UCLouvain

epl

École polytechnique de Louvain

Domain-To-Text: improving Domain Generalization using Natural Language

Co-supervised by Politecnico di Torino with Tatiana TOMMASI and Silvia BUCCI

Author: **Martial VAN DEN BROECK**
Supervisor: **Benoit MACQ**
Readers: **Jean-Charles DELVENNE, Maxime ZANELLA**
Academic year 202Z–2023
Master [120] in Data Sciences Engineering

Abstract

A common assumption in machine learning is to suppose that the training and the testing distribution are the same. Unfortunately, this assumption is not always true, especially in computer vision: one of the biggest challenges that visual recognition systems must face in the real world is the ability to have good performances on visually different data samples. The interest in this problem is growing in the computer vision community, and an increasing number of data scientists focus on that to develop models that can generalize well to unseen distribution. In this project, we investigated how the natural language description of the visual domains could be helpful to improve the generalization ability of a recognition model. In particular, we propose a metric learning based approach to learn a common embedding over the images and text such that nearby image and phrase pairs in the embedding space are related. Then we use the learned representation to improve the prediction on the testing data based on its similarity with the training distribution. We obtain encouraging results on the PACS dataset showing how a multi-modal learning based on natural language can improve the generalization ability of a model.

Contents

List of acronyms

AI Artificial Intelligence. 7

BERT Bidirectional Encoder Representations from Transformers. 15

CLIP Constrastive Language-Image Pre-training. 24

CNN Convolutional Neural Network. 12

DA Domain Adaptation. 17

DG Domain Generalizaton. 31

DTD Describable Textures in Detail Dataset. 34

ML Machine Learning. 8

MLP Multi-Layer Perceptron. 11

NLP Natural Language Processing. 23

PACS Photo - Art - Cartoon - Sketch dataset. 39

PCA Principal Component Analysis. 52

SOTA State of the art. 8

ViT Vision Transformer. 24

VL Visual-Language. 23

Chapter 1

Introduction

One of the most popular task in machine learning is the supervised learning, the paradigm for problems where the data is labelled and the goal is to learn a function mapping features vectors to labels, in other words, link the inputs to the outputs. Supervised learning has achieved success over success in many applications as object recognition, speech translation, object detection or classification in general. One factor explaining in part those success is the availability of massive and labelled datasets such as ImageNet [40] but this kind of dataset represents a significant cost and the process of collecting sample is prone to include bias [58]. In object recognition, for example, the training samples are often collected under specific environments such as a laboratory. This kind of environment provides ideal conditions; static lightning, standardised product, fixed camera viewpoints, fixed background, ... This provides classifiers unable to be directly applied to other related dataset because the set of images is too "perfect". Indeed, it is clear that a model trained only on laboratory images of strawberries for example as can be seen on the image below (Figure 1.1) on the left will have difficulty performing as well on images from daily life as can be observed on the right part of the image.

Among all the challenges that visual recognition systems must face, one is retaining our attention, the ability to have good performances on visually different data samples in the real world[59]. Developing learning algorithms able to reduce the discrepancy between images belonging to different domains is therefore a compelling problem. In this context, a domain represents a probability distribution from which the samples are drawn. Domain adaptation and domain generalization have been proposed to address this issue. Usually, the domain can be of two types: the source domain which is labelled and the target domain where the samples are unlabelled. The fact that those domains are related but still different limit the applicability of using standards learning methods on the target domain. A basic assumption in machine learning is to suppose that the training and the testing

Figure 1.1: Ideal conditions (laboratory) versus realistic conditions (daily life). Illustrations from [65] and [1]

distribution are the same. This assumption here does not stand and is violated [22].

This project focuses on Domain Generalization (DG) which is a formalization of this scenario: it is composed of several labeled source domains and a single unlabeled target domain, never seen during training. It aims to incorporate knowledge from multiple source domains into a single model that could generalize well on unseen target domains.

Note that the distances between the target and each source domain are not equal for all the sources (e.g., the domain shift between sketch and photo is larger than the domain shift between cartoon and photo). A first solution would be to compute the average of the different outputs generated by the models trained on each source domain. Alternatively, we can weigh the contribution of the models trained on each source domain in function of their similarity with the target. In this project, we investigate how a textual description of the visual domain could be helpful in measuring the source-target distance to properly weigh the contribution of each source domain in the target prediction. We use textual descriptions of visual domains to perform metric learning on the PACS dataset in order to build a reliable and coherent metric between the different visual domains. This idea and approach was suggested to us by Tatiana Tommasi and Silvia Bucci from the Politecnico di Torino University. We thank them for their welcome, the supervision provided as well as the access to high-end computer equipment to carry out the experiments.

The document is composed by a theoretical part followed by an expermimen-

tal part. The first includes explanations of the foundations of Machine Learning and Deep Learning by reviewing the main historical and current models in image classification. Then we will discuss the current state of research on Domain Generalization and models combining Visual and Language with particular attention to models taking advantage of massive use of data, "Zero-Shot Learning" type models such as CLIP or Dalle-2, its generative evolution. These models have a strong power of generalization and are able to predict samples for classes never seen during training. This is why we will briefly discuss the usefulness of our method when dealing with this kind of models in practical applications. Then we will present our classification method in Domain Generalization scenario and a variant, as well as the architecture and the training method of the embedder, the model which allows, thanks to the natural language, to send each image in a feature space coherent with reality and which allows us to measure the relative distance between each domain and to give a weight to these domains. For the second part, we present the creation of the dataset, the results obtained using the metric learning approach and a visualization part to assure a certain coherence in the metric.

Chapter 2

Basics of Deep Learning

2.1 Artificial Intelligence, Machine Learning and Deep Learning

2.1.1 Artificial Intelligence

The definitions of Artificial Intelligence (AI) vary from source to source, it could be defined as “a system’s ability to interpret external data correctly, to learn from such data, and to use those learnings to achieve specific goals and tasks through flexible adaptation.[25]”. It could aslo be "intelligence—perceiving, synthesizing, and inferring information—demonstrated by machines, as opposed to intelligence displayed by animals and humans" [90]. Or more simply as the study of rational agents[21]. An intelligent agent is a system that takes rationale decisions, so carries out the action with the best outcome after considering all the possible actions in its environment to reach its goals as we can see on the Figure 2.1. An AI system is composed of an agent and its environment. An agent is anything that can be viewed as : "perceiving its environment through sensors and acting upon that environment through actuators"[21].

One of the first artificial intelligences known to the general public is in the world of chess[54]. Indeed the agent here compares the possibles moves through his perception (sensors) of its environment and picks the one to play wih pre-defined heuristics (actuators). This notion of heuristic is interesting and underlines the limitations of memory and computation time. Today chess is still not a solved game as we are unable for the moment to compute a tree of possible moves deep enough.

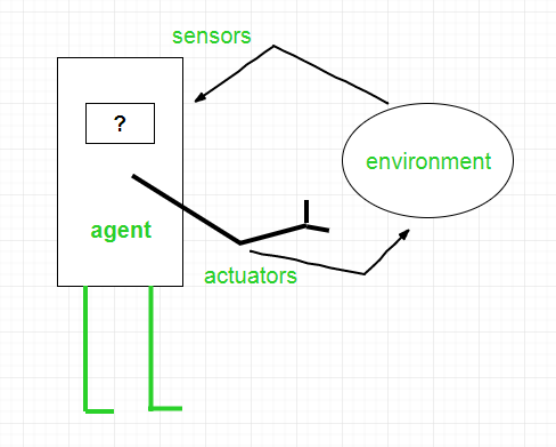


Figure 2.1: Schema of the agent perceiving its environment through sensors and acting upon that environment through actuators. Illustration from [21]

2.1.2 Machine Learning

Machine Learning (ML) is a sub-field of AI in where the agents update their parameters by learning from data. Machine learning algorithms build a model based on data called the training set in order to learn. When a machine learning algorithm "learns from data," it means that it is able to analyze patterns in the data and use those patterns to make predictions or decisions. These algorithms are able to automatically improve their performance as they are exposed to more data. For example, if we have a dataset of customer information, we would be able to use a machine learning algorithm to identify patterns that might indicate which customers are likely to buy a certain type of product. Once the training has been done, the model can be used to make predictions about new customers unseen before.

There exists a multitude of algorithms, the most popular are among other things; Linear regression[41], decision trees[70], support vector machines[27] and K-means[60]. Those algorithms were mentioned for the context but this document is focused on Deep Learning as state of the art (SOTA) for the classifcation of images is Neural Network and specifically Transformers[14].

2.1.3 Deep Learning

Deep Learning is a subset of Machine Learning as we can see on Figure 2.2. It is a type of machine learning using neural networks with multiple layers to learn patterns in data. These algorithms are made up of multiple layers of interconnected neurons performing simple calculations on the input data and pass the results to

the next layer. The final output of the network is a prediction based on the input data. To train a deep learning algorithm, data is fed through the network and the output is compared to the desired result. A process of backpropagation of the errors allows the network to iteratively minimze the error by adjusting the weights and biases of the neurons. These operations are repeated until the network is accurate enough on the training data[38].

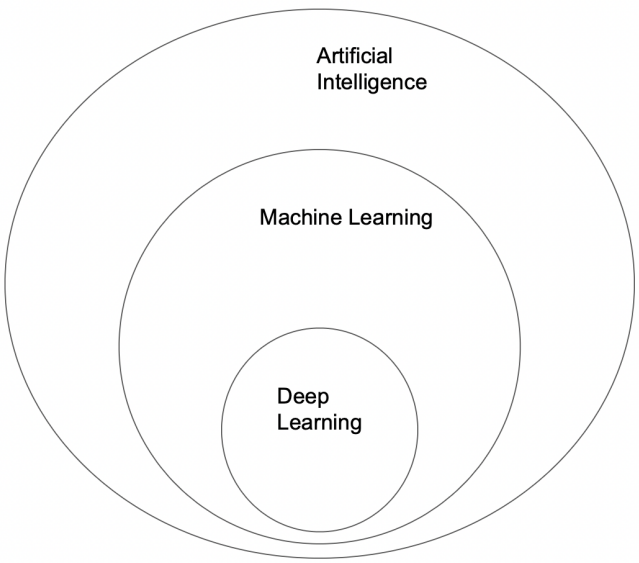


Figure 2.2: Deep Learning as a subset of Machine Learning as a subset of Artificial Intelligence. Illustration inspired from [42]

In ML, a model is possibly trained using a single layer of features. While in deep learning, algorithms can learn to extract features from raw data using multiple layers of interconnected nodes. This is why deep learning algorithms can learn more abstract and hierachical representations of the data. For example, a model specialized in image recognition would be able to identify edges and shapes in the first layers and then combine those features to identify complex objects like animals or buildings in the last layers of the network.

Link to the human body and perceptron

The term "neural" is inspired by the brain, in biology, the neuron receives the signal and treats it with basic computations and then sends the result to the next one thanks to synapses (connections between two neurons). The first two mathematicians to model these operations were McCulloch and Pitts[55] which served as the basis for the first learning algorithm, the Frank Rosenblatt's perceptron(1957)[61].

The neuron has a propagation function, also called transfer function, that transforms the inputs of the neuron with a weighted sum. The output of this function is passed to an activation function which activates the neuron or not if its input exceed a treshold value. The perceptron performed pattern recognition and learned to classify labeled examples. Thanks to the labeled data, the learning algorithm is able to slightly update the weights on the inputs implementing gradient descent. Rosenblatt proved the theorem below:

Theorem 2.1.1 *An elementary perceptron can separate any two non-intersecting sets of binary images[39].*

To be sure to correctly classify binary images is a good start but in most cases we need to classify many classes and single-layer perceptron is only capable of learning linearly separable patterns. This is one factor that made the research about AI stagnate for many years before the feedforward neural network with two or more layers (also called multi-layer peceptron) arrived[92] which is the arrival of deep learning. We will explain the multi-layer perceptron[43] more in details further.

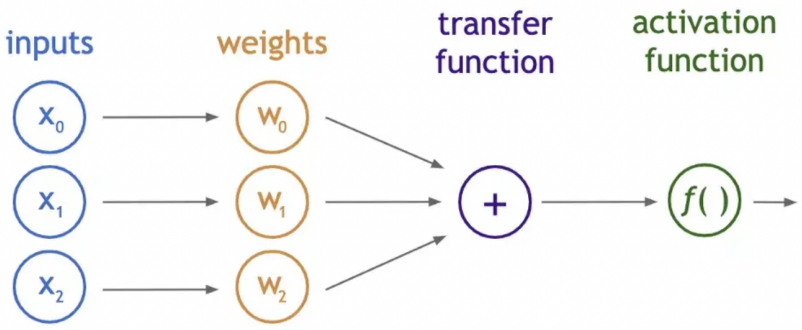


Figure 2.3: Schema of Perceptron. Illustration from [38]

Attention is drawn to the fact that the perceptron is inspired by the brain and not copied it. We can draw a parallel with the invention of the airplane, we were indeed inspired by the bird and its wings, but it would be wrong to say that we copied the bird. The same is true for Deep Learning. The human brain functions in a similar way but network is far more complex and each neuron of the network perform a separate task. Our understanding and our ability of deduction is superior. The deduction in Deep Learning is used in a linear and one-dimensional way[72]. We haven't found any very recent paper about the topic but it could be interesting to reevaluate this ability of deduction for modern models like ChatGPT.

2.2 Variety of Neural Networks: from MLP to modern architectures

2.2.1 Multi-layer Perceptron

A Multi-layer Perceptron (MLP)[43] is the evolution of the Single-layer Perceptron and a fully connected feedforward Articial Neural Network(ANN) where feedforward means that the connections between the nodes do not form a cycle. The network consists of at least thre layers of nodes: an input layer, one or multiple hidden layers and an output layer as we can see on the Figure 2.4. Its multiple layers and non-linear activation functions distinguish MLP from the Single-layer perceptron wich is totally linear.

