ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Multi-model disentanglement of object representations

Research Project Thesis

Eric Stavarache

July 19, 2019

Advisors: Frederik Benzing, Asier Mujika

Department of Computer Science, ETH Zürich

Abstract

Being able to decompose a scene into elementary components is a mark of intelligence. In this work we present a novel approach to learning a disentangled representation of scenes, by having several VAE's wired up in parallel, with each of them representing a different type of object.

Contents

Chapter 1

Introduction

Humans reason about the surrounding world by decomposing it into orthogonal components. The idea of an object is decoupled from the qualities of the object: it is very easy for us to imagine a pink elephant, even though we have never observed such an animal, and these two words suffice for us to imagine a visual scene. The words are a latent representation of the scene.

Variational Autoencoders (VAE) [5] are a powerful framework for learning latent representations. Much work has already been done on disentangled representations, where we require that any two latent variables are uncorrelated.

Our work focuses on model-based disentanglement of objects. In particular, for each type of object we would like to have a VAE which is trained to recognise it: in a normal setting, we could have one VAE which represents chairs, another VAE which represents tables, and so on. We will call our approach the **ParallelVAE**.

Chapter 2

Related work

Most similar in spirit is the [2], where a single VAE model iteratively recognises objects from a scene by using attention masks. The main difference is that the MONet uses a single VAE for all objects, whereas we use multiple VAE's in parallel. Another similar idea is contained in [3], where they have a single VAE which tries to model the scene, and where they iteratively refine the latent parameters.

Chapter 3

Model

3.1 ParallelVAE

The ParallelVAE network tries to reconstruct a scene by feeding the image to some VAE's connected in parallel, whereby each VAE models a separate part of the scene.

Let us say that there are K VAE's, where each of them tries to model a different object. Then the image X will be fed independently to each of them.

Each VAE will then model a distribution over the latent variables, $q_{\theta_k}(\mathbf{z}_k|\mathbf{X})$.

By sampling from these distributions, they will generate two outputs of the same size as the image: $\hat{X}_k$, which is the k'th model reconstruction of the image, and $\hat{m}_k$, which is a confidence mask.

This confidence mask represents how sure a VAE is about its output for a given pixel. Higher confidence means that the pixel is part of an object which is of the type the VAE is modelling.

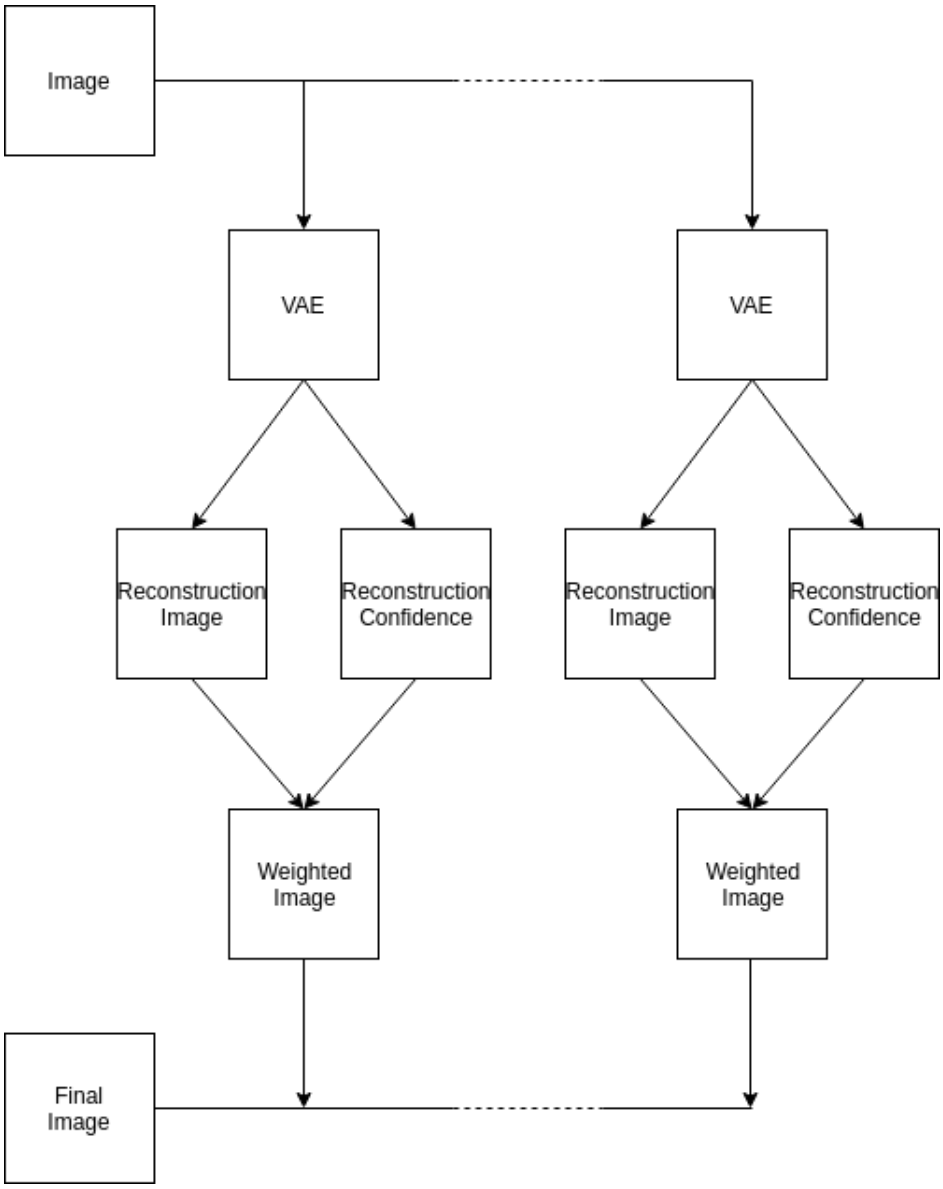
These confidence masks are produced by first having the models generate some "raw" confidence values, and then taking a pixel-wise softmax across all of the k models.

