Deep Learning for automatic target recognition in synthetic aperture radar images

Nuno Ferreira
nuno.barros.ferreira@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa, Portugal

October 2020

Abstract

Automatic target recognition (ATR) in satellite images is an important application in maritime surveillance. Annually, thousands of refugees lose their lives in the sea; illegal fishing highly contributes to the over-exploitation of fish resources, perturbing ecosystems; besides, other activities, such as drug and firearms trafficking, are conducted through the sea. Hence, it is clear that a good maritime surveillance is key to mitigate these problems.Usually, ATR from satellite images is performed by supervised machine learning algorithms, making use of labels during training. However, the daunting process of labelling images is often expensive and time consuming. On the other hand, unsupervised learning techniques allow the extraction of relevant information from data, without making use of labels, being particularly interesting for the analysis of satellite images, given the huge amount of available data, and its constant increase. This work addresses anomaly detection in synthetic aperture radar (SAR) images, using Variational Autoencoders (VAEs), in an unsupervised manner. In the proposed framework, the encoder of the VAE is trained to map normal images in a latent space. To perform classification of a given set of images, with and without anomalies, these are encoded in a latent space by the encoder, where normal images are clustered together and the anomalous images are spread across the space. Finally, a clustering algorithm is applied to this generated space, being able to identify the anomalous images among the set.

Keywords: Deep learning, Variational autoencoders, Anomaly detection, Unsupervised learning, SAR

1. Introduction

Maritime surveillance is an important activity for many countries. From the early detection of oil spills to the identification of clandestine vessels, the relevance of an effective monitoring of the sea is evident. Annually, thousands of refugees lose their lives in the sea. According to [1], from 2015 to 2018, over 14000 deaths of migrants trying to reach Europe by crossing the Mediterranean were confirmed. Illegal, unreported and unregulated (IUU) fishing highly contributes to the overexploitation of fishing, perturbing ecosystems and fish populations. According to [2], there is a correspondence between the regional estimates of illegal and unreported fishing and the number of depleted stocks in those regions. Several other illegal activities are conducted by sea, such as cocaine trafficking, which, according to the United Nations Office on Drugs and Crime, is trafficked to Europe mostly by sea.

It is self-evident that a good sea monitoring is crucial in the prevention of the aforementioned events and in the control of the impact of their consequences, and it is within this scope that remote sensing image scene classification comes into play.

Traditional target recognition methods in SAR images are based on constant false alarm rate (CFAR) methods, which use a threshold to keep the false alarm rate constant [3]. In these methods, the sea clutter background is modeled according to a suitable distribution and a threshold is set to achieve an assigned probability of false alarm (PFA) [4]. One issue of CFAR algorithms is that they are not able to detect targets with intensity values close to the sea clutter and the threshold computation can represent a time-consuming procedure [4]. Moreover, this methods perform poorly in rough sea conditions. Variations of the CFAR algorithm have also been proposed, such as the bilateral CFAR algorithm [5], which uses a combination of the intensity distribution and the spatial distribution of the SAR images.

Other novel ship detection methods that also rely on the modeling of the clutter and/or the signal backscattered from the ship - such as a detector based on the generalized-likelihood ratio test (GLRT) [4] and the depolarization ratio anomaly detector using dual-polarization SAR images [6] [7] - have been proposed.

In accordance to the emerging paradigm toward data intensive science, machine learning techniques have become increasingly important. In particular, deep learning has proven to be both a major breakthrough and an extremely powerful tool in many fields, namely in the field of computer vision, where deep learning algorithms have managed to equal or even surpass human-level performance in tasks that computers used to struggle when trying to solve them. Object recognition, for instance, a seemingly effortless and rapid task for humans to perform, was very challenging for computers up until the appearance of deep neural networks (DNNs). Prior to the usage of such algorithms, traditional machine learning algorithms for object detection were roughly divided into region selections, such as scale-invariant feature transform (SIFT), and histogram of oriented gradients (HOG), and classifiers, such as support vector machines (SVMs). The beginning of the deep learning booming is often credited to the Image Net's image classification challenge winner in 2012, where a deep convolutional neural network called AlexNet surpassed all other contestants' algorithms, winning the challenge with impressively high accuracy and performance [8].

In the wake of the success of deep learning, in conjunction with the increasing availability of data, deep learning algorithms naturally started to take off in remote sensing. A series of supervised deep learning methods for ship detection as well as for ship classification have been proposed. Among the most successful methods are the you only look once v2 (YOLOv2) [9, 10], the faster region-based convolutional neural network (Faster RCNN) [11, 12, 13] and ResNet [14]. Despite the good results achieved by these methods, they are supervised methods which require large amounts of labelled data to be trained. Since the annotation process is time-consuming and requires domain knowledge from SAR experts, these methods do not take advantage of the huge and increasing amount of accessible SAR data. For instance, the Sentinel satellites, launched in 2014, had already collected about 25 PB of data, as of December 2017 [15].

It is clear that an algorithm able to process SAR images and perform target recognition regardless of annotation is highly important in order to take advantage of this incredible amount of data. And it is upon this reasoning that this thesis is developed.