Like explained before, the neurons from each layer receive the inputs through from the previous layers except for the first one. Each neuron from this layer treats the signal by summing the weighted values and then applies an activation function. The output of the output layer is a prediction based on the input data. During the training a process of backpropagation of the errors allows the network to iteratively minimze the error by adjusting the weights and biases of the neurons. The values of the weight give less or more power to a neuron and so control the influence of this one. These operations are repeated until the network is accurate enough on the training data[38].

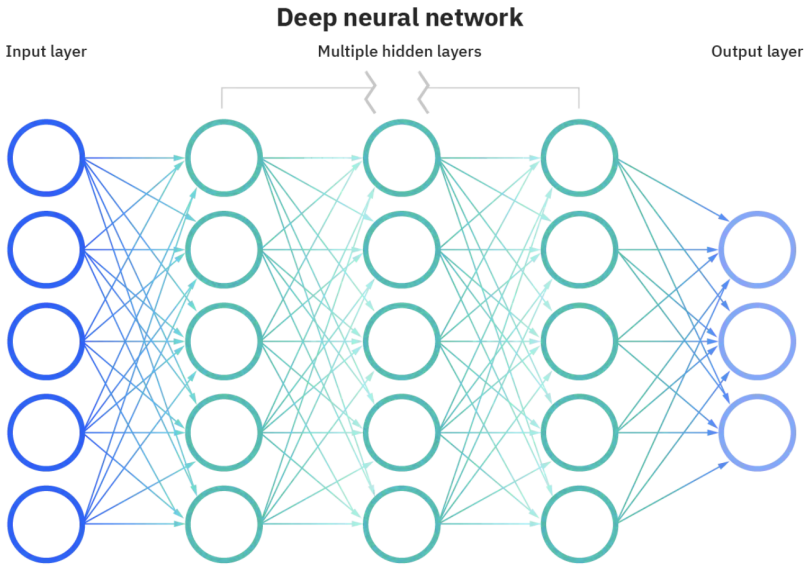


Figure 2.4: Example of structure of a Multi-layer Perceptron. Illustration from[32]

2.2.2 Convolutional Neural Network

Let's suppose that we want to classifiy images composed of a grid of M x N pixels, the input of a MLP vector would be a vector of size M x Nas the flattened version of the image. Knowing that the size of the M x N vector will probably be greater than one million, the number of parameters to be trained would be considerable for a MLP. In addition to being computationally intensive, such a number of parameters requires a large dataset and this algorithm does not take advantage of the image structure. To deal with these issues, the Convolutional Neural Network was introduced taking inspiration from Digital Image Processing and its main operation the image convolution [75].

Let's consider a monochrome image as a N x M matrix x_{ij} with i and j the entries for $i = 1, .., M$ and $j = 1, ..., N$. The convolution y of the image x with another image H, the kernel, of size K x K (with K typically a small integer like 5 or 7) is defined like that[5]:

$$y_{ij} = \sum_{u=1}^{K} \sum_{s=1}^{K} H_{us} x_{a(i,u),b(j,s)} \tag{2.1}$$

Where we can see a(i,u) and b(j,s) as i+u and j+s.

As we can see on the Figure 2.5 (first step of the convolution illustrated), a new image will appear by shifting the kernel H all over the image x. In this case, the kernel chosen is a filter to detect vertical edges. The goal is to combine such kernels to detecte more and more abstract and complex shapes of the image.

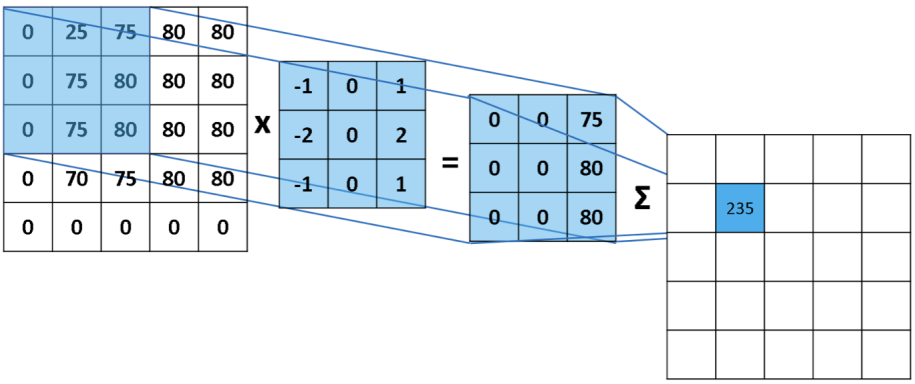


Figure 2.5: First step in the convolution process. Illustration from [56].

‘ We have seen that convolution is able to find the features of an image if we already know which kernel to use. That's why we need a learning algorithm and a

training dataset to learn which features to look for on the image and which kernels to use. That's the main idea behind the Convolutional Neural Network (CNN), to combine as best as possible a combination of kernels to spot the right shapes and textures in an image. This time, the input vector is a tensor of dimension M x N x D where M x N is the size of the image and D are the three channels (Red, Green and Blue). The architecture of the CNN is composed of three main types of layers:

1. **Convolutional layer:** this is the first layer to extract featyres from the input image. The convolution layer in CNN passes the result to the next layer once applying the convolution operation presented just before. It is important to note the choice of the dimension is crucial because this operation if usually reducing the size of the image. Indeed, if a m * m image is convolved with a n * n image the output image is of size (m - n + 1) * (m - n + 1). To avoid the image shrinks too much and that the edges of the image are less involved in the process, concept of padding and stride are used[63].

2. **Pooling layer:** this layer is used to reduce the size of the input by dividing the image into small clusters applying an operation on each cluster to combine neuron into one neuron. Typically, the cluster is of size 2 x 2 and the max of the four neurons or the average are computed to end up with a single neuron. These two process are called Max Pooling and Average Pooling. The main idea of this layer is to summarize the information into a small image.

3. **Fully-connected layer**: these layers are usually the last layers to map the layers to the output classes. The neurons are fully connected to together using on are two MLP layers.

An example of CNN structure for the classification of number is given below on Figure 2.6. As we can see, this architecture is composed of two module of convolution layer followed by a Max Pooling layer to end by two Fully-Connected layers to map the output of the two first modules to the 9 classes. At the end we obtain a tensor of probability for each class. This tensor is the probability estimated by the model that the input (number two in this case) belongs to each class.

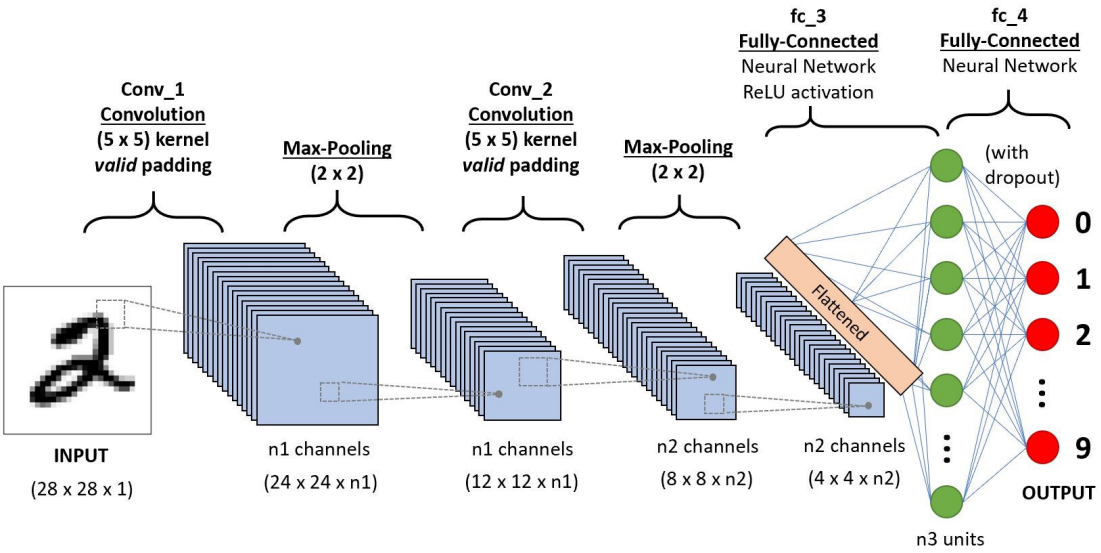


Figure 2.6: Example of usual CNN structure. Illustration from [69]

‘ In this document, we use four CNN models for the four domains composing the PACS dataset (Photo, Art, Cartoon and Sketch). Those models were provided by Politecnico di Torino and serve as witnesses. The goal is to optimize the accuracy on each target domain without optimizing these four CNN classifiers.

2.2.3 Transformers

In 2017, Google introduced by [84], a new model called transformer, a model composed of an encoder and a decoder (as we can see in the figure 2.4) and that is designed to process sequential input data, such as natural language. This architecture has deeply revolutionized the field of machine learning research. The succes of this model resides in the attention mechanism that looks at an input sequence and decides at each step which other parts of the sequence are important. Indeed, this notion of context was already present in the RNN's and the LSTM's architecture [73] but the reference window, that represents the context was much smaller. While with transformers, attention has theorically an infinite reference window given enough computational power [84]. Moreover, transformers use attention to boost the speed with which these models can be trained, now the input sequence can be passed in parallel. Parallelization allowed training on larger datasets and this led to the development of pretrained systems such as BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer), both trained on significant datasets in term of size.[93]

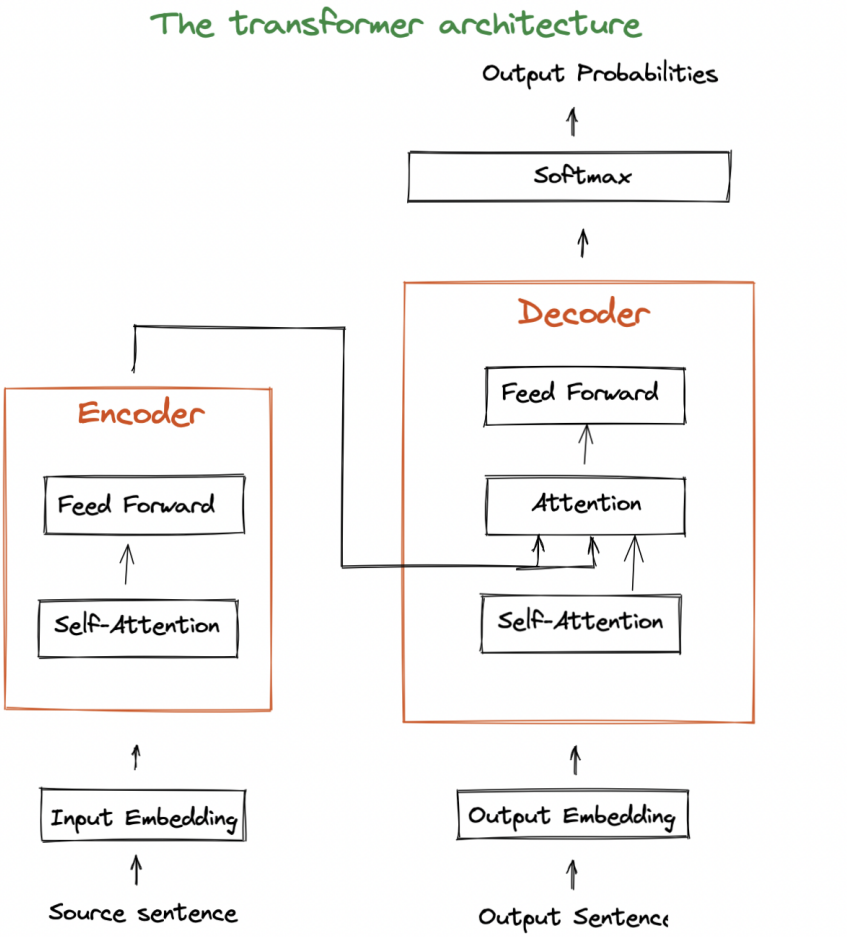


Figure 2.7: Representation of the transformer architecture

2.2.4 BERT

Bidirectional Encoder Representations from Transformers (BERT) is a transformer-based machine learning technique for natural language processing (NLP) pre-training developed by Google composed of a consequent dataset with 3.3 Billion words mostly from Wikipedia [91]. While NLP tasks have traditionally been solved by individual models created for each specific task, BERT made the difference by outperforming the state of the art models in the 11+ common NLP tasks like sentiment analysis, question answering, text prediction, ... [9] BERT is designed to pre-train deep bidirectional representations from unlabeled text. Than, the pre-trained BERT model can be fine-tuned with just one additional output layer[13]. The pre-training is done by jointly conditioning on both left and right context in all layers. Indeed, historically, NLP models could only read text input

sequentially – either left-to-right or right-to-left while BERT is able to read in both directions at once. That's why the word Bidirectional stand in Bidirectional Encoder Representations from Transformers (BERT)[51].

To pre-train the model, BERT uses two training strategies: Masked Language Model (MLM) and Next Sentence Prediction (NSP). For the first strategy, 15% of the words in each sequence are replaced with a [MASK] token before being fed into BERT and the model attempts to predict the original value of the masked words, based on the context provided by non-masked words.The final hidden vectors corresponding to the mask tokens are fed into an output softmax over the vocabulary, as in a standard LM[29]. The second strategy is the NSP, as input, the model receives pairs of sentences and learns to predict if the second sentence is the subsequent sentence in the original document. In the training, 50% of the second sentences are the subsequents sentences in the original document, while the other 50% are random sentences from the corpus. The disconnection between the random second sentence and the first sentence is taken as assumption. To help in this problem, A sentence embedding indicating if its Sentence A or Sentence B is added to each token. For a given token, The input representation of each token is constructed by summing the corresponding token, segment, and position embeddings [13] as seen in the Figure 2.6. During the training of the BERT model, MLM and NSP are used together with the minimization of the combined loss function of the two strategies.