Thus, the masks are a probability distribution, where for each pixel we have what is the probability that it belongs to an object modelled by a VAE:

$\sum_{i=1}^{k} \hat{m}_k = \mathbf{1}$.

Our loss function is

Figure 3.1: A diagram of how the model operates.



$$\begin{aligned}\mathcal{L}(X;\theta_1,...,\theta_k) = &\sum_{i=1}^{k}(\hat{X}_i - X)^2 \odot \hat{m}_i \\ &+ \beta \sum_{i=1}^{k} D_{KL}[q_{\theta_k}(\mathbf{z}_k|\mathbf{X})||\mathcal{N}(0,\mathbf{I})] \\ &+ \gamma \cdot -\log(\frac{1}{\log K} \cdot -\hat{m}\log\hat{m})\end{aligned}$$

The first term of the loss is a L_2 loss weighted by each model's confidence. The intuition behind this is that the more responsability a VAE takes for drawing a certain pixel, the more it should be penalized for making mistakes.

The second term is the standard KL loss term, weighted by a β hyperparameter, as first introduced in [1]. We note that the KL loss can be interpreted as how much information is stored in the bottleneck; the presence of this term means that in an optimal solution, no information should be duplicated between VAE's: if one of them remembers that there is a chair in the picture, then another VAE remembering the same thing would only increase the KL loss, while not providing any additional reconstruction gains.

The last term is our original contribution, which is a cross-entropy loss. The idea is that the VAE's should try to be unassuming, and they should incur a cost if they decide to take on a lot of responsability for representing a pixel. We have the $-\hat{m}\log\hat{m}$ term, which is the normal cross-entropy, where we interpret the masks as component-wise distributions over VAE's. We divide it by the maximal value it can take ($\log K$, where K is the number of VAE's) in order to obtain a value between 0 and 1. We want to penalize situations where one VAE takes over everything, and in these situations the cross-entropy would be close to 0. As such, we take the negative log of the normalized cross-entropy. This whole term is weighted by another hyperparameter, γ. One reason for our introduction of this loss is that, without it, if a VAE would "take over" a part of the image (have really high confidence), then the other VAE's would have really small gradients, and they would never have the chance to learn. With this loss, VAE's are very heavily penalized for being overly confident.

We compare this with the approach of [2], where they have a single VAE which tries to model all objects, whereas we have a single VAE for each object type. Furthermore, in their model, the VAE is instructed what to model by the U-Net attention mask, whereas in our approach each VAE decides "independently" what to learn. In our approach, the only communication happening between the VAE's is via the softmax operation of the confidence masks.

3.1.1 Training

In order to train our ParallelVAE, we split the training into multiple stages. Let us suppose that we know there are K item types, and also our ParallelVAE is composed of K VAE's. Then the training program is:

1. In stage 0, VAE-0 trains on reconstructing images containing only the item 0. The other models is active (so they are contributing to the confidence masks), but their weights are frozen, and only VAE-0 is learning.
2. In stage 1, VAE-1 trains on images containing items 0 and 1 . In this time, VAE-0, VAE-2, ..., VAE-k are frozen, *however* they are still contributing to the confidence masks. Because of this, places where the item 0 appears are already assigned high confidence values by VAE-0, and so VAE-1 will not be motivated to learn them. On the other hand, places where item 1 appears will not be recognised by VAE-0, and so VAE-1 will learn them.
3. ...
4. In stage $2n$, VAE-n trains on the item n, with the others frozen.
5. In stage $2n + 1$, VAE-0, VAE-1, ..., VAE-n train together on images of items 0, 1, ..., n.

3.1.2 Intuition

We will now try to provide some explanations for why the ParallelVAE model previously described should work.

Because of the KL loss, an optimal model should not duplicate information. One instance when this happen is if each VAE learns a different object. Another instance is that one VAE learns everything, and the others do nothing. This scenario can occur if, while training, one VAE just has very high confidence, and so the others never have enough gradients in order to learn anything. In order to avoid this happening, and to encourage the models to split, we introduced the entropy loss – paying a price for taking responsability. This way, the solution when one model takes care of everything is no longer optimal.

Multiple VAE's training all at the same time would pose great optimization instability, so in order to improve stability, we decided to go for the stage-based training approach, so that each VAE is given an opportunity to learn its object.

Chapter 4

Results

For our experiments, we have taken a supervised approach to training the models.

4.1 MNIST

In order to test our idea, we began with the MNIST dataset. To make the task more challenging, and to allow decomposition via objects, we construct our training data by putting two digits side-by-side, where the digits that we use are the same as the number of VAE's (so for 4 VAE's, we will use the digits $0, 1, 2, 3$). We also include the "empty digit" (which is just an empty black square). This is so that the VAE's also encounter scenarios when it is optimal to not output anything.