The goal of this thesis is to develop an unsupervised deep learning framework for anomaly detection in SAR ocean imagery, without resorting to image annotations during the training phase of the algorithm. The main idea is to learn image feature representations through a deep convolutional variational autoencoder, then followed by anomaly detection performed by a clustering algorithm.

It is expected that, after training, the developed model will be able to separate images with anomalies from images without anomalies (normal images), by learning key feature representations inherent to the images that comprise the training data.

2. Materials and methods

2.1. Autoencoders

Autoencoders [16] are feedforward neural networks trained to reconstruct their inputs at the outputs, in an unsupervised manner. Unsupervised learning is the process of training a neural network without using the labels of the training set. An autoencoder network is usually composed of two parts: an encoder and a decoder. The encoder is responsible to map the input data $x \in \mathbb{R}^d_x$ to a hidden space representation $z \in \mathbb{R}^d_z$, through some encoding function, such that $z = f(x)$. The decoder part of the network then maps back from the hidden code z to input space, producing a reconstruction of x, $\hat{x} = g(z)$. A regulariser term can be added to ensure that the model does not overfit the training set, and effectively learns a useful representation of the data.

Typically the hidden code z has a lower dimensionality than the input space x, which forces the autoencoder to learn a compressed representation of the input data. When an autoencoder possesses a hidden code with dimensions smaller than the inputs, it is called undercomplete.

Using a compressed code with lower dimensions can make the autoencoder learn the most salient features of the data. However, if the encoder and decoder of the model are given too much capacity, they can learn how to map the inputs to the outputs, regardless of the dimensionality reduction observed in the network. The same goes to architectures with hidden code dimensions equal to or higher than the input dimensions, cases in which even a linear encoder and a linear decoder could learn to mimic the inputs without learning anything useful. These situations can be avoided by applying regularisation to the autoencoder. Instead of limiting the power of the model by capping the capacity of the encoder and decoder, a regularising term that encourages the network to learn other properties can be added to the loss function, leading the model to optimise the cost taking into account a trade-off between good reconstruction and successful achievement of the goal set by the regularising term. The properties imposed by this term include sparsity of the representation, size of the derivative of the representation, and robustness to noise or missing inputs [17].

Generative models such as the variational autoencoder and the generative stochastic networks, on the other hand, can learn useful encodings without resorting to regularisation, although they can naturally learn high-capacity, overcomplete representations of the inputs. This is made possible because these models are trained to approximately maximise the probability of the training data, instead of copying it.

2.2. Variational autoencoders

As mentioned before, autoencoders are prone to overfitting if they are not regularised, meaning that they might fail when trying to build a meaningful encoded space, because their main goal is to learn the parameters that best reconstruct the input data, regardless of the encoding structure.

Variational autoencoders (VAEs) are built in a way so that they encode the inputs into latent variables, that exhibit a meaningful structure. With an architecture similar to the one of a standard autoencoder, it is composed by an encoder and a decoder. However, contrarily to a standard autoencoder, a VAE does not encode the data into points, encoding it into distributions instead, with a process that works as a form of regularization. The steps for training a VAE are the following:

- Encode input data as distributions over the latent space.
- Sample a point from the latent space.
- Reconstruct the data from the sampled point.
- Backpropagate the reconstruction error over the network in order to update the network parameters.

The VAE is a deep generative model composed of a stochastic encoder and decoder, that models the relationship between the input random variable x and the low-dimensional latent random variable z [18].

The marginal distribution over the inputs x and the latent variables z can be obtained with the joint distribution $p_\theta(x,z)$:

$$p_\theta(x) = \int p_\theta(x,z)dz\,, \tag{1}$$

where $p_\theta(x)$ is an approximation of the true distribution of the data, $p^*(x)$.

When the distributions of a latent variable model $p_\theta(x,z)$ are parameterised by neural networks, the term deep latent variable model (DLVM) is used. One important feature of DLVMs is that the marginal distribution $p_\theta(x)$ can be very complex regardless of the complexity of the conditional distribution in the directed model, allowing good approximations for potentially complicated underlying distributions $p^*(x)$. One downside of DLVMs, however, is the intractability of the marginal probability of data under the model, which makes maximum likelihood learning very difficult. This is due to the lack of an analytical solution or an efficient estimator for the integral in equation 1, which makes it indifferentiable with respect to its parameters. Considering the relation:

$$p_\theta(z|x) = \frac{p_\theta(x,z)}{p_\theta(x)}\,, \tag{2}$$

and the fact that $p_\theta(x,z)$ is efficient to compute, $p_\theta(x)$ could be computed through this relation. However, this is not possible since the posterior $p_\theta(z|x)$ is also intractable in DLVMs.

To address the problem of calculating the posterior, the VAE framework provides a way to efficiently optimize DLVMs along with a corresponding inference model, using an optimiser such as stochastic gradient descent.

