University of Pisa

Master's Degree in Computer Engineering: Enterprise System

Design and Development of a Visual Anomaly Detection System Based on Attention

July, 2021

Supervisors
Claudio Gennaro
Fabrizio Falchi
Nicola Messina

Author
Filippo Scotto

Advisor
Fabio Valerio Massoli

Abstract

Transformer (ViT), they have started to gain popularity also in the field of computer vision for what concerns image classification. In this thesis, a visual anomaly detection system based on attention is proposed. Generally, in literature when dealing with visual anomaly detection, the Convolutional Neural Networks (CNNs) represent the dominant approach and they are at the present day the state-of-the-art. This is mostly due to their intrinsic ability to extract local spatial information from images. To obtain the same amount of knowledge, Vision Transformers are required to see much more data examples.

In the first part of this work, I focused on the feasibility of a visual anomaly detection system based on ViTs when dealing with small data sets. In particular, a hybrid solution composed of a Vision Transformer and a convolutional decoder was tested on the *UCSD Pedestrian Anomaly Detection Dataset*.

In the second part of the thesis, I proposed a novelty anomaly detection method built on the features extracted using a self-supervised pre-trained Vision Transformer. The ViT was not trained to recognize the video frames coming from the UCSD Dataset nor for anomaly detection in general, however, it showed impressive capacity at recognizing patterns in data helpful to achieve a satisfying detection. The proposed architecture, achieved good results on the UCSD PED2 Anomaly Detection Dataset, even if it failed to outperform the state-of-the-art. The outcome is encouraging considering the nature of the approach and how new these technologies are. CNNs have been using for more than a decade, they are a fully mature technology, whereas Vision Transformers have been using only for several months.

Keywords: *computer vision, video anomaly detection, attention mechanisms, transformers, vision transformers*

Contents

1

Introduction

Introduced in the 1960s, *Computer Vision* is an interdisciplinary scientific field that aims to understand and reproduce tasks that the human visual system can do: automatic extraction, analysis and understanding of useful information from a single image or a sequence of images. In the last decade with the introduction of deep learning and *Convolutional Neural Networks (CNNs)*, the huge amount of data and the availability of hardware resources (Graphics Processor Units (GPUs)), a significant breakthrough has been observed in the field of Computer Vision which gained a lot of popularity becoming nowadays one of the most important application for Artificial Intelligence (AI) and Machine Learning (ML); countless applications strongly relies on intelligent video analysis such as: self-driving cars, manufacturing production line management, cancer detection.

Machine Learning is about developing a computer program that is said to learn from experience with respect to some task and some performance measure; if its performance improves with experience then we say that it has learnt the task. Major issues with traditional machine learning come from the necessity for the programmer to handcraft the optimum features required to feed the model. This is both a time demanding and knowledge-hungry activity especially for computer vision in which it is required to deal with complex data such as images and videos. When Krizhevsky et al. (2012) published their insightful results, the old machine learning paradigm completely changes in favour of deep learning. The approach described in Krizhevsky et al. (2012) not only uses automatically discovered representation of the data, but it also performs dramatically better than what used to be the state-of-the-art in image classification. It was a real milestone that started the golden age of CNNs that became in the following years the state-of-the-art for all computer vision tasks. To understand why Deep Learning was so successful for computer vision we just need to read the definition of Deep Learning

given by LeCun et al. (2015):

> it is a subset of representation learning with multiple levels of representation composing simple but not linear modules that each transform the representation at a higher, slightly more abstract level

we can easily find a lot of analogies with biological vision, in fact, the first CNN [Lecun et al. (1998)] was deeply inspired by animals' visual systems.

In Vaswani et al. (2017) was introduced an architecture based on the self-attention mechanism for Natural Language Processing (NLP) which obtained immediate success due to the great performance achieved for the language translation task. The new architecture, named *Transformer*, turned out to provide a strong mechanism for discovering patterns in sequences of data. The Transformer was with respect to the state-of-the-art a simple architecture, more parallelizable and capable to achieve better performance with less time to train.

In Dosovitskiy et al. (2020) a further step forward has been done with the introduction of the *ViT*: a transformer-based model for image recognition; the paper shows that reliance on CNNs is not necessary for image classification to achieve state-of-the-art results. This new architecture was acclaimed with great enthusiasm by researchers and the scientific community and many works have been published in the last few months all claiming to have achieved impressive results. In this work, ViTs are analyzed in order to find out if it is possible to use them to perform Anomaly Detection in videos and achieve state-of-the-art performances. For further details on the techniques that were quickly introduced in this chapter, please read the chapter 2.

1.1 | Anomaly Detection

Anomaly Detection (AD) is the process of identifying unexpected events which differ from the normal behaviour. When we talk about normality in the means of Anomaly Detection, we simply refer to "how the majority of samples behaves", which is unrelated to the Gaussian definition of normal. It is a crucial task in many application domains (security cameras, medicine, industry), due to the fact that untreated anomalies often produces critical consequences. In medicine an anomalous pattern in bio-image might indicate the presence of cancer; anomalies in credit-card transactions may indicate a fraud; in a self-driving car detecting an anomaly can mean a difference between life and death. In literature there can be found several strategies to treat AD: unsupervised learning, supervised learning, semi-supervised learning. In *unsupervised learning*, generally, the most adapted, the entries in the data set are not labelled, but it is assumed that

normal data is the majority, and anomalous entries are just outliers. In *supervised learning* instead, we model is fed using labelled data, this approach is discouraged in literature [Chandola et al. (2007)], mostly because it is hard to provide a good representation of what is anomalous, models trained using this strategy fail to recognize anomalies that differ too much from what was given as input during training time. *Semi-supervised* learning is about giving no knowledge to the model about what is an anomaly, we just try to teach the model to recognize what is normal; during test time, if the model is not able to recognize something as normal, it classifies it as anomalous. An important aspect for any AD technique is how the anomalies are reported, this can be performed in one of the following two ways: by assigning an *anomaly score* to each instance in the test set and then with a threshold it is possible to discriminate between normal data and anomalous data; by assigning a *label* to each entry and then performing a classification task.

With respect to classification-based AD, choices are multi-class classification or one-class classification whether if there are multiple classes for normal data or not. Such techniques learn a discriminating boundary around normal data using algorithms like Support-Vector Machines (SVMs) [Manevitz and Yousef (2002)]. All the test instances that fall out of the boundary are declared anomalous. One-Class Classification is particularly interesting because it required to have only one label for the training data, the normal class. This makes this approach suitable for imbalanced data sets and it has become very popular in Computer Vision [Chandola et al. (2007)]

Several factors make AD very challenging, such as the boundary between normal and anomalous behaviour is not precise; sometimes if the anomaly is the result of malicious action, it is masked in order to make it look as much normal as possible; there is no definition of an anomaly, there is not a singular well-defined behaviour; information about what is anomalous is extremely limited (this is also a reason to discard supervised learning); normal data can suffer from noise that makes them look anomalous. Due to these challenges, it is safe to state that in its most general form, Anomaly Detection is not an easy task.

1.1.1 | Video Anomaly Detection

In this work, we will focus in particular on *Video Anomaly Detection*. Anomaly detection with videos is generally achieved with extensive use of CNNs given their great effectiveness with images. Unfortunately those solutions are also pretty expensive in terms of computational costs [Suarez and Jr. (2020)]. Three main routes can be pursued in order to perform Video Anomaly Detection:

- *Frame reconstruction*: the basic assumption here is that reconstruction error for normal samples would be lower because they should be similar to training data. When dealing with frame reconstruction, Autoencoders [Baldi (2011)] are quite popular literature [Manevitz and Yousef (2002), Hasan et al. (2016), Ribeiro et al. (2018)] due to their capability to encode the input into a more compact representation and finally to recreate the input from the representation using prior knowledge from training data.

- *Future frame prediction*: in this approach, a model is fed with a sequence of frames and it is trained to predict the frame which is immediately after the sequence. When the prediction significantly differs from the actual frame that may be the sign of the presence of anomalies. Some examples are Liu et al. (2018), Ravanbakhsh et al. (2017) and Park et al. (2020b).

- *Deep features*: deep neural networks are used to extract significant features that are then forwarded to a classifier (for example a Support-Vector Machine, a Multi-Layer Perceptron or a K-Nearest Neighbours Classifier). A deep feature is a representation of a portion of a video and encodes both spatial and temporal information. The idea is to exploit the fact that deep features associated with anomalous portions of a video differ from those associated with normality. Some examples are: Ionescu et al. (2018), Abati et al. (2019), Sabokrou et al. (2017).

1.1.2 | Metrics for Anomaly Detection

In this work we will consider anomaly detection as a task of binary classification this allows evaluating the considered models using metrics that are well known in machine learning literature such as ROC and AUC. *Receiver Operating Characteristic (ROC)* is a graphical plot that shows the ability of a binary classifier to correctly distinguish between the two classes in the function of how the discrimination threshold is varied. This method was originally introduced for operators of radar receivers hence the name. The curve is obtained by plotting the true-positive rate against the false-positive rate while changing the threshold. The closer the curve is to the top-left corner the better the screening ability of the test, the closer it is to the first bisector the worst the classifier is (a ROC aligned to the first bisector means a totally random classifier).

On the other hand, *Area Under the Curve (AUC)* as the name suggest is the area under the ROC curve, it is the probability that the model ranks a random positive example more highly than a random negative example. Its value ranges in value from 0 to 1, the

value for a random classifier is 0.5. AUC is scale-invariant and most importantly it is threshold-invariant

1.2 | Motivation

Vision Transformers (ViTs) have gained a lot of popularity around the computer vision community, various recent works in the literature suggest that there are reasons to believe they may work well with Video Anomaly Detection as well. In Naseer et al. (2021), the authors investigate the properties of ViTs which makes them so impressive in performing Computer Vision tasks. The study shows that ViTs are highly robust to severe occlusions, perturbation and domain shifts. They are also less biased toward local features with respect to CNNs. When properly trained Vision Transformers show shape recognition capability comparable to that of the human visual system. The authors also analyze the effectiveness of the representations encoded by ViTs for what concerns classification accuracy. Most of the reasons behind the success of ViTs over CNNs are due to the differences between the operation of convolution and the self-attention mechanism: convolutions excel at learning local interactions, whereas self-attention effectively learns global interactions [Hu et al. (2019), Ramachandran et al. (2019b)]. Further details on the self-attention mechanism will be provided in chapter 2.

Recent works such as Mishra et al. (2021b) show that ViTs are capable to achieve state-of-the-art results also for what concerns Image Anomaly Detection. In conclusion, all of the above arguments are strong enough to justify the work described in this thesis on the use of Vision Transformers (ViTs) in Video Anomaly Detection.

1.3 | Contributions

The work done in this thesis consists of the following contributions in the field of video anomaly detection:

- a study on the feasibility of Transformers Based solution for small data sets with a hybrid Attention-Convolutional proposal;
- a proposal of a novelty anomaly detection system that operates on the features extracted using a self-supervised Vision Transformer;
- a One-Class Classification method that performs anomaly detection using the features extracted by the same system mentioned in the previous contribution.

1.4 | Thesis Structure

- In Chapter 2 will be described in detail some of the technologies and techniques that were introduced in the introduction. In particular, it will be focused on CNNs, Auto Encoders, Recurrent Neural Networks, Transformers, ViTs and the *MNAD*: a work based on Convolutional Neural Networks that is at the time of writing the state-of-the-art for anomaly detection in videos.
- In Chapter 3 will be proposed a variation of the MNAD network that uses a ViT instead of a Convolutional Encoder. At the end of the chapter, it will be discussed the feasibility of Transformer-based architecture in small datasets.
- In Chapter 4 will be described an architecture for Self-Supervised Vision Transformers and it will be shown a study on the quality of the features extracted using such architecture. It will be proposed an anomaly detection system that operates on those extracted features. Finally, in the last part of the chapter, it will be described a one-class classification anomaly detection method applied to the same features.

2

Background & Literature Overview

In this chapter is reported an overview of the techniques adopted in this work, with particular focus on Vision Transformers and the Memory-guided Normality for Anomaly Detection (MNAD) method introduced by Park et al. (2020b).

2.1 | Convolutional Neural Networks

Introduced for the first time by Lecun et al. (1998), *Convolutional Neural Networks (CNNs)* are one of the most significant networks in deep learning. Since their first appearance, they had allowed people to accomplish brilliant achievements such as face recognition, autonomous vehicles and intelligent medical treatment. They become very popular in the field of Computer Vision after that Krizhevsky et al. (2012) achieved the best classification result using a network with extensive use of convolution (*AlexNet*). Such impressive results were possible because CNNs leverage three important ideas that can help improve a machine learning system:

- sparse interactions: each neuron is no longer connected to all neurons of the previous layer contrary to what happens in in Multi-Layer Perceptrons (MLPs)
- parameter sharing: using the same weights for more than one function in the model, thus reducing the number of parameters
- down-sampling dimensionality reduction: the input is down-sampled reducing the amount of data.

To build a CNN-based classifier three components are typically needed: the convolution layer, the pooling layer (the down-sampling layer) and the fully connected layer.

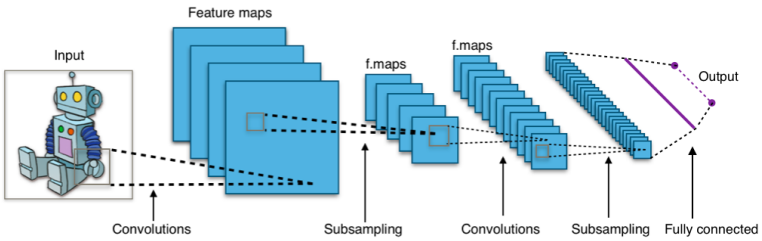

Figure 2.1: Typical architecture of a Convolutional Neural Network (CNN)[1].

The *convolution* operation is the most important aspect of CNNs, it is a mathematical operation that requires to perform a dot product between the input data and a *convolution kernel matrix* (also known as *filter*). The kernel is smaller than the input matrix, and it slides along the input data, at the end it generates a *feature map* which will form the input for the next layer. In between convolution layers there may be a pooling layer which will reduce the dimensions of data by applying an aggregating function (usually max or avg) to small clusters of the features map. At the end of this process of multiple convolution-pooling, there is a simple classifier made with a set of fully connected layers to condensate all the data into a finite number of neurons each identifying a class of the data set. This is the basic mechanism behind a CNN, it was deeply inspired by the biological visual system, and it can also be expanded with more complex operations (for example by adapting several filters at each convolution layer in order to let the network focus on different aspects of the input data), it just depends on the application for the network.

