Gabriel Santos Luz

Contrastively-trained Structured World Models (C-SWMs): Modifications and Evaluation on Downstream Tasks

Belo Horizonte, Minas Gerais
2020

Gabriel Santos Luz

Contrastively-trained Structured World Models (C-SWMs): Modifications and Evaluation on Downstream Tasks

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

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.

However, there is still much to be explored, especially in the context of using video data. Videos are a more powerful source of visual information than static images and have not been fully explored to improve computer vision methods through self-supervision. Videos offer spatial and temporal information, such as presenting an object from multiple views or differentiating a moving object from the background, that can be used to get a better understanding of the world.

In this context, the algorithm Contrastively-trained Structured World Models (C-SWMs) [Kipf, Pol e Welling 2019] was proposed. It tries to model the current world state using objects extracted from observations, representing it as a set of vectors, one for each object. The algorithm learns by trying to predict the next state representation using a contrastive loss function. It combines the successful contrastive self-supervised learning with object based approaches, opening interesting possibilities such as applying the contrastive loss on the level of objects, and not just of images or videos. However, C-SWM was evaluated using ranking metrics, whose correlation with downstream task performance is not clear, and it has not been evaluated against strong self-supervised learning algorithms. Furthermore, the algorithm does not use data augmentation and uses an older loss function that is different from the recent successes in contrastive learning.

The main contributions of this work are to propose a better evaluation protocol for C-SWM and to use it to evaluate modifications that include successful ideas from recent self-supervised learning literature. This work does not aim to validate C-SWM as a competitive algorithm, as it will not be compared to strong self-supervised algorithms.

2 Theoretical Background

2.1 Fundamental Concepts

2.1.1 Basic 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.

2.1.2 Pretext Tasks

The pretext tasks for self-supervised learning are essential and determine the method approach. We follow [Jing e Tian 2020], which summarizes the pretext tasks in four non exclusive categories: generation-based, context-based, free semantic label-based and cross modal-based.

- Generation-based: Any method that uses image or video generation as a pretext task. The task might be to generate parts or entire images.

- Context-based: These methods use pretext tasks based on context features. The considered context can be, for example, other images in the dataset, patches of the same image, and frames in a video. An example of a context-based pretext task is given a set of frames from a video, the algorithm must put them in the correct order. The labels are easily generated. A very important example are methods based on context similarity, such as clustering ones or contrasting methods that use a loss function that compares entities, such as the loss Noise Contrastive Estimation (NCE) [Gutmann e Hyvärinen 2010].

- Free semantic label-based: These methods use automatically generated semantic labels as pretext tasks. These labels could have been generated by a non-supervised algorithm or by a game engine in a simulation. An example is to use semantic segmentation automatically generated by a game engine to train a model to do semantic segmentation. These methods may not be categorized as self-supervised as they require humans to build the simulations, but they should not be discarded, because we already have good simulators.

- Cross modal-based: These methods focus on correspondence by verifying if two inputs are corresponding to each other. An example is to train a model to verify if a video corresponds to a given audio.

As we want to evaluate the impact of using video data instead of static images, we will focus on methods that can learn only from RGB images or videos. Methods that use Ergo-motion are an example of methods that will not be included in this work, as they require extra data.

Many of the most successful methods are context-based and use a contrastive loss. In general terms these methods try to maximize the contrast between different images while maximizing the similarity between an image and some augmentations of it, such as rotations, filters, patches and other hand designed transformations. This leaves an interesting opening to explore how video data can improve these methods. Therefore, we want to explore this open question during the evaluation phase of the project.

2.1.3 Downstream Tasks

The survey [Jing e Tian 2020] gives the four most used downstream tasks for evaluating visual features learned by self-supervision:

- Semantic Segmentation: Given an input image, classify each pixel with a semantic label. The main datasets for semantic segmentation are: COCO [Lin et al. 2014], PASCAL VOC [Everingham et al. 2010], Cityscapes [Cordts et al. 2016] and CamVid [Brostow et al. 2008].
- Object Detection: Localizing the position of objects in images and classifying each localized object. One common way is to use bounding boxes to localize the desired objects. Two important datasets are: COCO [Lin et al. 2014] and OpenImages [Kuznetsova et al. 2020].
- Image Classification: Given an image, classify it. It is a classic Deep Learning task and ImageNet [Deng et al. 2009] is one of the most traditional datasets.
- Human Action Recognition: Given a video, classify it with a class from a list of actions. This commonly used as a downstream task for self-supervised learning on videos.

The evaluation of self-supervised learning is often done in specific downstream tasks and dataset, but, recently, diverse benchmarks were proposed with the goal of better evaluating the learned representations. Two recent benchmarks are introduced in the Related Work section.

2.2 Related Work

2.2.1 Evaluation of Self-Supervised Methods

The evaluation of the quality of learned visual representations has been the subject of a few recent papers. The majority of previous works choose to evaluate it on a few standard downstream tasks and use a simple transfer learning method. 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. However, some works question if this approach is appropriate.

One of the most important papers on this matter proposed the Visual Task Adaptation Benchmark (VTAB) [Zhai et al. 2019]. Instead of using only one or a few downstream tasks, VTAB uses 19 tasks that cover many important computer vision tasks. It is divided into three sets of tasks: natural, specialized and structured. One key idea is that VTAB aims to measure the representation quality by adapting it to diverse unseen tasks with few examples. VTAB-1k forces sample efficiency by providing only 1,000 examples for each task. The benchmark is not restricted to self-supervised methods and do not limit the transfer learning techniques used, the methods must simply obey the restrictions of not using the benchmark tasks or datasets for training and that hyperparameter search must not be task-dependent, meaning that the same hyperparameter search is used for all tasks.

The paper used VTAB to compare many different approaches for self-supervised learning, focusing on studying the self-supervised loss function. It showed that generative methods performed worse than discriminative ones, with BigBiGAN [Donahue e Simonyan 2019] being the only exception. In fact, they showed that generative methods performed worse than training a supervised learning algorithm from scratch on kenda-1k. The best self-supervised algorithms were: BigBiGAN, Rotation [Gidaris, Singh e Komodakis 2018] and Exemplar [Dosovitskiy et al. 2014]. It also showed that Kendall's correlation between linear evaluation protocol and fine-tuning is very low on many tasks of VTAB, indicating that fine-tuning, which yields better results, is a more appropriate evaluation.

Another important benchmark is the Facebook AI SSL challenge [Goyal et al. 2019]. Similarly to VTAB it use a diverse set of task, in this case 9. However, differently to other benchmark, it limits the transfer learning part, as it believes to be a better way to evaluate the learned representations. Another difference is that it does not turn every task into a classification task.

The paper [Newell e Deng 2020] compares self-supervised algorithms by generating synthetic data, which allows for generating different downstream tasks with varying complexities. It offers very useful tools for evaluating methods. It also defines utility of a self-supervision method as the ratio of additional labels used by a supervised method to achieve the same performance of the fine-tuned self-supervised method. It showed, once again, that the linear evaluation protocol is not a good indicative of fine-tuning performance.

2.2.2 Using Static Images

Self-supervised algorithms for learning visual representations from static images have been widely studied, here we focus on the state-of-the-art methods on self-supervised image classification on ImageNet and VTAB.

The three best performing methods on ImageNet are based on constrastive loss, even though. SimCLR (Simple Framework for Contrastive Learning of Visual Representations) [Chen et al. 2020] is one of the main contrastive learning algorithmm and it's second version [Chen et al. 2020] is the current state-of-the-art in self-supervised learning for ImageNet [Russakovsky et al. 2015]. The second is Bootstap Your Own Latent (BYOL) [Grill et al. 2020] and the third is Swapping Assignments between multiple Views of the same image (SwAV) [Caron et al. 2020].

In the VTAB paper, the authors showed that BigBiGAN [Donahue e Simonyan 2019] was the best generative method in VTAB-1k, among the ones tested, and it is also one of the best generative methods on ImageNet. The best generative approach on the latter is iGPTL [Chen et al. 2020], it draws inspiration from language models and scales computation and size to achieve good results.

2.2.3 Using Video Data

The literature on self-supervised learning from videos is extensive, but only a few evaluate the learned representations on static image downstream tasks. Here we will introduce recent papers that use video data and achieved very good performance.

[Purushwalkam e Gupta 2020] proposes a way to investigate the invariances learned by contrastive self-supervised methods. It discovered that common data augmentation techniques do not enforce important invariances when dealing with objects, such as viewpoint and deformation invariance. It proposes using video data and an unsupervised object tracker algorithm to produce better data augmentations. This work is an example of directly using videos to improve image-based methods.

[Zhuang et al. 2020] presented Video Instance Embedding (VIE) based on static image contrastive methods and compared it to other video-based methods, showing better performance on ImageNet, UCF101 and HMDB51.