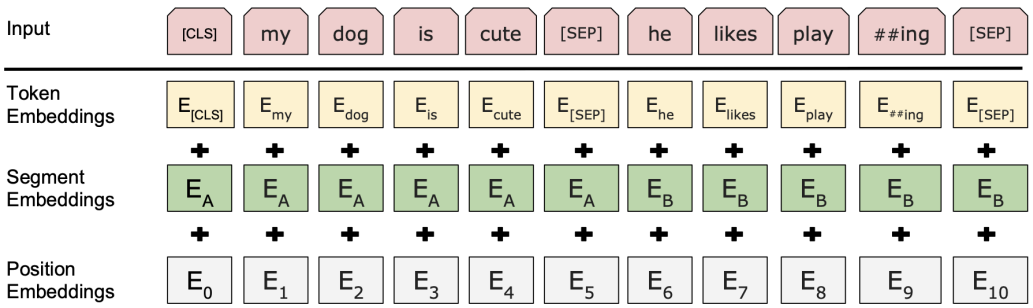


Figure 2.8: BERT input representation as the sum of the token embeddings, the segmentation embeddings and the position embeddings. Illustration from [13]

In this case, we will use Bert for the Text encoder part of our embedder where we will add a linear layer to map the embeddings to 256 dimensions compatible with the image embedding from the image encoder.

Chapter 3

Related works

Both, Domain Adaptation (noted as DA) and Domain Generalization (noted as DG) are concerned with reducing dataset bias. Their goal is to produce models that perform well on a target domain by training on labeled samples from the source domain(s). The difference between DA and DG is the availability of unlabeled samples from the target domain during training. In DG, there is no sample of the target domain available for the training [22]. Domain Generalization aims to incorporate knowledge from multiple source domains into a single model that could generalize well on unseen target domains.

In this section, like explained before, we will discuss the current state of research on Domain Generalization and models combining Visual and Language with particular attention to models taking advantage of massive use of data, "Zero-Shot Learning" type models such as CLIP or Dalle-2, its generative evolution.

3.1 Domain Adaptation

Machine learning includes several sub-discipline, one of them is domain adaptation wich deals with particular scenarios where a model trained on a source distribution is used on a different but related target distribution[28].Domain adaptation (DA) is the closest topic related to Domain Generalization.

Let's take the case where the data has been collected, annotated and the test set is from the same distribution as the training data. Those are ideal conditions. In practice, this may not always be the case and the test data comes from different distributions as the training data. That's where we can use domain adaptation in order to increase the efficiency of our model because Domain Adaptation align train and test extracted features and make them live in the same space[3].

To align the extracted features, the feature extractor has to fulfill two goals :

1. The features produced have to be meaningful for the classifier to optimize the choice of belonging to a class.

2. To make the features produced domain-wise indistinguishable, they must capture characteristics of the image regardless its specific domain.

That's where we introduce Domain-Adversarial Neural Network (DANN) as a good example of domain adaptaion because it is a specific implementation designed to satisfy both requirements.

DANN is based on a mapping between the source and the target domain. The goal of this mapping is to allow the classifier to perform on the source domain but also on the target domain. Features that are both discriminative and domain-invariant are the priority for the algorithm in term of learning [11].

The architecture is divided in three parts. First, the label predictor, that is the standard label classifier. Secondly, the domain classifier, its goal is to discriminate the features extracted between the two domains, thus the loss is normally backpropagated through that branch. And finally, the feature extractor which focuses on finding the most discriminative and domain-invariant features like we said earlier. It maximizes the Loss of the domain classifier negatively backpropagating the Loss. It tries to 'fool' the domain classifier by generating the most domain invariants features as possible.

3.2 Domain Generalization

The interest on domain generalization in the research field is more recent than domain adaptation[58] and is relatively new in the computer vision community. The first to study this issue was Blanchard et al., [16]. Indeed [16], was motivated by DG to work on a medical application called "solving automatic gating of flow cytometry data". The idea was to automate the process of classifying cells in patient's blood samples (e.g. to distinguish cells that are lymphocytes or not). To obtain this kind of technology is an issue of the utmost importance to improve the diagnosis of the health of patients. Especially than doing gating by hand is extremly time-consuming and requires expertise in the field. The research on the Domain Generalization problem comes from the fact that the distribution shift between different patient's data is substantial, so a classifier using data from historic patients does not generalize to new patients and collecting new data for model fine-tuning is not possible. That's how the research on the Domain Generalization problem is born. [103]

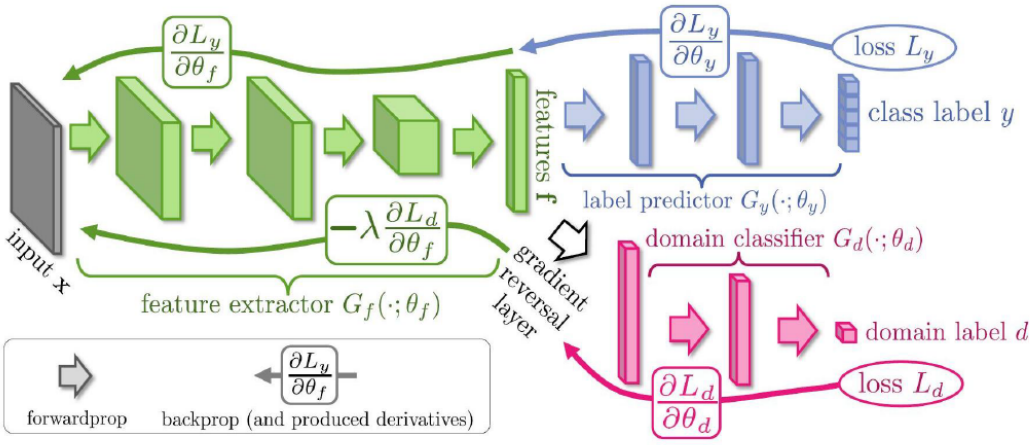


Figure 3.1: DANN architecture composed of a deep feature extractor (in green on the image), a deep label predictor (in blue) which together from a standard feed-forward network. To achieve Domain Adaptation, a domain classifier is added (in pink) connected to the feature extractor. The connection is made by the gradient reversal layer that multiplies the gradient by a negative constant during backpropagation to ensure that features distributions over the two domains are as difficult as possible to distinguish. Illustration and main ideas of the caption are from [98]

.

3.2.1 Domain Generalization approaches

Over the last ten years, numerous Domain Generalization methods have been proposed leading to a broad spectrum of methodologies, all of them studied in various applicaitons such as computer vision, NLP, medical imaging, reinforcement learning, etc. [103] In this section, we list existing DG methods into several groups based on the methology and the design suggested.

1. Domain Alignment methods are trying to make the model robust concerning the domain and that it should learn features invariant to the different source domains[101]. The central idea behind those methods is to minimize the difference among source domains by reducing the distance of learned representations between different domains. Eventually, if a feature is invariant to the source domain shift, it should also be robust to any unseen target domain [103].

Several statiscal distance metrics are used to measure the distance between distributions in order to achieve alignment such as the simple l2 distance, f - divergences, or the more sophisticated Wasserstein distance [85][103]. However,

designing an efficient domain alignment method requires important considerations such as "what to align?" and "how to align?".

To answer to the first question, let's recall that a domain is modeled by a joint distribution P(X,Y) given X the training data and Y the labelling data. Thanks to Bayes'theorem, this distribution can be decomposed into:

$$P(X, Y) = P(Y|X)P(X)$$

A common assumption in DG is to consider that the shift of distribution only occurs in the marginal distribution P(X) while the posterior distribution P(Y|X) remains relatively stable [57]. That is why many domain alignment methods are focused on the marginal distributions of the source domains [103]. However, other works suggest to align the posterior distribution [99].

There exists a lot of methods designed to answer the question "how to align?" that we will not develop in details in this document but here is a non exhaustive list of techniques used in the DG literature for distribution alignment: Minimzing Moments [34], Minimizing Contrastive Loss [52], Minimizing the KL Divergence [44], Minimizing Maximum Mean Discrepancy (MMD) [45], Domain-Adversarial Learning [49] [103]. Those methods have also been studied in the domain adaptation literature [19].

2. Meta Learning, also referred as learning-to-learning, can also be a great tool to deal with the DG problem. Meta-learning algorithms learn and make predictions by taking the output of other machine learning algorithms that learn from data [10]. The idea behind meta-learning is to expose a model to domain shift during training such that it can solve new learning tasks with only a small amount of training samples [103]. One of the meta learning approach most related to DG is Model-agnostic meta-learning (MAML) [15], splitting the source domains in meta-train and meta-test set. Than, a model is learned on the meta-train set with the goal of reducing the error on the meta-test set [8]. MAML was designed for parameter initialization such that the initial state of the model only requires a few gradients steps with a small amount of training data to produce good performance on a new task[15]. Meta-learning applied on Domain Generalization can only be applied to multi-source where domain labels are exisiting[103].

3. Data Augmentation Data scientists, machine learning engineer or any kind of profession linked to building a machine learning model is always desirous for more training data. Indeed, the generalization performance of the model often relies on the quantity and diversity of the training data[35]. One of the most popular

practice to avoid overfitting and improve generalization is data augmentation. This set of strategies allow models to be more robust against specific features on the seen domains in addition to being cheap computationally speaking. The field of possibilities in this matter is vast, it goes from random changing of color or background, rotation or even use of adversarial gradients.

Given x the images of the training data and y the corresponding labels, data augmentation is based on augmenting the pairs (x,y) with new pairs (A(x,y)) where A(.) is a transformation label-preserving. The challenge and the key to the success of the method resides in the design of A(.). According to [103], data augmentation methods are generally divided in four groups: image transformations, adversarial gradients, model-based augmentations and feature-based augmentations. *Zhou et al.* resume those categories with the Figure 3.2.

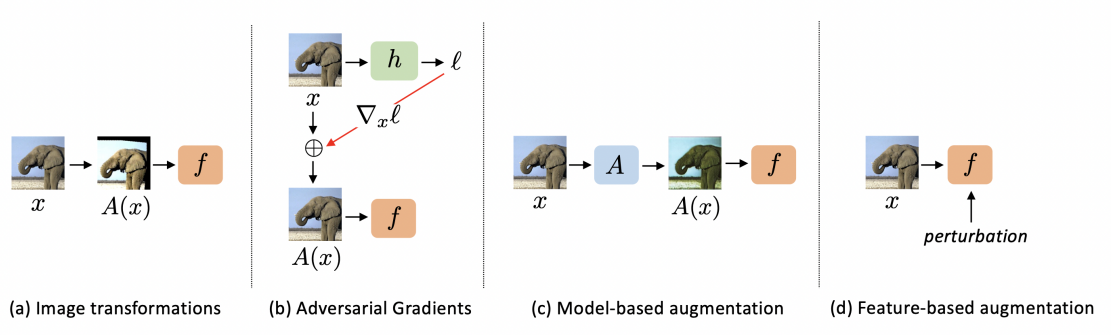


Figure 3.2: Data augmentation methods can be categorized into four groups; image transformations, adversarial gradients, model-based augmentations and feature-based augmentations. Illustration from [103].

This approach of Image Transformations is the most known and deals with a multitude of images transformations. Those augmentation operations include flipping, rotation, cropping, scaling, mirroring. Some are presented in Figure 3.3.

It is also possible to build a searching algorithm to determine a set of transformations that fit the best the target problem[86][103].

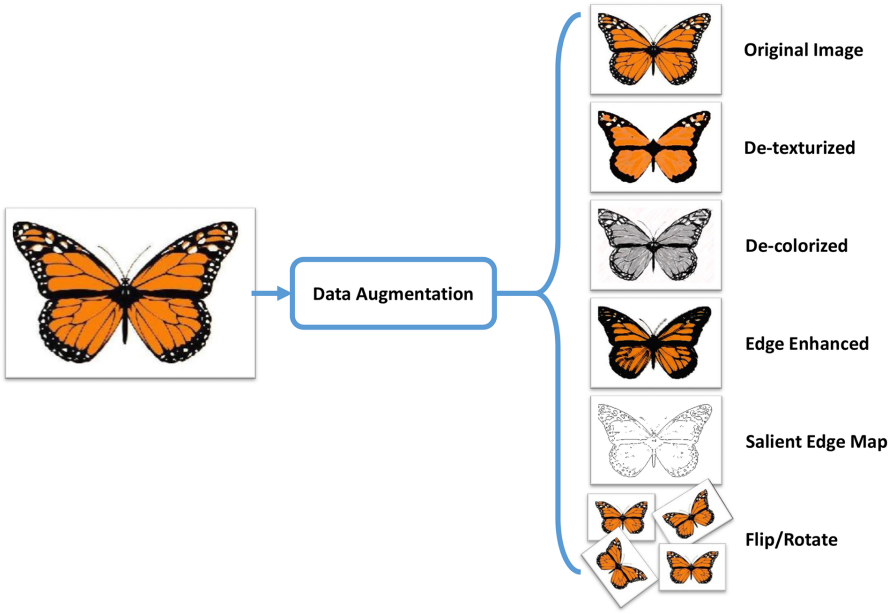


Figure 3.3: Common image transformations. Illustration from [2]

4. Regularization Strategies are based on some heuristics to focus more on features that capture the global structure and shape of objects instead on capturing local textures[103]. For this purpose, an example is that Huang et al. iteratively masked out over-dominant features with large gradients to force the model to be based on the remaining features[30]. Concerning CNNs, thoses model are indeed powerful but tend to overfit on source domains, that is why model regularization techniques like weight decay, early stopping and shake-skahe regularization are some potential ways to improve the Domain Generalization efficiency[30].