The encoder is a parametric inference model with parameters ϕ, trained to learn $q_\phi(z|x)$, an approximation of the intractable true posterior distribution $p_\theta(z|x)$, where ϕ are the parameters of the network (weights and biases) and θ are learned parameters. On the other hand, the decoder is trained to learn an approximation of the posterior distribution $p_\theta(x|z)$.

To evaluate the approximation $q_\phi(z|x)$ of the true posterior, the Kullback-Leibler (KL) divergence $D_{KL}\left(q_\phi(z|x)||p_\theta(z|x)\right)$ is used. Although it cannot be computed directly, it can be minimized by maximizing the sum of the Evidence Lower Bound (ELBO) on the marginal likelihood of the data points x_i. The ELBO for each data point is given by

$$\begin{aligned} ELBO_i = \mathbb{E}_{q_\phi(z|x_i)}\left[log\, p_\theta(x_i|z)\right] - \\ D_{KL}(q_\phi(z|x_i)||p(z))\,, \end{aligned} \tag{3}$$

where $p(z)$ is a prior distribution of z.

The loss function used to train the VAE is then given by

$$\begin{aligned} \mathcal{L} &= -\sum_i ELBO_i \\ &= -\sum_i [\mathbb{E}_{q_\phi(z|x_i)}[log\, p_\theta(x_i|z)] \\ &\quad -D_{KL}(q_\phi(z|x_i)||p(z))]\,, \end{aligned} \tag{4}$$

where the summation is calculated over all images in the training set. The first term in (4) is seen as a reconstruction error between the inputs and outputs, whereas the second term is a regulariser that prevents the network from assigning to each input a distribution in a different region of the latent space.

In order to use an optimiser to optimise the ELBO with respect to the parameters θ and ϕ, one needs to compute its gradients with respect to these parameters. Because the gradient of the ELBO with respect to the parameters ϕ is difficult to obtain, a reparameterisation trick is used. For the reparameterisation trick, the random variable $z \sim q_\phi(z|x)$ is represented as a differentiable and invertible transformation of an introduced random variable ϵ, given z and ϕ:

$$z = g(\epsilon, \phi, x) \tag{5}$$

where ϵ is a random noise sample $\epsilon \sim p(\epsilon)$. As a result, it can be shown that

$$\widetilde{ELBO}_i = log\, p_\theta(x_i, z) - log\, q_\phi(z|x_i) \tag{6}$$

is an estimate of the ELBO of the individual data point [19].

The operations of sampling ϵ, transforming z into $g(\epsilon, \phi, x)$ and calculating $\widetilde{ELBO}_i$ are differentiable with respect to the parameters θ and ϕ, and the resulting gradient is used to optimize the ELBO using SGD or other optimisation algorithm, with minibatches of data.

2.3. β-VAE

β-VAE [20] is a modification introduced to VAEs with the goal of discovering disentangled latent factors among the data. It does so by introducing an hyperparameter β that regularises the contribution of the KL divergence term in the loss function used to train the VAE. The resulting loss function is the following:

$$\begin{aligned} \mathcal{L} &= -\sum_i ELBO_i \\ &= -\sum_i [\mathbb{E}_{q_\phi(z|x_i)}[log\, p_\theta(x_i|z)] \\ &\quad -\beta\, D_{KL}(q_\phi(z|x_i)||p(z))] \,, \end{aligned} \tag{7}$$

Evidently, a $\beta - VAE$ with $\beta = 1$ is a regular VAE. When this parameter is increased above 1, it encourages the model to learn a more efficient latent representation of the data, that is disentangled if it contains some underlying factors of variation that are independent. One resulting caveat is that the model is pushed to focus more on the distributions learnt, creating a trade-off between reconstruction ability and disentanglement quality.

2.4. Convolutional Neural Networks (CNNs)

Convolutional neural networks (CNNs) or convnets are the standard go-to choice for computer vision tasks since 2012 [21], because of their ability to extract meaningful features from images, diminishing the need for feature engineering. In a CNN, the most important layers are the convolutional and the pooling layers. The convolutional layers have the ability to learn local patterns in the images and recognize them anywhere. The pooling layers are designed to reduce the size of the feature maps. By stacking several convolutional and pooling layers interleaved, CNNs can learn spatial hierarchies of patterns. Another key feature of this type of network is that they need fewer training samples to learn representations that have generalization power, when compared to dense feedforward neural networks. Since we wish to process SAR images and extract sufficient information in order to categorize them, CNNs were our choice.

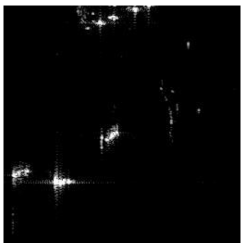
3. Implementation

In this section, the implementation details of our proposed approach are presented. First, the main characteristics of the dataset are discussed, as well as the processing it had to undergo before being used in our experiments. Subsequently, all the details of the process required to achieve the final results, from model training to testing are presented.

3.1. Dataset

The dataset used throughout this work is described in [22], and consists of 43,819 256x256 SAR images, all of them containing one or more ships, extracted from 102 Chinese Gaofen-3 images and 108 Sentinel-1 images, that vary in terms of polarisation, resolution, incidence angle, imaging mode and background complexity.