[Gordon et al. 2020] argues that videos provide data augmentation for free and proposes Video Noise Contrastive Estimation (VINCE) for modifying contrastive methods to use video data. It is a general method that can be integrated with state-of-the-art image-based algorithms. It also built a new video dataset Random Related Video Views (R2V2) that can be useful for comparing methods. This paper is a very good comparison for our work, as it explicitly tries to use videos to improve static images approaches.

Sequence Contrastive Learning (SeCo) [Yao et al. 2020] reported better results than VIE and VINCE. A problem is that SeCo and VIE were not compared to static image

methods and did not use a very diverse benchmark such as VTAB.

Other recent approaches, evaluated on different scenarios, are [Luo et al. 2020] and [Qian et al. 2020].

Even though contrastive approaches dominate the self-supervision scene, generative approaches are being actively researched. One approach to generative self-supervised learning from videos is video prediction. It consists in given a sequence of past frames we want to predict a sequence of next frames. The review [Oprea et al. 2020] proposes a taxonomy for Deep Learning techniques and datasets, laying a good foundation for understanding the state-of-the-art in 2020. The metrics and datasets are diverse, but we will focus on methods that perform well overall. Two remarkable methods are [Jin et al. 2020], which is a stochastic approach that uses the Wavelet Transform, and CrevNet [Yu et al. 2019].

2.3 Contrastive Learning

2.3.1 Noise Contrastive Estimation (NCE)

According to the original paper [Gutmann e Hyvärinen 2010], NCE is an estimation principle for parametrized statistical models. The original idea was to estimate the model parameters by discriminating real observed data from generated noise using nonlinear logistic regression. It was shown to be a convergent estimator of the parameters and to directly work for unnormalized models. An unnormalized model is one that the density function does not integrate to one. The normalization constant is usually very hard to compute.

The paper formulates the estimation problem as trying to model an unknown probability density function (pdf) $p_d(.)$, which is the data pdf. It models $p_d(.)$ using a parametrized distribution $p_m(.;a)$ from a family of functions. It assumes that $p_d(.)$ belongs to this family, and that $p_d(.) = p_m(.;a*)$. Therefore, the goal is to estimate the parameters a from observed samples from the data by maximizing some objective function. It imposes that $p_m(.;a)$ must integrate to one, which can also be fulfilled by redefining $p_m(.;a) = p^0_m(.;a)/Z(a)$, being Z(a) the normalization constant obtained from $p^0_m(.;a)$ and data. $Z(a)$ is usually intractable.

A way of dealing with $Z(a)$ is to consider it as an additional parameter of the model. This approach is not possible for Maximum Likelihood Estimation (MLE), as the likelihood can be made as big as one wants by setting $Z(a)$ closer to 0. As a result methods usually use the unnormalized function.

The NCE paper proposes an estimator that allows to estimate both a and $Z(a)$ from the same objective function. It does so by learning to discriminate between data samples and artificial noise. It presents a direct comparison to supervised learning to explain the estimator: it discriminates between data and noise, in a supervised manner, and

learns properties of the data. It basically outputs if the input example is from the data distribution or from noise. In practical terms it explicitly models the data distribution using a parametrized model and use a loss function that allows to learn these parameters by contrasting with noise. Therefore, in the end the result is a learned parametrized model of the data distribution. The noise distribution must obey to some technical constraints but, intuitively, it should be close to the data distribution in order to make the classification problem difficult.

NCE is the main theoretical base of many contrastive algorithms as mentioned in [Liu et al. 2020].

2.3.2 Simple Framework for Contrastive Learning of Visual Representations (SimCLR)

SimCLR [Chen et al. 2020] is the current (October, 2020) state-of-the-art self-supervised learning algorithm on the ImageNet dataset and illustrates the basic idea of a contrastive learning approach. It proposes a simpler algorithm than previous contrastive learning ones as it does not use memory banks or techniques such as clustering.

Among its major contributions, it showed that using a composition of data augmentations is very important for the pretext task, a nonlinear transformation between the learned representation and the pretext task output highly improves the representations and that contrastive learning benefits from larger batch sizes and more training steps than supervised learning. It was also found that representation learning with contrastive cross entropy loss is improved by normalized embeddings and a tuned temperature parameter, which are used in a loss function called Normalized Temperature-scaled Cross Entropy Loss (NT-Xent).

The data augmentation method was to sequentially apply 3 simple augmentations: random cropping, then resize to original size, random color distortions and Gaussian blur. The authors conclude that random cropping combined with color distortion is much better than the other augmentations tested.

ResNet was the neural network architecture chosen, but it can be easily replaced by another model. It also uses a projection head, which is an one hidden layer Multilayer Perceptron (MLP), between the learned representations and the contrastive output (the embedding vector used to calculate the similarity), which was found to increase downstream performance.

The loss function is defined as follows: A minibatch of N examples is randomly sampled, data augmentation is applied generating another N data points, yielding $2 \times N$ data points separated in N pairs. Each pair contains a data point and its augmentation, which is considered a positive pair. The negative examples for a data point are all the other $2 \times (N - 1)$ points in the batch. The loss function for a positive pair of examples (i, j) is

defined as:

$l_{i,j} = -log \frac{exp(sim(z_i,z_j)/\tau)}{\sum_{k=1}^{2N} Ind(k \neq i) \times exp(sim(z_i,z_k)/\tau)}$

Where z_i is the output of the neural network given example i as input, τ is a temperature hyper-parameter, $sim(x, y)$ is a similarity function, which in SimCLR is the cosine similarity, and $Ind(k \neq i)$ is a function that returns 1 if and only if $k \neq i$. Note that z_j is in the z_k's, which are the negative examples, even though z_j is the positive example. The loss is computed over all positive pairs (i, j) and (j, i). By using a big batch it does not require a memory bank.

Training with a larger batch size may be unstable with SGD and Momentum, so the paper used the LARS optimizer [You, Gitman e Ginsburg 2017]. Batch normalization was also used with some modifications.

The evaluation protocol was to use the linear evaluation and fine tuning on ImageNet using 1% and 10% of the labels.

2.3.3 Video Noise Contrastive Estimation (VINCE)

Video Noise Contrastive Estimation (VINCE) [Gordon et al. 2020] combines the idea of obtaining data augmentation by using videos and the Noise Contrastive Estimation loss (NCE). The basic principle is to maximize the similarity between an anchor data point and positive ones, while minimizing the similarity between the anchor and negative data points. VINCE obtains positive data points by using different frames from the same video, providing multiple viewpoints from objects, deformations and other views that serve as a more natural and complete data augmentation obtained from data.

VINCE tries to learn a semantic representation of the entire scene based on a single frame, in hope that if the network can represent different images from the same video with similar vectors, then the representation should encode information about the entire scene, temporal and visual.

It proposes a new dataset Random Related Video Views (R2V2), which is obtained by using ImageNet labels to guide a search on Youtube.

Multi-frame NCE is what the paper calls methods that use multiple frames from a video as positive pairs, in contrast to single-frame approaches that use augmentation from a static image.

VINCE is based on NCE learning, using the NCE loss and batch ideas. It states the NCE methods are improved by using large sets of negative examples as it increases the chance of finding a hard negative for a positive pair. These negatives can be sampled from a memory bank containing earlier outputs of the network, which are from prior batches. The NCE loss must be modified to accommodate this. Using outputs from prior batches can be a problem because the network might learn to contrast between its older outputs and the newer ones. MoCo [He et al. 2020] proposes modifications to alleviate this problem.

Differently from MoCo and SimCLR, VINCE uses more than n positive pairs in a batch of size n. It does so by using multiple frames from the same video, using all the possible pairs. Using v videos and k samples for each one, it can achieve $k^2 \times v$ positive pairs. The paper says that it could use one video per batch and compute n^2 positive pairs, but this leads to instability and extreme gradients. It calls it Multi-pair.

VINCE uses Multi-Frame, Multi-pair and the MoCo memory bank. It also uses data augmentation (crop, flip, color jitter). We could conclude that VINCE simply uses videos on the choice of positive pairs, without explicitly using temporal cues.

It evaluates MoCo, a static image method, on a video dataset, by not using the same frame technique and prevents from having multiple images from the same video for being in the MoCo memory back at the same time. It shows that VINCE achieves better performance than MoCo.

3 Contrastive Learning of Structured World Models (C-SWMs)

3.1 Algorithm

Contrastive Learning of Structured World Models (C-SWMs) [Kipf, Pol e Welling 2019] is an object centred approach that tries to understand the world in terms of objects, their relations and hierarchies. It uses Graph Neural Networks [Zhou et al. 2018] to model objects and relationships and to create state embeddings. The field of learning a structured description of the world using objects has been dominated by generative approaches. However, the most recent breakthroughs in self-supervised learning were, in general, made by contrastive approaches, making C-SWMs a promising innovation to the field.