An advantage of these methods is that they do not require domain labels for learning and are independent to the other Domain Generalization techniques as domain alignmnent or ensemble learning. Therefore, it is possible to combine those methods[103].

5. Ensemble Learning methods train multiple classifiers to solve the same problem, in opposition to traditional approaches which consist to train only one learner from training data. Thoses methods are also called comittee-based learning or multiple classifier system [105]. The ensemble setting usually contains a certain number of learners that are called base learners named after being trained on a base learning algorithm as decision tree or neural network for example. In reality, the interest of ensemble learning resides in the fact that it is often able to boost

the weak base learners that are very poor in term of efficiency to strong learners able to provide accurate predictions [105]. In practice, constructing an ensemble is not much more computationnaly costly than building base learners since we usually need to generate a significant amount of base learners for parameter tuning or model selection.

In 2011, the ensemble called Exemplar-SVMs, a collection of SVM classifiers has demonstrated generalization performance on the object detection task [53]. Seeing the interest of such a method, Xu et al. have extended Exemplar-SVMs to Domain Generalization. Indeed, given a test sample, the classifiers with the highest prediction scores are selected to build the ensemble prediction [103].

CNNs shine at discriminative feature learning, the natural following step is to replace the SVM classifiers by CNN-based models for ensemble learning. The idea is than to learn domain-specific neural network specialized each in a source domain[103]. More recently, Zhou et al. [102] proposed Domain Adaptive Ensemble Learning (DAEL), a model composed of a CNN feature extractor shared across domains and multiple domain-specific classifier heads. Each classifier is an expert to its own domain but a non-expert to others. Collaboration is at the center of DAEL, by teaching the non-experts classifiers by the experts classifiers themselves, so they can share complementary information from each other in the purpose of being more efficient for an unseen target domain [35].

An alternative strategy is to decide how to compute the prediction, a usual solution is to simply compute the average of the predictions over all the learners with equal weights to build a prediction. Alternatively, a source domain classifier aiming to measure the similarity between the target and each source domain can be trained to create a weighted average [103]. The contribution of this master thesis goes in this direction trying to find values for those weights.

3.3 Vision and Natural Language

Computer Vision and Natural Language Processing (NLP) have witnessed considerable changes during the last years, reaching models even surpassing the human level performance. While the performances were skyrocketing, they were seen as separate fields. Nowadays, researchers are combining those two tasks to solve problems across many disciplines and have developped several very popular mutlimodal models. Multi-modals refers to the fact of learning from different types of modalities using the same model (images and texts in this case)[37]. This new trajectory started with an increase of the population to share and use files that are multimedia, that contain interrelated images, videos and natural language text. For example, a usual online article mixes a text written by a journalist and a photo related to the news content. The combination of Vision and Natural Language

have became the basis of a clear and unambiguous communication.

3.3.1 BERT-like architectures

After the incredible rise of transformers in NLP, it was manifest to expect many people trying to apply them in VL. To process images and text at the same time, the majority of papers treating the topic have been used some version of BERT [13], resulting in a explosion of BERT-like multimodal models: VisualBERT[46], ImageBERT[64], VL-BERT[77], VilBERT[50], [31], VD-BERT[89], ...

All of those models are based on the idea of processing language and images at the same time with a transformer-like architecture. Indeed we are using BERT for our VL application in this document.

3.3.2 Differents tasks tackled

Vision-Language(VL) models have gained a lot in popularity over the recent years due to the number of potential applications. [83] distinguishes three kind of applications; the generation task, the classification task and the retrieval task.

Retrieval tasks

This kind of task consist of retrieve and recuperate an information based on another information. The three main tasks in this area are:

- **Visual Retrieval**, retrieves images based only on a textual description [66][50][48].
- **Vision-Language Navigation** is guiding an agent navigating through a space based on textual instructions[106][23].
- **Multimodal Machine Translation** consists of translating a description to another using an additionnal visual input[97][80].

CLIP

A model of this category (Visual Retrieval) that is greatly linked to our document is Clip due to his ability of generalization and his method of learning. Its name is an abbreviation for Constrastive Language-Image Pre-training. The idea behind is to train a contrastive model that is able to link pairs of texts and images on a massive dataset of 400M pairs scraped on from the Internet[66]. Let's dive deeper to understand why it is related to our document.

- **Contrastive Loss:** Contrastive Loss is quite is similar to triplet loss used in this document. Contrastive models produce high score (similarity) for an image and a text from the same pair and a lower score for mismatched texts-images. Both approaches, contrastive loss and triplet loss, use cosine distances between vectors as prediction probabilities in a classification network. The positive example's distance and the negative examples' distances are treated as output probabilities, and cross entropy loss is used. Essentially, the approach involves using cosine distances as prediction probabilities and minimizing the cross entropy loss between the predicted and actual probabilities[94].

  During training, CLIP is presented with a large dataset of images and text, and it uses this data to learn a common feature space that captures the relationship between the two modalities. The model consists of two encoders: a Text Encoder (several Transformers) and an Image Encoder (ResNet and ViT). Encoders produce embeddings and a dot product is calculated with both embeddings, and it results in a similarity score. That's where we use contrastive loss, the goal is to minimize the distance between linked texts and images and to maximize this one between non related texts and images as we can see on Figure 3.4 [66].

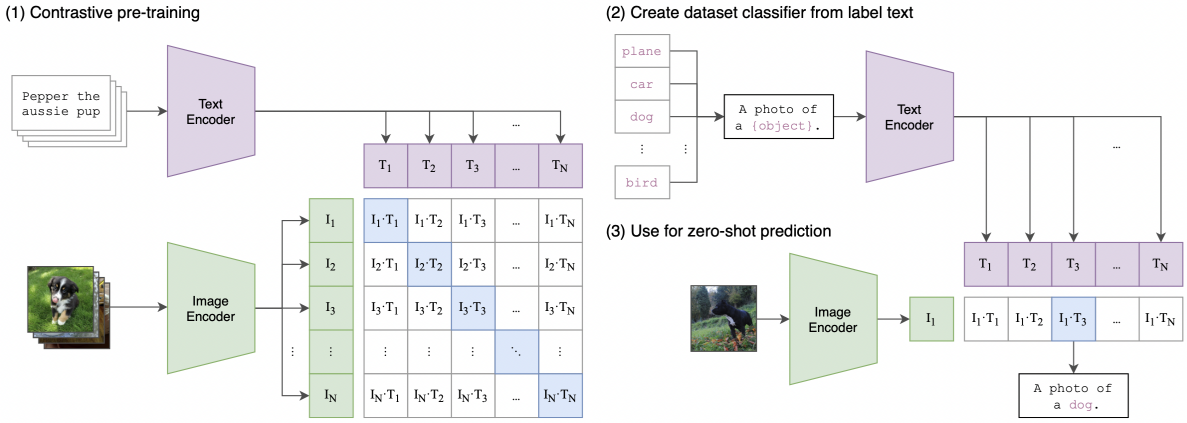


Figure 3.4: "Summary of the approach. While standard image models jointly train an image feature extractor and a linear classifier to predict some label, CLIP jointly trains an image encoder and a text encoder to predict the correct pairings of a batch of (image, text) training examples. At test time the learned text encoder synthesizes a zero-shot linear classifier by embedding the names or descriptions of the target dataset's classes." Illustration and caption from[66]

- **Generalization and resistance to data drift:** CLIP is a set of models that generalize for different vision tasks without being explicitly trained for these

tasks while usually SOTA models don't. CLIP is zero-shot classification as the model is not trained for a particular set of classes. As domain generalization, CLIP, is aiming to perform well across different domains. As we can see on the Figure 3.5, the performance of the best zero-shot CLIP model, ViT-L/14@336px, is compared with a model that has the same performance on the ImageNet validation set, ResNet-101. The results are outstanding compared to Resnet101 which is definitely not fitted to other domains.

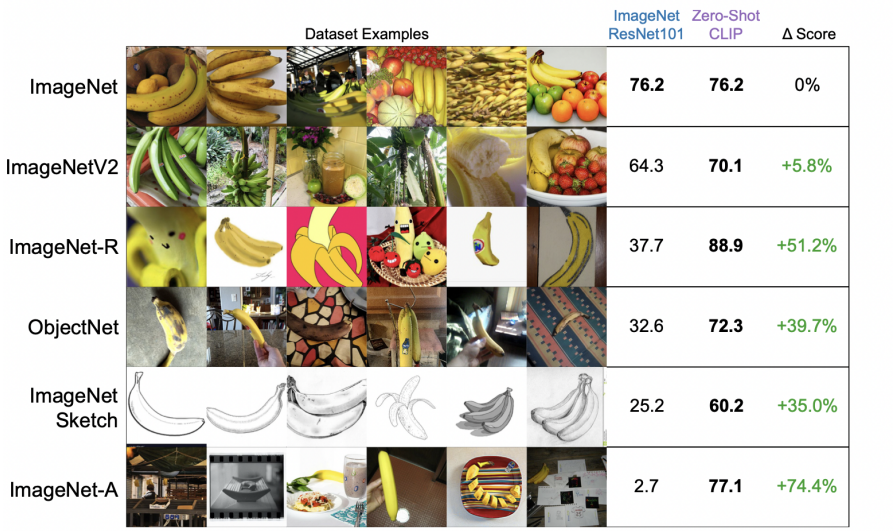


Figure 3.5: Comparison between CLIP and ResNet101 across several datasets. Illustration from [66]

.

Generation tasks

Another kind of task tackled by the VL models is the generative one that aims to create an text output from an image output or the opposite. In this category, as shown in Figure 3.6 created by [83], we can find in particular the following tasks:

- **Visual Question Answering**, process of providing an answer to a question given a visual input[87][4][76].
- **Visual Captioning** generates descriptions for a given visual input [88][18][36].
- **Visual Common-sense Reasoning** infers common-sense information and cognitive understanding given a visual input. [81][100].
- **Visual Generation** generates visual output from a textual input[67][47][68].

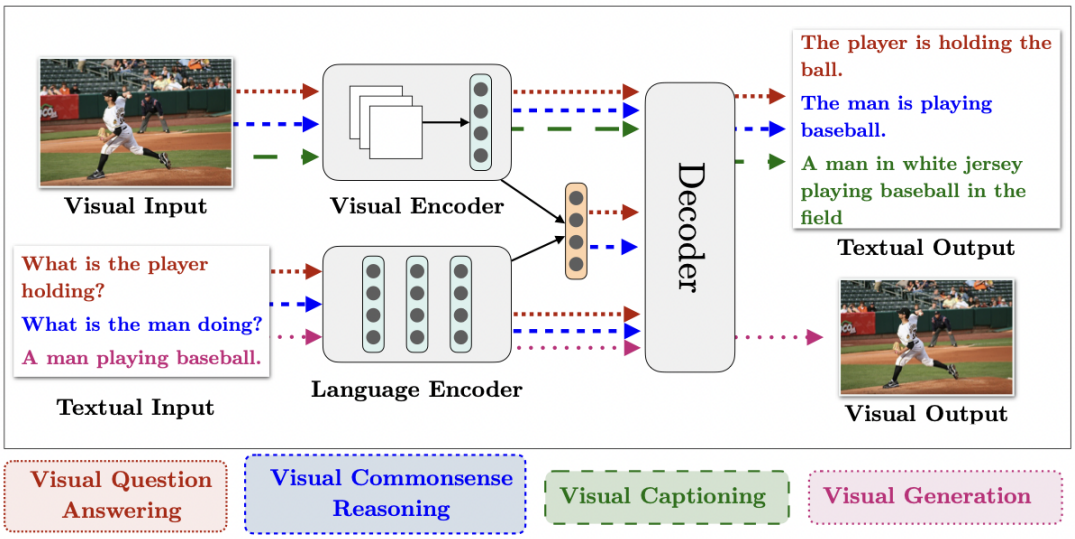


Figure 3.6: Representation of the most popular generative tasks. Illustration from [83]

This category was recently put in the spotlight and popularized by the general public thanks to DALL · E 2[67], which has been widely talked about, whether in philosophy, law or more generally on the place of art in society and its definition. Indeed DALL · E 2 which is a variant of CLIP is known for its ability to generate high-quality images from textual descriptions. It is a generative model that uses a transformer architecture to process input text and generate corresponding images. Some key features of DALL · E 2 include:

- The use of a large-scale dataset of text-image pairs, which enables the model to learn about a wide range of concepts and relationships between text and images[67].
- The generation of images from a large range of texts, including abstract concepts and combinations of words that have never been seen before, all of it with high-quality images with fine-grained details, such as textures, and background scenes[67].
- A transformer architecture, which allows the model to process input text in a more flexible and efficient manner compared to traditional convolutional neural networks.
- The use of a "discriminator" model to evaluate the realism and coherence of the generated images, which helps improve the overall quality of the generated

images. It is done by comparing the generated images to a large dataset of real images and learning to identify the characteristics that are common to real images but not to synthetic ones. The discriminator is trained using a process called adversarial training, in which it competes with the image generator to improve its ability to distinguish real from synthetic images. The generator tries to produce images that are as realistic as possible, while the discriminator tries to correctly identify which images are real and which are synthetic[82][6].

So DALL · E 2 is able to to generate images that are highly coherent with the given textual descriptions, making it a powerful tool for tasks such as image generation and data augmentation, we can see some examples of images output produced by this model on Figure 3.7.