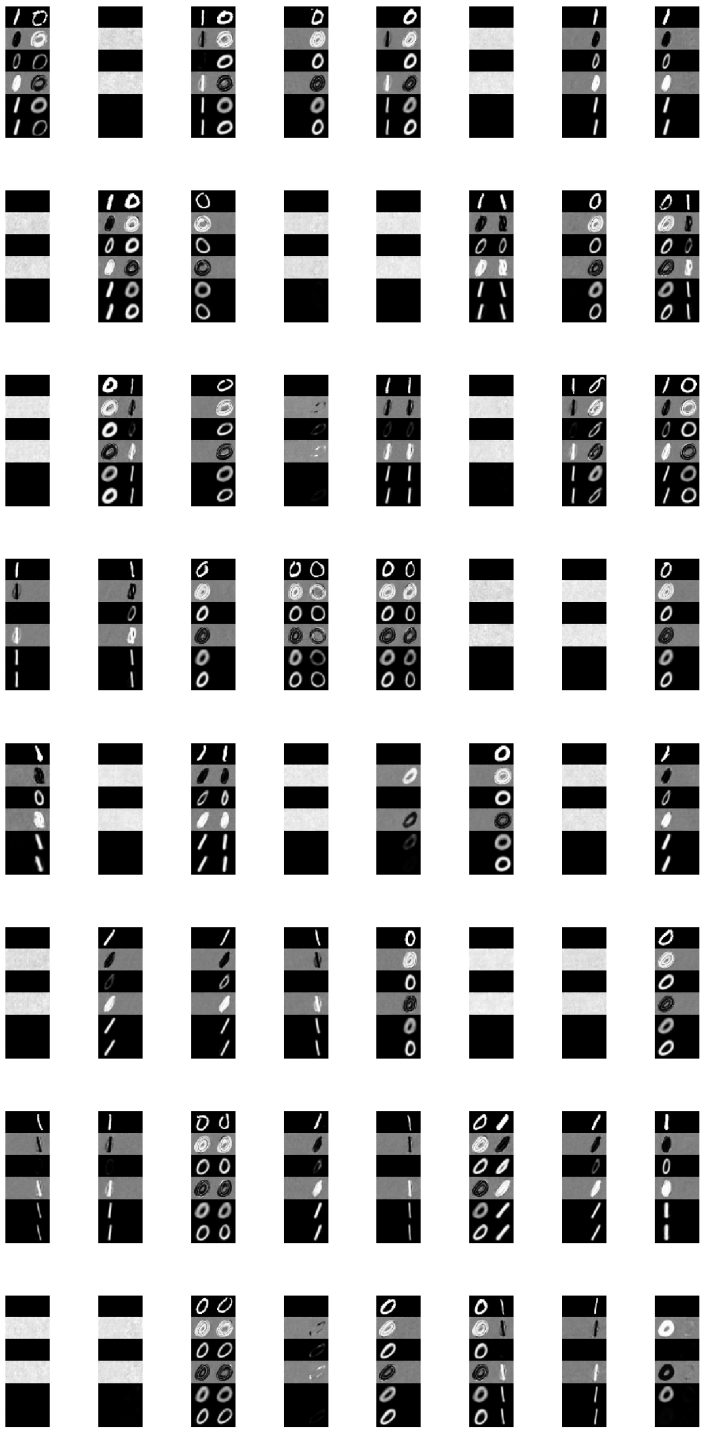
4.1.1 2-MNIST

At the start, we set up only 2 VAE's in parallel, where we want that VAE-0 learns to recognise and model the digit 0, and VAE-1 models digit 1.

In order to accomplish this, we split the training into two stages:

1. In stage 0, VAE-0 trains on reconstructing images containing only the digit 0. The other model is active (so they are contributing to the confidence masks), but its weights are frozen, and only VAE-0 is learning.
2. In stage 1, VAE-1 trains on images containing digits 0 and 1 (so of the two slots, each digit is sampled independently). In this time, VAE-0 is frozen, *however* it is still contributing to the confidence masks. Because of this, places where the digit 0 appears are already assigned high confidence values by VAE-0, and so VAE-1 will not be motivated to learn them. On the other hand, places where digit 1 appears will not be recognised by VAE-0, and so VAE-1 will learn them.

Figure 4.1: Results of training on 2-MNIST. Each picture has 6 rows: first row is input image X, second row is confidence mask of VAE-0 $\hat{m}_0$, third row is reconstruction of VAE-0 $\hat{X}_0$, fourth row is $\hat{m}_1$, fifth row is $\hat{X}_1$, and last row is weighted reconstruction.



As we can see in figure 4.1.1, we have the desired effect: VAE 0 has high confidences (indicated by very white portions of the confidence mask) where the digit 0 appears, and has VAE 1 has high confidences where the digit 1 appears.

We note that there are cases when, even though VAE 0 has low confidence, it still draws a 0. This is because it was trained only on zeroes, so when encountering a digit 1 it does the only thing it knows how to do: output a 0. Similarly, VAE 1 draws very generic zeroes when the input digit is a 0. Since it was trained on both images of 0 and 1, it has had a chance to learn how a generic 0 looks like. Furthermore, it only takes one bit of information to store if the input is a 0 (which is then included in the KL-loss), but it has a lot to gain by making its reproduction closer to the original image (which is the first term in the overall loss function).

After this experiment, we then proceeded to see what would happen with 5 VAE's in parallel.

4.1.2 5-MNIST

For this dataset, we again have the same two digit side-by-side training data. However, we now have 5 VAE's wired up together, where again we would want that VAE i learns to model digit i.

The training is done analogous to 2-MNIST, in 5 stages, where in stage k model k is learning by looking at images containing digits $0, 1, ..., k$, and all of the other models are frozen (but still contributing to the confidences).

From figure 4.1.2, we can see that the model achieves decent reconstructions on the digits which it was trained on, although not as good as we would expect.