It operates in a reinforcement learning context and learns state abstractions. A state abstraction is a latent representation of an environment state that encodes important information to predict the next state abstraction after taking a certain action. It can be thought of an encoder that transforms an observed state to a higher level representation and a transition model that takes a state abstraction and an action and predicts the abstraction of the next state.

It can be thought of as implementing the compelling idea of learning a model of the world, but it does so by using a contrastive approach rather than using a generative one. This brings the important notion of similarity and does not require to learn complex pixel based relations.

It is not well defined what is an object centred approach, but it can be vaguely understood as trying to create models with the prior knowledge that the world is composed of objects. An object can be a ball, a hand, an arm or even an entire person. These approaches usually try to capture the notion of compositionality. The idea becomes clearer in the C-SWM algorithm.

C-SWM is based on the graph embedding method TransE [Bordes et al. 2013]. Figure 1 shows a high level view of the model's architecture. The input which is an observation, which could be an image, is passed to an object extractor model that outputs object masks, which serve as input to an object encoder that outputs a set of vectors called abstract state descriptions (ASD). Each ASD is said to capture information from a single object of the scene, which is enforced, but not guaranteed, by the loss function and architecture. This set of vectors are said to describe the state of the world at that time step.

To each ASD, an action is associated. In the paper, the same action is copied to all the ASDs and is the action taken by agent in that state. However, this could be used to model actions taken by other agents in the environment. As another display of flexibility,

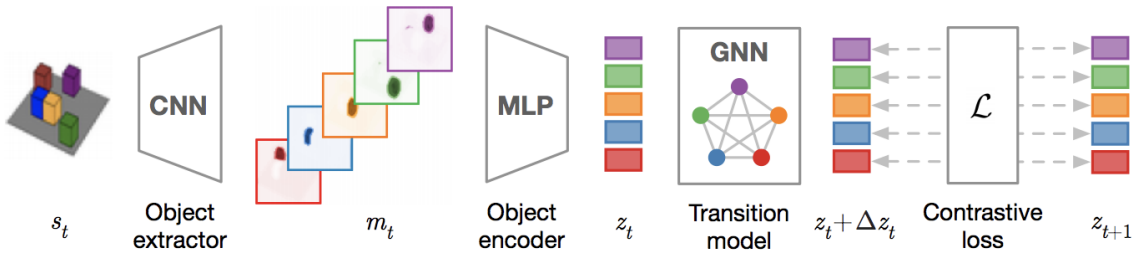


Figura 1 – C-SWMs model. Taken from the original paper [Kipf, Pol e Welling 2019]

C-SWMs can be easily adapted to a video scenario rather than a reinforcement learning one, by ignoring the actions.

The ASDs and actions are passed through a Graph Neural Network, that could model complex relations between objects, and are used to predicted the next set of ASD's. C-SWM is said to learn a structured world model because it represents the world using a structure of objects and use it to predict the next state. One could see this as predicting the future at a higher abstraction level in which useless features such as details in the background can be ignored. Conceptually, this should be a big advantage of C-SWM against pixel based generative approaches. This idea of higher level prediction has also been tried in the paper [Oord, Li e Vinyals 2018].

The contrastive part of the name comes from the fact that it uses a contrastive loss to enforce that the predicted state representation is similar to the actual state representation of the next step. Therefore, the self-supervision comes from the next frame of a video, or, more generally, from the next observation. However, the algorithm uses an energy-based hinge loss [LeCun et al. 2006], differently from the recent success in self-supervised learning.

3.2 Evaluation Metrics

The authors chose to not use downstream task performance as an evaluation metric. Rather, it used metrics that directly measure the quality of the predictions made by the model, more specifically, it uses raking metrics.

The ranking metrics used in the paper are: Hits at Rank 1 (H@1) and Mean Reciprocal Rank (MRR). The metrics are applied in the following context: given an observation and an action, the model predicts the representation $Rpred_{s+1,a}$ of the next state $s+1$ achieved by taking the action a. The algorithm keeps an experience buffer containing a set of different state representations. Then the model sees the true observation of the next state by taking action a in the environment and computes the true state representation $R_{s+1,a}$. The distances between $R_{s+1,a}$ and the other state representations are computed, computing a rank of the representations according to the distance to the true state representation.

The lower the rank, the closer it is to the true representation.

The paper also uses an extension in which the prediction is done after k steps and actions, evaluating a longer scale prediction, up to 10 steps. The model is only trained to predict one step ahead, then predicting k steps ahead is done in an auto-regressive way. The model predicts the next state and uses it as input to predict the next one.

Hits at Rank k (H@k): It is a binary score equal to 1 if the rank of $Rpred_{s+1,a}$ is lower or equal to k, and 0 otherwise. Therefore, H@1 is 1 if and only if $Rpred_{s+1,a}$ is the closest state to $R_{s+1,a}$. The results are averaged over the evaluation dataset. The higher the H@k, the better.

Mean Reciprocal Rank (MRR): It is a more relaxed metric than H@k as it is non binary. $MRR = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{rank_n}$, where $rank_n$ is the rank of $Rpred_{s+1,a}$ in the n_{th} example. The higher the MRR, the better.

3.3 Why C-SWM ?

C-SWM is a very interesting algorithm that combines two very promising approaches: contrastive learning and object based. It can be easily applied to videos, images and reinforcement learning. Its architecture can be modified to inject prior knowledge that could make learning more efficient, such as a special object extractor or transition model. The contrastive loss can be modified to take advantage of objects, which presents an interesting path for using not only entire images, but also specific objects to use as positive and negative examples for a contrastive loss. However, it is not clear if it is a competitive self-supervised learning algorithm and it has several shortcomings. Therefore, this work aims to follow the steps of C-SWM by proposing modifications and better evaluating it.

3.4 Slot Contrastive Networks (SCN)

A follow up to the C-SWM was the SCN [Racah e Chandar 2020]. It is another contrastive and object based approach very similar to C-SWM, but uses a different loss function and architecture.

The authors state that generative models have trouble dealing with small objects (as has been shown by previous works [Anand et al. 2019]) and waste capacity on modelling useless background pixels. In this scenario two self-supervised techniques have gained attention: contrastive learning and pretext tasks. It, other video based approaches, states that static images do not leverage important information such as movement and deformations. It aims to learn an object-centered representation, that instead of using a single embedding vector uses a set of vectors of separate entities. These vectors are called slot vectors.

Similarly to C-SWMs the proposed architecture uses a Convolutional Neural Network (CNN) to map the input frame to K sets of feature maps, called slot maps. It uses a convolutional layer followed by a MLP with shared weights to map each slot map to a slot vector, whereas C-SWMs use a more complicated graph neural network with an MLP object encoder. The two approaches are also different in terms of the loss function and training.

The paper divides the loss function into two parts: slot saliency and slot diversity.

Slot saliency: How to enforce that each slot vector captures a single object? One main idea of the paper is the assumption that objects and other important parts change in time, whereas the background usually does not. Therefore, it tries to ensure that the slot vector captures time dependent features, in hope that objects are dependent on time. It uses as positive pairs the same slot at two consecutive time steps and negative pairs the same slot at random, hopefully non consecutive, time steps. Note that this approach is very different from VINCE, which does not care about time and uses frames from the same video as always similar. However, here we are talking about slot vectors, so it wants the slot to capture a certain object. It tries to capture temporal differences between states. This is similar to the loss function of C-SWMs, as it uses a randomly chosen slot vector from the same slot as negative and the slot vector of the next frame of this slot is chosen as the positive.

Slot diversity: It wants to encourage diversity between slots so it tries to contrast between different slot representations in consecutive time steps. The same slot in consecutive time steps is chosen as a positive pair, whereas different slots in consecutive time steps are chosen as the negative examples.

It aims to directly measure slot representation quality. One way is slot compactness, which uses linear probing to output the coordinates of some objects in the scene, by using linear regression with the input being the concatenated vectors. Apart from this it uses slot modularity and slot accuracy.

In the discussion section the authors make an interesting claim about C-SWMs, which is that it is better at identifying predictable objects, whereas SCN is trained to find any object that moves. Another claim is that the slot diversity loss in SCN does little to actually increase slot diversity, as it does not change much compactness or modularity.

This paper is important for this work as it uses different metrics based on slots and shows an approach to use slot level contrastion. The methods used in this work are very different from the ones used in SCN, especially in terms on the evaluation metrics.

4 Problems and Goals

4.1 C-SWMs Evaluation

C-SWM cannot be proclaimed as a successful self-supervised learning algorithm because it has not been evaluated in challenging datasets, against strong algorithms or on common metrics.

The paper used "toy datasets" such as simple 3D and 2D shapes and Atari games (Space Invaders and Pong), which are good for an initial proposal but cannot guarantee that it will perform well on real world complex applications.