2.2 | Autoencoders

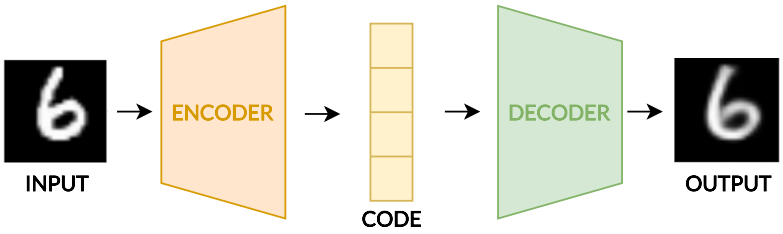

Figure 2.2: An Autoencoder example

An *Autoencoder* is specific type of neural network designed to encode the input into

[1] *Aphex34, CC BY-SA 4.0,* `https://commons.wikimedia.org/wiki/File:Typical_cnn.png`

a compressed meaningful representation (also known as *code*), which is then decoded to reconstruct the initial input data. The Autoencoder was introduced for the first time in Rumelhart and McClelland (1987), with the intention to build a network capable to obtain an informative representation without supervision, to be then used in tasks such as clustering.

An Autoencoder is typically built using an Encoder, a network that maps the input into a latent space, and a Decoder, which recreates the input from the information in the latent space. There is really no constraint on how these two networks should be made, they could be linear MLPs, CNNs or any other architecture. Their unsupervised nature make them particularly indicated to deal with Anomaly Detection; more precisely Convolutional Autoencoder are quite popular for what concerns Anomaly Detection in videos [Ribeiro et al. (2018)]. In such networks the encoder and the decoder are built using CNNs.

2.3 | Recurrent Neural Networks

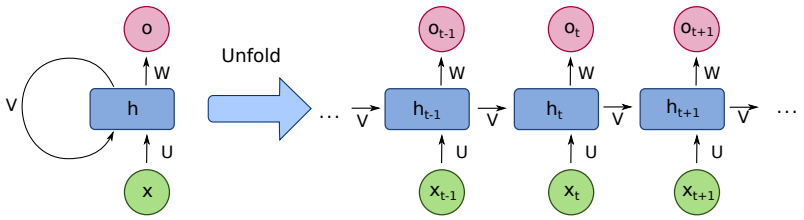

Figure 2.3: A basic Recurrent Neural Network (RNN)[2].

A Recurrent Neural Network (RNN) is a category of Neural Networks that allows to manage temporal dynamic behaviours of data due to the presence of an internal state (memory). For this reason, RNNs have been widely adopted in research areas concerned with sequential data, such as text, audio, and video. The typical feature of a RNN is the presence of cyclic connections (feedback connections) which enables the network to store a local state and to update it; the state is stored using a a set of recurrent cells, where each one has an hidden state connected to the previous and to the next hidden states. These networks have achieved great results, however they failed to keep

[2] *fdeloche, CC BY-SA 4.0, https://commons.wikimedia.org/wiki/File:Recurrent_neural_network_unfold.svg*

a significant state when the input data is relevant for too much time, in other words they are not able to handle long-term dependencies. More precisely, the problems rise when applying back-propagation to RNNs, in fact it may happens that either the gradients when back-propagated go to zero (*vanish*) or to saturation (*explode*). In practice, this means that the standard RNN fails to learn in presence of time lags greater than 5-10 discrete time-steps between relevant input events and target signals [Gers et al. (1999)].

2.3.1 | LSTM

RNNs have problems with long-term dependencies, to solve the problems related to the vanishing gradients, Hochreiter and Schmidhuber (1997) proposed the Long Short-Term Memory (LSTM). In this new architecture, the remembering capacity of the standard RNN cell was improved by introducing *multiplicative gates* into the cell which learn to open and close access to the cell. The original LSTM included two multiplicative gates, one at input and one at output. The input gate is capable to block irrelevant data to get inside the cell, similarly the output gate blocks not significant data to get out from the cell.

The problem with standard LSTMs and RNNs is that a continual input stream eventually may cause the internal values of the cell to grow indefinitely. In Gers et al. (1999) the original LSTM architecture was expanded by adding a new *forget gate* which is used to delete not needed information in the cell. In this way the cell learns to reset itself at appropriate times, thus releasing internal resources and achieving better results with longer sequences.

2.4 | Transformers: Attention is All You Need

In "Attention is All You Need", Vaswani et al. (2017) introduced the *Transformer* a revolutionary neural network architecture that outperformed the state-of-the-art for what concerns the language-translation task and that has achieved top results in any other Natural Language Processing (NLP) tasks. The Transformer is a simple network composed by an encoder and a decoder, that processes an input sequence of data to produce an output sequence, it is based solely on attention mechanisms and it is suited for parallelization.

2.4.1 | Attention Mechanisms

In machine learning, *Attention* is a technique that mimics cognitive attention in order to enhance the important parts of an input and ignore the rest. Attention mechanisms have gained popularity in various applications [Kim et al. (2017), Bahdanau et al. (2015)], allowing modeling of dependencies in sequences without regard to distance of the elements. The peculiarity of the Transformer, is that it uses only *self-attention mechanisms*, a class of attention functions in which a single input element of an input sequence is mapped against all the other elements of the same sequence, thus removing all the constraints of typical sequential computation; when dealing with sequential data it is hard to exploit parallelization, RNNs are an example of how time consuming and computationally inefficient it can be to train a model that learns from sequences, in fact RNNs require to encode the input one element at a time, in Transformers the encoding is done in one shot exploiting self-attention. Another huge difference with other architectures thought for working with sequential data is that in such networks, when working with long sequences, information from the earliest elements tends to get lost as the network focuses on latest elements of the sequence. This is a big problem, especially in NLP because closeness is not a sign of relationship between words, it is the contextual meaning that implies relationships between words. This means that when dealing with NLP, words that are very far from the considered one, can still be very important and so it should not be ignored; distance is not a good way to evaluate importance of elements in a sequence.

In general, an attention function maps a query and a set of key-value pairs to an output, which is computed as a weighted sum. Query, Value and key are terms borrowed from *information retrieval*[3] literature, when you perform a *query* (what you are looking for) on an information system, the search engine will map the query against a set of *keys* (each one identify an entry of the data set) and then present you the *values* (best matches). This also means that the attention operation can be thought of as an information retrieval process in which the query is the element of the sequence we are considering, the keys are the elements of the sequence that we are going to compare with the query and the values are the weighted elements of the sequence. The result of the operation is just a more contextual representation of the data, which tells how each element of the sequence is related to the other and how important those related elements are in the context of the chosen element.

[3]Information retrieval is the process of obtaining information that are relevant from a collection of resources.

2.4.1.1 | Scaled Dot-Product Attention

The authors of the Transformer proposed a particular attention function called "Scaled Dot-Product Attention" in which the weights for the attention evaluation are calculated with a dot product of the query with all the keys divided by the square root of the dimension of the keys, then a softmax is applied as a form of normalization. At the end the weights are multiplied by the values to obtain the attention scores (2.1).

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad \text{with Q,K,V matrices} \tag{2.1}$$

Once again we have the concept of query, keys and values, they are not conceptually different from what was described in the previous paragraph, in this formula the authors used matrices just to make things more compact, in fact the evaluation of the attention for each element of the sequence is independent from the others, they can all be performed in parallel at the same time and this is the reason why in the formula the authors used three matrices.

In practice, Q, K, V are obtained by multiplying the input matrix X (each element of the sequence is a vector of X, further details on how this matrix is obtained are described in 2.4.2) of the sequence by three weight matrices W^Q, W^K, W^V. The weight matrices are the learnable parameters of the Transformer and they are crucial to make the network capable of learning how to perform a specific task.

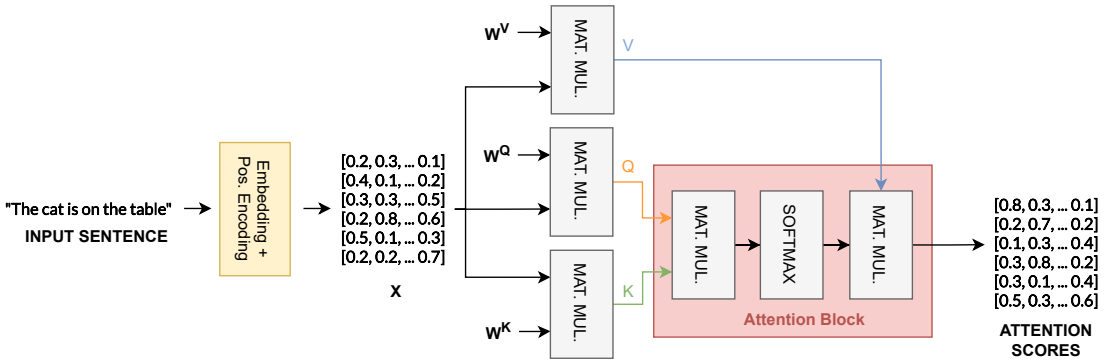

Figure 2.4: Schematic representation of the Scaled Dot-Product Attention algorithm

2.4.1.2 | Multi-Head Attention

Instead of performing a single attention function, the authors found it beneficial to implement multiple attention functions. The reason behind this choice can be simply explained by considering that relations between words in a sentence can be more than

just one type, it is legitimate to let the network focus on several aspects of the input sequence. In the Transformer as it is presented in Vaswani et al. (2017), the multi-head attention is obtained by running the Scaled Dot-Product Attention multiple times in parallel, then the independent attention outputs are concatenated.

2.4.2 | The Transformer Architecture

The architecture of the Transformer is composed by a stack of encoders followed by a stack of decoders. The *encoder* is composed by several identical layers, each one constituted by a multi-head self-attention mechanism and a position-wise fully connected feed-forward network. The *decoder* shared most of its internal structure with the encoder, with some differences:

- in the decoder there are two multi-head self-attention blocks:
  - the first has masked inputs, the reason for this will be more clear at the end of this chapter;
  - the second multi-head attention block has the keys and the queries for coming from the encoder output, whereas the values are coming from the first multi-head attention block of the decoder.

For what concerns internal structure there are not many differences, but there is an important distinction on how encoding and decoding are performed: the encoding is applied to the whole input sequence at the same time, whereas the decoding produces one element of the output sequence at the time.

As shown in figure 2.5, the input sequence is at the beginning transformed into an input matrix by evaluating the embedding vectors[4] for each element of the sequence. A positional encoding is added to each input embedding, this is crucial to account for the order of the elements in the input sequence. The positional encoding helps the Transformer to understand the distance between different input elements. It is also added an "end-of-sequence" tag at the end of the sequence.

The matrix obtained at the end of the embedding phase is inputted to the encoder that will produce the attention vectors by applying the self-attention mechanisms already described, the point of the encoding phase is to map the input sequence into an abstract representation in which any element of the sequence contain the information

[4]In NLP applications, it is common to convert each input word into a vector using an embedding algorithm. This transformation in necessary because many machine learning algorithms require their inputs to be vectors of numbers. This is also convenient when evaluating similarity scores.

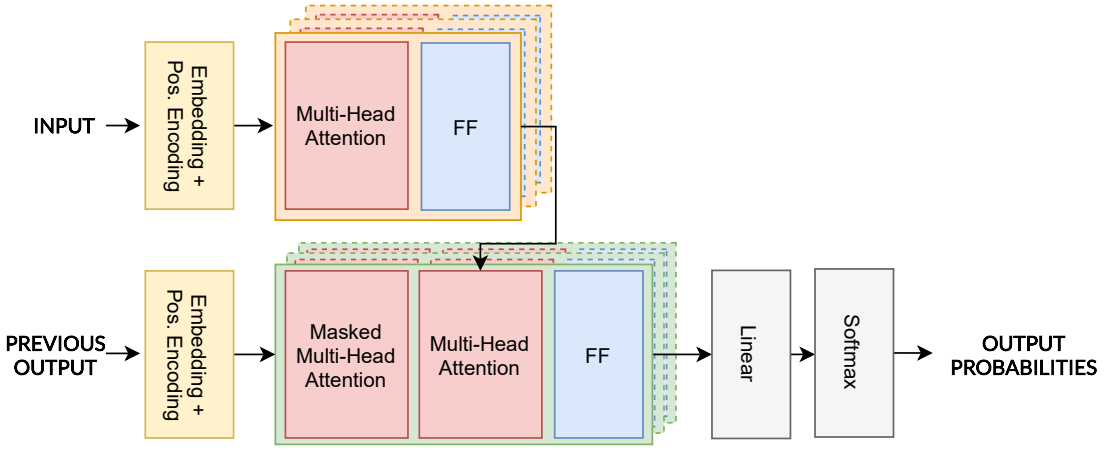

Figure 2.5: Simplified architecture of the Transformer network as described in Vaswani et al. (2017) with only the most important steps

about the element itself and its relationship with the whole sequence (in other words, the self-attention). The attention vectors will then be used in the decoding phase.

The stack of decoders will iteratively produce the output sequence, at each step a new element of the output sequence is evaluated by applying once again the self-attention function to an input matrix. This input matrix will be different from the one used in the encoding phase, the idea is still the same: applying self-attention to the whole sequence; the problem is that during decoding phase the output sequence is not known at the beginning. The authors defined a fictitious "start-of-sequence" element (complementary to the "end-of-sequence" of the encoding phase) for the first step and then they set all the other elements of the sequence to $-inf$ to mask them. As the decoding progresses the elements already decoded will be used as input for next steps and at the end no elements of the sequence will be masked. The output of the first self-attention block will be used to form the queries and keys for the second multi-head attention block, whereas the values will be formed by the output of the stack of encoders.

The final layer of the Transformer is composed by a linear layer followed by a softmax: the linear layer is a simple fully connected neural networks that projects the vectors produced by the stack of decoders into a much larger vector; the softmax layer finally turns the outputs into probabilities, the word associated with the highest probability is chosen (this output will then be used as input for the stack of decoders in the next step).