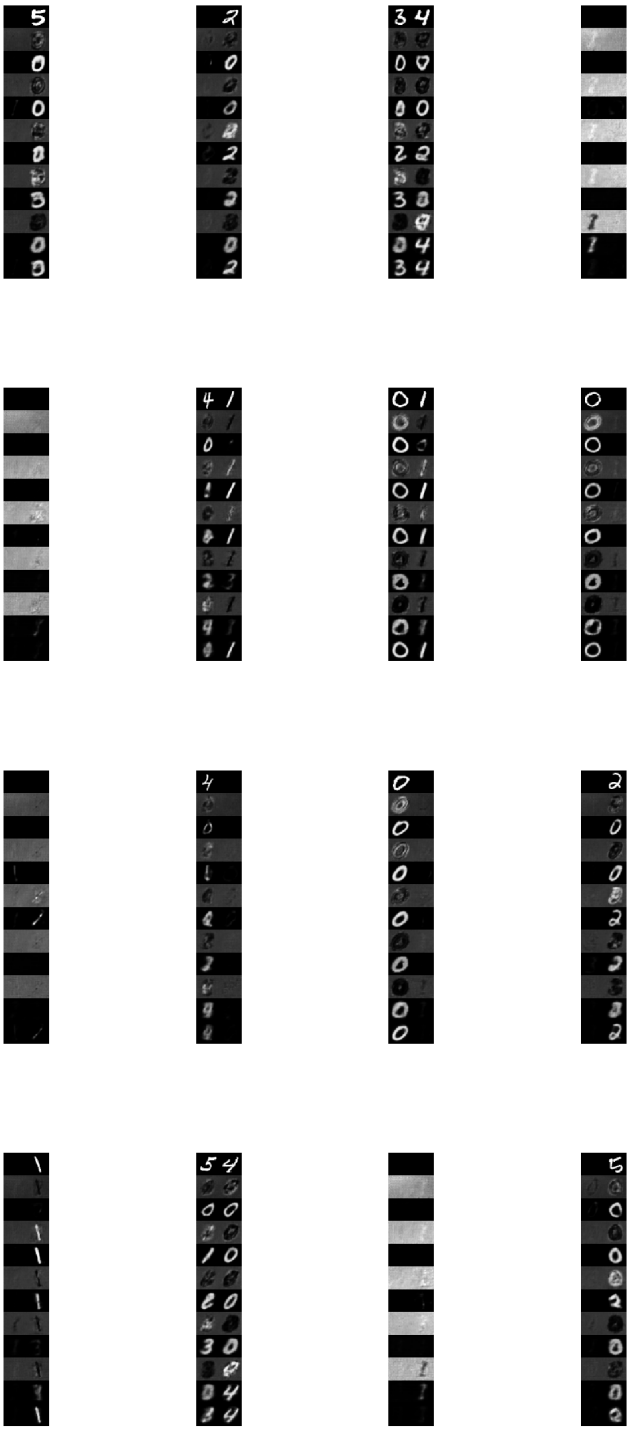
Again, we observe that when a digit 4 appears, only the last model has high confidences, and produces a good reconstruction; the other models generate a digit which they have learned, which minimizes the L_2 loss.

For digit 5, we observe that the model with the closest digit (in L_2 norm) has high confidence: sometimes the model for 2, and sometimes the model for 3.

4.1.3 Together training

Here we introduce "together-training", where we differentiate between two types of training stages: individual-training stages, and together-training stages. Our motivation for this modification is that, afer is that after VAE k has learned to represent the digit k the other models should then learn to not activate in case digit k is in the input.

Figure 4.2: Results of training on 5-MNIST. Each picture has 12 rows: first row is input image X, last row is reconstructed image. In between, each adjacent pair of rows represents the confidences $\hat{m}_k$ and the reconstructions $\hat{X}_k$ for each of the $K = 5$ VAE's. Note that even though the digit 5 appears as part of the input, none of the models were trained on it: this is just test data. We include it only to see how models generalize to unseen digits.



In the previous figures, a model which was trained for digits up to k tends to have low confidence when it receives as input one of the digits 0, ..., $k-1$ (which it has seen while training, and it was already handled by another model). The issue is that when it encounters one of the digits $k+1$, ..., K, it will have some false confidence in its reconstructions, because it never had the chance to learn how to react when it encounters these digits. This will cause chaos in training further models, as when we are at stage K, the model which is training has to overcome the false confidence provided by the first $K-1$ models, so the scale of the raw confidences which it outputs should be on the order of the sum of the confidences of the first $K-1$ models. As such, the more models we train in this fashion, the higher the required raw-confidences become in order to have a non-trivial contribution to the softmax. This is somewhat controlled by the crossentropy loss, but it still represents an instability in the training of models.

Our approach to overcome this is to introduce "together-training", so we differentiate between two types of stages: individual-training stages, and together-training stages. Let us presume that we have K VAE's which we want to train. We then modify the previous training schedule as such:

- VAE 0 trains on the digit 0.
- VAE 1 trains on the digits 0 and 1.
- VAE 0 and VAE 1 train together on the digits 0 and 1, with a lower learning rate.
- ...
- VAE K trains on the digits 0, 1, ..., K.
- VAE's 0, 1, ..., K train together on the digits 0, 1, ..., K.