It was evaluated against generative approaches, leaving many doubts about how it will perform against the more recent contrastive approaches.

Furthermore, it is not clear if good performance on the ranking metrics used corresponds to good performance on downstream tasks. In the context of representation learning, a representation is useful if it facilitates a certain task. As a result, downstream task performance is very important to evaluate the quality of the representations. Arguably, it is the ultimate test for a self-supervised learning algorithm for a specific task. Another problem with the metrics used is that no other important recent algorithm has used them, making it impossible to compare.

4.2 C-SWMs Shortcomings

The C-SWM paper lists some limitations of the algorithm. Firstly, it shows that the results can vary with the hyper-parameter that defines the number of objects and concludes that it has trouble dealing with a varying number of objects. A related limitation is that as it uses a simple CNN architecture to extract objects, it will not be able to disambiguate multiple appearances of the same object in the input image. In the literature there are approaches that deal with these problems and could be integrated in the C-SWM architecture, such as MONet [Burgess et al. 2019] and Slot Attention [Locatello et al. 2020].

Another limitation is that it does not model stochasticity, which is essential for complex data such as natural images. A probabilistic modification to the algorithm seems to be a very important step and it may be able to draw inspiration from the video prediction literature, in which there are several works [Oprea et al. 2020] that deal with this problem. Lastly, the transition model is based on the Markov assumption, which may limit how well it can predict and represent states.

Another possible shortcoming is that it does not use data augmentation or the contrastive loss functions that enabled recent success in self-supervised learning.

4.3 POC 1 Goals

The goal of the first part of this work, developed as POC 1, is to tackle some of C-SWM problems. More specifically we focus on the following goals:

- Evaluate C-SWM on a classification downstream task.
- Compare how different evaluation metrics, including ranking metrics, correspond to downstream performance.
- Replace the hinge based energy loss by the SimCLR loss (NT-Xent).
- Evaluate how SimCLR data augmentation affects performance.
- Integrate Slot Attention to the object encoder.

This part does not aim to validate C-SWM as a competitive algorithm. It aims to propose a better evaluation protocol and to use it to evaluate modifications that include successful ideas from recent self-supervised learning literature.

The POC 2 part will be shown in the Future Work section.

5 Methodology

5.1 Evaluation

Following VTAB [Zhai et al. 2019], we chose downstream task performance after fine-tuning as the main metric, as it provides a realistic scenario of transfer learning. The goal of this work is to use self-supervised learning to improve computer vision tasks and transfer learning was chosen as the way of using the learned parameters to make learning a downstream task more efficient in terms of data and training time. Therefore, improving fine tuning performance is the end goal of this work.

Following SimCLR [Chen et al. 2020], we use only 10% of the downstream task dataset for fine-tuning, as it presents a more challenging scenario in which the self-supervised pretraining is essential to efficiently achieve good performance on the task.

The fine tuning evaluation protocol is to train the model on a video dataset (pretext task) and then use the learned parameters as initialization for training on 10% of a static image dataset (downstream task), and then evaluate its performance on the full static image test set. The downstream model uses a different last layer suitable for the task, such as a one layer softmax. Both these phases require hyper-parameter tuning and we chose to use the same optimizer hyper-parameters for all evaluated models, but note that the hyper-parameters in pretraining and fine tuning may differ.

It is part of the transfer learning evaluation method to choose which part of the model will be used for fine tuning. Here we chose to use two options: the full model including the transition model or only the part that computes the state description (object extractor and object encoder).

We also used the linear evaluation protocol mentioned in the theoretical background, which consists in using the learned model as a feature extractor, in the sense that its parameters are not modified during this step. In other words its parameters are frozen. The model is applied on all the downstream training dataset examples and computes a representation for each one, generating a new dataset composed by the learned representations and the original labels. Using this new dataset, a linear classifier is trained on it and evaluated on the test set transformed by the same process. In this work we chose to use Support Vector Machine linear classifier trained using stochastic gradient descent.

We use the ranking metrics implemented in the original C-SWM implementation on *Github*. [1]

[1] `https://github.com/tkipf/c-swm`.

5.2 Datasets

In this work, due to computational resources and limited time, we chose to keep using "toy datasets" to train and evaluate the models. As a first step to propose and validate modifications to the original algorithm, a smaller and simpler dataset may be sufficient.

Here we use the Atari datasets (Space Invaders and Pong) from the C-SWM paper and modifications of the classic MNIST dataset [LeCun, Cortes e Burges 2010].

5.2.1 MNIST

We use the Moving MNIST [Srivastava, Mansimov e Salakhudinov 2015] [2], which is composed by videos of 20 frames showing 2 digits taken from the MNIST dataset bouncing on the image edges in a deterministic fashion. The frames are in grey scale and are 64×64 pixels. We use 60% of the Moving MNIST videos as training and use the other 40% as testing data to compute the ranking metrics. In self-supervised learning terms, we use Moving MNIST for the pretext task.

To use digit classification on MNIST as a downstream task it is necessary to modify the images dimensions because the original dataset dimensions are 28×28 pixels, whereas the model was trained to accept 64×64. To cope with this, we modify the original images by padding the images with more background pixels, in a way that the position of the digit in the image is random, uniformly distributed.

[2] `http://www.cs.toronto.edu/~nitish/unsupervised_video/`

6 Experiments and Results

6.1 Implementation

The implementation of the algorithms and experiments can be found in the following *Github* repository [1].

6.2 Re-evaluation on Atari Environments

The goal of this section is to point out inconsistencies in the evaluation method and datasets used in the C-SWM paper.

The original paper reported poor results and high variance on the Atari Environments. The original paper only evaluated it using four repetitions. We tried to repeat the results by using the same hyper-parameters and the original implementation. We evaluated using 6 repetitions, looking only one step after and using 3 object slots. The results are shown in table 1.

Our results were very different from the original ones, even though we used the same implementation, which may be explained by the randomness in generating the dataset and training. A more complicated issue arose when we trained the model ignoring the actions. The actions were supposed to be a helpful source of information, but ignoring them ended up improving the results. This result may invalidate the way the Space Invaders environment was used in the original paper.

Tabela 1 – Space Invaders environment results for 3 object slots and 1 step

Metrics	Original Results	Our Results	Our Results ignoring actions
Hits @ 1	46.2 ± 13.0	64.7 ± 6.5	84.6 ± 9.2
MRR	62.3 ± 11.5	78.8 ± 4.8	89.0 ± 7.3

We followed the same procedure in the Pong environment and found results similar to the original paper, except that we used more repetitions (6 instead of 4) and found higher variance. The results are shown in table 2.

Tabela 2 – Pong environment results for 3 object slots and 1 step

Metrics	Original Results	Our Results
Hits @ 1	36.5 ± 5.6	30.8 ± 11.9
MRR	56.2 ± 6.2	51.7 ± 9.5

[1] `https://github.com/gabrielsluz/c-swm`

Such poor results in "toy datasets" could indicate that these datasets could be used to evaluate modifications to the original algorithm. However, due to the ignore action results on Space Invaders, we considered that it was reasonable to use another dataset.

6.3 Moving MNIST Evaluation

A hyper-parameter search for neural network models can be extremely costly. As a result, we chose to use the original paper hyper-parameters except for the ones bellow, which were considered the most important and specific to an application.

- Number of objects
- Dimension of the abstract state description, referred as embedding dimension
- Object extractor model, referred as encoder
- Number of epochs

These parameters directly influence the object based implementation. We used the original object extractor models which are three simple CNN architectures. The one called *small* is a one layer CNN that we did not use as it is too simple. We evaluated the *medium* and *large* options. In both options, after each layer, it is applied Batch Normalization [Ioffe e Szegedy 2015] and the activation function used in the last layer is the Sigmoid.

The *medium* architecture is a two layer CNN with 9×9 filters in the first layer, and 5×5 filters with a stride of 5 in the second layer. The activation function used was the $LeakyReLU$ [Xu et al. 2015].

The *large* architecture is a four layer CNN with 3×3 filter and 16 feature maps per layer, except on the last layer which uses one feature map per object slot. The activation function used is the ReLU (Rectified Linear Unit).