The Transformer is a very powerful network architecture, it was introduced to solve

language translation tasks and it became one of the most popular architectures of the last five years. Here is a quick recap of how the Transformer processes the input sentence in order to produce the translated sentence:

1. each word of the sequence is embedded into a vector of real numbers and then a terminating special element is appended to the list of vectors;
2. each input vector is tagged with a positional encoding, the resulting list of vectors is stacked into a matrix called X;
3. X is given as input to the stack of encoder to initiate the encoding phase, each decoder will executed multiple times in parallel, one for each head i of the multi-head attention mechanism the following steps:
   - 3.1. three weight matrices ${W_{e,i}}^K$, ${W_{e,i}}^Q$, ${W_{e,i}}^V$ are multiplied to X to obtain K_e, Q_e, V_d;
   - 3.2. K_e, Q_e, V_e are inputted to the self-attention block which applies the Scaled-Dot Product Function as it was described in the previous paragraph;
   - 3.3. the attention scores are from each head are concatenated and the result is inputted to a feed-forward neural network.
4. the decoding phase can now begin, since at this moment there is no previous output to fed the decoder, a special start-of-sequence element is going to be used. The decoding is a step-by-step procedure, for each iteration h the Decoder will:
   - 4.1. an input matrix is formed, by considering only the items up to h, the following items of the sequence will be masked and the decoder will perform a masked multi-head-attention using K_d, Q_d, V_d (in the same way as it was done in the encoding phase);
   - 4.2. the attention scores from the previous step will be used to form a new V_d matrix that will be used to perform the second multi-head attention, whereas the K_d, Q_d matrices will be formed using the output from the encoder;
   - 4.3. the resulting attention scores will be inputted to a feed-forward neural network
   - 4.4. the output from the stack of decoders is concatenated and inputted to a final linear layer;
   - 4.5. finally a softmax is applied and the word with the highest probability is chosen as output, it will be used to fed the next iteration of the decoding phase.

2.5 | Vision Transformers

Attention mechanisms produced brilliant results in NLP causing a general enthusiasm about the Transformer architecture in the entire machine learning science community. In Dosovitskiy et al. (2020) the authors took a breakthrough step forward in Computer Vision by introducing a new Transformer-based network to solve image classification tasks, the *Vision Transformer (ViT)*. Inspired by the successes with NLP tasks, multiple works tried to combine the attention mechanisms with the convolutional neural networks [Wang et al. (2017), Carion et al. (2020)], and some tried to replace entirely the CNNs [Ramachandran et al. (2019a), Wang et al. (2020)] but they failed to outperform the state-of-the-art at that time (ResNets). With the Vision Transformer (ViT), the au-

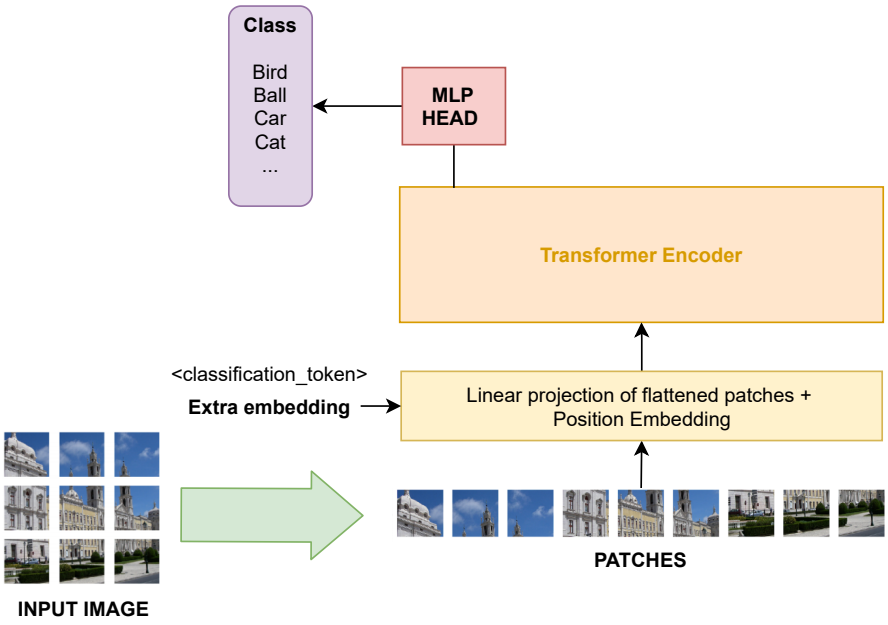

Figure 2.6: Overall architecture of the Vision Transformer as described in Dosovitskiy et al. (2020)

thors tried to apply a standard Transformer directly to images, with the fewest possible modifications. The input image is at the beginning split into patches, that are later flattened thus providing the sequence of linear embeddings of these patches as a valid input for the Transformer. The patches are then projected into a new space with a trainable linear projection and the resulting vectors will be exactly the same as what the word tokens were for the original Transformer. In this way, it is possible to exploit the attention mechanisms without applying them directly to the pixels of the images, which could be potentially very expensive in terms of computations. One of the few noticeable

differences with the Transformer, is that in the ViT the authors, inspired by the BERT architecture [Devlin et al. (2019)] included a learnable embedding to the input sequence, whose equivalent element of the output sequence will be used to feed a classifier (the additional token is also called "classification token"). The classifier used in the ViT is a regular MLP with one hidden layer. As shown in figure 2.6, in the ViT is present only the Encoder from the original Transformer architecture, because in this work the goal was to perform image classification, there was no reason to produce an output sequence.

Table 2.1: Comparison between ViT and the state-of-the-art on popular image classification benchmarks. proposed different version of their architecture: ViT-H/14 (JFT) has a 32 layers Transformer encoder, 14 attention heads and it was pre-trained on JFT-300M ; ViT-L/16 (JFT) has a 24 layers Transformer encoder, 16 attention heads and it was pre-trained on JFT-300M; ViT-L/16 (I21k) the same as ViT-L/16, but it was pre-trained on ImageNet-21k. In the table below are reported the mean and the standard deviation of the accuracies for each experiment [DBLP:journals/corr/abs-2010-11929].

Dataset	**ViT-H/14 (JFT)**	**ViT-L/16 (JFT)**	**ViT-L/17 (I21k)**	**ResNet 152x4**
ImageNet	**88.55**±0.04	87.76±0.03	85.30±0.02	87.54±0.02
ImageNet ReaL	**90.72**±0.05	90.54±0.03	88.62±0.05	90.54±0.00
CIFAR-10	**99.50**±0.06	99.42±0.03	99.15±0.03	99.37±0.06
CIFAR-100	**94.55**±0.04	93.90±0.05	93.25±0.05	93.51±0.08
Oxford-IIIT Pets	**97.56**±0.03	97.32±0.11	94.67±0.15	96.62±0.23
Oxford Flowers	99.50±0.02	**99.74**±0.02	99.61±0.02	99.63±0.03
VTAB (19 tasks)	**77.63**±0.23	76.28±0.46	72.72±0.21	76.29±0.70

In the table 2.1 are reported the results obtained by Dosovitskiy et al. (2020) with respect to some benchmark datasets; the ViT outperformed the state-of-the-art, but most importantly it showed that reliance on Convolutional Neural Network (CNN) is not necessary anymore, a pure Transformer applied directly to sequences of image patches can perform very well on image classification. The greatest drawback with the Vision Transformer (ViT) is that it requires to be pre-trained on huge datasets[5], in fact when tried with medium-range datasets[6] the results was not able to outperform the state-of-the-art. The CNNs encode prior knowledge about the image domain that reduces the need of data with respect to Transformers, which requires to discover such knowledge from very large-scale datasets otherwise the Transformer will fail to generalize well. The

[5]For this work the authors used the JFT-300M, an internal Google 300-million examples dataset for image classification models.

[6]In this case, the authors tried to pre-train the ViT using the Image21K dataset, which has 14 million entries and 21 thousand different classes.

reason for this is that in the Vision Transformer there is no prior bias on the data: for example, in a CNN the network knows how to deal with images because its architecture is intrinsically thought to perform convolution on local small portions of the input; another example, the LSTM knows that the data is a sequence of elements, and it knows the direction of the data flow. In the Vision Transformer there is none of this, and for this reason to be able to discover this kind of information about the input data it requires to see a lot of examples.

2.5.1 | Data-efficient image Transformers

The major drawback of the Vision Transformer is necessary to pre-train the network on huge datasets to outperform carefully tuned CNN designs. In Touvron et al. (2021) the *Data-efficient image Transformer (DeiT)* is introduced a transformer based network capable to obtain state-of-the-art results in large-scale image classification without relying on any external large dataset. The authors claims the can get such results using mid-sized datasets (e.g. 1.2 million examples compared to the 300 million examples dataset that was used in the ViT work) and with a relatively shorter training time. The authors was capable to achieve their results by adapting a novel distillation-learning approach for Transformers.

Knowledge Distillation introduced by Hinton et al. (2015), refers to the training paradigm in which a *student* network is trained by transferring the knowledge from a *teacher*, a larger network, without loss of validity. The distillation procedure adopted to train the DeiT uses a CNN as teacher network (RegNetY-16GF [Radosavovic et al. (2020)]) and it requires to add a distillation token to the input patch embeddings (similarly to the classification token). The DeiT is then trained using a cross-entropy loss function on the output class and a distillation loss to match the distillation token with the output from the teacher. The DeiT was capable to compare well against the top-performing CNN architectures (reaching 84.44% accuracy, very close to the best result achieved by a CNN-based network, 85.2%). The authors were able to achieve this impressive result using only 1.2 million examples of data due to:

- distillation: using a CNN as teacher network may have helped the ViT to discover those patterns in data that CNNs can easily see due to their structural bias. Does the ViT inherit existing inductive bias that would facilitate the training? The authors find it hard to formally answer this question, but their experiments suggest so;

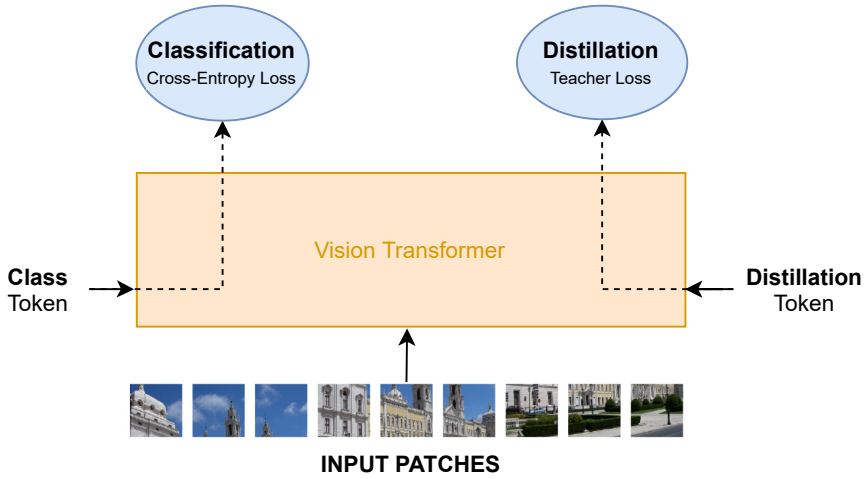

Figure 2.7: Overview of the Data-efficient image Transformer (DeiT) approach as described in Touvron et al. (2021)

- data augmentation: standard techniques such as flipping the images, slightly changing the contrast or the saturation of the images;
- optimization: extensive tuning during the cross-validation of the model to find the best hyper-parameters, this may require a huge amount of computational power.
- fine-tuning at different resolution: the input images used in the fine-tuning were up-scaled while keeping the L2-norm of the embedding tokens invariant.

2.5.2 | DINO: Self-supervised learning with Vision Transformers

Encouraged by the results obtained with the DeiT, the Facebook AI researchers introduced in Caron et al. (2021) a new learning strategy for Vision Transformers based on self-supervised learning. *Self-supervised learning* is a learning method that does not require labeled data, it however differs from *unsupervised learning*, which is totally unsupervised, by obtaining supervisory signals from the data itself, leveraging the underlying structure in the data (for example a common practice, for what concerns NLP is to hide some parts of the input sentence and let the model guess what is missing). AI system relying on labeled data require to see a lot of examples in order to perform classification tasks with success, this is a major issue of such systems, practically you can not label everything. Self-supervised learning, according to Caron et al. (2021), works well with the ViT architecture, in particular they found out that self-supervised ViT features contain explicit information about the semantic segmentation of an image, which

is not as clear with supervised ViT features. Furthermore these self-supervised features are great for k-NN classifiers. The resulting method, called *self-DIstillation with NO labels (DINO)*, is an impressive learning algorithm capable to produce models that can discover and segment objects in an image or a video with no supervision and without being given any prior knowledge on the object or any human-generated annotation.

Figure 2.8: Example of object segmentation with DINO on the right using the attention-map, in the middle is a segmentation example generated by a supervised method, *Caron et al. (2021)*

It is interesting to notice that this approach can also be used with success also with CNNs, the authors tried the DINO framework with a ResNet-50 architecture achieving the state-of-the-art for image classification on ImageNet. The DINO framework is a form of knowledge distillation with some differences with respect to traditional distillation: self-supervised learning (no labels); same network architecture for both the student and the teacher, but different parameters; the teacher network is trained along with the student network, not before it. Finally, here are reported the main steps in the DINO learning method:

1. The input image is augmented to produce multiple global-view images and it is cropped to produce local-view images;

2. Both the student and the teacher networks are trained at the same time, but for the teacher the back propagation is disabled;

3. The student network is fed with both local crop and global augmentations, then a softmax is applied to evaluate the output probabilities (p_s);

4. The teacher network is fed only with global augmentations and the difference here is that right before applying the softmax there is a centering operation which the authors proved being beneficial for the accuracy. The output probabilities of the teacher branch (p_t) will be then used to evaluate the loss using a cross-entropy loss function: $\ell = -p_t \log p_s$;

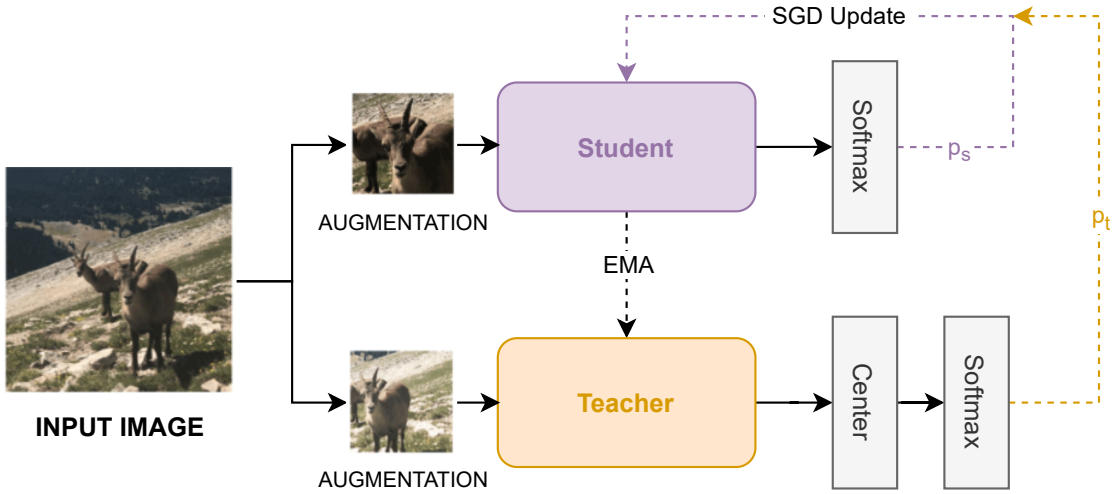

Figure 2.9: Overview of the self-DIstillation with NO labels (DINO) framework as described in Caron et al. (2021)

5. Back-propagation only on the student network;

6. The teacher network's parameters are updated using an Exponential Moving Average (EMA) of the student's parameters: $\theta_t = \lambda\theta_t + (1 - \lambda)\theta_s$.