Figure 3.7: Some samples of images produced by DALL · E 2. Illustration from [67]

Classification tasks

The last category of tasks adressed by VL is the classification approach. It aims at the prediction of a class label among a set of classes for a given input data. Two popular tasks of classification related to VL are:

- **Multimodal Affective Computing**[20][74][24] consists at interpreting visual affective activity from visual and textual input. (similar to multimodal sentiment analysis) [37]
- **Natural Language for Visual Reasoning** [104] [78][79]determines the correctness of a statement regarding a visual input.

Chapter 4

Method suggested

4.1 Formalization of the problem: Domain Generalization

Like stated in the introduction, Domain generalization (DG) aims to incorporate knowledge from multiple source domains into a single model that could generalize well on unseen target domains.

Let's formalize Domain Generalization according to the definitions gave by Jindong Wang[35];

Definition: A domain is composed of data that are sampled from a distribution. We denote it as $S = \{(x_j, y_j)\}_{j=1}^{n} \sim P_{XY}$, where $x \in \chi \subset \mathbb{R}$, $y \in \mathcal{Y} \subset \mathbb{R}$ denotes the label, and P_{XY} is the joint distribution of the input sample and output label. X and Y denote the corresponding random variables.

Definition: Domain Generalization As shown in the figure below, in domain generalization, we are given M training (sources) domains $S_{train} = \{S^i | i = 1, ..., M\}$ where $S^i = \{(x_j^i, y_j^i)\}_{j=1}^{n_i}$ denotes the i-th domain. The joint distributions between each pair of domains are different: $P_{XY}^i \neq P_{XY}^j, 1 \leq i \neq j \leq M$.

Domain generalization tries to identify a generalizable predictive function $h : \chi \to \mathcal{Y}$ from the M training domains to achieve a minimum prediction error on an unseen test domain S_{test} (i.e., S_{test} cannot be accessed in training and $P_{XY}^{test} \neq P_{XY}^i$ for $i \in 1, ..., M$):

$$\min_h \mathbb{E}_{(x,y)\in S_{test}} |\ell(h(x), y)| \tag{4.1}$$

where $\mathbb{E}$ is the expectation and $\ell(.,.)$ is the loss function[35].

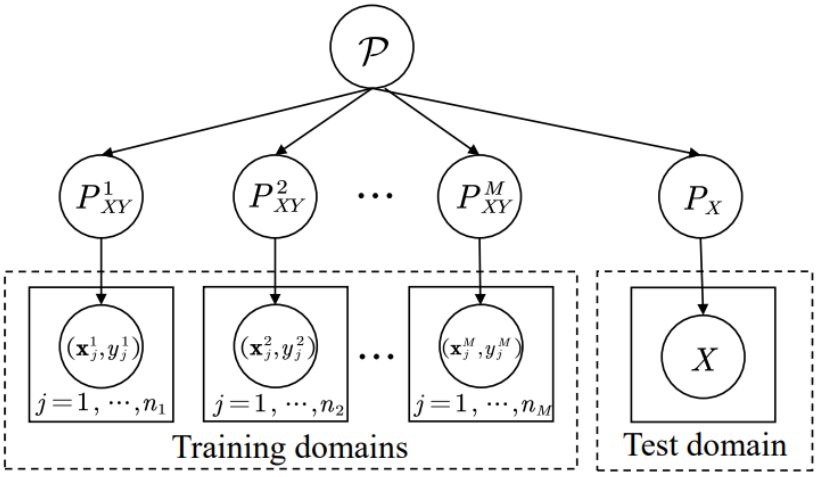


Figure 4.1: Illustration of domain generalization from [35] adapted from[7]

4.2 Improving DG using natural language

As explained before, one common way to deal with DG is to train a classifier on each source domain and the target prediction is simply the average of the output given by each classier (with a softmax function applied at the end). This method has the advantage to be simple but doesn't reflect the relative distances between each source domain as you give as much importance to each one of them. Our solution is to train a coherent embedder able to reflect the distances between each domains and to give a weight to each domain thanks to those distances. The target prediction is than the weighted average where we apply a softmax layer.

Our contribution and our work is than:

- The annotation of the dataset PACS.
- The analysis and the adaptation of the code produced by [96] for our task and our dataset.
- The training of the embedder based on the code of [96].
- The conception of the method to classify a new image of the target domain.

- The experimental part including the optimization of the differents parameters and the comparison with variants of the method.

4.3 Method of classification

In order to classify a new image from the target domain, we designed the next steps:

1. Train one classifier f on each source domain i, $S_{train} = \{S^i | i = 1, ..., M\}$.
2. Send each image j from each source domain in the image encoder z to embed the image and obtain the embedding z(j).
3. Compute the barycenter a of each domain i; $a_i = \frac{\sum_{j=1}^{n} z(j)}{n}$.
4. Compute the distances in the feature space between the embedding of the new image k and each barycenter with a cosine similarity function between the tensor of the barycenter and the tensor generated by the image encoder with the new image. Let's denote b as the embedding of the new image k, $b = z(k)$. Than the relative distance is $w_i = cosine\_sim(a_i, b)$
5. Send the new image we want to classify k on each classifier to have a number of output equal to the number of source domains $output_i = f_i(z(k)$.
6. Compute the total output as the weighted sum with the distances as weights $output = \frac{\sum_{w_i}^{M} w_i * output_i}{\sum_{w_i}^{M} w_i}$.
7. We obtain a vector of likelihood for each category and we take the maximum to get the category predicted $l = softmax(output)$.

A variant of this method that we also tested is to modify the step 4. In this variant, we are not computing $w_i = cosine\_sim(a_i, b)$ where a_i is the embedding of the barycenter of the source domain i and b is the embedding of the image to classify but $w_i = cosine\_sim(a_i, c)$ where c is the embedding of the barycenter of the target domain. This means that in this case, the weights w_i keep the same values for every images but the outputs change.

4.4 Embdeder detailed and training

Preliminary to these steps, we have to train the best model as possible to obtain a coherent embedder composed of an image encoder and a text encoder. We will

explain in this section how we trained it (the metric learning approach and a variation call hard negative mining) and where this way of training comes from.

4.4.1 The basis of the embedder: Describing Textures using natural language

Our embedder and his architecture is mostly insipired by Chenyun Wu[96] where we modified some functions and most of the parameters. Let's resume briefly this paper before explaining why it is pertinent to use this architecture as basis.

Textures in natural images can be characterized by color, shape, periodicity of elements within them, and other attributes that can be described using natural language. In the article of Chenyun Wu [96], they study the problem of describing visual attributes of texture on a novel dataset containing rich descriptions of textures. The definition of texture in this context is this one : Local patterns such as material and color that excludes the effetcs of shape, object category, or other high level cues[95].

For this purpose, 5 different annotators on Amazon Mechanical Turk were hired to create the descriptions, asking them to describe the texture using natural language with at least 5 words. They worked on the dataset: Describable Textures in Detail Dataset(DTD) coming from the article: "Describing textures in the wild"[12].

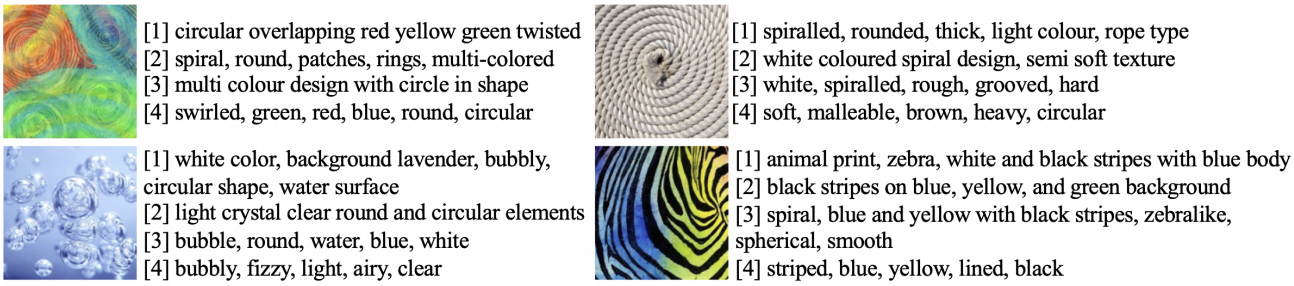


Figure 4.2: Examples of images annotated from [96]. The dataset is composed of textures images. with natural language description that provide fine-grained details for various aspects of texture such as color composition, shapes and materials.

Using this dataset they defined three tasks:

1. **Phrase retrieval**, given an image, rank the 655 description.
2. **Image retrieval**, given a description, rank the images.

3. **Description generation**, given an image, generate a new description.

Even if we don't work on the same tasks, they used one method to solve thoses tasks that is useful in our case, the metric learning approach explained in more details further. The metric learning approach aims to learn a common embedding over the images and phrases such that nearby image and phrase pairs in the embedding space are related. That's exactly what we need, to get reliable distance between our image from an unseen domain and the other domains, we have to be able to generate a coherent and realistic feature space.

The choice of this algorithm and architecture is based on several factors. Firstly, The results of the [96] paper are promising and their way of treating the annotations was easily adaptable for our case. But the most important is that the dataset used is abstract and low level objects composed of textures, shapes and colors nuances. Indeed, in our experiments we tested two setups, the training of the model from zero and the fine-tuning. This is why having a good representation of low level aspects of images is important for the fine-tuning.

A next step could be to try other architectures like CLIP[66] for example that is using Contrastive Loss (similiar to Triplet Loss) and that contains already a good internal representation of many common domains. For the differences between Triplet Loss and Constrative Loss, Triplet Loss is less greedy because the condition is already satisfied when the differents samples are already well spaced. Also, Contrastive Loss reaches a local minimum earlier due to the consideration of the similar pairs. Indeed, Constrastive Loss consider the margin only when it compares dissimilar pairs and doens't consider the position of the similar pairs at this moment[71].

4.4.2 Architecture of the embedder

As mentioned before, the architecture is the same as the one used in [96].The embedder is composed of two components, the image encoder and the text encoder detailed below:

1. **Image encoder:** For embedding images, the encoder part from ResNet101[26], specifically is used and activations from layer 2 and layer 4 of ResNet101 are selected. We add an additional linear layer with 256 units resulting in the embedding dimension $\psi(I) \in \mathbb{R}^{256}$.

2. **Text encoder:** For embedding phrases (textual descriptions) we use pre-trained BERT[13] model with its own tokenizer, and outputs the average of last hidden states of all tokens in the phrase P. To compute the final embedding of the phrase $\phi(P)$, we add a linear layer to map the embeddings

to 256 dimensions compatible with the image embeddings. The pre-trained BERT model is frozen during training.

4.4.3 Metric Learning approach: Triplet Loss

The metric learning approach aims to learn a common embedding over the images and phrases such that nearby image and phrase pairs in the embedding space are related. In the paper "Describing Textures using natural language" [96], they use the metric learning for learning a joint embedding of images and phrases. These embeddings are then used to retrieve the nearest images from a specific description or the opposite, to retrive the nearest description from a specific image. In our projetct, we will use these embeddings and especially the textual embeddings to create a coherent a reasonable feature space across the visual domains of the PACS dataset.

We adopt the standard metric learning approach based on triplet-loss. Consider an embedding of an image $\psi(I)$ and of a phrase $\phi(P)$ in $\mathbb{R}^d$. Where d is the dimension of the embedding (usually 256). Denote $\| \psi(I) - \phi(P) \|_2^2$ as the squared Euclidean distance between the two embeddings. Given an annotation (I, P) consisting of a positive (image, phrase) pair, we sample from the training set a negative image I' for P, and a negative phrase P$'$ for I. We consider two losses which are RelU functions; one from the negative phrase:

$$L_p(I, P, P') = max(0, \alpha + ||\psi(I) - \phi(P)||_2^2 - ||\psi(I) - \phi(P')||_2^2) \tag{4.2}$$

and another from the negative image:

$$L_i(P, I, I') = max(0, \alpha + ||\psi(I) - \phi(P)||_2^2 - ||\psi(I') - \phi(P)||_2^2) \tag{4.3}$$

Where α is the margin loss, a parameter used to add more robustness to the RelU functions and to keep negative samples far apart.

The metric learning objective is to learn embeddings ψ and ϕ that minimize the loss $L = \beta_1 L_p + \beta_2 L_i / \beta_1 + \beta_2$ over the training set [96]. Where the parameters β_i are the weights to give more importance to L_p or to L_i. If β_1 is bigger than β_2 it means that the model gives more importance to the negative phrase and vice versa.

To resume it more simply, we select a pair of related image and text I-P. We select randomly an image called I' and randomly a phrase called P'. For the first step, we define I as the anchor, P as the positive and P' as the negative and we will minimize the distance between then anchor and the positive and maximize the distance between then anchor and the negative. This step is illustrated on the Figure 4.3 below. For the second step, we will do the same but this time, we define

P as the anchor, I as the positive and I' as the negative. The total loss function is the average of the first step and the second step averaged by the β_i.

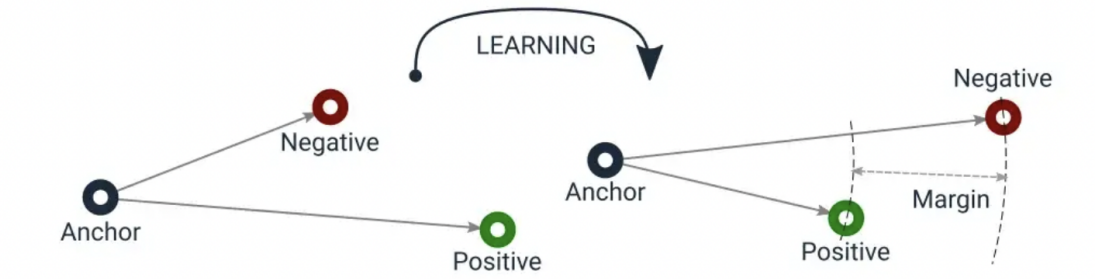


Figure 4.3: Triplet loss illustrated from [71]

4.4.4 Hard Negative Mining

Before, the negative image I' and the negative phrase P' are chosen randomly in the dataset. In order to get more "interesting" tuples to train the network, we will make the problem harder using the technique of Hard Negative Mining. This technique is based on selecting P' such that $\phi(P')$ is close to $\psi(I)$ and I' such that $\psi(I')$ is close to $\phi(P)$[33].