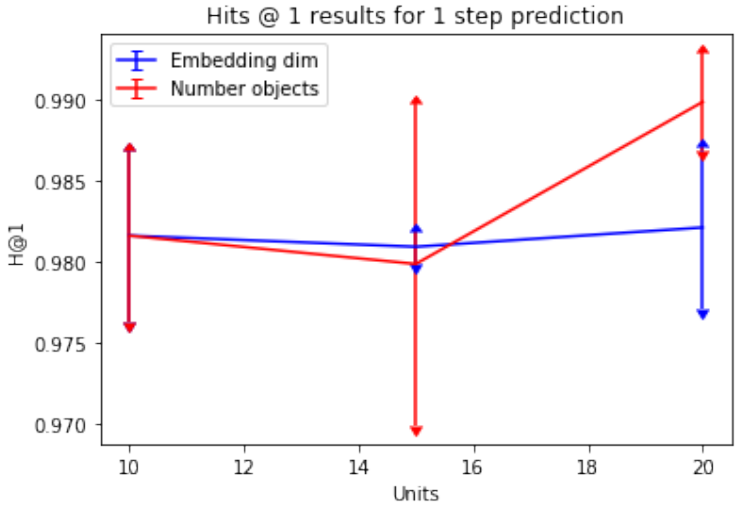
6.4 Encoder and Hidden Dimension

The dimension of the abstract state description (embedding dimension) defines the size of the vector that represents a single object. As a result, a state representation that uses 10 object slots and an embedding dimension of 10, is a matrix of dimension 10×10. As a result, a question arises: Is it better to use a higher embedding dimension or more object slots ?

A simple experiment, using the ranking metrics, was used to try to answer this question. At first the embedding dimension was fixed to 10 and the number of objects was varied. Then the opposite was done. It was repeated 4 times for 50 epochs each. The results are shown in figure 2. Firstly, the results show that the hyper-parameters used were able to

achieve very good performance in the Hits @ 1 ranking metric. Furthermore, it shows that increasing the number of objects in this small scenario was much better than increasing the size of the hidden dimension, even though it had higher variance and lower mean on the middle value.

Figura 2 – Comparison of the impact of the number of objects and embedding dimension



In order to choose between the *medium* or *larger* encoder, an experiment was performed using 10 objects with a embedding dimension of 10, varying only the encoder. It was repeated 4 times for 50 epochs each. The results in table 3 show that the *large* encoder was considerably better. Therefore, we use the *large* encoder for all the next experiments.

Tabela 3 – Results for different encoders

Encoder	H@1
Large	0.982 +- 0.005
Medium	0.950 +- 0.004

6.5 Number of Objects

The goal of this section is to evaluate how the number of object slots impacts fine tuning performance and to check how the evaluation metrics are correlated. Here we evaluated both the model with and without the transition model in the transfer learning step. In this section the expression "number of objects" refers to the number of objects the model is prepared to identify, in other words, it refers to the value of the hyper-parameter that determine how many object slots will be used.

The important hyper-parameters used were:

- Number of objects: 2, 5, 10, 15, 20, 25

- Embedding dimension: 10
- Encoder: Large
- Number of epochs: 10
- Experiment repetitions: 4

The experiments were conducted by training the model using self-supervision on the Moving MNIST dataset for 10 epochs and then the trained was evaluated using the evaluation metrics. The experiment was repeated 4 times for each unique set of hyper-parameters.

The fine-tuning was done using the padded MNIST dataset, divided into training and test sets, using the same division of the original MNIST. The model was fine-tuned for 30 and 60 epochs and then its accuracy on the test set was measured. The fine-tuning process was done for the model with and without the transition model.

By transfer learning we refer to the fine-tuning and linear evaluation methods. The results for these metrics are shown in figure 3. Firstly, there is a positive relation between the number of objects considered and fine-tuning performance. The more objects, the better the performance. One possible interpretation is that by using more objects slots, we are also using a larger model, which increases performance. This matter will be further evaluated on the section "Number of objects versus Embedding dimension". Secondly, the fine-tuning performance was much greater than the linear performance, indicating that in a practical scenario the former would be preferred.

This results contrast with the original paper decision to use a small number of object slots.

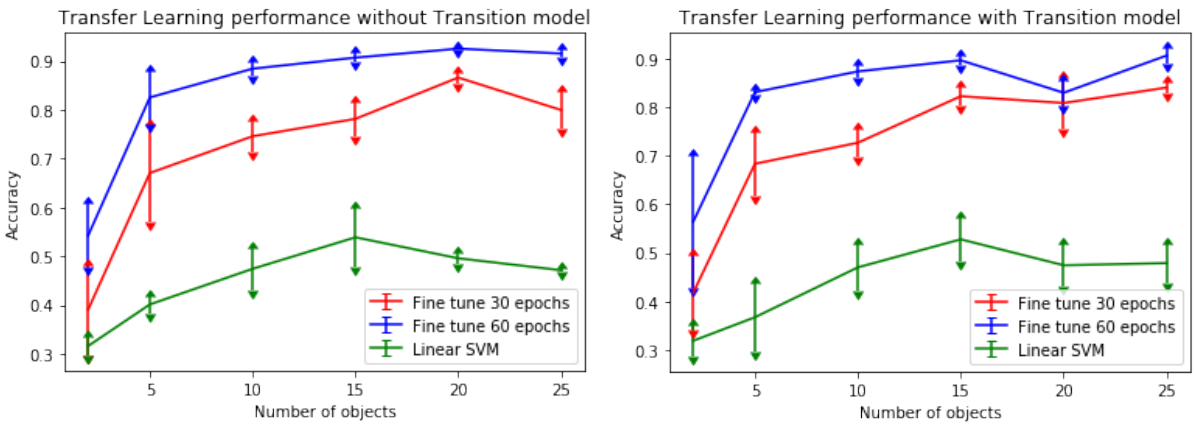
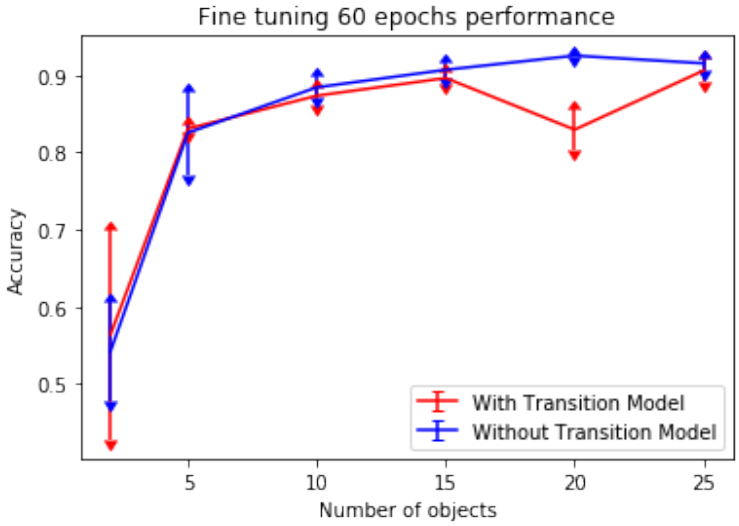


Figura 3 – Transfer Learning evaluation for varying number of objects

In order to compare if the transition model is beneficial during fine-tuning time, we compare the model with and without it in figure 4. The model without the transition

model was better, which is good as the model without is more efficient to fine-tune and to make inferences.

Figura 4 – Comparison of the best models with and without the Transition Model



We also evaluated using the ranking metrics in figure 5. The H@1 and MRR for 1 step prediction show near perfect results, but for 10 steps it gets much worse. The H@1 and MRR metrics show very similar trends.

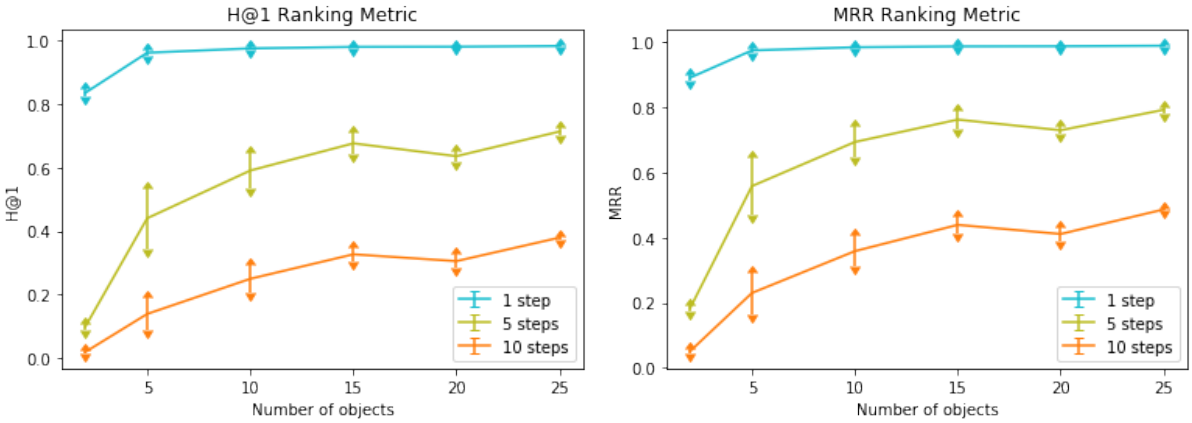


Figura 5 – Ranking metrics evaluation for varying number of objects

In order to evaluate how different metrics correspond to a better fine-tuning performance, we calculated the Pearson's correlation coefficients for all metrics in respect to fine-tuning for 60 epochs in the without transition model scenario. It evaluates how two random variables are linearly correlated, ranging from -1 to 1. A value of -1 indicates that there is total linear negative correlation, whereas a value of 1 indicates that there is total positive correlation. Therefore, the most correlated metric will be the one with the highest value.