Figure 1 contains two samples of the aforementioned images, with one example obtained by the Gaofen-3 satellite and one example obtained with the Sentinel-1.


(a) Gaofen-3 image

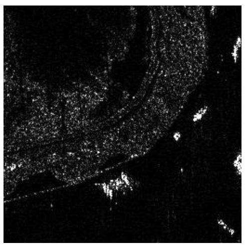

(b) Sentinel-1 image

Figure 1: Images from the original dataset [22].

To build a no-ship image dataset, we randomly cropped portions of these images, all of which contained no ships, which was possible since the ships location and size were labelled. This process resulted in a total of 43,789 images without ships - 42,789 images for the training set and 1,000 images for the test set - all with varying width, height, and aspect ratio. To complete the test set, 500 images with ships were cropped using the ship labels, resulting in a test set composed of 1,500 images: 500 with ships and 1,000 without ships. Figure 2 contains some examples of the resulting dataset.

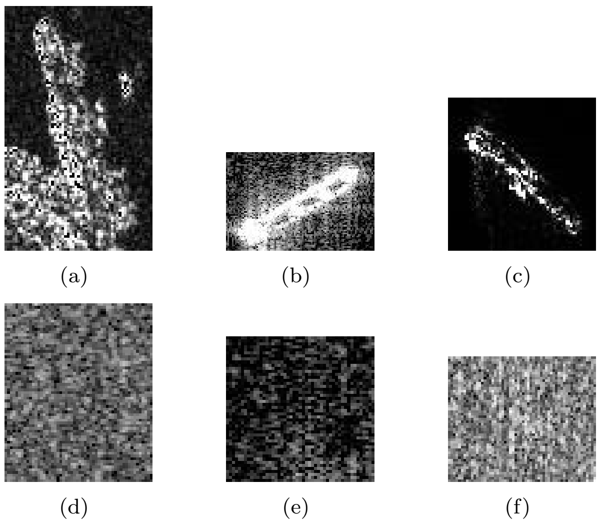


Figure 2: Examples of images with ships (a, b, c), and without ships (d, e, f), from the resulting dataset.

3.2. Representation learning

The first part of the experimental setup consisted in building the VAE network. As stated before, the job of the VAE is to extract features relevant enough so that the resulting low-dimensional latent space for each sample provides sufficient information for the clustering algorithms to assign each image to its respective cluster.

3.2.1 VAE architecture

The chosen architecture for the encoder part of the network is shown in table 1, whereas the architecture for the decoder part is shown in table 2, with each layer's specifications.

Regarding the encoder specifications, Conv2D corresponds to the convolutional layer, whereas MaxPool2D corresponds to the max pooling operation, and d_z is the desired size of the latent space. The outputs of layer 6 are wired to both layers 7 and 8, which output the sets of means and log variances, respectively. Layer 9 is a custom layer (or lambda layer) that performs the reparameterization trick to sample a point z from the latent space, that is assumed to generate the input image, and will feed it to the decoder part of the network. This sampling layer should be regarded as an intermediate layer between the encoder and the decoder, cleverly placed to allow the usage of gradient descent over the network.

After sampling from the latent space, the point z is fed to the decoder, that attempts to reconstruct the original input image based on the sample. Regarding the details of the decoder architecture, Conv2DTranspose layers are convolutional layers that also learn the best way to upsample their inputs, although here the dimensions are kept constant. The upsampling job is conducted by the UpSampling2D layers, using nearest-neighbor upsampling to scale the image up. One should note that for the last layer, the activation function implemented is the sigmoid, since we wish to output an image with pixel values between 0 and 1.

3.2.2 VAE training

The network was trained with the Adam optimiser, with a learning rate of $lr = 3 \times 10^{-4}$ and a mini-batch size of 128. The reconstruction error function chosen was the mean squared error (MSE).

In order to study the influence of varying the penalty associated with the KL divergence term, the network was trained several times with different values of β. In the end, it is expected that the set of different results for each value of β, evaluated on the test set, will allow us to draw some conclusions about the behaviour of the representation learning task in situations in which optimisation focuses more on the reconstruction versus situations in which the optimisation objective is more oriented towards learning a good distribution for the data.

Additionally, for each different value of β, the network was trained with multiple values for the latent space dimensions, in order to analyse how the dimensionality of the generated space influences the results of the model. Table 3 summarises the set of values used in each training routine.

3.3. Anomaly detection

After the aforementioned process, we had our model trained with 35 different configurations of hyperparameters, ready to be tested with the test set of 1500 images with and without ships, and try to cluster them based on the features extracted by the resulting trained encoder. For this task, two clustering methods were used to automatically separate our data into two groups.

3.3.1 Data visualisation

As was shown in previous sections, each input image will generate a set of d_z means and a set of d_z variances, with d_z ranging from 4 to 256. Since visualising each dimension of such high-dimensional data would be infeasible, we opted to employ PCA with two components to the generated space, in order to visualise the resulting data in two dimensions.