In other words, $\forall(I,P), P' = argmin_P^* ||\psi(I) - \phi(P^*)||_2^2$ and $I' = argmin_I^* ||\psi(I^*) - \phi(P)||_2^2$ At each iteration, we perform this kind of mining to build our minibatch. This method is computationally demanding because we have to embed each image and each description for each positive pair image-phrase (I,P). It's worth it, without Hard Negative Mining, I' and P' are chosen randomly and it happens quite often that I',P' are already well spaced from I,P. This scenario leads to a lack of learning because since we use a RelU function if $||\psi(I) - \phi(P)||_2^2 \leq ||\psi(I) - \phi(P')||^2$ and/or $||\psi(I) - \phi(P)||_2^2 \leq ||\psi(I') - \phi(P)||^2$ than the Loss will be equal to 0. We can see graphically on the figure below 4.4 this kind of scenario. To avoid this scenario we use the parameter α but sometimes it is not enough and the Hard Negative Mining is necessary.

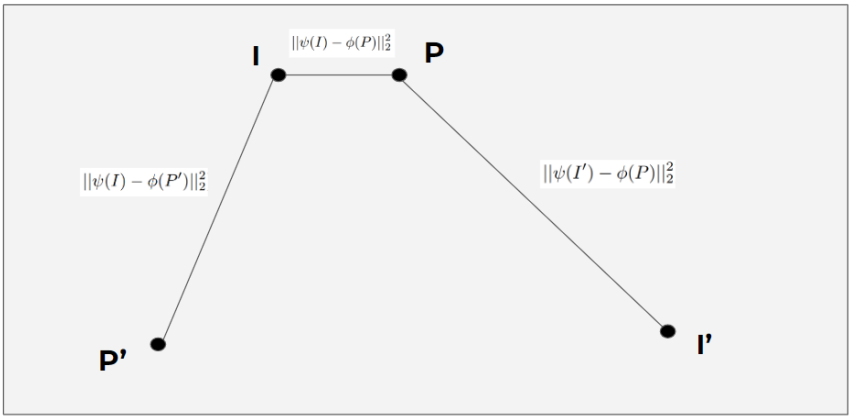


Figure 4.4: Representation of a possible output of positive and negative images and sentences in the feature space where I and P are already well spaced.

Chapter 5

Dataset : PACS

5.1 Presentation

The reference dataset here is PACS, it is an image dataset for domain generalization. PACS consists of four domains, each one for each letter of the name PACS. Namely Photo (1,670 images), Art Painting (2,048 images), Cartoon (2,344 images) and Sketch (3,929 images). Each domain contains 7 categories; dog, elephant, giraffe, guitar, horse, house and person[62]. Out of this dataset, we manually corrected textual descriptions in natural language for 3200 images of the PACS dataset, 800 of each domain and the distribution between each category is also respected.

Each description is made of 9 criteria :

1. **Level of Details**: If the image is rich in detail or it is a raw representation (e.g., low level details/mid level details/high level details)
2. **Edges**: Description of the contours of the objects (e.g., definite/precise/neat strokes, definite/precise/neat brush strokes etc.)
3. **Color saturation**: The intensity and brilliance of colors in the image (e.g., low saturation/ high saturation, light reflections, etc.)
4. **Color shades**: If there are shades of colors in the image (e.g., colorful, contrasting/black and white/bland colors, etc.)
5. **Background**: Description of the background (e.g., monochrome/white/grayscale, etc.)

6. **Single instance**: : If the image is composed by a single instance or multiple instances of the same object (e.g., single instance/multiple instances, etc.)
7. **Text**: If there is text in the picture (e.g., without text/sparse text/dense text in bottom left corner, etc.)
8. **Texture**: If there is a visual pattern that is repeated over the whole picture (e.g. without texture/painting strokes, etc.)
9. **Perspective**: If the three-dimensional proportions of the object parts are realistic (e.g. without perspective/with perspective, realistic/without perspective, unrealistic, etc.)

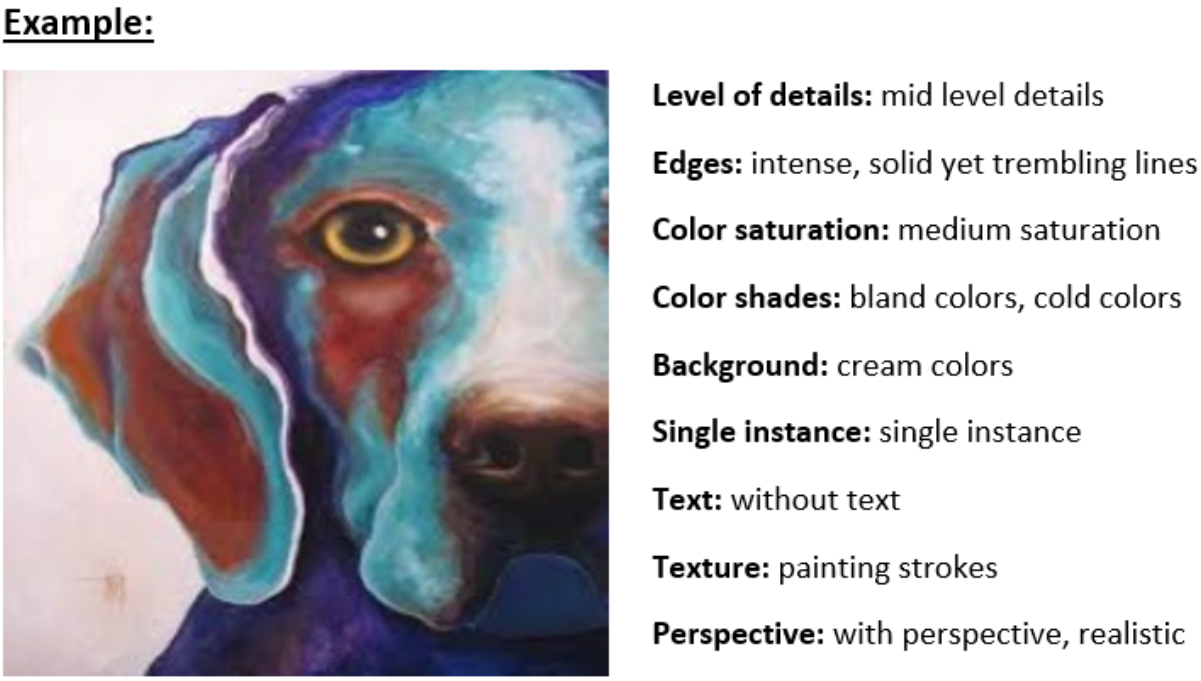


Figure 5.1: Sample from PACS dataset [62]

5.2 Annotations

For the class of Machine Learning at Polito, eight groups of students manually wrote the descriptions for 400 images each. A significative part of this work was to

correct the annotations made by the students and especially to put some coherence between them in order to create a training dataset of 3200 images, so 28800 descriptions/annotations . In total we had to modify 61,47% of the annotations, more than 17.500 annotations. To annotate faster, we created a small program available on this github: `https://github.com/Martial-Wsl/PACS-Annotater`. It allows to type very quickly the most popular annotations using key combinations. In the appendix section, a more precise analysis of which type of descriptions are the most modified is available for each group (except the groups; Group 5 AML, Group 2 DAAI, Group 3 DAAI who didn't respect the format of the 9 criteria).

For example, we can see on the Figure 5.1 below that we had to modify and correct 50,60% of the annotations especially for the criteria; details, saturation, shades and perspective.

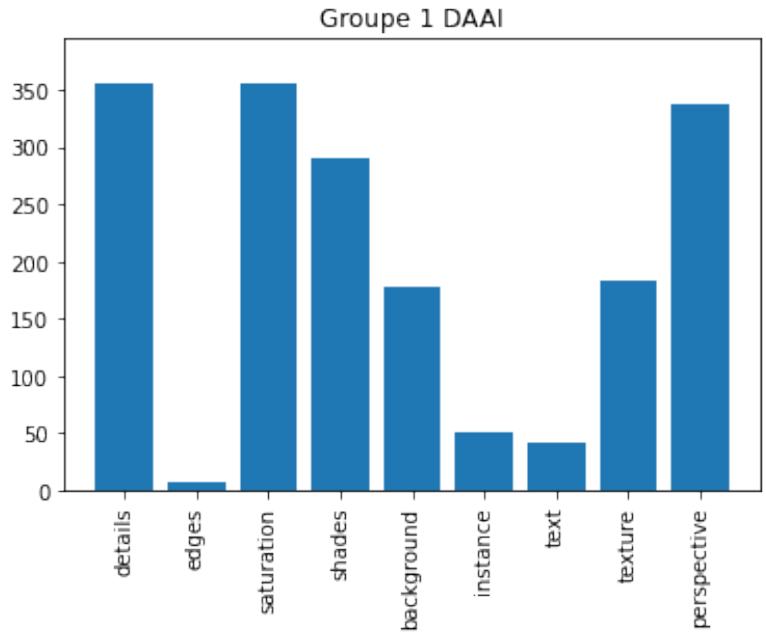


Figure 5.2: Number of differences between student of Groupe 1 DAAI and actual used descriptions (50,60%)

Chapter 6

Results

6.1 General overview

Let's define the baseline, the value to beat to attest if our method is promising or not. The baseline is the common method of averaging the outputs wihtout weights. The prediction on the target domain is the average of the three predictions generated by the classifiers trained on the three sources visual domains of the PACS dataset. Specifically, these classifiers consist in a linear layer (that give as outputs prediction's probabilities for the 7 object classes of the PACS dataset) on top of a pretrained ResNet18 (used as feature extractor). Finally the final outputs get passed inside a softmax function. The goal of this project is to improve the accuracy provided by the baseline.

On the table below, we compare 3 different approaches. (a) is the baseline explained just above. The second model, (b) is the model trained by the team of Chenyun Wu on the DTD dataset, so without using the PACS dataset in the training of the model used to embed. Finally (c), is our best model. It is the model generated from scratch (no fine-tuning of the DTD model), using Hard Negative Mining and with the parameters : Margin Loss = 1.5 and the weights of the Loss function $\beta_1 = 0.5$ and $\beta_2 = 1.5$.

	Art Painting	Cartoon	Sketch	Photo	Average
(a) baseline	67.63	57.12	60.40	94.49	69.91
(b) DTD	70.21	57.98	62.03	94.85	71.27
(c) our best	**72.07**	**60.27**	**63.91**	**95.88**	**73.03**

6.2 Analysis and path traveled step by step

In this section, we will go through the path of the different models trained to obtain the best model. Let's detail first, a few characteristics of the training method:

- **Training time:** For each of these models it takes approximately 24 hours and one hour to evaluate the model. For the models trained with Hard Negative Mining, this time raises to 3 days of training.
- **Epoch and evaluation:** The model is trained on maximum 6 epoch and the size of the batches is 16. Every 100 steps, we evaluate the efficiency of the model. The evaluation is the two first tasks of the "Describing Textures using natural language" paper, so Phrase retrieval and Image retrieval. The metric is than the sum of the average of the first task and the average of the second task.
- **Decay:** At each evaluation, if the metric outpasses the best one until this moment, the training pursue. If the best metric met does not increases 10 times in a row, there is a decay of the learning rate parameter, the learning rate decreases by a factor 10. After 20 times, the learning rate decreases again by a factor 10. Thoses decays optimize the convergence of the model. After 40 times, the training stops even if it didn't reach 6 epochs.
- **Difference BEST checkpoints and LAST checkpoints:** In the tables below, there is for each set of parameter two models. The first one is the BEST checkpoints, those are the checkpoints saved when the evaluation was at the higher score. The second row, is the LAST checkpoints, those are the checkpoints saved when the training stops. The checkpoints are the weights of the neural network at a precise moment, it represents the state of the model at this specific moment.
- **Difference "From Scratch" and "Fine tuning":** In the tables below the titles begins either by " From Scratch" either by "Fine Tuning". "From Scratch" means that we didn't use the model generated on the DTD dataset as base. While "Fine Tuning" means that we used transfer learning, we used the model generated on the DTD dataset as base and we fine-tuned this model.
- **Difference "First Method" and "Second Method":** Also for the titles, they contain two types of methods. The first method is the method explained in the method of classification. To compute the output of the image we want to classify, $output = w1 * output1 + w2 * output2 + w3 * output3 / w1 + w2 + w3$. Where $w_i = cosine\_sim(ai, b)$, ai is the embedding of the barycenter of

the source domain i and b is the embedding of the image to classify. The second method is the variant explained in the method of classification. The modification in the second method is that this time, $w_i = cosine\_sim(ai, c)$ where c is the embedding of the barycenter of the target domain. In this case, the weights keep the same values for every images but the outputs change. While in the first method the weights changed for every image.

6.2.1 Learning Rate

The first parameter explored is the Learning Rate. The initial learning rate is 0.0001. As we can see on the table below, each model beats the baseline but is still below the DTD model. The values of this parameter are chosen arbitrarily, as this parameters is goint to automatically decay during the training, we decided to increase it. We can observe that the models on the rows "LAST" are less or even accurate as the models "BEST".