Table 4 shows the results associated with the number of objects hyper-parameter. It shows that all the metrics are well correlated with the true metric. One possible conclusion

is that, in respect to the number of objects, the H@1 and MRR may offer a very good approximation to the true metric. An explanation for this is that increasing the number of objects leads to better results in all metrics. However, a parameter such as the number of epochs may have very different results, as it may lead to over specialization in the self-supervision task, increasing performance on the ranking metrics, but decreasing in the downstream tasks.

Tabela 4 – Sorted Pearson's correlation coefficients for the evaluation metrics in respect to Fine Tuning for 60 epochs on the model without the transition model

Evaluation Metric	Pearson's coefficient
Fine Tuning - 60 epochs	1.000
MRR - 1 step	0.993
H@1 - 1 step	0.992
MRR - 5 steps	0.987
Fine Tuning - 30 epochs	0.985
H1 - 5 steps	0.977
MRR - 10 steps	0.937
Linear Evaluation	0.913
H1 - 10 steps	0.906

6.6 Number of epochs

The goals of this section are to evaluate how the models performance varies with the number of training epochs and to evaluate how the metrics are correlated in respect to this hyper-parameter.

Due to computational constraints we chose to use a number of objects equal to 15 and not 20, to evaluate the number of epochs. The other hyper-parameters and the experimental protocol were the same as the number of objects experiments.

Figure 6 displays the results for the transfer learning metrics. It shows very similar results to the number of objects experiments. All the metrics achieve a steady performance with few epochs and do not display signs of overfitting.

Figure 7 shows that the variations with and without the transition model have very similar results, but this time the with variation achieved better performance. In comparison to the number of objects experiments, we can note that increasing the number of epochs from 10 to 20 was not enough to surpass the model that used 20 object slots, instead of 15.

The results for the ranking metrics, shown in figure 8, were very similar as well. Training the model for more epochs allowed to increase these metrics at a steady pace, but the results remained low. It was expected that the model would be able to achieve a good 5 step prediction on the Moving MNIST dataset, but it did not occur. The reason for this should be further explored, as it shows that the model could not learn a, possibly,

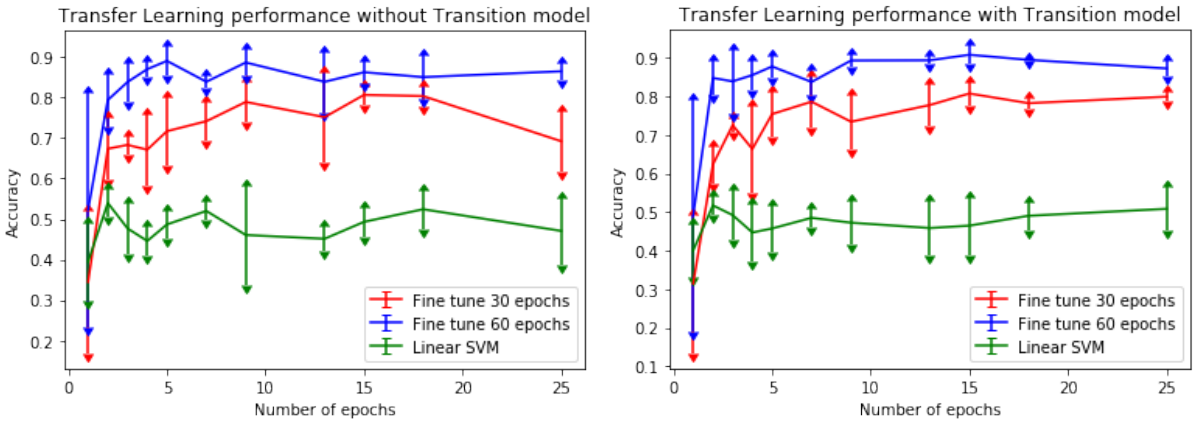
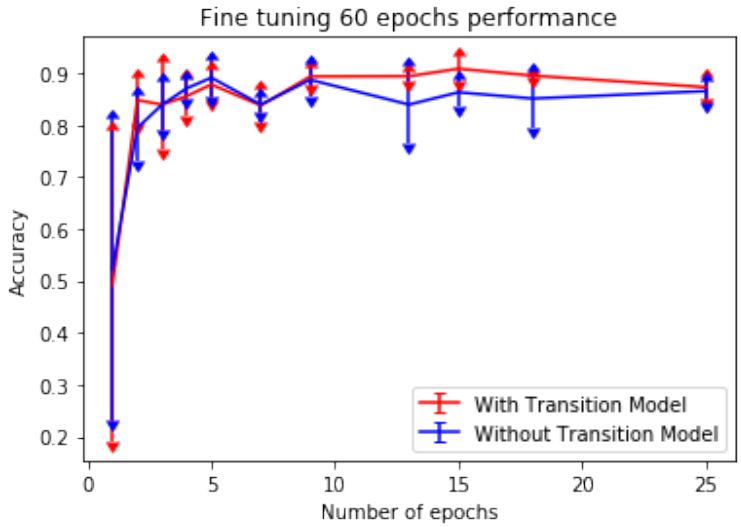


Figura 6 – Transfer Learning evaluation for varying number of epochs

Figura 7 – Comparison of the best models with and without the Transition Model



easy temporal prediction task. C-SWM only uses the next frame as a temporal cue, a modification that uses different time scales may perform better. This matter may be deeply related with the Markov assumption shortcoming, as the model assumes that the current state contains all the information to predict the next state, which may not hold for longer term predictions.

The Pearson's correlation coefficient between all the metrics and fine-tuning accuracy after 60 epochs was calculated, and are shown in table 5. In comparison to the number of objects experiments, the correlation coefficients were much lower. It can be seen that the linear evaluation protocol is almost not linearly correlated with fine-tuning performance, indicating, once again [Zhai et al. 2019], that it is not a good evaluation metric for self-supervised learning algorithms. The metrics H@1 and MRR for 1 step prediction achieved a stable high value, similarly to fine tune performance. The harder metrics (prediction at 5 and 10 steps) achieved a much lower correlation value. Therefore, some of the ranking metrics are not good approximations of downstream task performance.

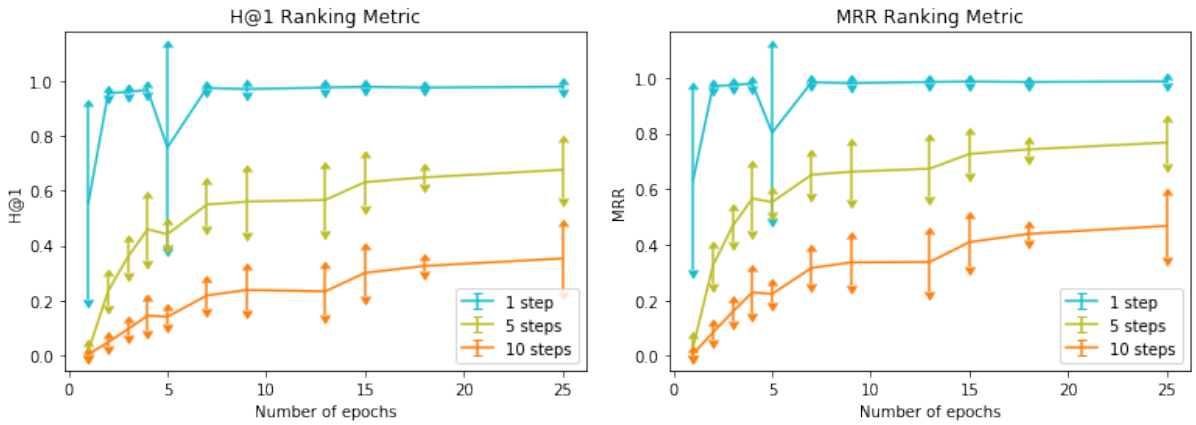


Figura 8 – Ranking metrics evaluation for varying number of epochs

Tabela 5 – Sorted Pearson's correlation coefficients for the evaluation metrics in respect to Fine Tuning with transition model for 60 epochs

Evaluation Metric	Pearson's coefficient
Fine Tuning - 60 epochs	1.000
Fine Tuning - 30 epochs	0.778
H@1 - 1 step	0.660
MRR - 1 step	0.655
MRR - 5 steps	0.281
H1 - 5 steps	0.156
Linear Evaluation	0.129
MRR - 10 steps	-0.100
H1 - 10 steps	-0.180

6.7 NT-Xent Loss Function

The first modification to the original algorithm was to use the NT-Xent loss, used in SimCLR. It was adapted to the C-SWM model by using the pair predicted next state and real next state as positive pairs, whereas all the other images in the mini-batch are considered negative examples. The negative examples may include frames from other videos or any other frame from the same video except the current and next observation.