3.3.2 Clustering in the generated space

After training, the 1500 images put aside for testing are passed through each trained encoder to generate their corresponding sets of means and variances. Then, K-means clustering with $n_{clusters} = 2$ and GMMs clustering with $n_{components} = 2$ are used to

Index	Type	Kernel size	Filters	Stride	Padding	Activation	Output shape
1	Conv2D	(3,3)	16	1	Same	ReLU	(56,56,16)
2	MaxPool2D	(2,2)	-	2	Valid	Linear	(28,28,16)
3	Conv2D	(3,3)	32	1	Same	ReLU	(28,28,32)
4	MaxPool2D	(2,2)	-	2	Valid	Linear	(14,14,32)
5	Conv2D	(3,3)	64	1	Same	ReLU	(14,14,64)
6	MaxPool2D	(2,2)	-	2	Valid	Linear	(7,7,64)
7	Dense	-	d_z	-	-	Linear	(d_z)
8	Dense	-	d_z	-	-	Linear	(d_z)
9	Custom	-	-	-	-	-	(d_z)

Table 1: Encoder architecture.

Index	Type	Kernel size	Filters	Stride	Padding	Activation	Output shape
1	Dense	-	3136	-	-	ReLU	(3136)
2	Conv2DTranspose	(3,3)	64	1	Same	ReLU	(7,7,64)
3	UpSampling2D	(2,2)	-	2	-	Linear	(14,14,64)
4	Conv2DTranspose	(3,3)	32	1	Same	ReLU	(14,14,32)
5	UpSampling2D	(2,2)	-	2	-	Linear	(28,28,32)
6	Conv2DTranspose	(3,3)	16	1	Same	ReLU	(28,28,16)
7	UpSampling2D	(2,2)	-	2	-	Linear	(56,56,16)
8	Conv2DTranspose	(3,3)	1	1	Same	Sigmoid	(56,56,1)

Table 2: Decoder architecture.

split the data into two clusters. The smallest cluster assigned by these algorithms is categorized as a cluster of images with ships (anomalies), whereas the biggest cluster is labelled as a cluster of images without ships, or normal images.

3.4. Results evaluation

To evaluate the performance of the framework, F1-score and accuracy are used as evaluation metrics to compare the labels assigned by the clustering algorithms with the original labels of the test set. Moreover, a CNN with an architecture similar to the encoder is built and trained with the 42789 images used for training the VAE, plus 58536 images with ships cropped from the original dataset, in order to compare the performance of a supervised model with the proposed unsupervised framework.

4. Results

4.1. Results of anomaly detection on the test set

In this section, we perform an evaluation of the framework in terms of its ability to detect anomalies in the test set. The results obtained by applying the two chosen clustering algorithms to the 35 mean spaces generated by the trained models are presented.

4.1.1 K-means clustering in the approximate posterior mean space

Table 4 presents the accuracy of the models for every pair of β and d_z, using K-means clustering in the approximate posterior mean space. The highest accuracy (lowest error rate) for each value of β is highlighted in bold, whereas the best accuracy values for each value of latent dimensions d_z are underlined.

Inspecting table 4, it is evident that clustering had the most success using the space generated by the model trained with $\beta = 0.3$, outperforming all the other models trained with different values of β in terms of accuracy, with the highest value of all being the one obtained with $d_z = 256$. Increasing β above 0.3 makes the accuracy drop consistently for almost all values of d_z. The effect of decreasing β below 0.3 cannot be determined properly, since no intermediate values between this number and 0 were tested. We can, however, conclude that setting it to 0 has a very negative impact on the classification accuracy, since all models trained with this value of β report the highest error rate, except for the one trained with $d_z = 4$. Further testing could be conducted with values of β within the interval [0, 0.7], since there might be a value in this range that leads to better accuracy results. It is possible to observe that for values of $\beta = \{0.3, 0.7, 1\}$, the

Optimiser	lr	β	d_z	Loss	Batch size
Adam	3×10^{-4}	$\{0, 0.3, 0.7, 1, 2, 4, 10\}$	$\{4, 16, 64, 128, 256\}$	MSE+KL Divergence	128

Table 3: Training hyperparameters.

		Accuracy (%)				
		d_z				
		4	16	64	128	256
β	0	**82.80**	64.67	66.20	63.73	63.40
	0.3	93.80	95.67	95.60	96.20	**96.53**
	0.7	92.67	94.87	95.33	95.60	**95.87**
	1	93.73	93.87	94.60	94.53	**95.07**
	2	92.47	**94.07**	93.13	92.13	92.80
	4	73.20	**92.93**	91.60	90.40	89.60
	10	76.33	76.87	**77.53**	77.33	75.60

Table 4: Accuracy of anomaly detection obtained with K-means using generated mean space.

increase in latent space dimensions promotes an increase in the accuracy, having its highest value for $d_z = 256$. Additional studies could be done with $d_z > 256$, in order to check if there are latent space dimensions that produce better results, since the obtained values suggest that training with higher dimensions would possibly improve accuracy. Since clustering of the mean space generated by the model trained with $\beta = 0.3$ and $d_z = 256$ showed the highest F1-score too, it was considered the best among the 35 tested.