2.6 | Related works: MNAD

Memory-guided Normality for Anomaly Detection (MNAD) introduced in Park et al. (2020a) is, at time of writing, the state-of-the-art in terms of video anomaly detection. It is a method based on CNNs strongly influenced by the U-Net architecture [Ronneberger et al. (2015)], whose main characteristic is the presence of a memory module that allows to perform unsupervised learning by considering the diversity of normal patterns. In traditional CNNs based approaches for anomaly detection the main drawback is that they do not consider the diversity of normal patterns explicitly. The authors assumed that a single prototypical feature is not enough to represent the various patterns of normal data, so a memory module was added such that individual items in the memory correspond to prototypical features of normal patterns. In practice, the memory module reduces the capacity of the CNN of producing representative features of the data. To properly train the memory module in the network, the authors introduce a two loss functions: a *feature compactness loss*, mapping the features of a normal video frame to the nearest item in the memory and encouraging them to be close; a *feature separateness loss*,

which minimizes the distance between each feature and its nearest item, while maximizing the discrepancy between the feature and the second nearest one. The method also prevents the memory from recording features of anomalous samples at test time (during test time the abnormality of a frame is evaluated using a score defined by the authors in order to update the memory only when dealing with presumed normal frames).

MNAD relies on two different methods to perform anomaly detection:

- *frame reconstruction*: the network tries to reconstruct the frame provided as input;
- *frame prediction*: several consequent frames are used to predict the following one. It is somehow a special case of the reconstruction, the method's goal is to reconstruct the future frame using the previous ones.

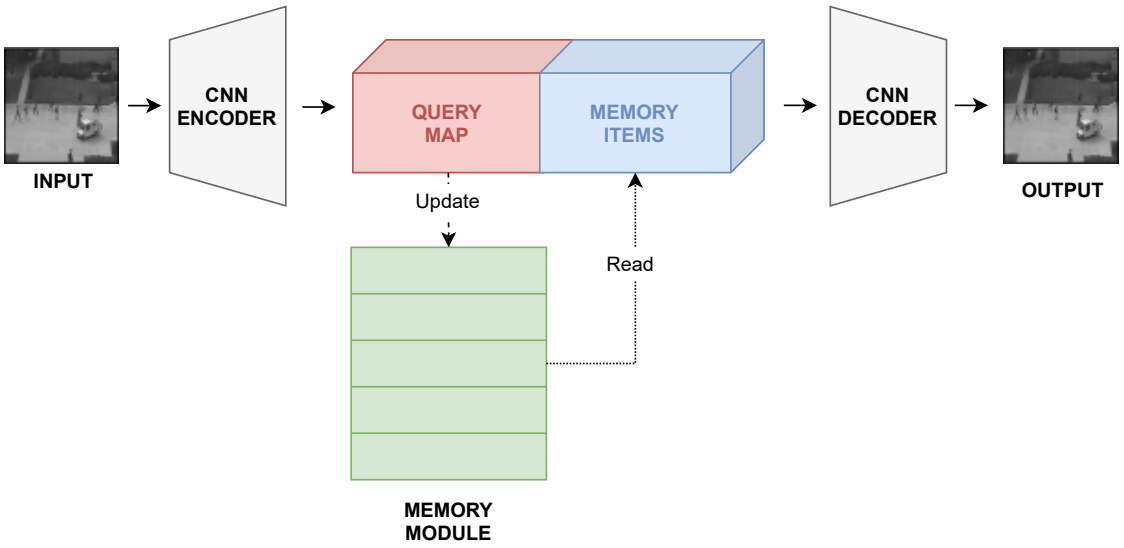

Figure 2.10: The architecture of the Memory-guided Normality for Anomaly Detection (MNAD) for the frame reconstruction task

Since the prediction task can be considered as a special case of the reconstruction task, the MNAD model has in both scenarios mostly[7] the same **architecture** consisting of three components:

- an *encoder*: a convolutional neural network used to extract the features from a frame of a video. The features will be then used as a *query map* to access the mem-

[7] the only architectural difference is regard skip-connections that were not inserted for the reconstruction task, according to the authors this choice produces better results

ory to retrieve prototypical normal patterns and at the same time to update the memory;

- a *memory module*: in literature there are many ways to capture long-term dependencies in sequences, LSTM is an example. In this work, the authors opted for a memory network [Weston et al. (2014)], such architectures have shown better performances than RNNs. The memory module in MNAD is queried using the features extracted by the decoder, and it outputs the memory items containing the normal patterns (the read operation is done by computing the cosine similarity between the query and all the memory items). The memory is updated both in training and test time, since normal patterns may be different (e.g., due to illumination or occlusion). The peculiarity of the MNAD's memory module is the fact that updating is performed also during test time, but in this case it is necessary to avoid storing anomalous features;

- a *decoder*: the query map produced by the encoder is concatenated with the memory items outputted by the memory module thus forming a transformed features map, that is then inputted to the decoder, a convolutional neural network that will recreate the target frame.

The **training** of the whole architecture is performed by composing three different loss functions: the reconstruction loss, the feature compactness loss and the feature separateness loss. The *reconstruction loss* makes the video frame reconstructed by the decoder as much similar as possible to its ground truth, by minimizing the distance between the twos. The *feature compactness loss* and the *feature separateness loss* are both used for the memory network: the first one is to encourage the queries to be close to the nearest item in the memory; the second is to keep similar query allocated to the same item in order to reduce the memory size, whereas at the same time, items in memory should be far from each other when considering different aspect of normality.

The last interesting part of the MNAD architecture is how the **abnormality score** is evaluated. This is crucial in any anomaly detection application, in this work, the authors exploit the memory items stored in the memory module to implicitly the abnormality score (the Peak signal-to-noise ratio (PSNR) of how well the frame is reconstructed using the memory items) and then this value is added to the normalized distance between the query map and the nearest item in memory.

2.7 | Summary

In this section were reported all the technologies and the methods that were used in this work: from a brief introduction of CNNs to a survey of the latest publications on Vision Transformers. In the last part of the chapter the Memory-guided Normality for Anomaly Detection (MNAD) was introduced, a convolutional architecture, which is at time of writing the-state-of-the-art in terms of video anomaly detection. All this knowledge will serve as the pillar of the works described in the following chapters.

3

Transformer-Based MNAD

In this section, it will be described a straightforward method to use the attention-mechanisms in video anomaly detection. The idea is simply to modify the original architecture of the Memory-guided Normality for Anomaly Detection (MNAD) described in the previous chapter, in order to have a Vision Transformer as encoder instead of the convolutional neural network that used by Park et al. (2020a).

3.1 | Method

The idea is to define an hybrid architecture with a Vision Transformer as the encoder and a convolutional neural network as decoder, while preserving the memory module. The new encoder was obtained by removing the classification layer from the Vision Transformer implementation. However, this is not sufficient to provide an encoded representation compatible with the rest of the MNAD architecture, some adjusting steps are required to obtain the right data-shape:

1. the first step is once again the definition of the input patches for each image of the input batches. The original MNAD architectures uses 256x256 three-channel images in batches of four;
2. the vision transformer will require an additional classification token, even if this is not required for the purpose of the anomaly detection;
3. from the transformer output sequence it is removed the vector corresponding to the classification token;
4. the final step requires to reshape the output sequence back to a two-dimensional vector, so that it can be handle correctly from the rest of the MNAD architecture.

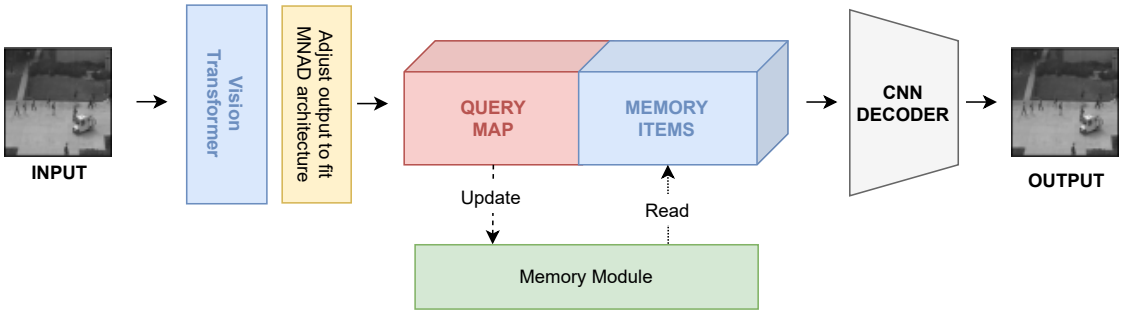

Figure 3.1: The new MNAD architecture with a Vision Transformer as encoder for the frame reconstruction task

It is important in the first step to choose the right value for the patch size in order to reconstruct the output with no issues.

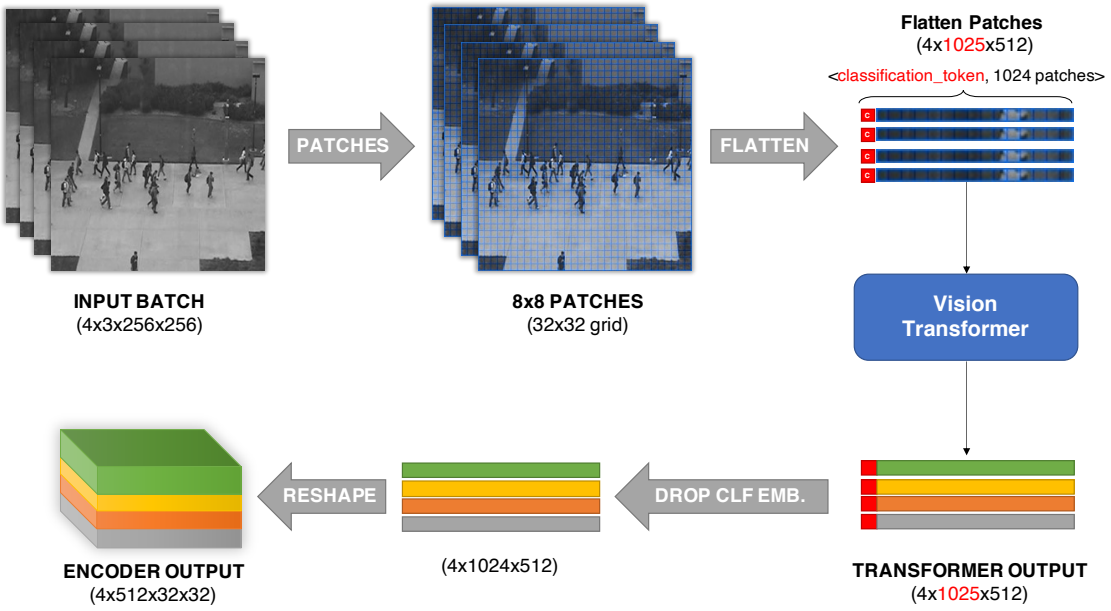

Figure 3.2: Steps to adjust the Vision Transformer output for the MNAD architecture

3.1.1 | Frame Prediction

The steps described to adjust the data outputted from the Vision Transformed are required for both the methods used in the MNAD architecture, however, for what con-

cerns frame prediction there are some additional changes to do in order to have the network working without problems. In fact the original architecture adopted skip-connections between the encoder and the decoder, this is indeen not possible in the version of the MNAD architecture that relies on transformers for the encoding of the input. So it was necessary to remove the skip connections in the decoder (in the original architecture both encoder and decoder had the same amount of convolutional layers, and each pair of corresponding layers was connected with a skip-connection).

3.2 | Results

Here are reported the results for the experiments done with the proposed MNAD architecture with a Vision Transformer as Encoder. Before that, however, it is reported the reproduction for the original MNAD architecture experiments, in order to have a comparable baseline for the work. All the following experiments were performed on the same machine with two *GeForce GTX 1080* GPUs and using *PyTorch 1.1.0* as recommended by the authors of the original MNAD architecture.

3.2.1 | The Dataset

In both scenarios, the original architecture and the proposed Transformer-based version, the experiments were performed on the *UCSD Anomaly Detection Dataset* [Mahadevan et al. (2010)]. The dataset was acquired with a stationary camera in correspondence of pedestrian walkways. Abnormal events are due to: the circulation of non-pedestrian entities (bikes, skateboards, small carts) or anomalous pedestrian motion patterns. The dataset is split in two subsets, each corresponding to a different scene:

- **PED1**: clips of groups of people walking towards and away from the camera. It contains 34 training video samples and 36 testing video samples, all having the same duration (200 frames). This subset is more challenging than the other.
- **PED2**: clips of groups of people walking parallel to the camera plane. It contains 16 training video samples and 12 testing video samples. In this subset, the anomalous events are only those containing non pedestrian entities.

In both subsets, the training video samples contains only normal events, whereas the testing videos contains at least one anomalous event at some point (it could be also a full anomalous video, with no normal events at all). Each video frame was resized to

the size of 256x256 and then normalized to the range of [-1, 1] before they were fed into the MNAD architecture.