This time the experiments were done using a different methodology. We only evaluated fine-tuning performance, as it is the desired metric, and we did not fine-tune for a certain amount of epochs and then test the models accuracy on the test set. Instead, we chose to fine-tune for 60 epochs and test the accuracy every 5 fine-tuning epochs. The best accuracy result is returned as the fine-tuning test accuracy. The idea is that choosing a fixed number of epochs, such as 30 or 60, would not be used in practice as it is possible to get a better result by choosing the model that achieves the best validation error during fine-tuning.

The NT-Xent loss has a temperature hyper-parameter, following SimCLR, we evaluate

the temperature on two promising values: 0.1 and 0.5 . The results shown in figure 9 indicate that using the transition model during fine-tuning is considerably better. This result is different from SimCLR, as the MLP head is used for the self-supervision part, but taken out in the transfer learning part. Here we kept the transition model, analogous to the MLP head, achieving better results than fine-tuning without it.

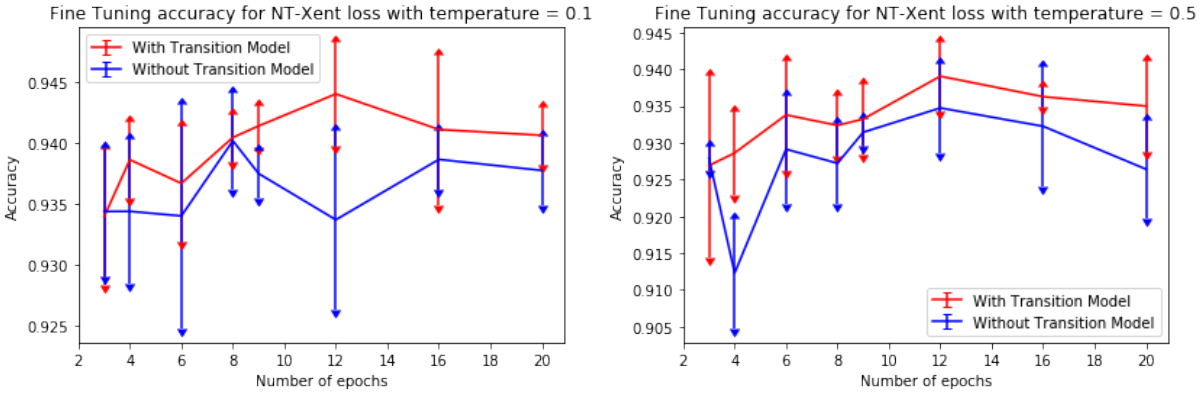
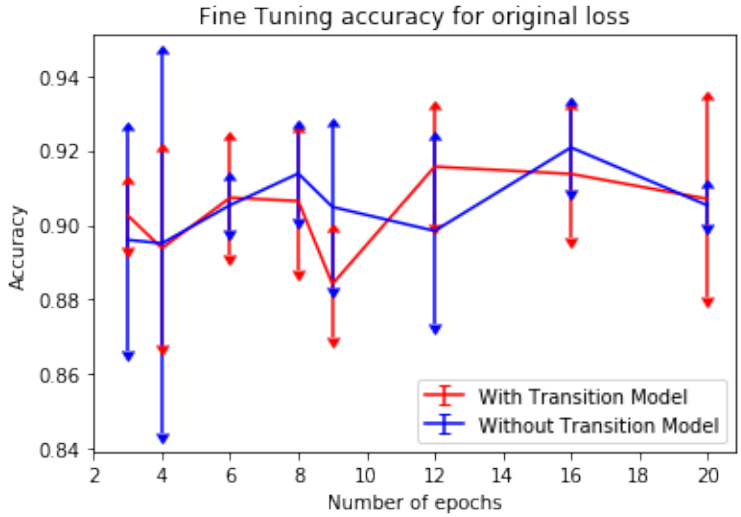


Figura 9 – Fine Tuning evaluation for NT-Xent loss

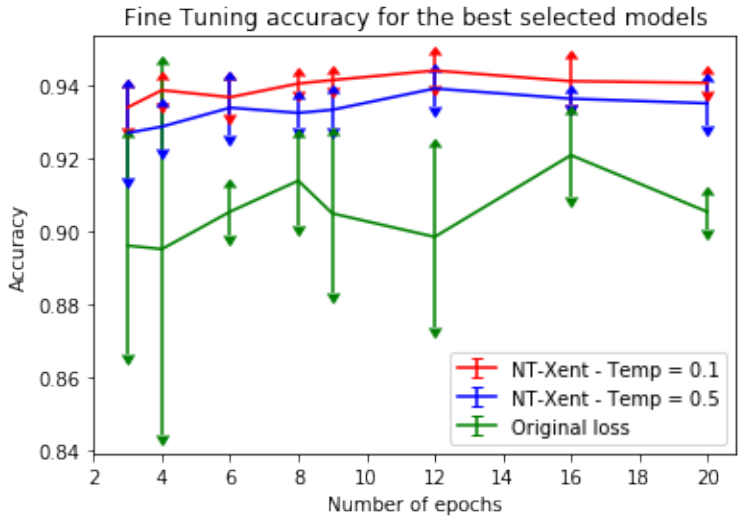
Figure 10 shows the results for the original C-SWM loss function, which was re-evaluated using the new fine-tuning protocol. The results are very similar to the ones found in the number of epochs experiments, but with the different evaluation protocol, the results were slightly better, achieving an accuracy higher than 0.9 .

Figura 10 – Fine Tuning evaluation for the original loss



The main result of this section is shown in figure 11. The models trained using the NT-Xent loss were considerably better than the model trained with the energy based hinge loss. The results for the new loss were stabler and displayed lower variance.

Figura 11 – Fine Tuning evaluation comparing NT-Xent loss with the original loss



6.8 Data Augmentation

For these experiments we used the entire Moving MNIST dataset to train the model, as the validation part was only used for the ranking metrics.

We used almost the same data augmentation composition as SimCLR, which showed that randomly cropping, resizing and then applying random color distortion greatly improved the quality of the learned representations. We used these transformations to augment every image on a mini-batch. For the random cropping we chose a crop size of 50 pixels, as the images of the dataset are largely composed of black pixels, and using smaller crop sizes could result in fully black crops.

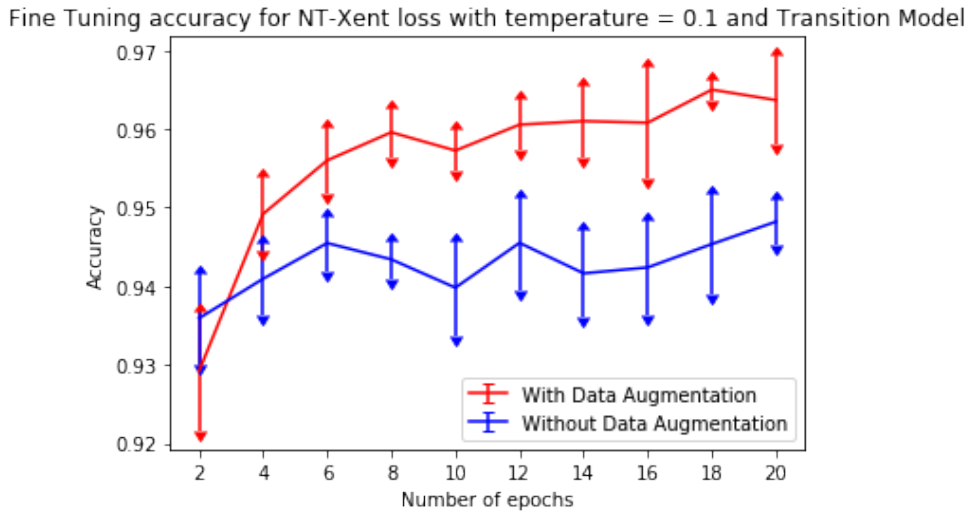
Figure 12 shows the results for the model with and without data augmentation trained on the entire Moving MNIST dataset. The results were greatly improved and, the model trained with data augmentation benefited more from more epochs than the previous ones.

One very important result, shown in table 6, is that the model trained with the NT-Xent loss function and data augmentation achieves a higher fine tuning performance, but a lower ranking metric performance, in comparison to the original C-SWM loss. This shows that for different losses the ranking metrics are not a good indicator of downstream task performance.

Tabela 6 – Comparison between the loss functions in different metrics. Both models were trained for 20 epochs.