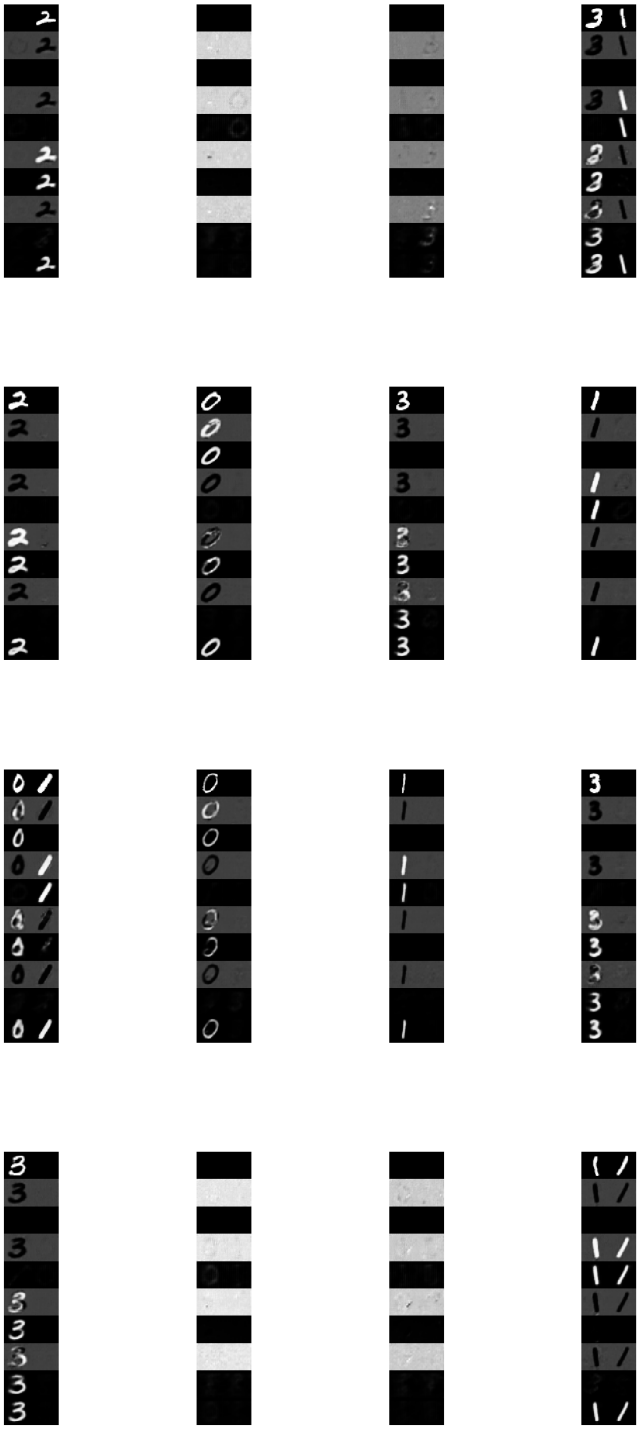
Every other stage is a collective training on a prefix of length k of the VAE's.

4.1.4 KL loss as information

One interesting consequence of the way our model works is that, since each VAE models a separate type of object, we are able to infer if an object of that type is present in a picture by looking at the KL loss of the corresponding VAE.

When generating an empty image, all of the models have a KL-loss close to 0, indicating that they did not need to store any information. However, when a digit that a VAE has learned is present in the input, then its KL loss increases to reflect that, as can be seen in table 4.1.

Figure 4.3: Results of together-training on 4-MNIST. We can see that the VAE's indeed learn to not output anything unless the inpit is their specific digit.



Digits	KL-0	KL-1	KL-2	KL-3
ee	0.63403076	0.6190877	1.4198456	0.480307
00	33.911064	0.7044331	20.23064	0.5375141
11	0.6985266	15.11555	1.8148736	0.84338933
22	1.9036595	0.99384606	30.392303	1.0314183
33	1.6360512	2.186792	28.371572	7.6027184
01	17.265297	7.5015736	11.151404	0.8603339
12	0.8505232	8.492853	14.917055	0.3921642

Table 4.1: KL losses for different scenes. Digit "e" refers to nothing (i.e., empty).

4.1.5 Conclusions from MNIST

When looking at figure 4.1.3, we can see that the model responsible for digit 2 (so the third pair in one picture) also learns digit 3, but not any others. Our explanation for this is that digits 2 and 3 are fairly similar to one another (at least when compared to 0 and 1), and so VAE 2 "competes" with VAE 3 for representing the digit 3. Indeed, this is also supported by the results in table 4.1, where model 2 has high kl-loss both when the digits 2 and 3 are present.

If this hypothesis is correct, then the problem should disappear if the object types (digits, in this case) were to be sufficiently different from one another.

In order to test this, we adapted our model to more complex datasets.

4.2 Fashion MNIST

Fashion MNIST [7] is a modern take on MNIST, where the digits are replaced by clothing items.