In Figure 3 are depicted two plots of the 2-component PCA for $d_z = 256$ and $\beta = 0.3$, one with the ground truth labels (a) and another with the labels assigned by the K-means algorithm (b). It is noticeable how data points further away from the cluster of images without ships were classified as positive ("contains ship"), whereas data points closer to this cluster were classified as negative ("no ships").

Figure 4 contains a few examples that were mislabelled as not containing ships by the algorithm, that actually contained ships. Inspecting these images, it is hard to identify a clear ship shape, and for that reason the model may have struggled to find enough features to trigger detection.

4.1.2 GMMs clustering in the approximate posterior mean space

To facilitate the reading of this part of the section, we will denote the model trained with $\beta = 0.3$ and $d_z = 256$ by model A, and the model trained with $\beta = 1$ and $d_z = 256$ by model B.

Table 5 presents the accuracy results using GMMs clustering applied to the generated latent space for each parameter pair (β, d_z), with the highest accuracy for each value of β highlighted in bold, and the best accuracy values for each value of latent dimensions d_z underlined.

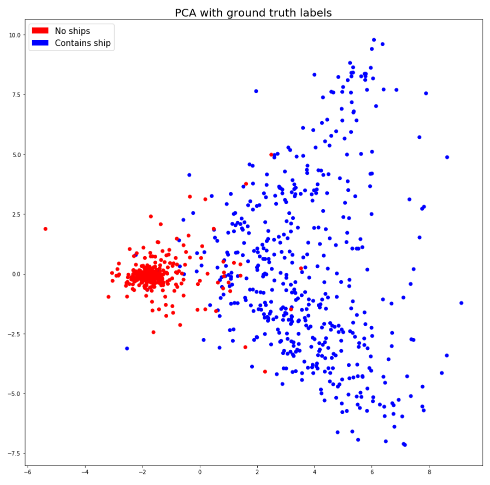
		Accuracy (%)				
		d_z				
		4	16	64	128	256
β	0	**89.07**	86.40	67.00	63.73	63.53
	0.3	77.60	68.80	80.47	86.33	**96.73**
	0.7	64.80	77.67	86.40	89.67	**96.53**
	1	71.60	79.33	88.20	92.27	**96.67**
	2	74.20	84.27	92.07	93.67	92.73
	4	82.80	88.60	93.73	**93.87**	88.93
	10	87.00	89.87	**92.87**	91.60	85.80

Table 5: Accuracy of anomaly detection obtained with GMMs using generated mean space, with the highest value highlighted in bold.

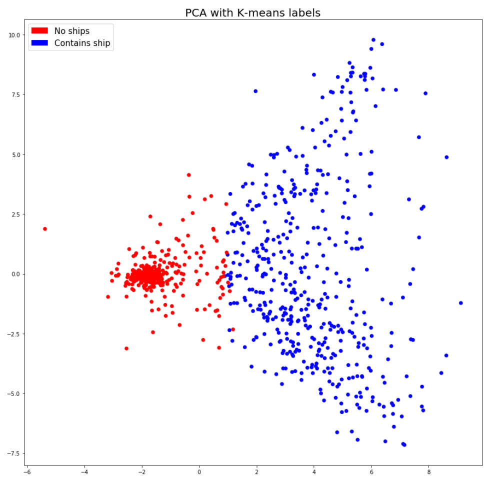
Similarly to what occurred with K-means, the encoded z spaces that showed the best error rate with GMMs were the ones obtained by the models trained with $d_z = 256$ and $\beta = \{0.3, 0.7, 1\}$, with considerably higher accuracy than the others. Furthermore, for these values of β there was the same tendency of increasing the accuracy as we increased the latent space dimensions d_z. The overall top accuracy value was again obtained by applying the clustering algorithm to the mean space generated by model A, reaching the 96.73% mark, a slightly better value than the one obtained by applying K-means.

There was, however, a slight deterioration in the error rates for smaller values of d_z, specially for $d_z = 4$ and $d_z = 16$. Also, as opposed to the previous case, $\beta = 0.3$ does not yield the best accuracy values for all the latent space dimensions.

The tendency of decreasing the accuracy for values of β above 0.3 verified for K-means clearly does not hold for GMMs. Instead, it increases for the majority of the latent space dimensions, except for $d_z = 256$. Namely, for the models trained with latent space dimensions $d_z = \{16, 64, 128\}$, the high-

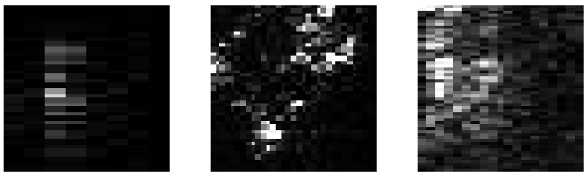


(a) PCA with ground truth labels



(b) PCA with K-means labels

Figure 3: Comparison between ground truth and K-means generated labels for $\beta = 0.3$ and $d_z = 256$.