From Scratch: Learning Rate

Learning Rate	Checkpoints	Art Painting	Cartoon	Sketch	Photo	Average
0.0001						
	Baseline	67.63	57.12	60.40	94.49	69.91
	DTD	70.21	57.98	62.03	94.85	71.27
0.0001						
	BEST	70,16	57,42	61,80	94,91	71,07
	LAST	69,38	57,38	61,00	64,97	70,61
0.0005						
	BEST	68,55	57,17	61,03	94,49	70,31
	LAST	67,82	51,17	60,37	94,55	69,98
0.001						
	BEST	68,21	51,12	60,47	94,61	70,10
	LAST	68,21	51,12	60,47	94,61	70,10

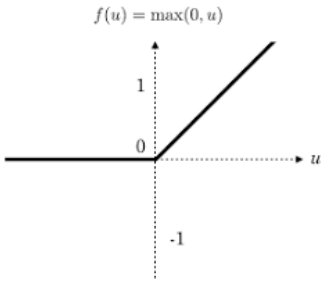
6.2.2 Margin Loss

The second parameter that we explored is the Margin Loss. The Margin Loss is the parameter α of the two equations composing the Loss using the triplet loss approach:

$$L_p(I, P, P') = max(0, \alpha + ||\psi(I) - \phi(P)||_2^2 - ||\psi(I) - \phi(P')||_2^2) \tag{6.1}$$

$$L_i(P, I, I') = max(0, \alpha + ||\psi(I) - \phi(P)||_2^2 - ||\psi(I') - \phi(P)||_2^2) \tag{6.2}$$

The Margin Loss translates the RelU function to the right. It makes the function more robust by penalizing the cases at the limit.



For this parameter, we can observe that the best results are obtained "From Scratch" and that this time, we have 4 models beating the DTD model: "From Scratch/Margin Loss = 1.5/BEST", "From Scratch/Margin Loss = 1.5/LAST", "From Scratch/Margin Loss = 1.6/BEST" and "Fine Tuning/Margin Loss = 1.5/BEST". The best one is "From Scratch/Margin Loss = 1.5/BEST" with an accuracy equals to 71,80. If a model beats the DTD model, it is underlined.

We can notice that the models LAST are again less accurate. We can also notice that the first method is consistently more efficient.

From Scratch: Margin Loss

Margin Loss	Checkpoints	Art Painting	Cartoon	Sketch	Photo	Average
0						
	BEST	67,63	57,25	60,40	94,55	69,96
	LAST	67,63	57,25	60,40	94,55	69,96
0.5						
	BEST	67,97	57,21	60,40	94,55	70,03
	LAST	69,38	57,00	60,83	94,67	70,47
1.3						
	BEST	70,36	56,31	62,79	95,15	71,15
	LAST	70,56	57,12	61,77	95,15	71,15
1.5						
	BEST	**71,53**	**57,38**	**62,84**	**95,45**	**71,80**
	LAST	**70,61**	**58,11**	**61,95**	**95,39**	**71,51**
1.6						
	BEST	**69,68**	**58,49**	**62,03**	**94,97**	**71,29**
	LAST	71,04	56,19	61,16	95,09	70,87

Fine Tuning: Margin Loss - First Method

Margin Loss	Checkpoints	Art Painting	Cartoo	Sketch	Photo	Average
1						
	BEST	70,80	57,64	61,57	94,79	71,20
	LAST	70,61	57,17	60,93	94,91	70,90
1.3						
	BEST	70,90	57,17	61,21	94,73	71,00
	LAST	68,07	51,21	60,40	94,61	70,07
1.5						
	BEST	**71,48**	**57,00**	**62,36**	**94,97**	**71,45**
	LAST	68,80	57,17	60,86	94,61	70,36
1.6						
	BEST	70,07	56,78	61,19	94,73	70,69
	LAST	71,24	57,21	61,47	94,97	71,22

Fine Tuning: Margin Loss - Second Method

Margin Loss	Checkpoints	Art Painting	Cartoon	Sketch	Photo	Average
1						
	BEST	70,75	57,12	61,64	94,67	71,05
	LAST	70,41	56,53	61,06	94,67	70,67
1.3						
	BEST	70,80	56,53	61,21	94,67	70,80
	LAST	68,12	57,21	60,40	94,55	70,07
1.5						
	BEST	71,44	55,84	62,20	94,97	71,11
	LAST	68,75	56,91	60,86	94,55	70,27
1.6						
	BEST	70,02	56,27	61,16	94,55	70,50
	LAST	71,09	56,53	61,39	94,85	70,96

6.2.3 Weights Image-Sentences

The next parameter explored is the "Weights Image-Sentence", that is the β_i in the Loss function $L = \beta_1 L_p + \beta_2 L_i / \beta_1 + \beta_2$. If β_1 is bigger than β_2 it means that the model gives more importance to the negative phrase and vice versa, if β_2 is bigger than β_1 it means that the model gives more importance to the negative image.

For this parameter we tested all the combinations with the features; Margin Loss = 1, Margin Loss = 1.5 (those margins gave the best results earlier), $\beta_1 = 0.5/\beta_2 = 1.5$, $\beta_1 = 1.5/\beta_2 = 0.5$ "From Scratch", "Fine Tuning". It appears that we have several models beating the DTD model.

The best model gave an accuracy of 72,24 with the combination of features; From Scratch, Margin Loss = 1.5, $\beta_1 = 0.5/\beta_2 = 1.5$.

From Scratch: Weights Image-Sentence - Margin Loss - First Method

W_IS	Checkpoints	Art Painting	Cartoon	Sketch	Photo	Average
wis0515_margin15						
	BEST	**70,65**	**59,34**	**63,32**	**95,63**	**72,24**
	LAST	70,12	57,00	61,70	94,85	70,91
wis1505_margin15						
	BEST	68,70	57,21	60,55	94,61	70,27
	LAST	68,90	57,21	60,96	94,61	70,42
wis0515_margin1						
	BEST	69,24	54,95	61,49	95,39	70,27
	LAST	69,48	56,74	61,77	95,09	70,77
wis1505_margin1						
	BEST	71,19	57,55	61,42	94,91	71,27
	LAST	70,46	57,51	60,75	95,03	70,94

From Scratch: Weights Image-Sentence - Margin Loss - Second Method

W_IS	Checkpoints	Art Painting	Cartoon	Sketch	Photo	Average
wis0515_margin15						
	BEST	**70,85**	**56,40**	**63,12**	**95,57**	**71,48**
	LAST	70,07	57,08	61,59	94,85	70,90
wis1505_margin15						
	BEST	68,60	57,12	60,55	94,55	70,21
	LAST	68,95	57,12	60,96	94,61	70,41
wis0515_margin1						
	BEST	68,90	53,11	61,31	94,49	69,45
	LAST	70,26	54,90	61,57	94,61	70,34
wis1505_margin1						
	BEST	70,85	56,91	61,54	94,85	71,04
	LAST	70,61	56,27	60,96	94,67	70,63

Fine Tuning: Weights Image-Sentence - Margin Loss - First Method

W_IS	Checkpoints	Art Painting	Cartoon	Sketch	Photo	Average
wis0515_margin15						
	BEST	70,75	55,55	63,12	95,27	71,17
	LAST	**71,14**	**56,83**	**62,41**	**95,86**	**71,56**
wis1505_margin15						
	BEST	69,09	57,51	61,11	94,61	70,58
	LAST	68,21	57,17	60,45	94,61	70,11
wis0515_margin1						
	BEST	**70,61**	**57,55**	**62,41**	**95,21**	**71,45**
	LAST	70,51	56,61	61,67	95,03	70,96
wis1505_margin1						
	BEST	68,85	57,08	60,98	94,61	70,38
	LAST	70,02	56,70	60,58	94,73	70,51

Fine Tuning: Weights Image-Sentence - Margin Loss - Second Method

W_IS	Checkpoints	Art Painting	Cartoon	Sketch	Photo	Average
wis0515_margin15						
	BEST	71,68	54,31	62,99	94,67	70,91
	LAST	71,48	55,72	62,38	95,33	71,23
wis1505_margin15						
	BEST	68,75	57,21	61,11	94,61	70,42
	LAST	68,26	57,12	60,47	94,61	70,12
wis0515_margin1						
	BEST	**71,24**	**56,14**	**63,37**	**94,97**	**71,43**
	LAST	70,90	56,31	61,62	94,73	70,89
wis1505_margin1						
	BEST	68,65	56,96	60,98	94,61	70,30
	LAST	69,98	56,53	60,55	94,67	70,43

6.2.4 Hard Negative Mining

Due to the long computation time with this method, we only tried the initial parameters, identical as the DTD model and the set of features that gave the best results without the Hard Negative Mining.

As we can notice, the best result with an accuracy of 73,03 is obtained with the combination of feature : From Scratch / Margin Loss = 1.5 / $\beta_1 = 0.5/\beta_2 = 1.5$.

We can comment this result, due to the significative amount of computational resources that this method requires, it was not possible to send for each positive images, the whole dataset in the image encoder and the sentence encoder to select the best image and the best sentence. Instead, for each positive image, we randomly select 32 images and 32 descriptions, the negative image and the negative description were chosen among those 32. So it is maybe possible to improve the performance by increasing this number of randomly images and sentences selected.

Hard Negative Mining: First Method

Hard Mining	Checkpoints	Art Painting	Cartoon	Sketch	Photo	Average
Init						
	BEST	70,02	58,36	60,63	94,91	70,98
	LAST	68,95	57,51	61,06	94,61	70,53
wis1505_margin15						
	BEST	**72,07**	**60,27**	**63,91**	**95,88**	**73,03**
	LAST	**71,57**	**59,36**	**62,29**	**94,97**	**72,05**

Hard Negative Mining: Second Method

Hard Mining	Checkpoints	Art Painting	Cartoon	Sketch	Photo	Average
Init						
	BEST	71,19	57,51	60,91	94,97	71,15
	LAST	68,85	57,42	61,14	94,61	70,50
wis1505_margin15						
	BEST	71,68	57,38	60,78	95,24	71,27
	LAST	**71,89**	**57,38**	**61,14**	**95,12**	**71,38**

6.2.5 Comparison

By testing all the combination between 0 and 10 for each w_i, we can have a decent approximation of the theorical maximum. This gives us the accuracy of 77,73 while our best model has a score of 73,03. By digging a bit further we can see, that this score is not reachable by our method. Indeed, the best combination of weights for the target domain Art Painting for example is [1, 1, 0]. We can see that we should give no importance to one of the source domain. This is impossible because it would mean that the distance between the target domain and this source domain is infinite. In any case, we notice that there is still place for amelioration.

Comparison

	Art Painting	Cartoon	Sketch	Photo	Average
Brute Force	78,13	66,68	68,83	97,31	77,74

	Art Painting	Cartoon	Sketch	Photo	Average
Our best model	72,07	60,27	63,91	95,88	73,03

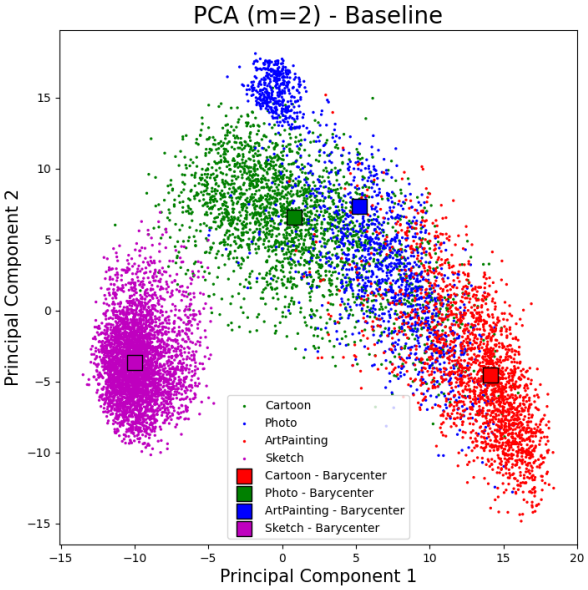
6.3 Visualization

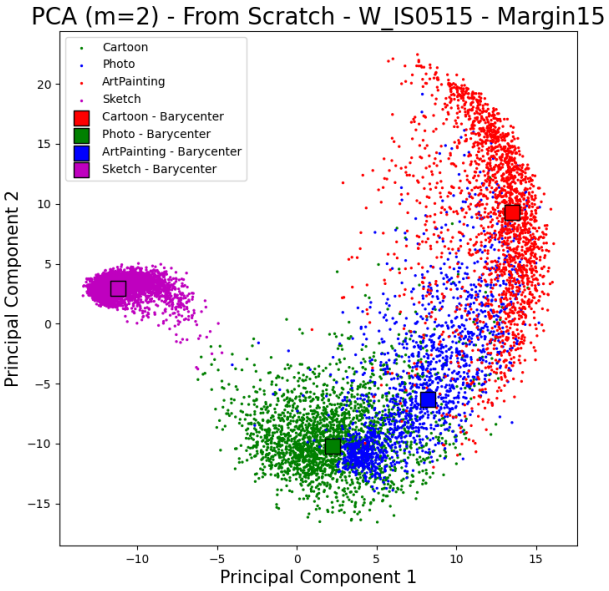
To conclude this section, we use a Principal Component Analysis (PCA) to visualize the coherence of the learned metric. In order to do so, we compute the image embedding for each image of the 9991 images of the PACS dataset and compute the tensors of the four barycenters. We obtain a vector of dimension (9995,256) and we apply PCA on this vector with m=2 to visualize clearer[17].