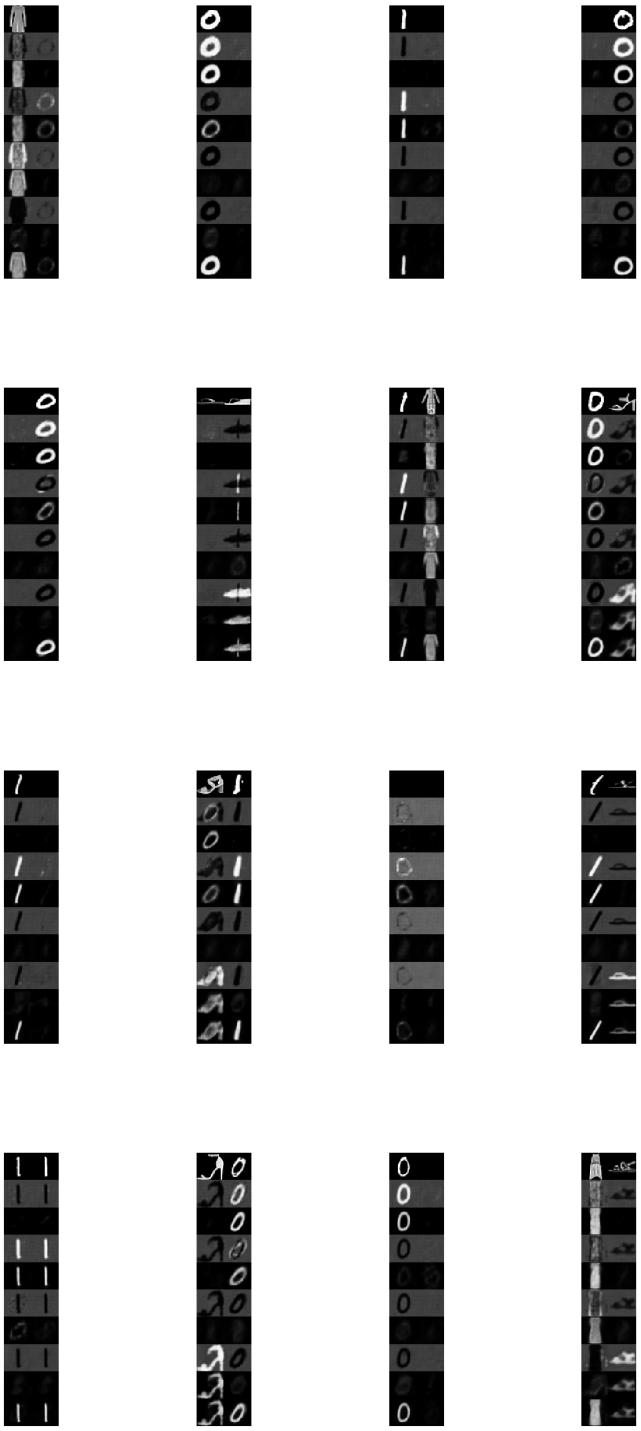
For this problem, we replace pictures of the digit 2 with pictures of dresses, and the digit 3 is replaced with shoes. In this way, all 4 types of objects will be very different from one another.

Looking at figure 4.2, we see that the last VAE is the only one which models shoes, and the VAE for dresses is the only one which handles dresses, which is evidence for our hypothesis being correct.

4.3 Clevr

The Clevr [4] dataset contains renderings of complex 3D scenes, containing objects such as cylinders or spheres. We took pictures which contained only cylinders, cropped and then resized the pictures in order to construct a simplified dataset.

Figure 4.4: Results of together-training on Fashion-MNIST. We can see that each VAE models one object (the one it is confident about).



First, we tested the capacity of a single VAE, to see if it was capable of recreating a single scene. We took the model from Fashion-MNIST and adapted it to the Clevr dataset.

The reconstructions, as can be seen in figure [**?**], are quite unsatisfactory. Training the model to reach this performance also took a significant amount of time.

4.3.1 Spatial Broadcast Decoder

The spatial broadcast decoder (s.b. decoder) [6] is a new technique for VAE's, whereby the deconvolutional decoder is replaced with a s.b. decoder: the latent variables are tiled as to create a cube, which is then passed through normal unstrided convolutions to produce the final image. This decoder has much fewer parameters to learn, and supposedly it also produces better disentangled representations. In papers such as [**?**], they claim the use of this decoder was essential to the results they obtained.

As be seen in figure 4.3.1, the reconstructions are much more accurate.

At last, we decided to make the dataset black and white, with the motivation that since we as humans are able to recongise objects regardless of their color, then this should also be a good starting point for our model.

In figure 4.3.1, the results are yet more accurate than on colored clevr (we also note that for the black and white dataset, more epochs were completed in the same amount of time, since there was 3 times less data movement). We considered these reconstructions satisfactory enough in order to go on to testing the scenario with two VAE's in parallel.

4.3.2 Tandem VAE's on Clevr

Using an idea similar to the MNIST scenario, the digit 0 is replaced with cylinders, and the digit 1 is replaced with spheres.

Training a single VAE to accurately reconstruct a scene takes a really long time, and we supposed that after so many epochs of training a single VAE, it would just have high confidence everywhere, and be good at reconstructing everything (since spheres and cylinders are similar shapes), and so the second VAE would not have the chance to learn anything.

To get around this, we decided to train the model in alternating stages (which we call tandem training):

```
for T steps:
  for E epochs:
    train VAE-0 on cylinders
```

Figure 4.5: Results of one VAE on Clevr. The first row is full white, because the VAE is responsible for modelling everything.