Metric	NT-Xent loss	Hinge energy based loss
H@1	0.689	0.980
MRR	0.727	0.987
Fine Tuning accuracy	0.954	0.895

Figura 12 – Fine Tuning evaluation comparing the impact of data augmentation on the best model so far.



6.9 Slot Attention

The Slot Attention [Locatello et al. 2020] is a neural network module based on an attention mechanism that receives perceptual representations, such as features maps outputted from a CNN, and outputs a set of vectors, called slots. The reason to consider using Slot Attention is that it is a specialized object extractor, which achieved better results than other strong approaches, such as MONet [Burgess et al. 2019] and IODINE [Greff et al. 2019]. Furthermore, it is an iterative object encoder, which was recommended in the original paper as an alternative to increase robustness to a varying numbers of objects.

The Slot Attention module fits with the C-SWM model, as it has an object extractor that outputs feature maps and an MLP encoder that outputs a set of vectors. A straightforward modification is to replace the MLP encoder for a Slot Attention module. The results for this modification are shown in figure 13. We did not use hyper-parameter search and used 3 attention iterations as in the original paper. The results were much worse than the model with the MLP encoder. We also tried a modification in which the Slot Attention received the outputs of the MLP encoder, but the results were even worse, achieving no more than 0.2 accuracy. Two possible reasons for the failure is that the object extractor only outputs one feature map per object slot and we did not use positional embeddings.

6.10 Number of objects versus Embedding dimension

The goal of this section is to go back to the question: Is it better to use a higher embedding dimension or more object slots ? But now with better hyper-parameters and metrics to evaluate it.

Figura 13 – Comparison between the best model and the modification with Slot Attention replacing the MLP encoder

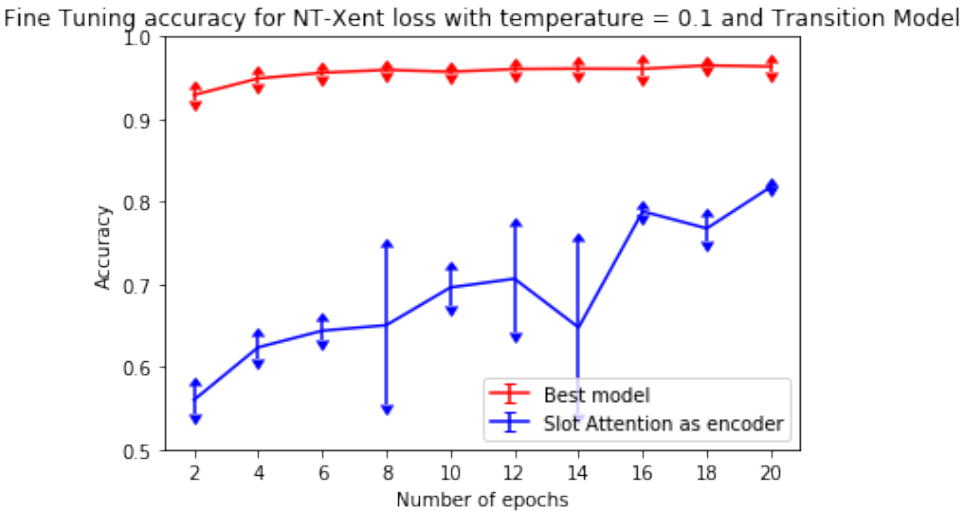


Figure 14 shows the results comparing how different combinations of number of vectors and vector dimension impact fine-tuning accuracy. The red lines show the results for a higher number of object slots and the blue ones for a higher embedding dimension. The plot on the left shows the results for closer values, whereas the one on the right show for more distant ones. A larger number of object slots was more beneficial. One reason may be that the transition model will have more nodes and, as a result, more capacity. Another reason may be that by recognizing simpler objects the model may learn to recognize small parts that are useful for several examples, whereas a more complex object may be restricted to few examples.

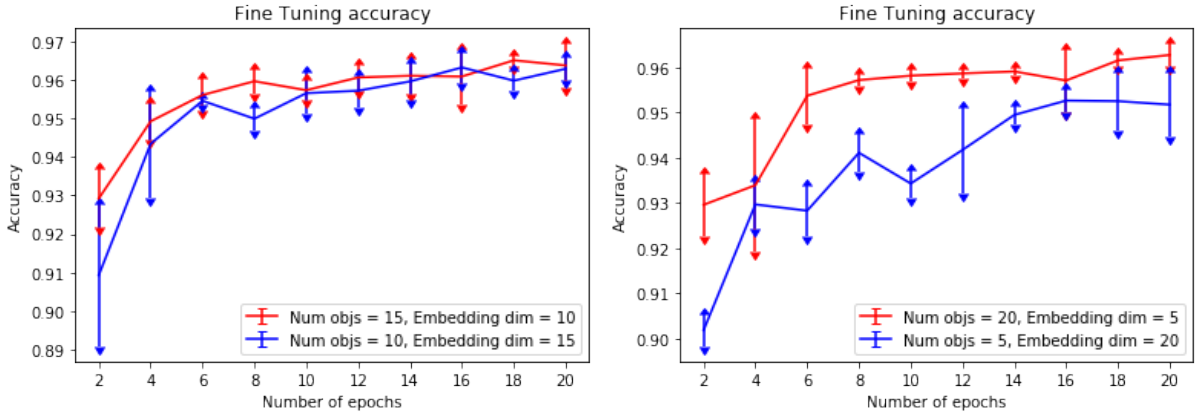


Figura 14 – Comparison between using more object slots and using a higher object slot dimension.

7 Conclusion

7.1 Main Results

In respect to the evaluation metrics, the linear protocol was shown again to be unrealistic and to have low correlation with downstream task performance. The ranking metrics used in the original paper were shown to not always be a good approximation to downstream task performance, especially when evaluating different loss functions. However, the ranking metrics at 5 and 10 steps were shown to be challenging to the current algorithm on a simple dataset, which may indicate that by focusing on improving them, the quality of the learned representations could also improve.

The NT-Xent loss function and data augmentation modifications were successful in improving the models performance and showed that recent developments in self-supervised learning can be applied to C-SWM.

The results showed that using more object slots with less dimensions is better than using fewer slots with more dimensions.

7.2 Future Work

In order to do proclaim C-SWM as a competitive self-supervised learning algorithm, we propose to evaluate it against strong algorithms in complex real world computer vision tasks. One interesting question is if there are tasks that are better suited for object based approaches. One possible answer may be the Clevrer dataset [Yi et al. 2019], in which the algorithm is asked to reason about object's interactions. A task in which object based may have advantages will be preferred.

Another possible future work is to develop ways to use constrative learning at the level of objects, similarly to what was done in Slot Contrastive Networks [Racah e Chandar 2020]. In this context, downstream performance may not be the best metric to evaluate new methods, as it does not allow to directly evaluate the learned representations in terms of objects. However, the end goal is to use contrast at object level to improve downstream performance.

The shortcomings of C-SWM mentioned in the original paper are still open problems. In respect to the Markov assumption limitation, a possible solution may be to use more than one state to base the predictions for the next one and to use ranking metrics at 5 and 10 steps to evaluate it. One possible source of inspiration is the video prediction literature, as the problem is very similar and there are attempts to deal with stochasticity and the Markov assumption [Oprea et al. 2020].

Referências

[Anand et al. 2019]ANAND, A. et al. Unsupervised state representation learning in atari. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2019. p. 8769–8782.

[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.

[Bordes et al. 2013]BORDES, A. et al. Translating embeddings for modeling multi-relational data. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2013. p. 2787–2795.

[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.

[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.

[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.

[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.

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

[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 et al. 2019]GREFF, K. et al. Multi-object representation learning with iterative variational inference. *arXiv preprint arXiv:1903.00450*, 2019.

[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. 2020]HE, K. et al. Momentum contrast for unsupervised visual representation learning. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2020. p. 9729–9738.

[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.

[Ioffe e Szegedy 2015]IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[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.

[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.

[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>.

[LeCun et al. 2006]LECUN, Y. et al. A tutorial on energy-based learning. *Predicting structured data*, v. 1, n. 0, 2006.

[LeCun, Cortes e Burges 2010]LECUN, Y.; CORTES, C.; BURGES, C. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, v. 2, 2010.

[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.

[Locatello et al. 2020]LOCATELLO, F. et al. Object-centric learning with slot attention. *arXiv preprint arXiv:2006.15055*, 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.

[Oord, Li e Vinyals 2018]OORD, A. v. d.; LI, Y.; VINYALS, O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[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.

[Racah e Chandar 2020]RACAH, E.; CHANDAR, S. Slot contrastive networks: A contrastive approach for representing objects. *arXiv preprint arXiv:2007.09294*, 2020.

[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.

[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.

[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.

[Xu et al. 2015]XU, B. et al. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[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.

[Zhou et al. 2018]ZHOU, J. et al. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.

[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.