3.2.2 | Original MNAD Reproduction

In order to have comparable results, it is necessary to reproduce the experiments described in Park et al. (2020a) on the same environment considered for the work described in this thesis. The MNAD architecture is characterized by many hyper-parameters and different configurations:

- Method: the MNAD architectures can perform video anomaly detection using a frame-reconstruction approach or a frame-prediction approach;
- Batch size: the number of samples that will be propagated through the network at each forward step, in this work this value was set to 4;
- Features map shape: the height, the width and the number of channels of the query map produced by the encoder, in this work this was set respectively to 32**,** 32**,** 512;
- Memory items: the number of entries in the memory module, set to 10;
- Learning rate: a tuning parameter regulating the step-size at each back-propagation step. In this work, there will be two different values depending on the adopted method for the anomaly detection. $2e-5$ for the prediction task and $2e-4$ for the reconstruction task;

In both reconstruction and prediction tasks, it was used the *Adam optimizer* [Kingma and Ba (2017)] and 60 epochs for the training. In this reproduction of the experiment, almost all the parameters initially chosen by the authors were kept unchanged, however due to hardware limitations it was necessary to implement a *batch-aggregation* mechanism, the training was done using virtual batches of size 4, but the actual batch size was set to 1 to avoid memory issues. In the table below are reported the best AUC values that were obtained for the MNAD architecture compared to the claims of the authors.

As you can see from the table 3.1, the reproduction of the experiment failed to achieve the same performance, this may be due to the differences in the environments. Furthermore Park et al. (2020a) did not try their architecture on the PED1 subset, in this work however both subsets of the UCSD Anomaly Detection Dataset were considered. From this point on, the future results will be compared to the performance of the

Dataset	Method	Park et al. (2020a) AUC (%)	Reproduction AUC (%)
PED1	Prediction	—	**75.44**
PED1	Reconstruction	—	—
PED2	Prediction	**97**	94.93
PED2	Reconstruction	**91**	88.86

Table 3.1: AUC values for the original MNAD architecture compared to what was obtain during the reproduction of the same experiments described in Park et al. (2020a)

original MNAD that were observed during the reproduction of the experiments of the original authors (this values will be indicated as *Park et al. (2020a)**).

3.2.3 | Transformer-Based MNAD

In this new version of the MNAD architecture, it is used a Vision Transformer to encode the input images, whereas the rest of the architecture is kept unchanged as much as possible. When dealing with ViTs there are several parameters to tune in order to maximize the performances:

- Patch size: the size of the input patches for the Vision Transformer. In literature there are evidence that keeping this value small is better for the performance [Caron et al. (2021)]. In this work this was set initially to 8, this also allows to keep the same features shape (with $8x8$ patches there are exactly 1024 output embeddings, that can be then reshaped to a $32x32$ two-dimensional data, the same values as the height and width of the convolutional features map of the original MNAD;
- Feature dim: the size of the embedding vector, it was set to 512, this together with the $8x8$ patches allow to reconstruct a query map with $32x32x512$ shape;
- Number of heads: this value specifies the number of heads of the multi-attention blocks within the Vision Transformer, in this work several values were tested $[2, 4, 8, 16]$;
- MLP Dim: this value specifies the size of the feed-forward neural network in the Vision Transformer, several values were tested $[1024, 2048, 3072]$;
- Vision Transformer Depth: the size of the encoders stack within the Vision Transformer, the values tried were $[6, 8, 12]$

3.2.3.1 | Frame Reconstruction Task

Indeed there are many parameters, trying every possible combination would not be feasible, for this reason at first it was tried an exploratory phase in which several combination were tried in order to evaluate the importance of the factors. From the literature it is known that when working with Transformers the most important factors are the number of heads and the size of the MLP layer. In table 3.2 are reported the AUC values for the proposed hybrid architecture for the PED2 dataset and only for the task of frame reconstruction, while varying the number of heads and the MLP dim parameters. From the experiments it emerges that both factors are quite important in achieving better results, but the MLP dim is dominating over the number of heads of the multi-attention blocks. Interestingly, choosing a greater number of heads did not imply better performance, but it always implied longer training time (figure 3.3).

Method	Dataset	Patch size	# Heads	MLP Dim	ViT Depth	AUC (%)
Recon	PED2	8x8	16	2048	4	88.69
Recon	PED2	8x8	16	1024	4	85.75
Recon	PED2	8x8	8	2048	4	86.49
Recon	PED2	8x8	8	1024	4	85.69
Recon	**PED2**	**8x8**	**4**	**2048**	**4**	**88.79**
Recon	PED2	8x8	4	1024	4	86.78
Recon	PED2	8x8	2	2048	4	88.62
Recon	PED2	8x8	2	1024	4	87.77

Table 3.2: Results of the experiments done with the hybrid transformer-convolutional MNAD architecture on the PED2 dataset

3.2.3.2 | Frame Prediction Task

Starting from the knowledge derived from the reconstruction task, several tests were done to find the best parameters configuration for the frame prediction task. In table 3.3 are reported the resulting AUC values and the adopted values for the parameters.

3.2.3.3 | Pre-Trained Vision Transformer

The biggest problem with Vision Transformers is that they require quite a huge amount of data to learn those patterns in data that Convolutional Neural Networks learn im-

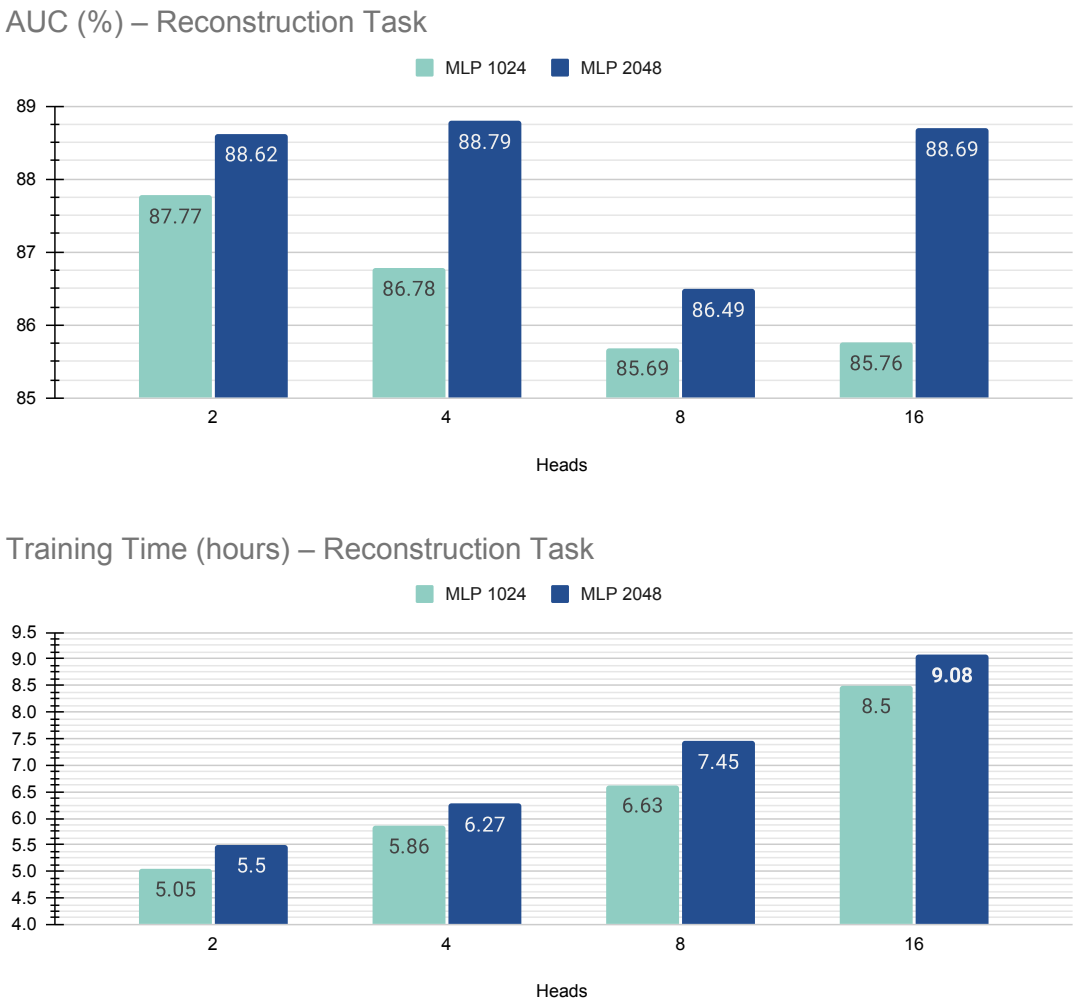

Figure 3.3: AUC values and training time while varying the MLP Dim and the number of heads in the Vision Transformer for the frame reconstruction task of the Transformer-based MNAD.

plicitly due to their architectures. Unfortunately due to the extremely specific nature of the anomaly detection task, it is hard to find a pre-trained Vision Transformer for such applications. At time of writing, there can only be found pre-trained Vision Transformers for image classification tasks. In table 3.4 are reported the results of the tests that were done with pre-trained Vision Transformers[1], and as you can see their performance

[1]The Vision Transformer was pre-trained on ImageNet21k, source: `https://github.com/lukemelas/PyTorch-Pretrained-ViT`

Method	Dataset	Patch size	# Heads	MLP Dim	ViT Depth	AUC (%)
Pred	PED2	8x8	4	1024	6	90.46
Pred	PED2	8x8	4	2048	12	88.39
Pred	PED2	8x8	16	2048	12	83.52
Pred	PED2	8x8	8	2048	12	82.06
Pred	PED2	8x8	4	1024	8	86.35
Pred	PED2	8x8	4	2048	6	89.68
Pred	**PED2**	**8x8**	**16**	**1024**	**6**	**92.12**
Pred	PED2	16x16	4	1024	6	87.78
Pred	PED2	16x16	4	2048	6	83.04
Pred	PED2	16x16	12	3072	12	91.65
Pred	PED2	16x16	12	3072	12	80.58

Table 3.3: Results of the experiments done with the hybrid transformer-convolutional MNAD architecture on the PED2 dataset for the frame prediction task

are comparable to the untrained versions.

Method	Dataset	Patch size	# Heads	MLP Dim	ViT Depth	AUC (%)
Pred	PED2	16x16	12	3072	16	90.99
Recon	PED2	16x16	12	3072	16	87.56

Table 3.4: Results of the experiments done with the hybrid transformer-convolutional MNAD architecture on the PED2 dataset using a Pre-Trained Vision Transformer

3.2.4 | Final comparison

In table 3.5 is reported a final comparison between the two architectures, for what concerns the AUC values of the original MNAD work, the values reported are those obtained in the reproduction of the experiment and not the claims by Park et al. (2020a). As you can see, the proposed architecture can only perform slightly better in the case of frame reconstruction for the PED2 dataset, in all the other configurations it failed to outperform the fully convolutional architecture proposed by Park et al. (2020a). The main problem here is that both PED1 and PED2 datasets are quite small, not even comparable with the 300-Million entries dataset used by google; however, the results that were

obtained are not that far from the state-of-the-art, and this is an encouraging outcome, considering how new these technologies are.

Dataset	Method	Park et al. (2020a)* AUC(%)	This work AUC(%)
PED1	Prediction	**75.44**	74.00
PED1	Reconstruction	—	—
PED2	Prediction	**94.93**	92.12
PED2	Reconstruction	88.47	**88.62**

Table 3.5: Final comparison between the AUC values for the MNAD architecture as observed during the reproduction of the original experiments and the Transformer-based version of the MNAD decribed in this work.

4

Anomaly Detection with Self-Supervised ViT Features

In this section, a novelty video anomaly detection system based on attention mechanisms is presented. The core of this work is represented by the DINO learning method introduced in Caron et al. (2021). As described in 2.5.2, DINO is a revolutionary self-supervised method based on knowledge distillation that produces Vision Transformers that are particularly good at extracting features from input data that was never been seen before. The idea is to exploit this Vision Transformer pre-trained using the DINO algorithm (from this point on the term DINO will be used to refer to this Vision Transformer and not the learning method), to extract the features from the input frames and then perform anomaly detection on those features. In particular this work was composed by the following contributions:

- a study on the quality of the DINO features;
- an unsupervised anomaly detection method combining DINO with an LSTM network;
- a One-Class Classification anomaly detection system built using DINO.

It was decided to keep using the same *UCSD Anomaly Detection Dataset* (see 3.2.1) that was adopted for the previously described experiments on the MNAD.

4.1 | Analysis of the DINO features

The DINO learning method introduced in Caron et al. (2021), was used by the authors to train a Vision Transformer (ViT) with the configuration described in table 4.1.

Input image size	Patch size	# Heads	MLP Dim	ViT Depth	# Params
224x224	8x8	6	384	12	21M

Table 4.1: Configuration for the Vision Transformer trained with the DINO method. The *MLP Dim* will also be the size of each features vector; # *Params* is the total number of parameters of the Vision Transformer; for further details on the other hyper-parameters, please see 3.2.3.

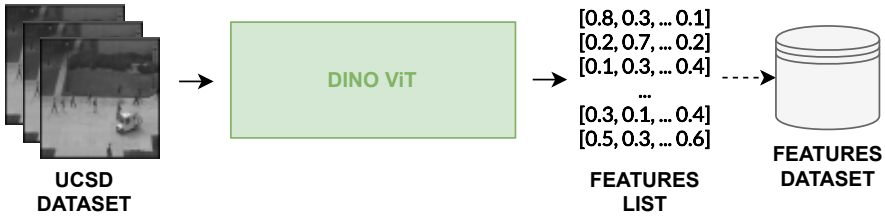

Figure 4.1: DINO Features extraction workflow.

As described by its authors, the features extracted using a ViT trained with the self-supervised DINO method are particularly good to feed a k-NN classifier; furthermore the self-supervised ViT features contains explicit information about the scene layouts. Their results were encouraging enough to try building a novelty approach based on self-supervised ViT features for video anomaly detection. In order to achieve such system, it was necessary to perform a study on the quality of those features when extracted from the *UCSD Anomaly Detection Dataset* (in particular for this preliminary phase, the study was focused on the features extracted from the *PED2* subset of the dataset).