(a) (b) (c)

Figure 4: Examples of K-means missed detections for the model trained with $\beta = 0.3$ and $d_z = 256$.

est accuracy values emerge when they are trained with values of β of 4 or 10. This suggests that the GMMs algorithm is probably taking better advantage of the disentanglement promoted by these values of β than the K-means algorithm. In addition, the fact that K-means assumes spherical clusters, while GMMs has more flexibility in terms of cluster shape, might justify these results. Nevertheless, these properties did not enable the algorithm to surpass the results obtained by the models trained with β values of 0.3, 0.7 and 1. Although model A has a slightly higher accuracy than model B, this difference is rather small to be used as a criterion of choice to elect the best between the two. Looking at precision and recall values, despite having lower precision (0.9611 as opposed to 0.9808 for model A), model B shows a considerably higher recall value (0.938 as opposed to 0.920 shown by model A), which is important if we want to minimise the number of missed detections. Since the main goal of this framework is to detect anomalies, we consider that trading off some precision (meaning it will produce more false alarms) for better anomaly detectability is very reasonable (considering the magnitude of the difference), so this is the model chosen as being the best when applying GMMs among the 35 models.

The two plots in Figure 5 depict the 2-component PCA applied to the mean space generated by model B, one with the ground truth labels (a) and another with the labels assigned by the GMMs algorithm (b). In Figure 3 (b) (depicting the two-component PCA representation with K-means labels) it was possible to observe that the separation made by the K-means algorithm was heavily expressed along the first dimension (x-axis in the figure) of the PCA representation. In Figure 5, this is not observed so evidently with the GMMs labels. Instead, it appears the model was able to make use of additional features to perform its classification, since it was capable of finding normal images that are further away from the main cluster, and ship images which are closer to it, which might justify its better performance when compared to K-means.

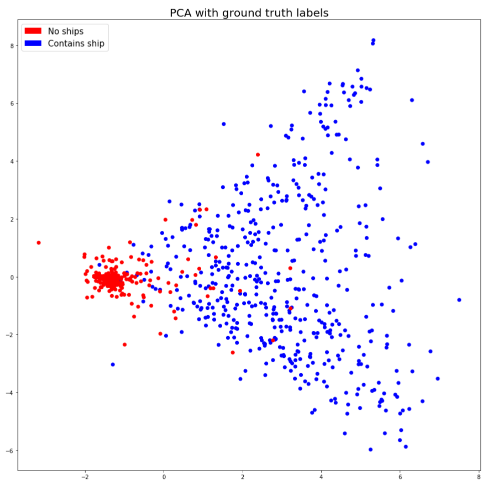
Figure 6 contains 3 examples of ship images with incorrect labels assigned by GMMs applied to the mean space generated by model B. The model might have failed to identify these shapes due to the dimness of the pixels that compose the ships, probably mistaking them with noise.

4.2. Comparison with a supervised CNN

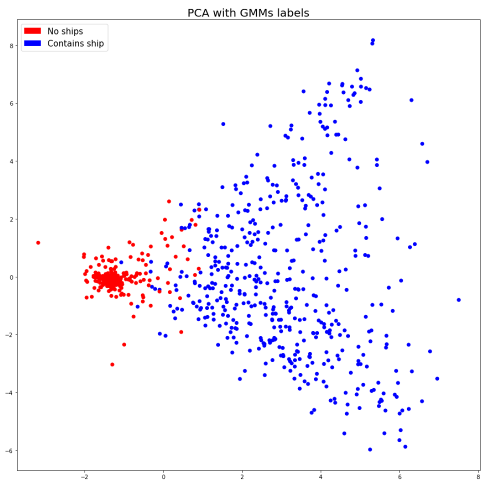
Table 6 shows an overview of the results for the suggested framework and the results of the supervised CNN. As expected, the supervised CNN performs better, but the results obtained by the proposed unsupervised method are very close.

5. Conclusions

The presented thesis aimed at developing a completely unsupervised machine learning framework to detect anomalies in SAR images, resorting to the representation learning ability of variational autoencoders. By the time we engaged in this endeavour, we believed that unsupervised learning tools



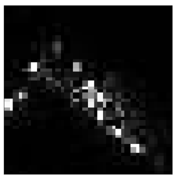
(a) PCA with ground truth labels



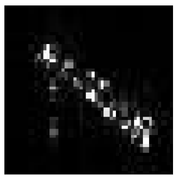
(b) PCA with GMMs labels

Figure 5: Comparison between ground truth and GMMs generated labels for $\beta = 1$ and $d_z = 256$.

(a)



(b)



(c)

Figure 6: Examples of GMMs missed detections for the model trained with $\beta = 1$ and $d_z = 256$.

Table 6: Results of ship detection

Evaluation Metric	**Clustering Model**		**CNN**
	K-means	***GMMs***	
Accuracy	0.9653	**0.9697**	0.9760
F1-score	0.9452	**0.9494**	0.9641

were especially important in the field of remote sensing, namely in the field of SAR image analysis, given the huge and ever-increasing amount of data available. This work came to confirm our belief. Since very few previous works were found on the application of VAEs to SAR images, arriving at the final model was no easy task. A lot of experimenting was conducted, often led by trial and error, in order to achieve satisfying results. Nonetheless, the final model achieved very interesting results, with a performance very close to the one of a supervised CNN, so it is fair to conclude that the proposed objective for this thesis was reached with success.

Despite the good results obtained, there are a few things that should be considered for possible future developments. As previously mentioned, the network should be trained with $d_z > 256$, in order to check for possible improvements. Regarding the applicability of the solution to real case scenarios, further testing should be conducted to evaluate other performance metrics, such as the time needed to classify each image, in order to perform a benchmark with current state of the art methods, and assess the feasibility of applying the framework in a live surveillance system. This would also require a study of possible optimisations to be applied to the classification pipeline, in order to achieve minimum classification time. This work focused on analysing the generated mean spaces of each trained model. Future developments could try to also take advantage of the generated sets of variances, that could possibly contain useful information to help the model perform even better. Further studies could also include training the model using the triplet loss [23], and the usage of adversarial VAEs [24]. Finally, it would be interesting to train and test the network in other scenarios, such as in land surveillance applications, in order to analyse how the model would behave in those conditions.

Acknowledgements

This work was supported by Portuguese funds through FCT (Fundação para a Ciência e Tecnologia) under LARSyS - FCT Project UIDB/50009/2020, and PT2020 project 24534 - INFANTE.

References

[1] UNHCR. *Desperate Journeys.* Online, 2019. Available: https://www.unhcr.org/desperatejourneys/.

[2] David J. Agnew, John Pearce, Ganapathiraju Pramod, Tom Peatman, Reg Watson, John R. Beddington, and Tony J. Pitcher. Estimating

the worldwide extent of illegal fishing. *PLoS ONE*, 2009.

[3] David Crisp. The state-of-art in ship detection in synthetic aperture radar imagery. *DSTO Information Science Laboratory*, 2004.

[4] Pasquale Iervolino and Raffaella Guida. A novel ship detector based on the generalized-likelihood ratio test for SAR imagery. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(8):3616–3630, 2017.

[5] X. Leng, K. Ji, K. Yang, and H. Zou. A bilateral CFAR algorithm for ship detection in SAR images. *IEEE Geoscience and Remote Sensing Letters*, 12(7):1536–1540, 2015.

[6] A. Marino, W. Dierking, and C. Wesche. A depolarization ratio anomaly detector to identify icebergs in sea ice using dual-polarization SAR images. *IEEE Transactions on Geoscience and Remote Sensing*, 54(9):5602–5615, 2016.

[7] A. Marino and P. Iervolino. Ship detection with cosmo-skymed pingpong data using the dual-pol ratio anomaly detector. In *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2017.

[8] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 2012.

[9] H. Khan and C. Yunze. Ship detection in SAR image using YOLOv2. In *Proceedings of the 37th Chinese Control Conference*, 2018.

[10] Yang-Lang Chang, Amare Anagaw, Lena Chang, Yi Chun Wang, Chih-Yu Hsiao, and Wei-Hong Lee. Ship detection based on YOLOv2 for SAR imagery. *Remote Sensing*, 11(7):786, 2019.

[11] Jianwei Li, Changwen Qu, and Jjjj Sun. Ship detection in SAR images based on an improved faster R-CNN. In *SAR in Big Data Era: Models, Methods & Applications*, 2017.

[12] Miao Kang, Xiangguang Leng, Zhao Lin, and Ke Ji. A modified faster R-CNN based on CFAR algorithm for SAR ship detection. In *International Workshop on Remote Sensing with Intelligent Processing*, 2017.

[13] J. Zhao, Z. Zhang, W. Yu, and T. Truong. A cascade coupled convolutional neural network guided visual attention method for ship detection from SAR images. *IEEE Access*, 6:50693–50708, 2018.

[14] Y. Li, Z. Ding, C. Zhang, Y. Wang, and J. Chen. SAR ship detection based on Resnet and transfer learning. In *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*, 2019.

[15] X. X. Zhu, D. Tuia, L. Mou, G. Xia, L. Zhang, F. Xu, and F. Fraundorfer. Deep learning in remote sensing: A comprehensive review and list of resources. *IEEE Geoscience and Remote Sensing Magazine*, 5(4):8–36, 2017.

[16] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[17] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[18] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[19] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 2019.

[20] I. Higgins, Loïc Matthey, A. Pal, C. Burgess, Xavier Glorot, M. Botvinick, S. Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.

[21] François Chollet et al. Keras. https://keras.io, 2015.

[22] Yuanyuan Wang, Chao Wang, Hong Zhang, Yingbo Dong, and Sisi Wei. A SAR dataset of ship detection for deep learning under complex backgrounds. *Remote Sensing*, 11(7):765, 2019.

[23] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(2), 2009.

[24] Lars M. Mescheder, Sebastian Nowozin, and Andreas Geiger. Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. *CoRR*, 2017.