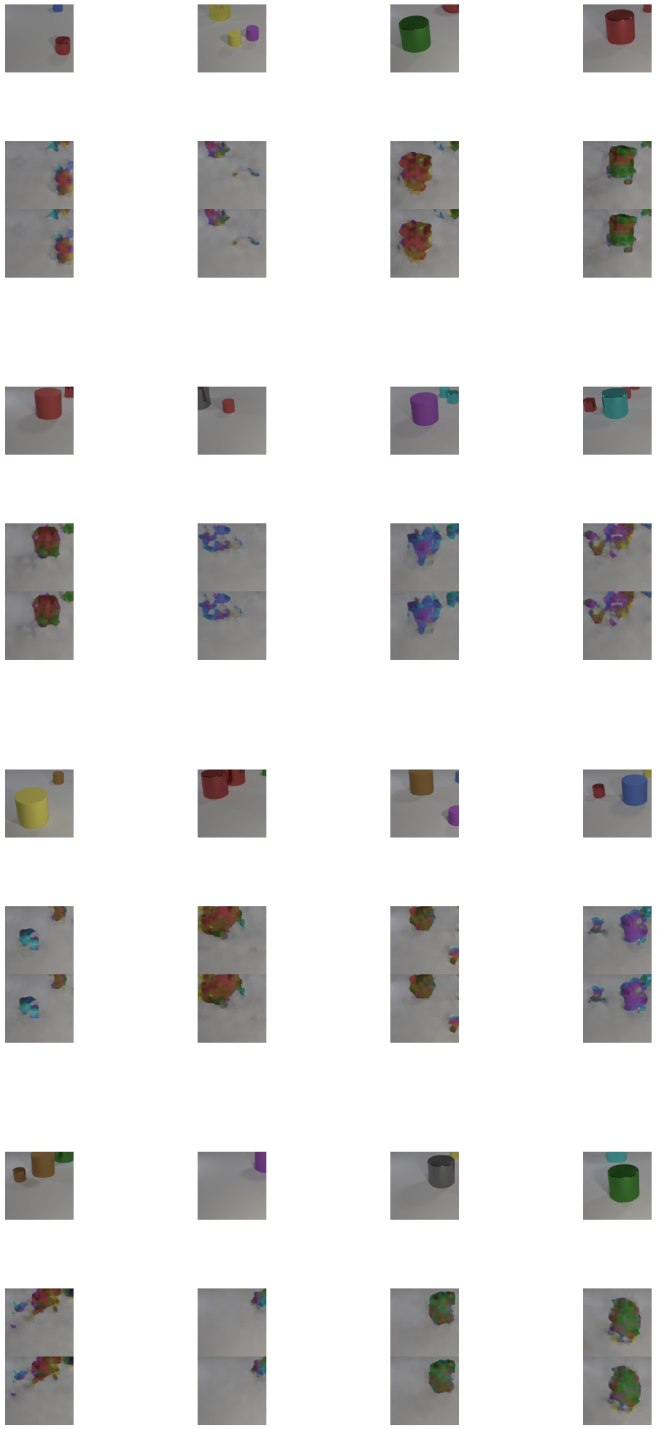
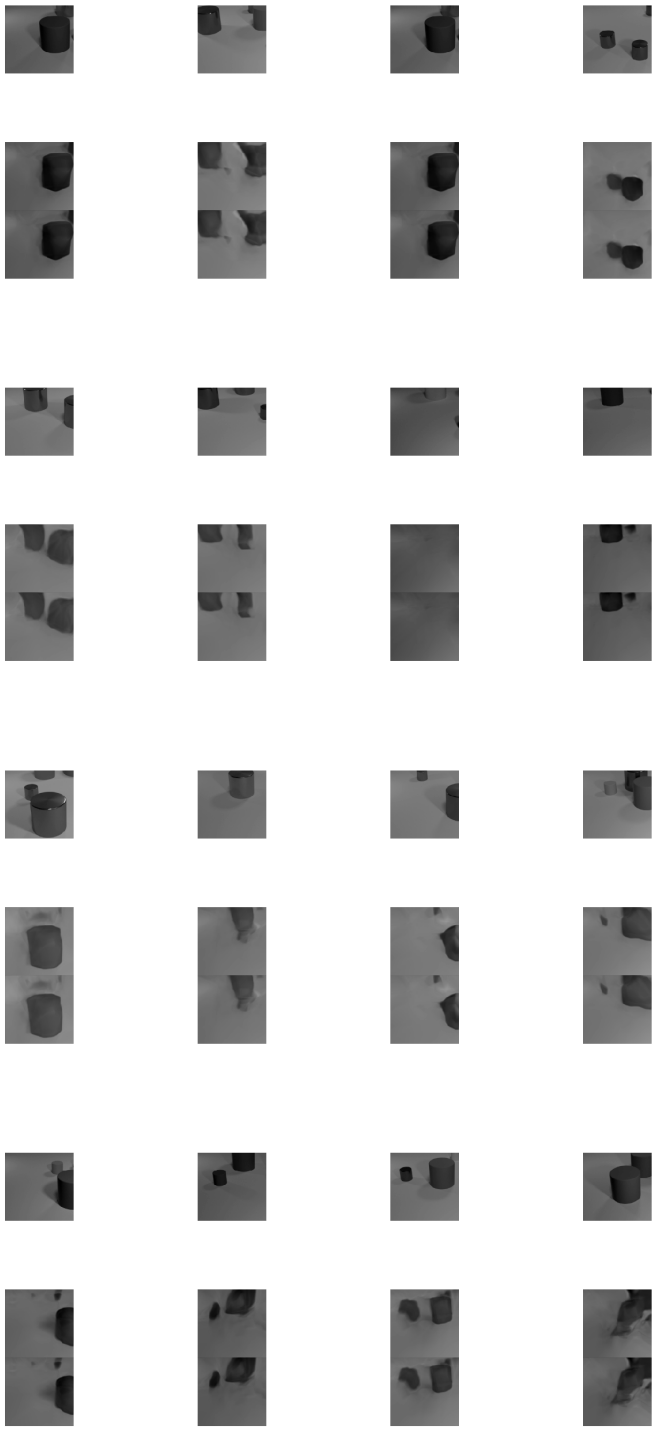


Figure 4.6: Results of one s.b. decoder VAE on Clevr.

Figure 4.7: Results of one s.b. decoder VAE on black & white Clevr.

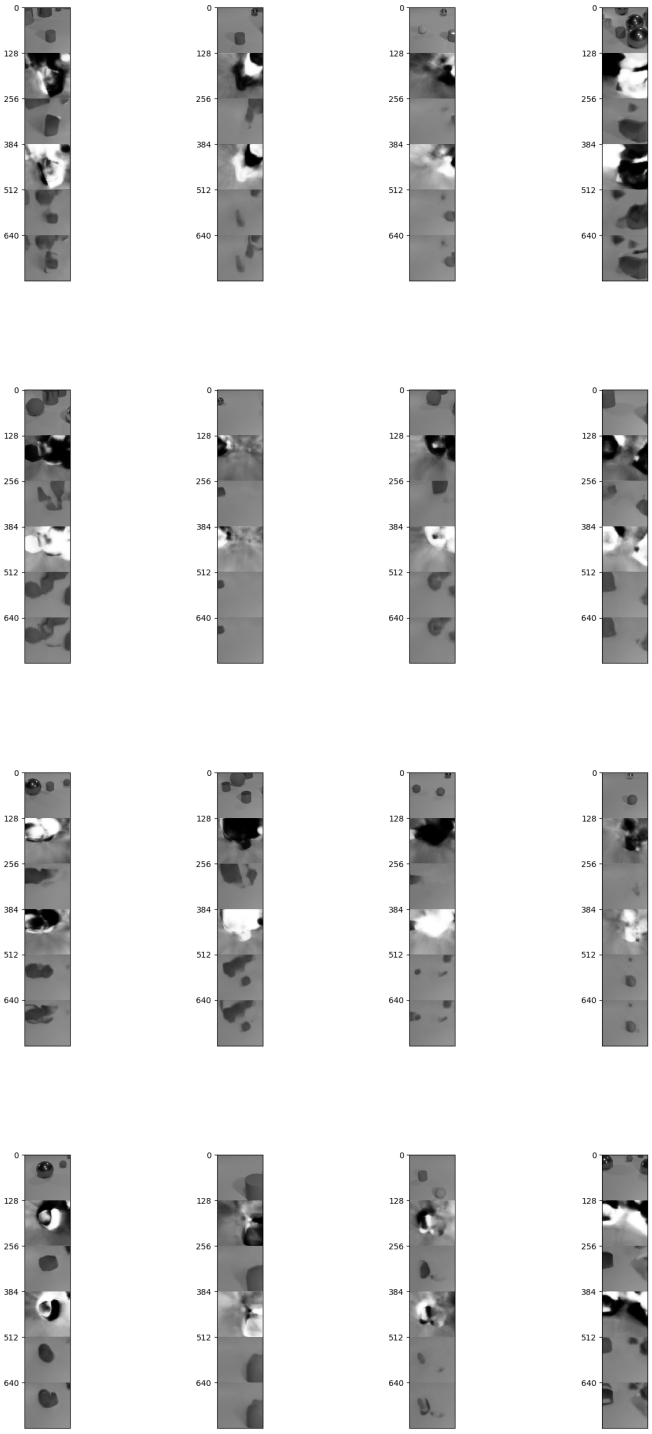


```
for E epochs:
  train VAE-1 on cylinders and spheres
```

Where one epoch means a fixed number of parameter updates.

The results, from figure 4.3.2, are quite unsatisfactory when compared to the Fashion-MNIST dataset. We were not able to figure out if this is caused by the models not being powerful enough to model such a difficult dataset (although we have tried to reproduce the architecture used in [2]). Furthermore, this could also be caused by a bad choice of hyperparameters (stage length or epoch length, for example); or achievieing this sort of disentanglement requires a much more elaborate approach than ours.

Figure 4.8: Results of two s.b.-decoder VAE's trained in tandem on black & white Clevr.



Chapter 5

Conclusions & Future work

We found that our approach was capable of learning disentangled objects in simple datasets such as MNIST, but it did not produce good results on Clevr. It would be interesting to further investigate this failure, and see if the model can be adapted to work.

Furhtermore, we took a supervised approach to learning, based on different stages. We expect that it should be possible to have an unsupervised learning process, where a model's weights are costlier to update the higher its KL loss is (so that after a VAE's has learned an object type, it does not learn anymore, but instead lets other VAE's learn).

Appendix A

Methods

The experiments were performed using the ADAM optimizer, with a learning rate of $1e-3$ for the individual learning stage, and a rateof $1e-4$ for the training together stage.

The hyperparameters were $\beta = 0.5$ and $\gamma = 0.009$.

MNIST

For the MNIST experiments, the architecture consistend of a Convolutional-Deconvolutional autoencoder, with 2 layers on each side. The convolutions had a kernel size of 3, a stride of 2, and were followed by batch normalization and relu activation.

The final convolutional layer would go into a dense fully connected with 64 units and relu activation, and then this dense layer would feed into two other dense fully connected layers, one for the mean and the other for the variance of the latent distribution, with a latent dimension of 32.

CLEVR

For Clevr, we used the Convolutional - Spatial Broadcast Decoder architecture, with 4 convolutional layers with a kernel size of 3 and stride of 2 Each convolutional layer was followed by batch normalization, and then a LeakyReLU with $\alpha = 0.3$. The LeakyReLU was chosen because the normal ReLU version seemed to occasionally stagnate while training. The final convolutional layer would be followed by a dense fully connected with 256 units, and then this would then go into two separate dense layers, one for the mean and one for the variance, with a latent dimension of 128.

For the decoding part, we had a spatial broadcast decoder, followed by 4 convolutiona layers with stride 1.

Bibliography

[1] Christopher P. Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in β-VAE. *arXiv e-prints*, page arXiv:1804.03599, Apr 2018.

[2] Christopher P. Burgess, Loïc Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matthew Botvinick, and Alexander Lerchner. Monet: Unsupervised scene decomposition and representation. *CoRR*, abs/1901.11390, 2019.

[3] Klaus Greff, Raphaël Lopez Kaufmann, Rishabh Kabra, Nick Watters, Christopher Burgess, Daniel Zoran, Loïc Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference. *CoRR*, abs/1903.00450, 2019.

[4] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross B. Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. *CoRR*, abs/1612.06890, 2016.

[5] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv e-prints*, page arXiv:1312.6114, Dec 2013.

[6] Nicholas Watters, Loïc Matthey, Christopher P. Burgess, and Alexander Lerchner. Spatial broadcast decoder: A simple architecture for learning disentangled representations in vaes. *CoRR*, abs/1901.07017, 2019.

[7] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.