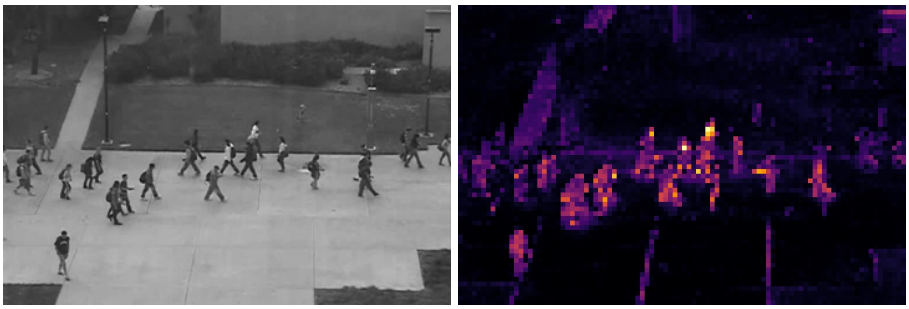

Figure 4.2: On the left a frame from the *UCSD PED2 Dataset*, on the right the attention map of a the frame produced by the last self-attention block of the ViT pre-trained with the DINO method.

Each video frame of the *UCSD PED2 dataset* was inputted to the DINO pre-trained

Vision Transformer to produce a vector of 384 real numbers. The extracted features were stored in a file on disk to allow an offline anomaly detection using only this information, without any knowledge about the original videos. This will make the detection extremely quick and the final models will be smaller and easier to train, considering how small the data are with respect to a full set of videos.

4.1.1 | k-Means Clustering

A quick and simple way to find out if the DINO features are good enough to be used in an anomaly detection system, is to perform a k-Means Clustering. The *k-Means Clustering* is a clustering algorithm that aims to partition the input data into k clusters (k is a given hyper-parameter of the algorithm). The algorithm is distance based and produces the resulting clusters by minimizing the distances each vector and the center of the closest cluster. At the beginning, k points of the input data set are randomly chosen as the initial centers of the clusters, at each iteration the centroid are then recalculated. When the assignments no longer change, the algorithm has converged.

The idea was to apply a 2-Means clustering and then to analyze the resulting clusters: they were expected to be unbalanced, with one of the two cluster dominated by normal features vectors and one dominated by anomalous features vectors.

Label	**Cluster 1**	**Cluster 2**
Normal	2233	679
Anomalous	324	1324

Table 4.2: An example of 2-Means Clustering on the DINO features extracted from the *UCSD PED2 Dataset*. As it is shown the clustering achieved an overall satisfying separation between the two different classes of data.

In the table 4.2 are reported the resulting clusters of the 2-Means Clustering, as it is shown, the results are overall satisfying, but as expected the separation is not perfectly sharp. In order to obtain a better separation, it was tried a 4-Means Clustering expecting to produce a better data separation because of the multiple typologies of anomalous events in the *UCSD PED2 Dataset*.

As shown in table 4.3, the resulting clusters of the 4-Means Clustering have two well defined anomalous clusters and two mixed clusters. After analyzing the frames whose DINO features vectors were placed in *Cluster 1* it emerged that they all came from videos in which the pedestrian walkway is little crowded. For what concerns the *Cluster 4* there was no actual meaning, apart from a generic visual similarity.

Label	Cluster 1	Cluster 2	Cluster 3	Cluster 4
Normal	1966	0	0	946
Anomalous	188	876	90	494

Table 4.3: An example of 4-Means Clustering on the DINO features vectors extracted from the *UCSD PED2 Dataset*

4.1.2 | t-SNE of the DINO Features

t-distributed Stochastic Neighbor Embedding (t-SNE) is a statistical visual tool for visualizing high-dimensional data by giving to each point a location in a two-dimensional map. It is a nonlinear dimensionality reduction technique, which models the datapoints in such a way that similar objects get mapped closed together in the two-dimensional space.

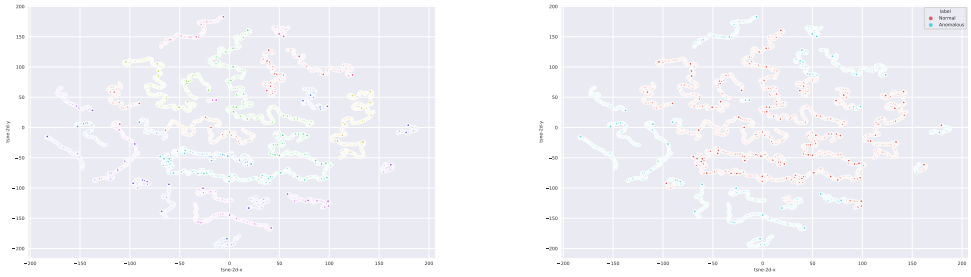

Figure 4.3: t-SNE of the DINO Features: on the left the colour indicates a different video, on the right the colour indicates the class (red is normal, blue is anomalous). The t-SNE was performed by setting $numiter = 5000$ and $perplexity = 10$.

In figure 4.3, it is shown the result of the t-SNE applied to all the features vectors of the dataset, as you can see the resulting clusters separate well the different videos of the *UCSD PED2 Dataset*, but it failed to show a sharp separation between anomalous and normal frames, mostly because the most similar frames to the anomalous frames of a video will be the other frames of the same video.

In figure 4.4, it is shown the result of the t-SNE applied to the aggregated features vectors of the dataset, one features vector per video. The features vectors associated to each frame of the same video were aggregated using a *max* function, thus producing only one vector. As you can see, in this case the separation between classes is sharper and well defined, apart from two anomalous videos. Those two anomalous videos ($Testing - 12$ and $Testing - 11$ in the *UCSD PED2 Dataset*), are in fact quite te-

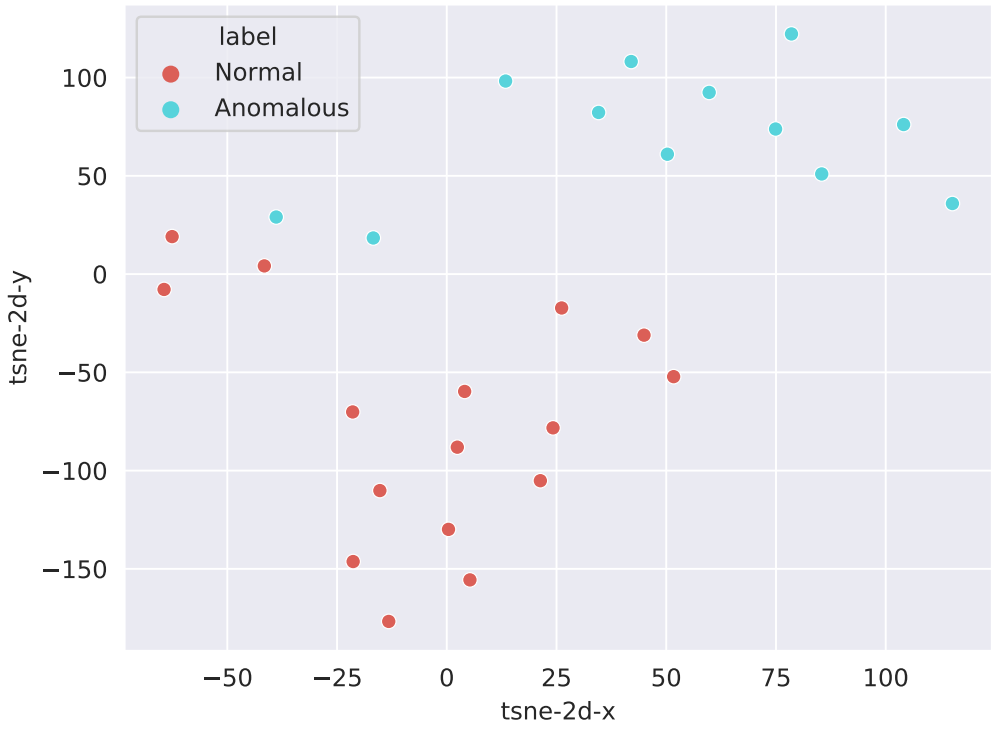

Figure 4.4: t-SNE of the aggregated DINO Features.The t-SNE was performed by setting $numiter = 5000$ and $perplexity = 10$, whereas the aggregating function is the max.

dious to classify, even for a human being it could be not so easy to spot an anomaly (see figure 4.5).

4.1.3 | k-NN Classification using the DINO Features

In this paragraph the features previously extracted using the Vision Transformer pre-trained with the DINO method will be analyzed when dealing with a k-NN classifier. Since here the only objective is to study how good the features are, it doesn't matter if it is necessary to change the original dataset and to go for a supervised approach.

The *k-NN Classification* is a classification method based on similarity between the examples of the data set. The most important thing of the k-NN Classification is the definition of the similarity function. A target example is compared to all the training data and the k nearest neighbors of the target example are returned. The classification

Figure 4.5: Examples of hard to classify anomalous frames from the *UCSD PED2 Dataset*. On the left a frame from the video $Testing - 11$, the presence of a biker is the anomalous event. On the right a frame from the video $Testing - 12$, the presence of a skateboarder is the anomalous event.

is then performed by doing a majority vote of the neighbors. k is an hyper-parameter of the classifier, it is typically a small positive integer and it represents the size of the neighborhood to be considered when performing the election of the classification label. In this work are reported the results that was obtained using the following values of k: $[1, 3, 7, 9, 11]$.

Since the k-NN classification is a supervised method, the *UCSD PED2 Dataset* can not be used so as it is defined, it is required to shuffle the videos (not the frames) of the dataset in order to have both anomalous and normal classes in the training data. In fact, originally the *UCSD PED2 dataset* contains anomalous frames only on the test set, and this is not good for a supervised learning method such as a k-NN classifier. The important thing when shuffling is to avoid to have frames coming from the same videos in both the training and the test set. It was decided to follow several approaches to perform the k-NN classification:

- repeated cross-validation;
- repeated cross-validation with aggregated features);
- leave-one-out.

In all of those methods, the similarity function to correctly perform the classification was always the cosine similarity (applied to each target features vector of the test set with respect to the whole training set). Furthermore, the k-NN classification was performed using both the DINO features vectors and the features vectors extracted using a pre-

trained convolutional architecture (*ResNet-50*). In this way it is possible to compare the quality of the DINO features with the convolutional features.

4.1.3.1 | Repeated Cross-Validation

In this first training strategy, it was opted for a repeated N-Fold stratified cross-validation. The number of iterations was set to 60, whereas the number of folds was set to $[10, 4]$. In this scenario each fold corresponds to a different video, because it is important to keep the features coming from the frames of the same video either in the training set or in the test set, but not in both at the same time. In table 4.4 are reported the mean values of the accuracy of the classification with 95-ConfidenceInterval.

k	# Folds	DINO Accuracy (%) – 95 CI	ResNet-50 Accuracy (%) – 95 CI
1	10	**85.06** $\pm$ 2.09	74.56 $\pm$ 2.22
3	10	**85.60** $\pm$ 2.11	76.05 $\pm$ 2.33
5	10	**85.76** $\pm$ 2.12	75.40 $\pm$ 2.32
7	10	**85.68** $\pm$ 2.13	76.18 $\pm$ 2.43
9	10	**85.89** $\pm$ 2.13	76.10 $\pm$ 2.45
11	10	**85.88** $\pm$ 2.17	76.39 $\pm$ 2.53
1	4	**84.16** $\pm$ 1.24	73.63 $\pm$ 1.25
3	4	**84.73** $\pm$ 1.22	73.98 $\pm$ 1.31
5	4	**85.02** $\pm$ 1.24	73.45 $\pm$ 1.31
7	4	**85.08** $\pm$ 1.25	74.86 $\pm$ 1.34
9	4	**85.22** $\pm$ 1.25	74.88 $\pm$ 1.35
11	4	**85.26** $\pm$ 1.26	75.19 $\pm$ 1.38

Table 4.4: Results of multiple N-Fold Cross-Validation for the k-NN classification for the DINO features compared to the ResNet-50 features, both extracted from the *UCSD PED2 Dataset*. In the table the mean accuracy values are reported together with their confidence interval evaluated with 95% confidence level. It is interesting to notice here that the DINO features always outperform the ResNet-50 features, which is an indicator of how good those features are.

The results were promising, indeed this was just a preparatory analysis, in fact such results are not comparable with the literature, since anomaly detection is generally achieved in literature using self-supervised/unsupervised approaches.

4.1.3.2 | Repeated Cross-Validation with Aggregated Features

In this second approach, it was decided to repeat one more time the same steps that were done in the first method (100 times repeated 4-Fold cross-validation), but using only one feature vector per video. The label for this aggregated vector is "Anomalous" if there is at least one frame in the video that is anomalous, otherwise it is set to "Normal". The idea is that aggregating the features vectors of every frame of the video, the anomalies would emerge if present. The following aggregating function were tried: *min*, *max*, *mean*.

k	Aggr. Function	DINO Accuracy (%) – 95 CI	ResNet-50 Accuracy (%) – 95 CI
1	*min*	93.04 ± 1.83	92.00 ± 1.86
1	*max*	$\mathbf{93.07} \pm 1.83$	87.43 ± 2.13
1	*mean*	$\mathbf{89.82} \pm 2.33$	87.36 ± 2.27
5	*min*	$\mathbf{92.79} \pm 1.86$	77.04 ± 2.61
5	*max*	$\mathbf{92.54} \pm 1.94$	74.00 ± 2.74
5	*mean*	$\mathbf{84.79} \pm 2.48$	76.07 ± 2.67
9	*min*	$\mathbf{89.68} \pm 1.99$	68.54 ± 3.09
9	*max*	$\mathbf{90.28} \pm 1.94$	72.18 ± 2.89
9	*mean*	$\mathbf{84.89} \pm 2.33$	66.71 ± 2.96

Table 4.5: Results of multiple 4-Fold Cross-Validation for the k-NN classification for the DINO aggregated features compared to the ResNet-50 aggregated features, both extracted from the *UCSD PED2 Dataset*. Also in this scenario the DINO features outperforms almost every time the ResNet-50, only the 1-NN with minimum as aggregation function produced with both features comparable results.

In table 4.5 are reported the mean accuracy values for several configurations. As you can see the accuracy in both DINO and ResNet-50 features are greater, this is not surprising, the previous approach failed to correctly classify only small subsets of features vectors for each video. In table 4.6 are reported the videos that were wrongly classified by the aggregated k-NN classification using *min*.