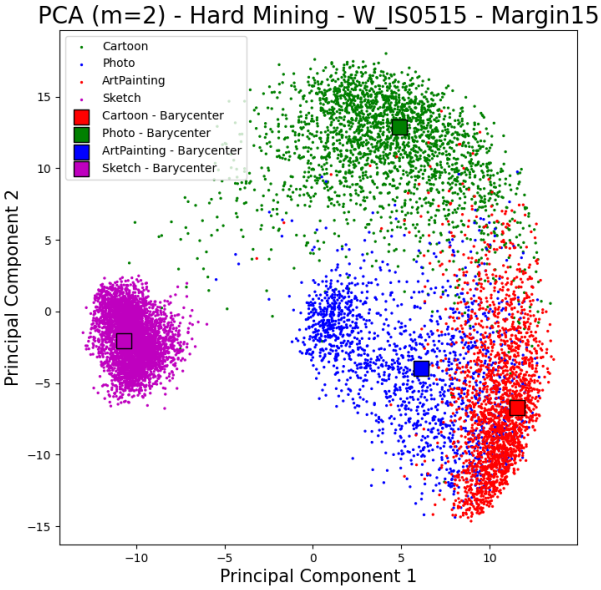
In the first PCA graph, which is the DTD model, we can notice that visual domains distances are reasonable. The barycenters of Art Painting and Photo are closer together than the others. We can also observe that except for the domain Sketch, the embeddings overlap between the 3 other domains. This could mean that doing more training could have lead to better results. This seems to be confirmed by our 2 best models where we can clearly see the four different clusters.

One must be careful with the interpretation of these graphs because if we look at the explained variance ratio, we see that the variance is explained in total between 50% and 60%. For example for the first graph, the explained variance ratio is equal to [0.34648998 0.16060676], so it means that the first axis only explains 34% of the variance and the second axis explains 16%. This ratio is equal to [0.37337189 0.23475738] for the second graph, which is slightly better and [0.3061959 0.20916035] for the third.

However, the low explained variance ratio did not influence our model because no action was taken a posteriori. These graphs were created for the sole purpose of visualization to observe the consistency of learned metric and the evolution of the relative position of each barycenter and each cluster of points.


PCA (m=2) - Baseline
Principal Component 2
Principal Component 1
Cartoon
Photo
ArtPainting
Sketch
Cartoon - Barycenter
Photo - Barycenter
ArtPainting - Barycenter
Sketch - Barycenter


PCA (m=2) - From Scratch - W_IS0515 - Margin15
Principal Component 2
Principal Component 1
Cartoon
Photo
ArtPainting
Sketch
Cartoon - Barycenter
Photo - Barycenter
ArtPainting - Barycenter
Sketch - Barycenter


PCA (m=2) - Hard Mining - W_IS0515 - Margin15
Cartoon
Photo
ArtPainting
Sketch
Cartoon - Barycenter
Photo - Barycenter
ArtPainting - Barycenter
Sketch - Barycenter
Principal Component 2
Principal Component 1
15
10
5
0
−5
−10
−15
−10
−5
0
5
10

Chapter 7

Discussion, Potential applications and Further work

We showed that the method was effective at least on the PACS dataset. In the end, we obtain an increase of 3 points compared to the baseline which consists of averaging the 3 classifiers without giving weight to each domain. In a useful practical application in industry or otherwise, a difference of 3 points may be negligible or major depending on the context. It is clear that most of the accuracy achieved comes first from the quality of the data, from cleaning the data, from managing outliers and the type of data, from the selection of features and from the quality classifiers of each source domain. This method would come at the end of the chain in the event of optimization and requires substantial work, namely the training of an additional embedder model and the annotation of the source domains data.

Moreover, we have seen that many "zero learning shot" type models such as CLIP have been developed in recent years and have a very good ability to generalize and are likely to do better than many models developed specifically for a dataset. However, these models are often better than SOTA in the domain for what might be called 'general' datasets but underperform in very specific datasets that might be called 'specialized'. This is precisely the case with CLIP which, as can be seen on Figure 7.1 below, underperforms on specialized datasets like EuroSAT (satellites images) or PatchCamelyon (medical scans) compared to Resnet50.

After these considerations, does this method have its place in a practical application? We think so, because despite the need for a precise scenario (DG + specialized dataset) and additional work required, this method brings a real gain. In an application requiring the best accuracy, this method is useful for optimizing models. This could be the case for example in the medical field and the analysis of different types of lymphocites. Field where the production of data is not easy and

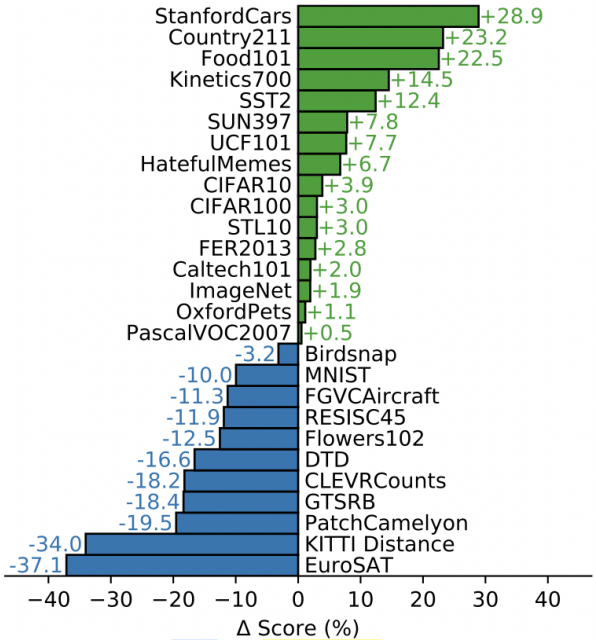


Figure 7.1: Zero-Shot CLIP vs. Linear Probe on ResNet50. Illustration from [66]

where the gain of each point of accuracy is essential.

As far as possible improvements and future work are concerned, here are the avenues that seem to us to be the most appropriate:

- A further exploration of the parameters with the method: Hard Negative Mining to try to achieve better accuracy.
- A study on the annotation system to estimate its influence
- Try other architectures for embedder training like CLIP with its contrastive loss and analyze the effects.
- Perform the experiments on a panel of datasets to estimate the average gain and see if the 3% gain is stable.

Chapter 8

Conclusion

To summarize, we implemented a metric learning approach based on triplet loss to train a model able to embed the source domains of the PACS dataset with reliability. We achieved a score of 73,03%, doing better than the baseline or the DTD model by training a model using the Hard Negative Mining and optimizing the parameters; Margin Loss and Weights Image-Sentence. We noticed that the checkpoints BEST are almost always better than the checkpoints LAST and that it is preferrable to use the first method consisting of computing the weights of the weighted sum in this manner : $w_i = cosine\_sim(ai, b)$, where a_i are the barycenter of the source domain and b the embedding of the image to classify.

If we observe at the PCA graphs, we clearly see that the learned metric is coherent, the domains with more similarities in practice were also closer in the PCA graph. We can also notice that there is a clear visual link between the accuracy of a domain and the ease to define clear clusters in the PCA graph.

Finally, we discussed the potential of the method in the event of practical applications and we believe that the method, even if usable in the event of a very specific scenario, can find its place. We have also suggested several avenues for possible improvements and we believe that the study of different architectures would be the best compromise between the additional work required and the potential gain in accuracy.

In conclusion, we showed that the addition of textual descriptions in natural language improves the Domain Generalization. We look forward to seeing if this method or a similar method will be exploited alone or in combination in future research work to improve Domain Generalizaton or if Zero-Shot Learning type models will become so powerful that they can adapt to any type of domain limiting the need for Domain Generalization techniques. Only the future will tell us.

Chapter 9

Repository

The entire code and data for the project are available in this github repository (look at README.md for the instructions to setup everything):

`https://github.com/Martial-Wsl/Domain-Generalization`

The code used to annotate and correct the students annotations is available here:

`https://github.com/Martial-Wsl/PACS-Annotater`

It contains also more small tools in link with the dataset and the cleaning.

Bibliography

[1] Femme Actuelle. *Origines, usages : ces infos que vous ignorez sur la fraise*. URL: https://www.femmeactuelle.fr/cuisine/tendance-cuisine/origines-usages-ces-infos-que-vous-ignorez-sur-la-fraise-2099323.

[2] Muhammad K Ahmad J and Baik SW. "ata augmentation-assisted deep learning of hand-drawn partially colored sketches for visual search". In: *PLoS ONE 1* (2017). URL: https://doi.org/10.1371/journal.pone.0183838.

[3] Antonio Alliegro. *MLAI - HOMEWORK 3 (Course of Machine Learning at Polito*. 2019 (2 July).

[4] Ankan Bansal, Yuting Zhang, and Rama Chellappa. "Visual Question Answering on Image Sets". In: *CoRR* abs/2008.11976 (2020). arXiv: 2008.11976. URL: https://arxiv.org/abs/2008.11976.

[5] Sarat Moka Benoit Liquet and Yoni Nazarathy. *The Mathematical Engineering of Deep Learning, 5 Convolutional Neural Networks*. URL: https://deeplearningmath.org/convolutional-neural-networks.html.

[6] Loz Blain. *OpenAI's DALL-E 2: A dream tool and existential threat to visual artists*. Jul 22, 2022. URL: https://newatlas.com/computers/dall-e-2-ai-art/.

[7] Gilles Blanchard et al. *Domain Generalization by Marginal Transfer Learning*. 2017. DOI: 10.48550/ARXIV.1711.07910. URL: https://arxiv.org/abs/1711.07910.

[8] Francesco Cappio Borlino, Antonio D'Innocente, and Tatiana Tommasi. "Rethinking Domain Generalization Baselines". In: *CoRR* abs/2101.09060 (2021). arXiv: 2101.09060. URL: https://arxiv.org/abs/2101.09060.

[9] Muller Britney. *BERT 101 State Of The Art NLP Model Explained*. March 2, 2022. URL: https://huggingface.co/blog/bert-101.

[10] Jason Brownlee. *What Is Meta-Learning in Machine Learning?* 2020. URL: https://machinelearningmastery.com/meta-learning-in-machine-learning/.

[11] Silvia Bucci. "Multimodal Deep Domain Adaptation (Sapienza Universita Di Roma)". In: (2018). URL: https://arxiv.org/pdf/1807.11697.pdf.

[12] Kokkinos I. Cimpoi M. Maji S. "Describing textures in the wild". In: *In: IEEE Conference on Computer Vision and Pattern Recognition* (2014). URL: https://arxiv.org/abs/1311.3618.

[13] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805 (2018). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

[14] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2020. DOI: 10.48550/ARXIV.2010.11929. URL: https://arxiv.org/abs/2010.11929.

[15] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks". In: *CoRR* abs/1703.03400 (2017). arXiv: 1703.03400. URL: http://arxiv.org/abs/1703.03400.

[16] G. Lee G. Blanchard and C. Scott. "Generalizing from Several Related Classification Tasks to a New Unlabeled Sample". In: *NIPS, pp.2178-2186* (2011).

[17] Michael Galarnyk. *PCA using Python (scikit-learn)*. 2017. URL: https://towardsdatascience.com/pca-using-python-scikit-learn-e653f8989e60.

[18] Chuang Gan et al. "StyleNet: Generating Attractive Visual Captions with Styles". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 955–964. DOI: 10.1109/CVPR.2017.108.

[19] Y. Ganin and V. S. Lempitsky. "Unsupervised domain adaptation by back-propagation". In: *ICML* (2015). URL: https://arxiv.org/abs/1409.7495.

[20] Jose Garcia-Garcia et al. "Multimodal Affective Computing to Enhance the User Experience of Educational Software Applications". In: *Mobile Information Systems* 2018 (Sept. 2018), pp. 1–10. DOI: 10.1155/2018/8751426.

[21] Geeks for Geeks. *Agents in Artificial Intelligence*. 22 Aug, 2022. URL: https://www.geeksforgeeks.org/agents-artificial-intelligence/.

[22] Muhammad Ghifary. "Domain Adaptation and Domain Generalization with Representation Learning". In: *Victoria University of Wellington* (2016). URL: https://researcharchive.vuw.ac.nz/xmlui/handle/10063/5181.

[23] Jing Gu et al. “Vision-and-Language Navigation: A Survey of Tasks, Methods, and Future Directions”. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2022. DOI: 10.18653/v1/2022.acl-long.524. URL: https://doi.org/10.18653%2Fv1%2F2022.acl-long.524.

[24] Yue Gu et al. “Multimodal Affective Analysis Using Hierarchical Attention Strategy with Word-Level Alignment”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 2225–2235. DOI: 10.18653/v1/P18-1207. URL: https://aclanthology.org/P18-1207.

[25] Michael Haenlein and Andreas Kaplan. “A Brief History of Artificial Intelligence: On the Past, Present, and Future of Artificial Intelligence”. In: *California Management Review* 61 (July 2019), p. 000812561986492. DOI: 10.1177/0008125619864925.

[26] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[27] M.A. Hearst et al. “Support vector machines”. In: *IEEE Intelligent Systems and their Applications* 13.4 (1998), pp. 18–28. DOI: 10.1109/5254.708428.

[28] Branislav Hollander. *Deep Domain Adaptation In Computer Vision)*. 2019 (2 July). URL: https://towardsdatascience.com/deep-domain-adaptation-in-computer-vision-8da398d3167f.

[29] Rani Horev. *BERT Explained: State of the art language model for NLP*. Nov 10, 2018. URL: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270.

[30] Zeyi Huang et al. “Self-Challenging Improves Cross-Domain Generalization”. In: *CoRR* abs/2007.02454 (2020). arXiv: 2007.02454. URL: https://arxiv.org/abs/2007.02454.

[31] Zhicheng Huang et al. “Pixel-BERT: Aligning Image Pixels with Text by Deep Multi-Modal Transformers”. In: *CoRR* abs/2004.00849 (2020). arXiv: 2004.00849. URL: https://arxiv.org/abs/2004.00849.

[32] IBM. *What is a neural network?* URL: https://www.ibm.com/topics/neural-networks.

[33] Di Domenico A. Iurada L. Strippoli S. “Domain-To-Text: improving Domain Generalization using Natural Language”. In: *Politecnico di Torino* (2021-2022).

[34] