the paper title, the developed algorithm beat neuro-symbolic approaches on CLEVRER, which were argued to be better than a fully neural approach for reasoning tasks. The paper highlights three aspects of the algorithm that complement each other: self-attention for temporal information integration, object-based discretization of the input frames and self-supervised learning.

In Natural Language Processing (NLP) the self-attention mechanism has been shown to be extremely powerful for dealing with sequences of discrete entities. Videos are inherently sequential, but do not have a direct meaningful discretization. The paper hypothesized that dividing a frame into a set of objects would generate a soft-discretization that would enable the application of self-attention mechanisms. As a result, each input frame is passed through MONet [Burgess et al. 2019], which outputs a set of vectors, each representing an object, generating a sequence of object representations that is fed to a Transformer.

In the CLEVRER dataset the input videos are paired with a natural language question. To deal with this the proposed algorithm embeds each question word and includes in the transformer input sequence, with a one-hot vector indicating if the input is a word or an object. If the question is a multiple-choice, it also feeds the transformer with the choices embeddings. It applies positional encoding. The output depends on the task.

The self-supervised learning strategy used is inspired by the loss used in BERT, which is made better by the use of object representations. The BERT loss is based on masking words, whereas this paper's loss is based on masking objects following different strategies. The self-supervised loss function is added to the task loss function and weighted by a constant, which generates the full loss function used. As a result, the self-supervised learning happens simultaneously with the supervised learning.

2.6 MAC Network

The Memory, Attention and Composition (MAC) network [Hudson e Manning 2018] was proposed as a compositional attention neural network for machine reasoning and applied successfully on the CLEVR dataset, in which it was shown to be computational and data efficient. It is modularized and uses attention in a way that makes the model more interpretable than a regular black box neural network.

It is a recurrent network divided into three main modules: Control Unit, Read Unit and Write Unit. In the CLEVR dataset, The Control Unit process the question at each recurrent reasoning step, the Read Unit extracts information from a knowledge base, which is composed of features extracted from the input image, guided by the control and memory states, and the Write Unit computes the result of the current reasoning step and stores it in the memory state.

It achieved state of the art results on CLEVR and due to it's modularity and flexibility can be adapted to many different tasks. It uses abstractions such as control, memory and knowledge base, giving inspiration of how a reasoning algorithm can be built using neural network blocks.

3 Methodology

The initial goal of the project was to re-implement the base algorithm using the paper description and then to make modifications in order to improve it. However, I failed to re-implement and had to rethink the goals and the approaches I used. As a result, the project changed over time and passed through 4 distinct phases. The phases were:

1. Attempts to implement the base algorithm.
2. Search for alternatives.
3. Implementation of baselines.
4. MAC Network.

This report will follow the structure of phases and will describe the most important parts of the project. The code was implemented in Python, mainly using the Pytorch library, and is publicly available in a github repository [1].

In order to choose the models we took inspiration in the paper [Yi et al. 2019] that proposed the CLEVRER dataset and offered several baselines. And in order to evaluate the models we chose to only use test and validation performance, as developing a good algorithm for the CLEVRER dataset is the main goal of the work.

[1] `https://github.com/gabrielsluz/SlowFast`

4 First Phase

4.1 Phase Goals

The goal of the first phase was to implement from scratch the base algorithm as described in the paper [Ding et al. 2020] and reproduce the results. The reported results from the paper are shown in table 1

Tabela 1 – Reported results from the base algorithm for each question type. The results were repeated 5 times.

Question type	Descriptive	Explanatory	Predictive	Counterfactual
Accuracy (%)	94.0 ± 0.4	96.0 ± 0.6	87.5 ± 3.0	75.6 ± 3.8

4.2 Development

As a first step I decided to implement a simpler version of the base algorithm using only MONet, question embeddings and the Transformer. The original paper reported an accuracy of 91% on descriptive questions using this version.

The authors did not release the source code used and did not give all details of the Transformer network used.They did not specify if they used the entire Transformer or only the encoder part and they did not specify some internal parameters, which must have been different from the original Transformer [Vaswani et al. 2017] as the parameters they informed were incompatible with the original. Given other works and the use of the CLS token as in BERT [Devlin et al. 2018], I assumed that the paper only uses the encoder part as BERT does. To implement this part of the model I used the standard Pytorch Transformer Encoder.

MONet is used in the base algorithm to extract 8 objects slots (real valued vectors) for each one of the 25 uniformly sampled frames from the video. The object extractor was pre-trained following the paper [Burgess et al. 2019], with 1,000,000 iterations with a batch size of 64. I used an implementation of MONet publicly available in github [1].

I could not reproduce the reported accuracy and the maximum validation accuracy achieved was of only 11%, indicating that the implementation was not correct. In order to find and correct errors we tried to list where the errors could be, coming up with the following candidates:

- Model architecture mistake, for example the wrong number of layers or dimensions, using the output incorrectly, model too complex or too simple.

[1] `https://github.com/stelzner/monet`

- Bad neural network practices, for example not using proper weight initialization, bad learning rate policy and warm-up, activation functions, regularization.
- Efficiency, the model needs more epochs to learn, but time and memory resources available are not enough.
- Pytorch details, for example setting the gradients to zero, tensor shape transformations, data loader issues.
- Errors in data processing, for example wrong color normalization, frame sampling rate, crop size.
- Dataset errors, for example using different vocabularies for training and testing.
- Bad hyperparameters, for example too high weight decay, learning rate, optimizer, batch size, dropout.

One aggravating problem is that the model is very time consuming to train, as it uses the Transformer and MONet every iteration, making every hyperparameter small search very expensive.

At first I tried to use good practices for neural networks, starting with trying different weight initialization methods, different learning rate policies and different combinations of weight decay and learning rate. However, the results did not improve.

I did not use a method for testing and debugging the code, I relied on only printing and manually checking the results. I did this because the task of debugging a neural network with large inputs and outputs, that was constantly changing, due to testing different hyperparameters, seemed too hard to automatize using conventional unit tests. However, in hindsight this was a bad decision.

Another approach used was to try to implement simple baselines for the CLEVRER dataset reported in the paper [Yi et al. 2019]. The reasoning behind is that the base algorithm is more complex and more likely to have errors. I tried to implement three baselines:

The first, CNN + MLP, simply passes a CNN through all frames computing a sequence of latent encodings, which are then aggregated by summation generating a latent video encoding. The question is passed through an embedding layer generating a sequence of embeddings, which are then summed into a latent question encoding. The latent encoding for the video and question are concatenated and fed into a Multilayer Perceptron (MLP) that outputs the answer.

The second, CNN + LSTM, makes better usage of the temporal dimension by using a LSTM [Hochreiter e Schmidhuber 1997] to aggregate the latent encodings generating an output which is then passed to a MLP that generates the final answer. I tried to use different types of recurrent neural networks, such as GRU [Cho et al. 2014] and a

Bidirectional LSTM [Schuster e Paliwal 1997]. I also tried to use a single recurrent network to process simultaneously words and frames, and separated networks.

The last one, LSTM, only uses the text of the question, ignoring the video, and is a very weak and simple baseline.

However, I could not successfully train any of these networks, never surpassing 11% validation accuracy.

4.2.1 Mistakes

The first phase was full of mistakes both in the code and in the approaches of trying to fix it.

A big mistake was to index the dataset by video and randomly sample two questions, a descriptive and a multiple choice. As result each sample item was composed by a video and two questions. This made the implementation to support the two types of questions easier, as each sampled batch would have a well separated set of descriptive and multiple choice questions, which are processed differently by the model. However, there are only 10,000 training videos and over 200,000 questions, meaning that data, an essential part for machine learning, was not being efficiently used. 20 epochs in this easy to implement method would not be enough to pass through the entire dataset, as the questions were randomly sampled and could be repeated.

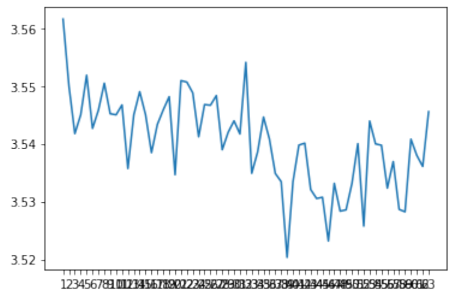
Another error was computing wrong multiple choice questions statistics, which were the metric used to evaluate the model. Before discovering this mistake, the model did not seem to overfit to the training set, even when we purposely tried to.

A proper way of evaluating the MONet model was not used. It was done by simply visually inspecting the generated frame reconstruction and decreasing the training loss. MONet was also trained without input normalization, which may have impacted performance. And as a final issue, MONet was fine tuned with the Transformer for the CLEVRER task, which is much more expensive than using MONet as a feature extractor to process the dataset before training.

Another problem was misinterpreting the training and validation curves as overfitting or underfitting scenarios. Figure 2 shows a supposedly underfitting scenario, in which both the training and validation losses never reach bellow 3.5. Whereas in figure 3 the training loss decreases to about 2.6 while validation loss increases, similar to a overfitting case. This mistake led me to train the model several times searching for the right combination of learning rate and weight decay (regularization) to find the point between the overfitting and underftitting scenarios. However, I could not find this point despite the large number of combinations tried. The problem, only discovered in phase three, was a bug in the code.

Another false judgement was that the models were presenting different behaviours in training and validation, but the problem found was that it was due to dropout or batch normalization layers, whose correct behaviour was different in validation and training.

(a) Training loss.



(b) Validation loss.

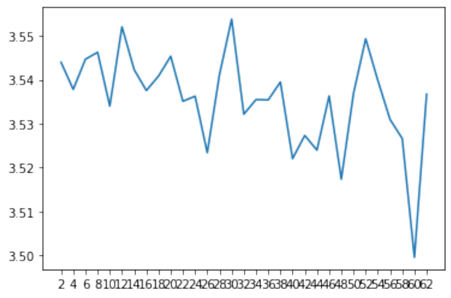
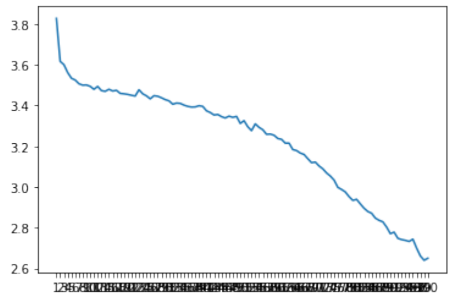


Figura 2 – Supposed underfitting scenario

(a) Training loss.



(b) Validation loss.

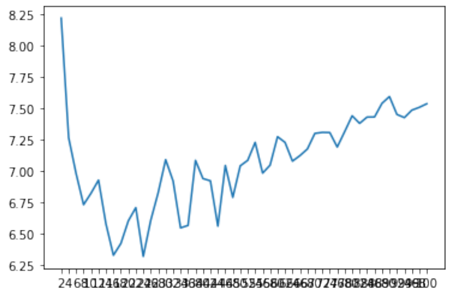


Figura 3 – Supposed overfitting scenario

4.3 Conclusion

The first phase failed to implement the base algorithm and simple baselines. I tried several approaches to improve the results, heavily focusing on modifying the algorithm, instead of trying to find errors in the software. After successful failed attempts I decided to look for alternative models for CLEVRER, moving to the second phase.

However, the first phase was not wasted. Firstly, it served as the base implementation for the third and fourth phases. Furthermore, important errors were corrected as mentioned above and good neural networks practices were added to the code.

5 Second Phase

5.1 Phase Goals

The goal of this phase was to find an alternative algorithm for CLEVRER to serve as a base for modifications.

5.2 Development

The leaderboard for CLEVRER is available in the website *Eval.ai*[1]. The current published state of the art is the base algorithm followed by the neuro-sybolic model Dynamic Concept Learner (DCL) [Chen et al. 2021].

The official implementation for DCL is available in Github [2] and with some modifications in the code and dataset can be used to train the model from scratch or use the pre-trained models to repeat the results. However, the training process was very slow and demanded many hours making it hard to use as a base of modifications. Therefore, this option was discarded.

In this stage of project all models tried were very expensive to train. As a result, I tried to look for alternatives in the CLEVR dataset [Johnson et al. 2017], which is very similar to CLEVRER, but uses only static images instead of videos. As it only uses images it is significantly faster to train.

One of the best models for CLEVR is the MAC network [Hudson e Manning 2018], about which a brief overview has been given in the Theoretical Background chapter. It is a very interesting algorithm and can be adapted to different applications. In the CLEVRER paper [Yi et al. 2019], a simple modification of MAC, MAC(V), was used as a strong baseline and the results are shown in table 2.

Tabela 2 – Reported accuracy results per question type in CLEVRER

Model	Descriptive	Explanatory	Predictive	Counterfactual
Base	94.0	96.0	87.5	75.6
DCL	90.7	82.8	82.0	46.5
MAC(V)	85.6	12.5	16.5	13.7

I used an implementation of MAC available in Github [3] and was able to get close to 95% accuracy on CLEVR in a much shorter training time than the CLEVRER baselines. Another reason for this faster training is that the MAC implementation uses ResNet50

[1] `https://eval.ai/web/challenges/challenge-page/667/leaderboard/1813`
[2] `https://github.com/zfchenUnique/DCL-Release`
[3] `https://github.com/rosinality/mac-network-pytorch`

[He et al. 2016] as a feature extractor. It first passes the CNN through the entire dataset generating another dataset composed of the extracted features. This has the downside of not fine tuning the CNN to the task, but it has the benefit of not requiring to compute the gradients or outputs of the large ResNet50 every iteration.

5.3 Conclusion

In the second phase the MAC network was considered to be a promising model for the CLEVRER dataset, as it is flexible and has shown good results on the dataset.

Another major point learned in this phase is that using a pre-trained CNN as a feature extractor is very important for achieving better results, as it makes training for more epochs possible.

6 Third Phase

6.1 Phase Goals

The goal of the third phase was to implement baselines for the CLEVRER dataset focusing on correcting software errors and using pre-trained models as feature extractors.

6.2 Development

The first step was to implement the LSTM using only text, which is the simplest baseline. This model is much faster than the others and can be easily trained using a GPU in Google Colab, facilitating debugging and hyperparameter tuning.

In order to better test the code I used three main approaches: unit testing with asserts for dataset and metrics functions, printing with manual checking for training and model functions, and asserting the the model parameters changed after one optimizer step.

The dataset has both questions and videos. To test videos I used the sum of the tensors as an heuristic to check if two sampled videos were different and manually opened a few random videos to check if they were correct. The questions are transformed into tensors of indexes according to a vocabulary. This was the source of a huge error, which was using a different vocabulary for training and validation. As a result, it was if the validation question words were scrambled, creating a huge performance gap between the training and validation splits. This was the last big error found in the code.

After testing the software, I tried to implement the text only and the CNN + LSTM baselines. the results are shown in table 3. In the CLEVRER paper, the authors tested several baselines including the LSTM and CNN + LSTM. The reported results are shown as the target values in table 3. Interestingly, both LSTM and BERT achieved equal performance, which may be a limit for text only approaches to CLEVRER.

The CNN + LSTM consisted of a ResNet50 pre-trained on ImageNet and a Bidirectional LSTM. ResNet50 was used as a feature extractor taken from the *pool5* layer, which outputs a 2,048 dimensional array for each frame. This approach allowed the model to be trained for over 10 epochs, achieving 45% accuracy on descriptive questions before starting to overfit. It was less than the 51.8% target, but much better than the previous results.

Tabela 3 – Baselines results on descriptive questions on the validation set. The target accuracy were reported in [Yi et al. 2019]

. .

Modification	LSTM	BERT	CNN + LSTM
Accuracy (%)	34	34	45
Target accuracy (%)	34.7	-	51.8

6.3 Conclusion

Focusing on correct software errors and using pre-trained models to speed up training, the third phase was crucial for developing a more trustworthy software base implementation to use more complex models such as the base algorithm and the MAC network.

7 Fourth Phase

7.1 Phase Goals

The goal of the last phase was to apply the MAC network [Hudson e Manning 2018] to the CLEVRER dataset. An overview of the model was given in the Theoretical Background chapter.

7.2 How to adapt MAC for CLEVRER ?

The MAC network is composed of a recurrent MAC cell, which is divided into three modules: Read Unit, Control Unit and Write Unit. Even though it is a rather general model, it was developed for the CLEVR [Johnson et al. 2017] dataset and it's original form is specialized in answering natural language questions about a static image. The Control Unit acts only on the question, the Read Unit acts on the knowledge base, which is a set of features extracted from the input image, and the Write Unit acts on the memory state given the information retrieved from the Read Unit and the control state. The main difference between CLEVR and CLEVRER is that one uses images and the other videos, keeping the questions and answers part almost equal. As a result, I judged that the Control and Write Units could remain the same, whereas the Read Unit and the organization of the knowledge base should be adapted to use video data.

It is important to note that the knowledge base is a set of vectors and the Read Unit operates by computing the similarity between each vector and the internal memory and control states. Therefore, the knowledge base and the mechanism to retrieve information from it do not take into account any temporal information or interactions between its elements. This works well for static images, but for videos it may be problematic.

I used an implementation of the MAC network available on github [1] because time was very limited in Phase 4.

7.3 Convolutional Approaches

In the paper that presented the CLEVRER dataset the authors used the MAC network as a baseline called MAC(V). The reported results are shown in table 4. Note that it is much worse than the base algorithm, but much better than the simple baselines. The paper describes the modifications done to MAC as using a pre-trained ResNet50 to extract 14×14 features maps and applying a temporal attention unit across the video frames

[1] `https://github.com/tohinz/pytorch-mac-network`

to generate a representation for the video. This attention unit seems to be too vague to re-implement exactly. What I understood is that this attention unit returns a 14×14 (height and width) with only one temporal dimension, effectively compressing the 25 frames into a single feature map. However, I cannot be sure about it.

Tabela 4 – Reported results from MAC(V) for each question type.

Question type	Descriptive	Explanatory	Predictive	Counterfactual
Accuracy (%)	85.6	12.5	16.5	13.7

I call MAC (V) a convolutional approach because it extracts features using a convolutional neural network and does not modify the Read Unit. I tried three convolutional approaches to modify the knowledge base.

The simplest one uses a pre-trained ResNet50 to compute the 2,048 dimensional output of the pool5 layer for each frame. As we follow the common approach of uniformly sampling 25 frames from the video, than we obtain a 25×2048 representation of each video. This matrix is then projected onto the number of dimensions used in MAC (512) generating a knowledge base of dimensions 25×512. Following the training procedures described in the MAC paper, I trained the network until validation loss convergence, achieving 55% accuracy in descriptive questions. This method is referred to as Res50 pool5.

The second approach is to replicate what is done in the original MAC, which uses the $1024 \times 14 \times 14$ feature maps from a ResNet101 and further process them passing through a two layer convolutional neural network (CNN), yielding a knowledge base of 196 vectors. However, if we apply this to 25 frames it yields a knowledge base of 4,900 vectors, being extremely costly to process using the Read Unit. Therefore, I modified the CNN to reduce each feature map dimension to 3×3 yielding 225 vectors. This approach is still much slower than the simplest one (225 against 25) and did not improve the accuracy, achieving only 53%. This method is referred to as Res101 conv4.

The last modification consisted in using the successful SlowFast [Feichtenhofer et al. 2019] network pre-trained on the Kinetics dataset, in hope that a state of the art architecture specialized in videos would be very helpful. However, we could not use the final pool layer output as it compresses the video into a single vector, generating a single element knowledge base. Then I chose to use a 16×16 feature map for the frames. The SlowFast network outputs, in an intermediary layer, feature maps for each frame, yielding $32 \times 16 \times 16$ (32 frames fed into the fast pathway) and $4 \times 16 \times 16$ (4 frames fed into the slow pathway), suffering with the same problem of inefficiency as ResNet101. I used the same approach of modifying the input layer CNN to reduce the dimensions. This approach achieved only 49% accuracy. This method is referred to as Slowfast.

The results for the convolutional modifications are shown in table 5. Note that the results are much worse than the 85.6% reported by MAC(V) and I could not re-implement MAC(V) as I did not understand the description given.

Tabela 5 – Convolutional modifications to MAC results on descriptive questions on the validation set.

Modification	Res50 pool5	Res101 conv4	Slowfast
Accuracy (%)	55	53	49

Another big problem in these approaches is that the method I used to store and retrieve the extracted features was extremely slow for training using the $25 \times 1024 \times 14 \times 14$ used in Res101 conv4. The dataset generated by pre-processing was about 189 GB and a large part of training was consumed by loading the examples from the storage. I opted to not try to make the method more efficient, because I wanted to expend more time working on using MONet with MAC, which seemed more appropriate given my POC 1 on object based methods and the base algorithm that uses MONet. As a result, I did not explore several other forms of aggregating the convolutional features from all frames in a video, a method that can be powerful as shown by MAC(V).

7.4 MONet

Returning to the formulation of the Binding Problem, which is divided into Segmentation, Representation and Composition, we can see a clear relation between MAC and MONet. The MAC network tries to solve the problem of reasoning given a task and a knowledge base, which configures the Composition part, whereas MONet segment, extracts and represents objects, configuring Segmentation and Representation. This same relation is used by the the base algorithm, which combines MONet with a Transformer. Therefore, these two models are complementary and it is natural to try to combine them.

7.4.1 Pre-training

Given the importance of pre-trained models for speed and accuracy, I chose another implementation of MONet [2], which contains a pre-trained model on CLEVR. Even though CLEVR and CLEVRER are similar, there may be some differences that impact MONet performance, then it is necessary to check the quality of the pre-training.

MONet learns to extract objects by decomposing an image into parts, given by attention masks, reconstructing each part and assembling them to reconstruct the original image. One way of checking the quality of the learned representations is simply to look at the reconstructed image. MONet uses images resized to a low resolution of 64×64.

Figure 4 shows the difference between the reconstructed frames by the model trained only on CLEVR and the model trained for 7 epochs (1 million iterations) on CLEVRER. This frame was chosen because it is challenging for these methods, due to multiple objects

[2] `https://github.com/baudm/MONet-pytorch`

of different colors and the contract between the green and yellow objects. Simpler frames, that contain only two or less separated objects are reconstructed better. We can note that both reconstruction lose shape information, being hard to differentiate between a cube and a cylinder. However, the model trained on CLEVRER is much better on reproducing the colors. The dataset has questions about color, sahpe and material, making it very important that these properties are maintained. These pictures show that the object representations learned by MONet are not perfect, but contain information enough to reconstruct color.

(a) Original frame. (b) Trained on CLEVR. (c) CLEVRER.

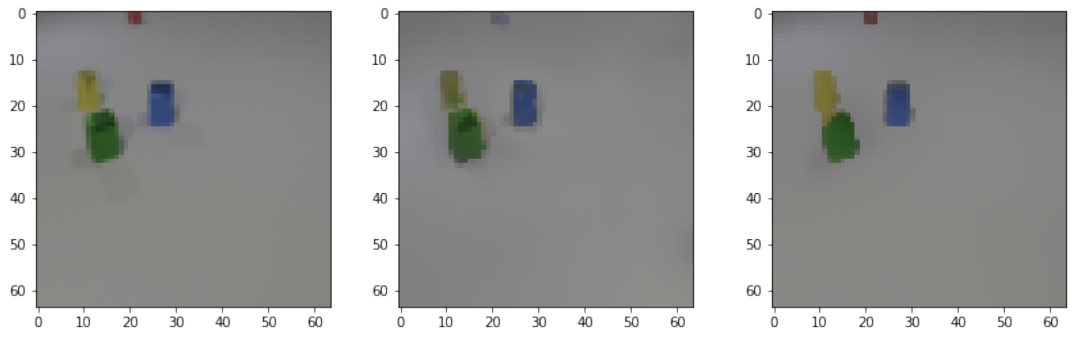


Figura 4 – Comparison of the quality of reconstructed images.

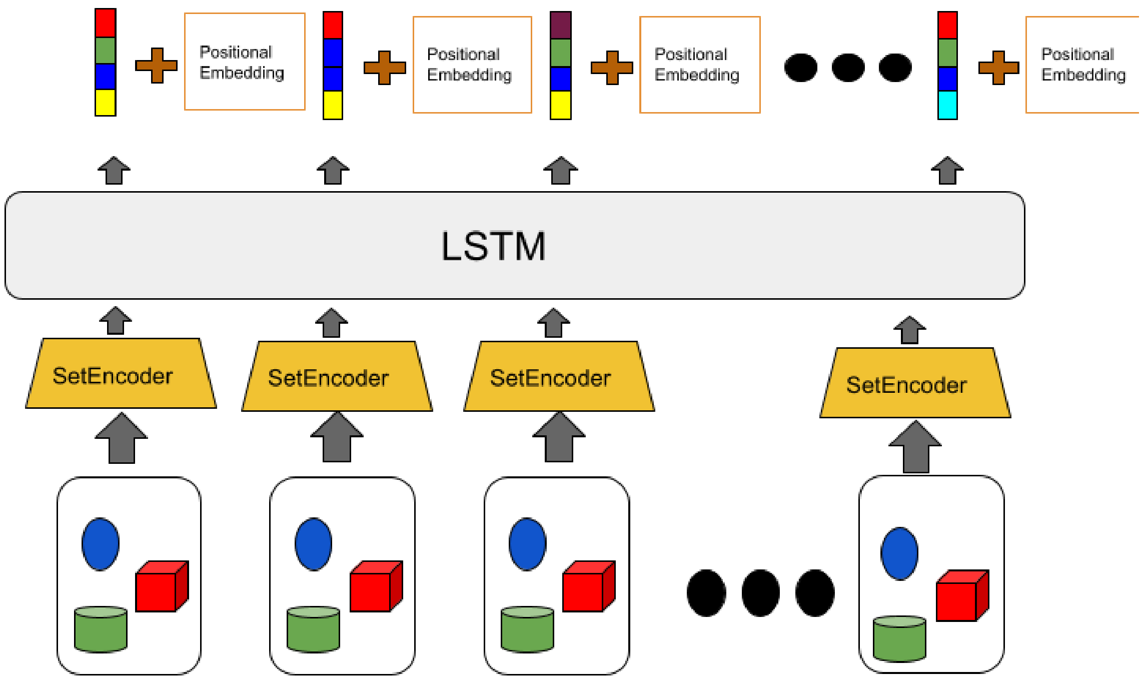
7.4.2 Modifications

A straightforward way to use MONet with MAC is to simply generate the objects slots for each frame and assemble a sequence of arrays representing the entire video. We follow the base algorithm and generate 8 slots per frame, that are 16 dimensional arrays. Therefore, a video is represented by a sequence of 25×8 arrays. This is the approach used by the base algorithm, but the sequence is passed through a Transformer with positional encoding, which is very different from the MAC network. The knowledge base is interpreted as a set of arrays, losing temporal information and to which frame an object slot belongs. The experiments confirm this problem as this approach only achieves 43% accuracy. Another problem with this approach is that the knowledge base is composed of 200 arrays, much more than the 25 used in *Res50 pool5*, making it slower.

For each frame, MONet outputs a set of object slots. As a result a video is represented as a sequence of sets. The CLEVRER dataset involves reasoning about objects and their interactions through time. In order to try to explore these two important factors, interactions between objects and time order, I tried to transform the sequence of sets into a set of vectors, one per frame, encoding the two factors.

The modification, illustrated in figure 5, is composed of three steps. Firstly it computes a single vector to represent the set of objects slots for a frame by processing the slots through a Set Encoder. It is a generic module that takes as input a set of vectors and outputs a single vector. In hope to take into accounts the interactions between objects in one frame, I implemented the Set Encoder as a small Transformer, whose self attention mechanism computes interactions between all possible pairs of objects. The generated frame representations are then passed through a Bidirectional LSTM in order to encompass the time order of events. As a last step, to the sequence of vectors outputted by the LSTM it is applied a Positional Embedding, following [Vaswani et al. 2017], with the goal to encode temporal information into the knowledge base.

Figura 5 – Model to process the MONet output into a knowledge base for MAC.



The Set Encoder seemed like a very promising modification due to the fore mentioned motivations and due to the fact that the generated knowledge base is of dimension $F \times D$, being F the number of frames (25) and D a hyper parameter that defines the size of the arrays. Making the MAC part as efficient as the *Res50 pool5* modification. However, the result, after using the standard MAC training and a small hyper parameter search, was 48% accuracy on descriptive questions.

Assuming that the generic structure of the modification is correct, which is apply the Set Encoder, pass through an LSTM and then apply positional embedding, then I believe that the two main reasons that this model failed were the chosen Set Encoder and the MONet representations. A simpler or different model for the Set Encoder should be tested and there are specialized models for dealing with sets. The issues concerning the MONet representation will be considered further on the text.

I also tried to use the Set Transformer [Lee et al. 2019], a specialized architecture for dealing with sets, as the Set Encoder. The module was used off-the-shelf without hyperparameter tuning. Both the encoder and decoder parts were used to receive a set with 8 elements and output a single array representing it. It achieved only 43.6% accuracy on descriptive questions. However, as no hyperparameter tuning was used, this result is not trustworthy.

Another modification using MONet was to maintain the knowledge base as a set of all object slots in the video but to change how the Read Unit process it. The modification was to apply the Read Unit separately at each frame, which is represented as a set of object slots, and to use an LSTM to aggregate the retrieved information. The knowledge base is of size $F \times S \times D$, being S the number of slots per frame, being less efficient than *Res50 pool5*. This modification did not generate good results.

7.4.3 Issues with MONet

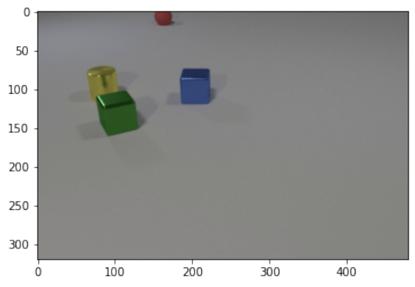
MONet was successfully used by the base algorithm to achieve state of the art results on CLEVRER, but in this project I could not use it effectively. A possible issue with MONet is the low dimension representation (arrays of 16 dimensions), but this was not a problem with the base algorithm and the frame reconstructions are of medium quality.

Another issue with MONet is that it resizes the frames to a low resolution of 64×64, which results in a loss of information, especially shape information. The figures 6 and 7 show the differences between different values of width and height. We can see that 64×64 is much worse than the others. Therefore, a possible solution might be to increase the resolution of the images used. However, it is important to note that the base algorithm also resizes the frames to this low resolution, and achieves much better results. It is important to note that the convolutional approaches use 224×224, as they use ImageNet pre-trained CNNs.

Given this issues that do not affect the base algorithm, on conclusion might be that the pre-training of MONet was not done properly. Trying to address this hypothesis, I trained MONet for another one million iterations, which marginally decreased the loss in training, but did not bring any improvement to accuracy in the CLEVRER task.

Therefore, I conclude that there might be two major problems. First, the pre-training of MONet was not done properly and a better evaluation method than visually inspecting the frame reconstruction and segmentation masks may be necessary. Secondly, the mechanisms implemented to take advantage of the MONet representations were not good enough. MAC network offers great flexibility to adapting to new tasks, but it is not trivial to make the modifications. The adaptations proposed in this section were very simple and did not change the Read Unit in a clever way, which is an area in MAC that deserves further experimentation.

(a) Original resolution.



(b) 224 x 224.

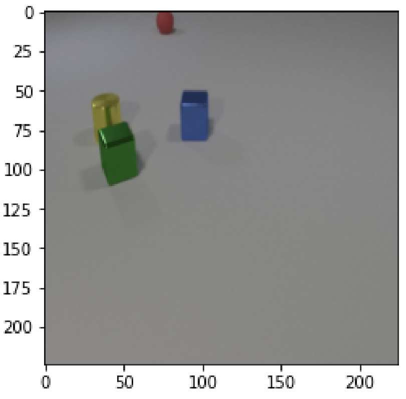
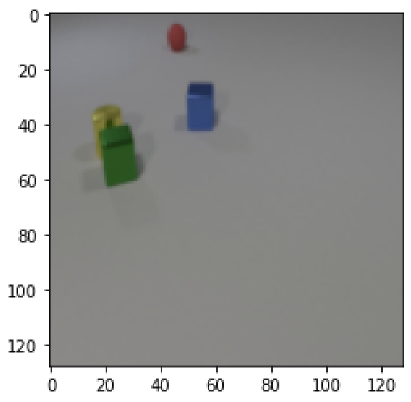


Figura 6 – Comparison of the quality of resized images.

(a) 128 x 128.



(b) 64 x 64.

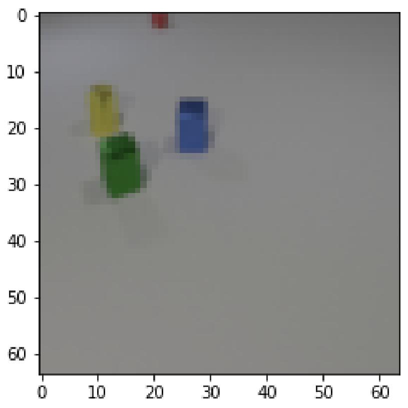


Figura 7 – Comparison of the quality of resized images.

7.5 Temporal Convolutional Approaches

The problems mentioned in the MONet section about how MAC does not use temporal information are also an issue with the convolutional approaches. Therefore, I implemented a simple modification to the *Res50 pool5* model in order to encode temporal information. The *Res50* features for the sequence of frames are passed through a LSTM yielding time encoded representations, to which positional embedding is added. This modification improved the accuracy from 55% to 60%, showing that taking into account temporal information is beneficial. The resulting model is called *Res50 pool5 + temp*.

Another modification was to simply concatenate the *ResNet50 pool5* features with the flattened object slots for each frame, and applying the model from *Res50 pool5 + temporal*, yielding *Res50 pool5 + mon + temp*. This modification achieved an accuracy of 62%.

7.6 MAC(V)

As a last attempt, I tried to re-implement the MAC(V) model mentioned before. The authors from [Yi et al. 2019] described it as extracting $1024 \times 14 \times 14$ feature maps with a ResNet50 and applies a temporal attention unit to the sequence of frames. I interpreted this attention unit in a simple way, which is using a trainable array of size equal to the number of frames, and during the forward pass this array is passed through a softmax function generating a probability distribution over the frames, which is the attention weights. Each frame feature map is than multiplied by the corresponding attention weight and are summed to form a single attention map of $1024 \times 14 \times 14$ dimensions, representing the entire video. Putting it in a simple format, the attention unit learns to weight each frame according to it's importance to answering the questions.

A problem with this implementation is the extremely slow method to load the large features from the storage, making training slow. Training for 11 epochs took around 20 hours, and achieved 76.8% accuracy on descriptive questions. This method is expected to achieve 85.6% with more epochs.

A difference between this re-implementation and the original MAC(V) will be explored in the next section.

7.7 Final Results

We have restricted ourselves to discussing only descriptive questions performance, as they are the main part of CLEVRER and it is easier to evaluate using only one metric. However, the other questions are extremely important and must be accounted for. The other questions are multiple choice true or false questions, in the sense that for one question there are at most four binary classification problems, each one presents an option that must be classified as true or false. A multiple choice question is considered correct only if all its options are answered correctly.

In this project I chose to transform the multiple choice into descriptive questions by separating each option into different questions that if the option is true then the answer is "yes"and if it is false then the answer is "no". In this implementation, to answer one multiple choice question at test time it is required to answer four transformed descriptive questions. The program does not differentiate between transformed descriptive and descriptive questions, as a result a true or false answer could be answered by the model with a non sense answers such as "metal"or any one of the 21 possible answers, instead of "yes"or "no".

Table 6 shows the final results for the best MAC modifications for CLEVRER. We chose to use per option accuracy for the multiple choice questions.

We can divide the methods in table 6 in two categories: spatial focused and temporal

Tabela 6 – Results for different MAC modifications for CLEVRER. Per option accuracy.

Modification	Descriptive	Explanatory	Predictive	Counterfactual
MONet SetEncoder	48.27	62.34	55.44	53.05
Res101 conv4	53.47	66.06	61.18	57.85
Res50 pool5	55.54	63.76	63.76	56.15
Res50 pool5 + temp	60.09	72.75	70.05	58.65
Res50 pool5 + mon + temp	62.00	71.56	70.57	58.99
MAC(V) all desc (11 epochs)	76.77	**88.95**	**77.71**	**73.03**
MAC(V)	**85.60**	59.50	51.00	54.60

focused. In the spatial focused methods, the Read Unit in MAC attends to parts of feature maps that correspond to different spatial locations, whereas in temporal focused, the Read Unit attends to features from different frames, each one from a certain point in time. The spatial focused methods are MAC(V), *MAC(V) all desc* and *Res101 conv4*. The temporal ones have the prefix *Res50*.

Res101 conv4 aggregates the frame features into a single feature map by using three dimensional convolutional operations, reducing the temporal dimension to 1 and the height and width to 3. This was done to speed up training, but yielded poor results. The best performing methods were MAC(V) and *MAC(V) all desc*, which are supposed to be identical, except by the fact that MAC(V) uses a different format for multiple choice questions and a separated prediction head, whereas the other uses the method of transforming these questions into yes or no descriptive questions. It is very important to note that *MAC(V) all desc* was only trained for 11 epochs, due to time constraints, and has not achieved full performance, and even so largely surpassed MAC(V) on the multiple choice questions. Therefore, the adaptation used is beneficial for improving multiple choice performance, and may be beneficial for all questions, but this can only be confirmed by training the model for more epochs.

The spatial focused methods performed better than the temporal ones. One reason may be that the spatial knowledge base is larger than the temporal, with 196 arrays against 25 for the temporal ones. Another important factor is that both use the same Read Unit, which was designed for spatial focused methods.

7.8 Analysis of the model answers

The CLEVRER descriptive questions have 21 possible answers, which can be divided into a few categories:

- Counting: 0, 1, 2, 3, 4, 5.
- Yes or No.
- Material: rubber or metal.

- Shape: cube, sphere, cylinder.
- Color: gray, brown, green, red, blue, purple, yellow, cyan.

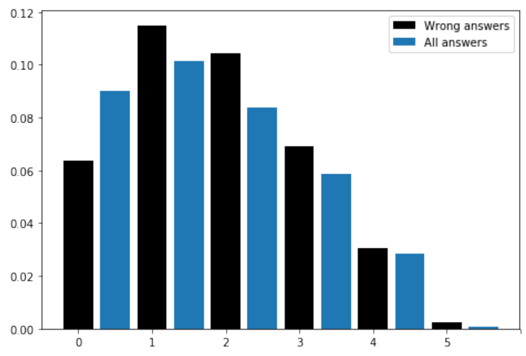
Analysing the distribution of wrong answers of the model can be useful to find areas of improvement. For example, in the MONet issues section it was argued that due to the low resolution and poor reconstructions, the object representations did not encode shape information properly.

The method to analyse the model answers was to select the questions that the model missed and plot histogram of the correct answers for these questions, and compare it to the histogram of all answers in the validation dataset. To facilitate the viewing of the labels the results will be presented separated per answer type. Only descriptive questions were used.

Only the model *Res50 pool5 + mon + temp* will be evaluated.

In figure 8, we can see that the model does slightly worse on counting questions than on the others. And in figure 9, we can see that the model is much better on yes or no questions, and by figure 10 we can see that the model is good at material questions and normal on shape questions. However, figure 11 shows that the model has a big problem with color questions. One possible interpretation of this result is that color is very important in CLEVRER, but using an ImageNet pre-trained CNN, which may have been trained using color modifications as data augmentation, might have a bad impact in answering color questions.

Figura 8 – Distribution of missed counting questions in comparison to ground truth answer distribution



One concern was that considering the multiple choice questions as descriptive questions with yes or no answers, this would bias the model towards answering yes or no more frequently. However, figure 12 shows that the model's output distribution and the true answers distributions were very similar, specially in the yes or no answers. Considering that the results found for our MAC modification was better than MAC(V) for multiple choice questions, and that it is good to answer yes or no descriptive questions without increasing the distribution of these answers, then I conclude that the decision to transform all questions into descriptive was beneficial.

Figura 9 – Distribution of missed yes or no questions in comparison to ground truth answer distribution

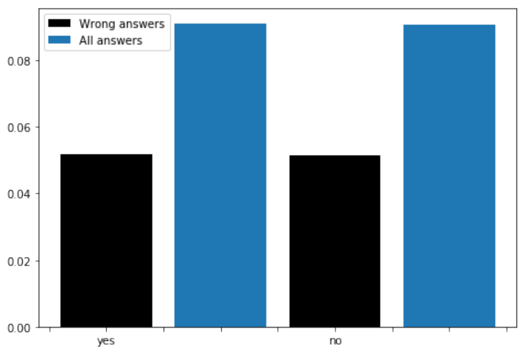


Figura 10 – Distribution of missed material or shape questions in comparison to ground truth answer distribution

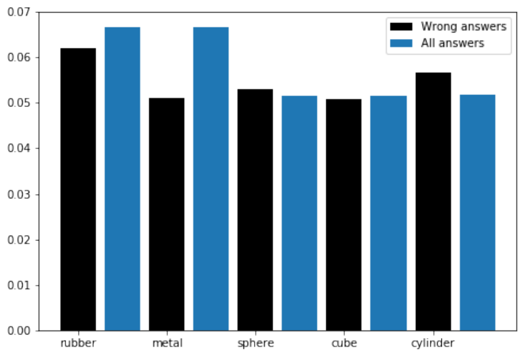
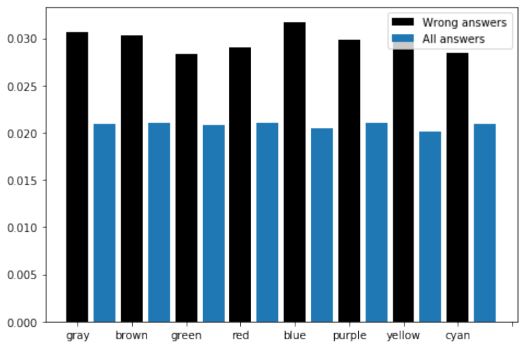


Figura 11 – Distribution of missed color questions in comparison to ground truth answer distribution

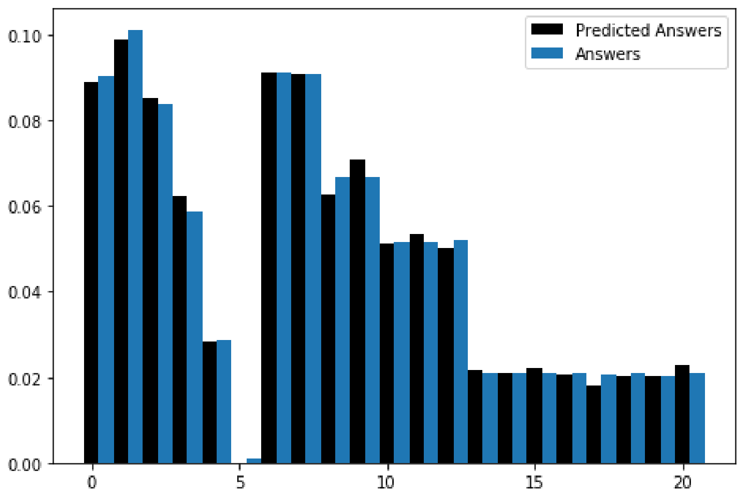


7.9 Transformer models

With a more trustworthy software implementation and MONet not being used effectively by MAC, I tried again to implement the base algorithm. This time I chose to use the Transformers HuggingFace [3] library to implement the ALBERT model [Lan et al. 2019], which is a lite version of the BERT model. However, the model only achieved 41% accuracy on descriptive questions. The problems might have been a poorly trained MONet or a

[3] `https://huggingface.co`

Figura 12 – Model predicted answers and true answers distribution on the validation set. The answers are indexed by integers from 0 to 20.



wrong implementation of the Transformer model, which could not have been replaced by ALBERT or need different hyperparameters.

8 Conclusion

The initial goal of implementing the base algorithm was not achieved. The main reason for this failure was not having an effective method to correct software errors. This resulted in wasting too much time on trying to tune hyperparameters or modifying models which could never work because of underlying errors. Other factors also impacted the development, for example the lack of experience in dealing with neural networks, the inherent difficulty of the CLEVRER dataset and the computational cost of video architectures.

The MAC network is a flexible model and I believe that there are clever designs for the Read Unit that can make it a very powerful model on different tasks, including CLEVRER. In this work and in the literature, MAC has not been effectively adapted to deal with the temporal dimension, with most adaptations leaving the Read Unit untouched and focusing on transforming the knowledge base. This work showed that using a LSTM and positional embedding to process the input can be beneficial to encode temporal information, but the improvements were minor. A more effective approach should be to fundamentally modify the Read Unit. Another important point is that most adaptations focused more on temporal or spatial dimensions, for example MAC(V) aggregates the temporal dimension into a single feature map, focusing on the spatial dimension, whereas *Res50 pool5 + temp* uses a single vector to encode each frame, focusing more on the temporal dimension. This difference in focus might explain why MAC(V) is better on descriptive questions and *Res50 pool5 + temp* is better on predictive and causal reasoning questions. However, as shown by *MAC(V) all desc* the modification to interpret multiple choice questions as descriptive yes or no questions is more likely to explain the improvement seen in multiple choice questions. An approach that can use the Read Unit to capture both spatial (attend to different patches of an image) and the temporal (attend to different frames) seems promising.

The *Res50 pool5 + mon + temp* model displayed poor result on color questions indicating a possible path for improvement. One hypothesis is that color augmentations used in pre-training the convolutional network used may be a cause of this problem. Testing this hypothesis would require a large scale training, which was not possible in this project.

Therefore, this project explored different architectures for the CLEVRER dataset, successfully implementing baselines and modifications to the MAC network, but without reaching the state of the art.

Referências

[Bengio, Courville e Vincent 2013]BENGIO, Y.; COURVILLE, A.; VINCENT, P. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 35, n. 8, p. 1798–1828, 2013.

[Brostow et al. 2008]BROSTOW, G. J. et al. Segmentation and recognition using structure from motion point clouds. In: *ECCV (1)*. [S.l.: s.n.], 2008. p. 44–57.

[Burgess et al. 2019]BURGESS, C. P. et al. Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019.

[Caron et al. 2020]CARON, M. et al. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv preprint arXiv:2006.09882*, 2020.

[Carreira e Zisserman 2017]CARREIRA, J.; ZISSERMAN, A. Quo vadis, action recognition? a new model and the kinetics dataset. In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2017. p. 6299–6308.

[Chen et al. 2020]CHEN, C.-F. et al. Deep analysis of cnn-based spatio-temporal representations for action recognition. *arXiv preprint arXiv:2010.11757*, 2020.

[Chen et al. 2020]CHEN, M. et al. Generative pretraining from pixels. In: *Proceedings of the 37th International Conference on Machine Learning*. [S.l.: s.n.], 2020.

[Chen et al. 2020]CHEN, T. et al. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[Chen et al. 2020]CHEN, T. et al. Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*, 2020.

[Chen et al. 2021]CHEN, Z. et al. Grounding physical object and event concepts through dynamic visual reasoning. In: *International Conference on Learning Representations*. [s.n.], 2021. Disponível em: <https://openreview.net/forum?id=bhCDO$_cEGCz$>.

[Cho et al. 2014]CHO, K. et al. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[Cordts et al. 2016]CORDTS, M. et al. The cityscapes dataset for semantic urban scene understanding. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 3213–3223.

[Deng et al. 2009]DENG, J. et al. Imagenet: A large-scale hierarchical image database. In: IEEE. *2009 IEEE conference on computer vision and pattern recognition.* [S.l.], 2009. p. 248–255.

[Devlin et al. 2018]DEVLIN, J. et al. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[Ding et al. 2020]DING, D. et al. Object-based attention for spatio-temporal reasoning: Outperforming neuro-symbolic models with flexible distributed architectures. *arXiv preprint arXiv:2012.08508*, 2020.

[Donahue e Simonyan 2019]DONAHUE, J.; SIMONYAN, K. Large scale adversarial representation learning. In: *Advances in Neural Information Processing Systems.* [S.l.: s.n.], 2019. p. 10542–10552.

[Dosovitskiy et al. 2014]DOSOVITSKIY, A. et al. Discriminative unsupervised feature learning with convolutional neural networks. In: *Advances in neural information processing systems.* [S.l.: s.n.], 2014. p. 766–774.

[Everingham et al. 2010]EVERINGHAM, M. et al. The pascal visual object classes (voc) challenge. *International journal of computer vision*, Springer, v. 88, n. 2, p. 303–338, 2010.

[Feichtenhofer et al. 2019]FEICHTENHOFER, C. et al. Slowfast networks for video recognition. In: *Proceedings of the IEEE international conference on computer vision.* [S.l.: s.n.], 2019. p. 6202–6211.

[Gidaris, Singh e Komodakis 2018]GIDARIS, S.; SINGH, P.; KOMODAKIS, N. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

[Girdhar et al. 2019]GIRDHAR, R. et al. Video action transformer network. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* [S.l.: s.n.], 2019. p. 244–253.

[Girdhar e Ramanan 2019]GIRDHAR, R.; RAMANAN, D. Cater: A diagnostic dataset for compositional actions and temporal reasoning. *arXiv preprint arXiv:1910.04744*, 2019.

[Gkioxari e Malik 2015]GKIOXARI, G.; MALIK, J. Finding action tubes. In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* [S.l.: s.n.], 2015. p. 759–768.

[Gordon et al. 2020]GORDON, D. et al. Watching the world go by: Representation learning from unlabeled videos. *arXiv preprint arXiv:2003.07990*, 2020.

[Goyal et al. 2019]GOYAL, P. et al. Scaling and benchmarking self-supervised visual representation learning. In: *Proceedings of the IEEE International Conference on Computer Vision*. [S.l.: s.n.], 2019. p. 6391–6400.

[Greff, Steenkiste e Schmidhuber 2020]GREFF, K.; STEENKISTE, S. van; SCHMIDHUBER, J. On the binding problem in artificial neural networks. *arXiv preprint arXiv:2012.05208*, 2020.

[Grill et al. 2020]GRILL, J.-B. et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

[Gutmann e Hyvärinen 2010]GUTMANN, M.; HYVÄRINEN, A. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. [S.l.: s.n.], 2010. p. 297–304.

[He et al. 2016]HE, K. et al. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 770–778.

[Hochreiter e Schmidhuber 1997]HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural computation*, MIT Press, v. 9, n. 8, p. 1735–1780, 1997.

[Hudson e Manning 2018]HUDSON, D. A.; MANNING, C. D. Compositional attention networks for machine reasoning. *arXiv preprint arXiv:1803.03067*, 2018.

[Iter et al. 2020]ITER, D. et al. Pretraining with contrastive sentence objectives improves discourse performance of language models. *arXiv preprint arXiv:2005.10389*, 2020.

[Jin et al. 2020]JIN, B. et al. Exploring spatial-temporal multi-frequency analysis for high-fidelity and temporal-consistency video prediction. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2020. p. 4554–4563.

[Jing e Tian 2020]JING, L.; TIAN, Y. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, 2020.

[Johnson et al. 2017]JOHNSON, J. et al. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2017. p. 2901–2910.

[Kalfaoglu, Kalkan e Alatan 2020]KALFAOGLU, M.; KALKAN, S.; ALATAN, A. A. Late temporal modeling in 3d cnn architectures with bert for action recognition. *arXiv preprint arXiv:2008.01232*, 2020.

[Kipf, Pol e Welling 2019]KIPF, T.; POL, E. van der; WELLING, M. Contrastive learning of structured world models. *arXiv preprint arXiv:1911.12247*, 2019.

[Kuehne et al. 2011]KUEHNE, H. et al. Hmdb: a large video database for human motion recognition. In: IEEE. *2011 International Conference on Computer Vision*. [S.l.], 2011. p. 2556–2563.

[Kuznetsova et al. 2020]KUZNETSOVA, A. et al. The open images dataset v4. *International Journal of Computer Vision*, Springer Science and Business Media LLC, v. 128, n. 7, p. 1956–1981, Mar 2020. ISSN 1573-1405. Disponível em: <http://dx.doi.org/10.1007/s11263-020-01316-z>.

[Lan et al. 2019]LAN, Z. et al. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[Lee et al. 2019]LEE, J. et al. Set transformer: A framework for attention-based permutation-invariant neural networks. In: PMLR. *International Conference on Machine Learning*. [S.l.], 2019. p. 3744–3753.

[Lin et al. 2014]LIN, T.-Y. et al. *Microsoft COCO: Common Objects in Context*. 2014.

[Liu et al. 2020]LIU, X. et al. Self-supervised learning: Generative or contrastive. *arXiv*, p. arXiv–2006, 2020.

[Luo et al. 2020]LUO, D. et al. Exploring relations in untrimmed videos for self-supervised learning. *arXiv preprint arXiv:2008.02711*, 2020.

[Newell e Deng 2020]NEWELL, A.; DENG, J. How useful is self-supervised pretraining for visual tasks? In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2020. p. 7345–7354.

[Oprea et al. 2020]OPREA, S. et al. A review on deep learning techniques for video prediction. *arXiv preprint arXiv:2004.05214*, 2020.

[Patrick et al. 2020]PATRICK, M. et al. Multi-modal self-supervision from generalized data transformations. *arXiv preprint arXiv:2003.04298*, 2020.

[Purushwalkam e Gupta 2020]PURUSHWALKAM, S.; GUPTA, A. *Demystifying Contrastive Self-Supervised Learning: Invariances, Augmentations and Dataset Biases*. 2020.

[Qian et al. 2020]QIAN, R. et al. Spatiotemporal contrastive video representation learning. *arXiv preprint arXiv:2008.03800*, 2020.

[Radford et al. 2019]RADFORD, A. et al. Language models are unsupervised multitask learners. *OpenAI blog*, v. 1, n. 8, p. 9, 2019.

[Ren et al. 2015]REN, S. et al. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015.

[Russakovsky et al. 2015]RUSSAKOVSKY, O. et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, Springer, v. 115, n. 3, p. 211–252, 2015.

[Schuster e Paliwal 1997]SCHUSTER, M.; PALIWAL, K. K. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, Ieee, v. 45, n. 11, p. 2673–2681, 1997.

[Soomro, Zamir e Shah 2012]SOOMRO, K.; ZAMIR, A. R.; SHAH, M. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[Srivastava, Mansimov e Salakhudinov 2015]SRIVASTAVA, N.; MANSIMOV, E.; SALAKHUDINOV, R. Unsupervised learning of video representations using lstms. In: *International conference on machine learning*. [S.l.: s.n.], 2015. p. 843–852.

[Sun et al. 2019]SUN, C. et al. Learning video representations using contrastive bidirectional transformer. *arXiv preprint arXiv:1906.05743*, 2019.

[Tan et al. 2018]TAN, C. et al. A survey on deep transfer learning. In: SPRINGER. *International conference on artificial neural networks*. [S.l.], 2018. p. 270–279.

[Vaswani et al. 2017]VASWANI, A. et al. Attention is all you need. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2017. p. 5998–6008.

[Yao et al. 2020]YAO, T. et al. Seco: Exploring sequence supervision for unsupervised representation learning. *arXiv preprint arXiv:2008.00975*, 2020.

[Yi et al. 2019]YI, K. et al. Clevrer: Collision events for video representation and reasoning. *arXiv preprint arXiv:1910.01442*, 2019.

[You, Gitman e Ginsburg 2017]YOU, Y.; GITMAN, I.; GINSBURG, B. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017.

[Yu et al. 2019]YU, W. et al. Efficient and information-preserving future frame prediction and beyond. In: *International Conference on Learning Representations*. [S.l.: s.n.], 2019.

[Zhai et al. 2019]ZHAI, X. et al. S4l: Self-supervised semi-supervised learning. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2019. p. 1476–1485.

[Zhai et al. 2019]ZHAI, X. et al. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*, 2019.

[Zhang, Isola e Efros 2016]ZHANG, R.; ISOLA, P.; EFROS, A. A. Colorful image colorization. In: SPRINGER. *European conference on computer vision*. [S.l.], 2016. p. 649–666.

[Zhang, Isola e Efros 2016]ZHANG, R.; ISOLA, P.; EFROS, A. A. *Split-Brain Autoencoders: Unsupervised Learning by Cross-Channel Prediction*. 2016.

[Zhuang et al. 2020]ZHUANG, C. et al. Unsupervised learning from video with deep neural embeddings. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2020. p. 9563–9572.