4.1.3.3 | Leave-One Out

In this final scenario for the k-NN classification, it is considered a special case in which the test set is composed by just one video and all the other videos are used as training

Video ID	ResNet-50 # Errors	DINO # Errors
12 – *Testing*	**207**	572
11 – *Testing*	**353**	496
10 – *Testing*	162	**0**
09 – *Testing*	115	**0**
08 – *Testing*	483	**0**
07 – *Testing*	517	**152**
06 – *Testing*	148	**0**
04 – *Testing*	492	**11**
03 – *Testing*	105	**0**
02 – *Testing*	156	**0**
01 – *Testing*	10	**0**
16 – *Training*	3	**0**
12 – *Training*	184	**0**
11 – *Training*	1	**0**
10 – *Training*	28	**0**
08 – *Training*	2	**0**
05 – *Training*	494	**0**
03 – *Training*	255	**78**
02 – *Training*	198	**114**
01 – *Training*	114	**93**

Table 4.6: Comparison between DINO and ResNet-50 of the numbers of total errors when doing k-NN classification in 4-Fold Cross-Validation of the aggregated features (*min*). The Video ID notation refers to the original organization of the data of the *UCSD PED2 Dataset*, which is composed by a set of training videos and a set of testing videos; as shown in the table, the DINO features outperforms the ResNet-50 features with almost all the videos. The videos whose classification produced zero errors with both DINO and ResNet-50 were not reported.

set. From the training videos only two features vectors are produced, one synthesizing all the features vectors coming from the normal frames and one synthesizing all the features vectors coming from the anomalous frames. This means that in this scenario there are two aggregating function, a local one (aggregating the features for each video) and a global one (aggregating the features for each class). This kind of approach can only be repeated one time per each video of the *UCSD PED2 dataset*.

Local Aggr.	Global Aggr.	DINO Accuracy (%)	ResNet-50 Accuracy (%)
min	*min*	57.14	57.14
min	*max*	**78.57**	53.57
min	**mean**	85.71	**92.86**
max	*min*	**60.71**	42.86
max	*max*	**85.71**	71.43
max	*mean*	**89.29**	71.43
mean	*min*	**89.86**	78.57
mean	**max**	**92.86**	71.42
mean	*mean*	75.00	75.00

Table 4.7: Comparison between DINO Features and ResNet-50 Features for what concerns the Leave-One-Out approach. With this method, DINO and ResNet-50 features produce comparable results when considering the best results ($min - mean$ for ResNet-50 and $mean - max$ for DINO). It is interesting how the best results were obtained using different aggregation functions.

4.2 | Supervised Anomaly Detection with DINO

In this section we propose a supervised anomaly detection system built on the DINO features described in the previous section. Supervised-learning is almost never a good choice for what concerns anomaly detection, mostly because it is too hard to correctly provide a data set that includes every type of anomalous events. What we are proposing here, is more of an additional analysis of the quality of the DINO features, than an actual proposal.

4.2.1 | Method

The anomaly detection task is solved here as a binary classification problem. The classifier that we propose is a simple MLP, in figure 4.6 is shown its architecture. As you can see it is a basic MLP with 384 input neurons, 128 hidden neurons and 1 output neuron. Since anomaly detection is often unsupervised, there is no benchmark data set as reference, so we decided to keep using the *UCSD PED2 Dataset* in a supervised manner: the training was performed by randomly splitting the dataset in two sub sets ($75 - 25$) and it was repeated 100 times. For what concerns the loss function, a simple *Binary Cross-Entropy* was used; lastly, *Adam* with a $1e - 4$ learning-rate was used to optimize the back propagation step.

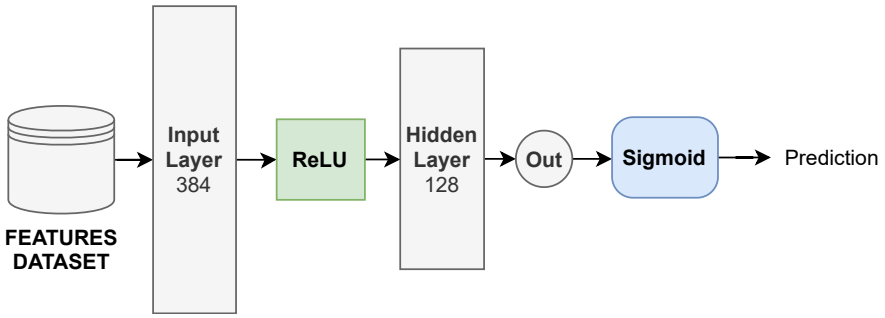

Figure 4.6: Architecture of the Multi-Layer Perceptron (MLP) used to test the DINO Features.

4.2.2 | Results

The biggest problem of this approach is that, since there is no supervised anomaly detection dataset, it is hard to compare the performance of the classifier with the literature. In this sense, this approach is merely a way to prove the quality of the DINO features. In table 4.8, the mean accuracy score is reported (with a 95% confidence interval).

# Runs	# Epochs	In Size	Hidden Size	Out Size	Mean Accuracy (%) – 95 CI
100	100	384	128	1	84.60 ± 1.52

Table 4.8: Accuracy score for an Multi-Layer Perceptron (MLP) classifier fed with the DINO Features. The training and the evaluation were repeated 100 times, in the table the mean value is together with the 95-ConfidenceInterval.

4.3 | Unsupervised Anomaly Detection with DINO

The preliminary analysis described in the previous chapters showed the quality of the features extracted using the self-supervised ViT trained using the DINO method. Several techniques were tried to test the DINO features in the context of an anomaly detection system, however they were mostly supervised methods applied to a data set (*UCSD Dataset*) that was not thought to be used in that manner. In this chapter we will present an unsupervised anomaly detection system based on the DINO Features. Unfortunately, the proposed method failed to achieve the state-of-the-art in terms of AUC score, but this is not a completely wrong result and it shows the potential of the self-attention mechanism applied to the anomaly detection. It is important to highlight the

fact that ViTs are a relatively new technology, whereas the CNNs have been widely using in computer vision since early 2010s, there are reasons to believe that in the future, the Transformers will be mature enough to eventually become the state-of-the-art for any aspects of computer vision.

4.3.1 | Method

The architecture proposed in this work is a simple anomaly detection technique based on prediction, in this situation the system will be fed with a sequence of DINO features vectors and it will try to predict the immediately next features vector. If the actual features vector remarkably differs from the prediction then it is marked as anomalous, otherwise it is considered normal.

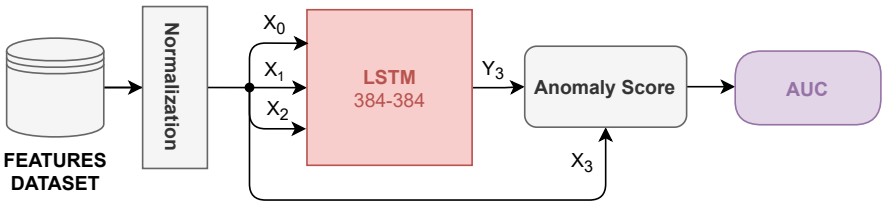

Figure 4.7: Architecture of the unsupervised Anomaly Detection System built using a Long Short-Term Memory (LSTM).

In its simplest form (figure 4.7), the anomaly detection systems relies on a two-layer LSTM for the prediction of the next element of the sequence of features vectors. Early experiments showed that normalizing (*L2-norm*) the DINO features before performing the prediction, significantly improves the performance of the system. For what concerns the length of the input sequence, several values were tried (ranging from 5 to 15), eventually the best results were obtained using 15-length sequences (where the fifteenth element is the one that should be predicted). One of the most important aspect such architectures, is the definition of the *anomaly score*. In this work, the anomaly score is obtained by normalizing (*min-max normalization*) the distances between targets and predictions. The model should be trained to reconstruct the target next features vector, in order to do so, it was used a regular *MSE* loss function.

In figure 4.8 is presented a more complicated version of the architecture (will be referred as *LSTM-AE*), which relies on two LSTM modules (each one having four layers instead of two and with a dropout with 0.7 probability after each layer), in a sort of auto-encoder configuration. The first LSTM module produce an encoded representation of the input sequence, then its hidden state is propagated as input to the second LSTM

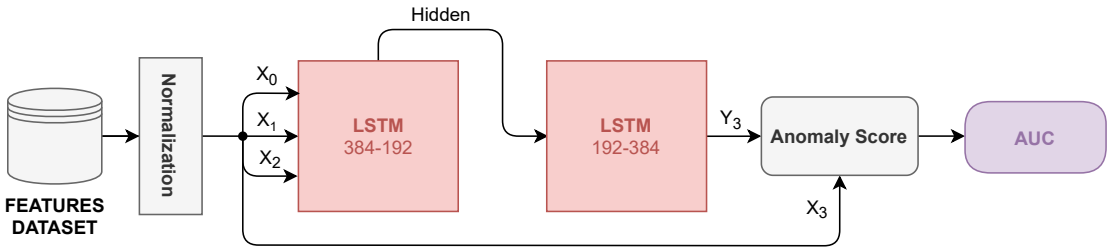

Figure 4.8: Variation of the anomaly detection system that uses two LSTMs blocks in a sort of Auto-Encoder configuration.

module that will produce the prediction for the next element of the sequence. The rest of the architecture is kept unchanged.

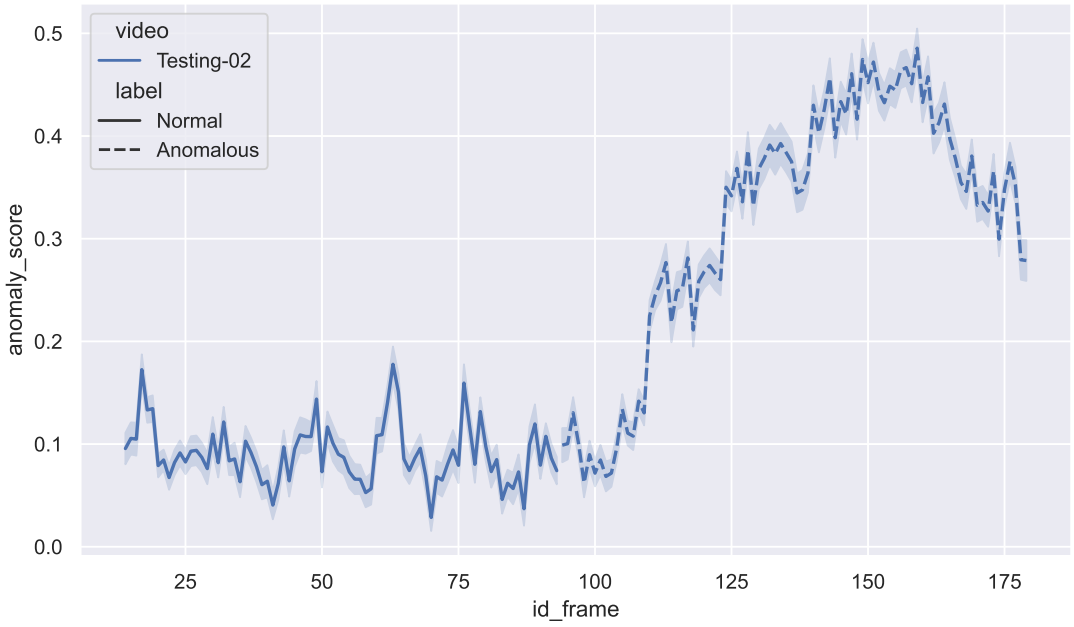

Figure 4.9: Example of anomaly detection using the DINO features and a basic LSTM module. In the example are reported the distances between the prediction and the target features vectors. The dotted line means that the features vector is coming from an anomalous frame, the continuous line instead means it is normal. The error is expressed using the standard deviation.

4.3.2 | Results

The two architectures were trained using an *Adam* optimizer with the learning rate set to $1e-4$, for 60 epochs. The experiments were repeated 60 times. Once again, every test was performed on the *UCSD Dataset* (see chapter 3.2.1). To complete the analysis of the system, the same architecture was tried also on the features of the same videos extracted using a convolutional neural network (*ResNet-50*).

The results of the experiments are reported in the table 4.9, as you can see the best performance was obtained by the $LSTM-AE$ architecture in both scenarios of the *UCSD Dataset*, this result however it is not enough to outperform the current state-of-the-art, which is still significantly better: the baseline considered in this thesis is the MNAD architecture described in 2.6, which is at time of writing the state-of-the-art for what concerns the anomaly detection on the *UCSD PED2 Dataset*, with 97% AUC. Park et al. (2020a) did not provide a result for the *UCSD PED1 Dataset*, in our tests the MNAD architecture was able to reach 75% AUC[1].

Type	Norm.	# Layers	Dropout	Seq. Len.	Features	AUC (%) 95-CI
LSTM	✓	2	0.0	15	$PED2_R$	**70.98** $\pm$ 0.26
LSTM	✗	2	0.0	15	$PED2_D$	80.16 $\pm$ 0.09
LSTM	✓	2	0.0	15	$PED2_D$	87.53 $\pm$ 0.22
LSTM	✓	2	0.0	10	$PED2_D$	87.35 $\pm$ 0.10
LSTM	✓	4	0.5	10	$PED2_D$	87.24 $\pm$ 0.09
LSTM-AE	✓	4	0.7	10	$PED2_D$	88.18 $\pm$ 0.10
LSTM-AE	✓	**4**	**0.7**	**15**	$PED2_D$	**88.54** $\pm$ 0.11
LSTM	✓	**2**	**0.0**	**15**	$PED1_D$	**61.54** $\pm$ 0.11
LSTM-AE	✓	4	0.7	15	$PED1_D$	60.50 $\pm$ 0.10

Table 4.9: AUC scores of the anomaly detection system. *LSTM* and *LSTM-AE* indicate respectively the architectures shown in figure 4.7 and in figure 4.8. $PED2_D$ and $PED2_R$ refer to the features vectors extracted from the *UCSD PED2 Dataset* using respectively the DINO method and the ResNet-50. $PED1_D$ refers to the DINO features from the *UCSD PED1 Dataset*

.

[1] for further details on the results of the MNAD architecture, please see chapter 3.2.2

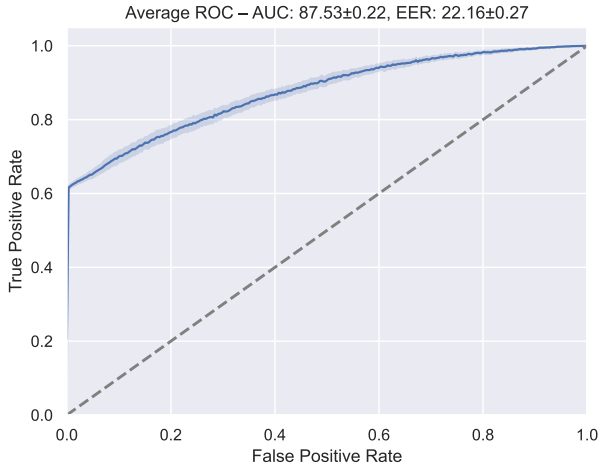

Figure 4.10: Average ROC curve for the single LSTM architecture (with normalized PED2 DINO Features, sequence length set to 15 and no dropout).

4.4 | One-Class SVM with DINO

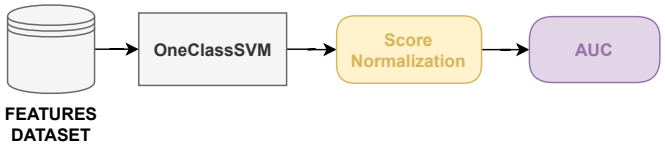

Figure 4.11: Architecture of the One-Class Classification approach with the OneClassSVM.

One-Class Classification is a widely adapted approach to perform anomaly detection tasks, it is a classification method that aims to identify objects of a specific class (the anomalous data) amongst all the other objects of the data set (the normal data). This is done by training the model only on the classes of the normal objects.

4.4.1 | Method

The One-Class SVM perform the one-class classification by identifying the smallest hyper-sphere consisting of all the data points. This method is called Support Vector Data Descriptor (SVDD). For this work, it was used the implementation of the OneClassSVM provided in the *Scikit-Learn Library*. To obtain an AUC score of the classification, the scores produced by the model for each input data were normalized in a $[0, 1]$ range, thus transforming them into probabilities.

4.4.2 | Results

In table 4.10 are reported the AUC scores for the one-class classification performed using the OneClassSVM. Once again, the reference dataset is the *UCSD PED Dataset*.

Dataset	DINO AUC (%)	ResNet-50 AUC (%)
PED1	62.96	–
PED2	**87.75**	60.53

Table 4.10: Results of the One-Class Classification with the OneClassSVM.

4.5 | DeepSVDD with DINO

Traditional one-class classification approaches often struggle when dealing with high-dimensional data, most of the times it is required to perform some features engineering procedure to overcome those issues. In Ruff et al. (2018) the authors proposed a one-class classification anomaly detection method based on a fully deep approach, the Deep Support Vector Data Descriptor (DeepSVDD).

4.5.1 | Method

This method, deeply inspired by the the SVDD method described in 4.4, learns useful feature representation of the data together with the one-class classification objective. To do this it is necessary to employ a specific neural network that is jointly trained to map the data into a hyper-sphere of minimum volume. Through this, DeepSVDD extracts common factors of variation from the data. For our purposes, the network responsible for mapping the features from input space to output space was a simple MLP with one hidden layer.

4.5.2 | Results

In table 4.11 are reported the results for the one-class classification performed using the DeepSVDD method. Each experiment was repeated 60 times and the results are expressed in terms of mean AUC with 95% confidence interval. The reference dataset is still the *UCSD PED Dataset*. Interestingly, as shown in table 4.11, this approach did not perform better than the regular SVDD of the OneClassSVM. In the end, both of the one-class classification approaches failed to achieve the state-of-the-art results; the AUC

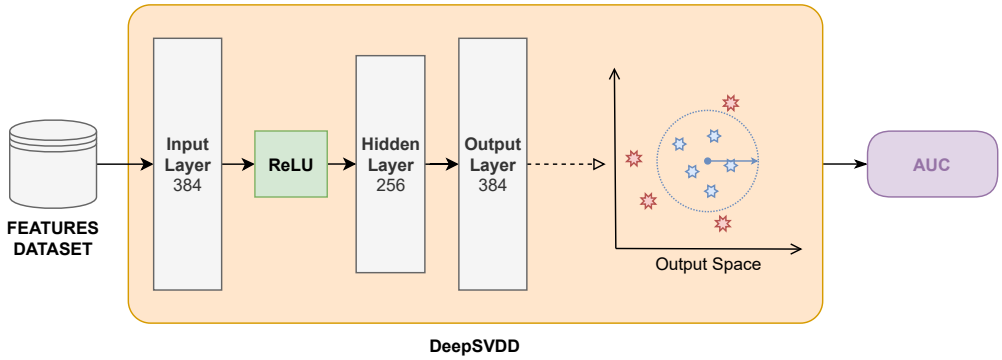

Figure 4.12: Architecture of the One-Class Classification method with the Deep Support Vector Data Descriptor (DeepSVDD).

scores obtained here are consistent with those described through all this section of the thesis.

Dataset	DINO AUC (%), 95-CI	ResNet-50 AUC (%), 95-CI
PED1	55.65 ± 2.12	–
PED2	76.72 ± 1.77	48.47 ± 3.52

Table 4.11: Results of the One-Class Classification method with the Deep Support Vector Data Descriptor (DeepSVDD). When using the ResNet-50 features, it was necessary to change the architecture of the MLP network, input size and output size were both set to 1000, whereas the hidden size was set to 512.

4.6 | Summary

In this section we presented several methods to perform anomaly detection relying mostly only on the features extracted using a pre-trained Vision Transformer. Such Vision Transformer was pre-trained using the DINO method, a revolutionary approach based on self-supervised learning and distillation. Unfortunately none of the proposed methods outperformed the state-of-the-art, CNNs are still topping video anomaly detection tasks. However the results were quite promising, especially if compared to the results obtained when using the features extracted with a ResNet-50.

5

Conclusions

With this thesis, we tried to implement a visual anomaly detection system based on attention mechanisms and Vision Transformers. Unfortunately, all of the proposed methods failed to outperform the state-of-the-art, which is still represented by Convolutional Neural Networks. However, it is important to remind that attention-mechanisms applied to Computer Vision is a relatively new approach [Dosovitskiy et al. (2020)], whereas CNNs have been widely adopted since the early 2010s. In Caron et al. (2021), the authors showed that with a different training method, they were able to achieve great results without relying on huge data sets, which is at the time of writing the major drawback when dealing with Vision Transformers. There are reasons to believe that in the next years, Vision Transformers will be mature enough to replace Convolutional Neural Networks in many computer vision tasks.

5.1 | Contributions and Limitations

After a brief introduction to the main methods to perform anomaly detection, in chapter 3 we presented a first straightforward way to bring attention mechanisms into visual anomaly detection by modifying the structure of the Memory-guided Normality for Anomaly Detection (MNAD) architecture, which is at the time of writing the state-of-the-art for what concerns visual anomaly detection. This first approach was a sort of hybrid architecture, composed by a Vision Transformer, without any pre-training, in the role of the encoder of the input, and a convolutional decoder. This variation in the architecture did not achieve the desired results, the main problem here was that the Vision Transformers have no architectural bias that helps them to recognize local data patterns in data (contrary to what happens with CNNs), for this reason, to be able to

understand that kind of information, ViTs require to see a lot of data examples. Anomaly Detection data sets unfortunately are not so huge.

In chapter 4 we tried to overcome the problem of small data sets by using an approach based on a Vision Transformer that was pre-trained using a self-supervised distillation learning method called DINO. This training method, exploit self-supervision and distillation to produce a model that is capable to obtain state-of-the-art comparable results in image classification while relying on small data sets. The Vision Transformer so as it was deployed by their authors was used to extract the features from an anomaly detection data set, and then we built several anomaly detection systems on those features (that we called *DINO Features*). We acknowledge that this kind of approach is not very popular for tasks like visual anomaly detection, however, the DINO features produced impressive results (even without any form of features engineering), especially if compared to what we obtained when repeating the same approaches, but using ResNet-50 features.

In table 5.1 is reported a final recap on the best mean AUC scores that were obtained through all the thesis work.

Dataset	MNAD	MNAD*	ViT + MNAD	DINO Prediction	DINO OCC
PED1	–	**75.44**	74.00	61.54	62.96
PED2	97	**94.93**	92.12	88.54	87.75

Table 5.1: Final recap of the AUC scores for all the Anomaly Detection methods used. *MNAD* refers to the results described in , whereas *MNAD* are the results that were obtained in our reproduction of the same experiment on our environment. *DINO Prediction* is the best AUC that was obtained using LSTM modules to predict the next features vector of a sequence of DINO features vectors. *DINO OCC* is the AUC result for a one-class classification method applied to the DINO features.

5.2 | Future Works

The Vision Transformer, introduced in Dosovitskiy et al. (2020), was a breakthrough method for what concerns image classification, since then several works relying on ViTs were published, generating a strong enthusiasm in the scientific community. The future of attention mechanisms for computer vision appears to be brighter than ever, it is safe to state that in the next years they will completely replace the Convolutional Neural Networks in almost any Computer Vision task. They are already achieving great results in image classification [Dosovitskiy et al. (2020), Touvron et al. (2021)], image generation

[Jiang et al. (2021), Chen et al. (2020)], object-detection [Carion et al. (2020)] and text-to-image synthesis [Ramesh et al. (2021)]; what is even more impressive is that all those works were published in late 2020 and early 2021. With regards to visual anomaly detection, are not widely adopted yet. However, recent works proposed hybrid methods combining Convolutional Neural Networks with ViTs (similar to our proposal of the Transformer-MNAD architecture described in chapter 3.2.3) [Mishra et al. (2021a)] and fully attention-based methods [Pirnay and Chai (2021)], which open new scenarios for attention-based visual anomaly detection, we hope this will drive further interest in the scientific community to leverage the potential of Transformer models and to overcome their current limitations – e.g. the necessity for high computational power and tons of data examples.

References

Davide Abati, Angelo Porrello, Simone Calderara, and Rita Cucchiara. Latent space autoregression for novelty detection, 2019.

Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. January 2015. 3rd International Conference on Learning Representations, ICLR 2015 ; Conference date: 07-05-2015 Through 09-05-2015.

Pierre Baldi. Autoencoders, unsupervised learning and deep architectures. In *Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning Workshop - Volume 27*, UTLW'11, page 37–50. JMLR.org, 2011.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *CoRR*, abs/2005.12872, 2020.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers, 2021.

Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey, 2007.

Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1691–1703. PMLR, 13–18 Jul 2020. URL `http://proceedings.mlr.press/v119/chen20s.html`.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020.

F.A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: continual prediction with lstm. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, volume 2, pages 850–855 vol.2, 1999. doi: 10.1049/cp:19991218.

Mahmudul Hasan, Jonghyun Choi, Jan Neumann, Amit K. Roy-Chowdhury, and Larry S. Davis. Learning temporal regularity in video sequences, 2016.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition, 2019.

Radu Tudor Ionescu, Sorina Smeureanu, Marius Popescu, and Bogdan Alexe. Detecting abnormal events in video using narrowed normality clusters, 2018.

Yifan Jiang, Shiyu Chang, and Zhangyang Wang. Transgan: Two transformers can make one strong GAN. *CoRR*, abs/2102.07074, 2021.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. Structured attention networks. *CoRR*, abs/1702.00887, 2017.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 1106–1114, 2012. URL `https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html`.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Yann LeCun, Yoshua Bengio, and Geoffrey E. Hinton. Deep learning. *Nat.*, 521(7553):436–444, 2015. doi: 10.1038/nature14539.

Wen Liu, Weixin Luo, Dongze Lian, and Shenghua Gao. Future frame prediction for anomaly detection – a new baseline, 2018.

Vijay Mahadevan, Weixin Li, Viral Bhalodia, and Nuno Vasconcelos. Anomaly detection in crowded scenes. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1975–1981, 2010. doi: 10.1109/CVPR.2010.5539872.

Larry M. Manevitz and Malik Yousef. One-class svms for document classification. *J. Mach. Learn. Res.*, 2: 139–154, March 2002. ISSN 1532-4435.

Pankaj Mishra, Riccardo Verk, Daniele Fornasier, Claudio Piciarelli, and Gian Luca Foresti. VT-ADL: A vision transformer network for image anomaly detection and localization. *CoRR*, abs/2104.10036, 2021a.

Pankaj Mishra, Riccardo Verk, Daniele Fornasier, Claudio Piciarelli, and Gian Luca Foresti. Vt-adl: A vision transformer network for image anomaly detection and localization, 2021b.

Muzammal Naseer, Kanchana Ranasinghe, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Intriguing properties of vision transformers. *CoRR*, abs/2105.10497, 2021.

Hyunjong Park, Jongyoun Noh, and Bumsub Ham. Learning memory-guided normality for anomaly detection. *CoRR*, abs/2003.13228, 2020a.

Hyunjong Park, Jongyoun Noh, and Bumsub Ham. Learning memory-guided normality for anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14372–14381, 2020b.

Jonathan Pirnay and Keng Chai. Inpainting transformer for anomaly detection. *CoRR*, abs/2104.13897, 2021.

Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces, 2020.

Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models. *CoRR*, abs/1906.05909, 2019a.

Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models, 2019b.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *CoRR*, abs/2102.12092, 2021.

Mahdyar Ravanbakhsh, Moin Nabi, Enver Sangineto, Lucio Marcenaro, Carlo S. Regazzoni, and Nicu Sebe. Abnormal event detection in videos using generative adversarial nets. *CoRR*, abs/1708.09644, 2017.

Manasss Ribeiro, Andr Eugnio Lazzaretti, and Heitor Silvrio Lopes. A study of deep convolutional auto-encoders for anomaly detection in videos. *Pattern Recogn. Lett.*, 105(C):13–22, April 2018. ISSN 0167-8655. doi: 10.1016/j.patrec.2017.07.016.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4393–4402. PMLR, 10–15 Jul 2018. URL `http://proceedings.mlr.press/v80/ruff18a.html`.

David E. Rumelhart and James L. McClelland. *Learning Internal Representations by Error Propagation*, pages 318–362. 1987.

Mohammad Sabokrou, Mohsen Fayyaz, Mahmood Fathy, and Reinhard Klette. Deep-cascade: Cascading 3d deep neural networks for fast anomaly detection and localization in crowded scenes. *IEEE Transactions on Image Processing*, 26(4):1992–2004, 2017. doi: 10.1109/TIP.2017.2670780.

Jessie James P. Suarez and Prospero C. Naval Jr. A survey on deep learning techniques for video anomaly detection. *CoRR*, abs/2009.14146, 2020.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers and distillation through attention, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation, 2020.

Xiaolong Wang, Ross B. Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. *CoRR*, abs/1711.07971, 2017.

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. 2014. cite arxiv:1410.3916.

List of Figures

List of Tables

List of Abbreviations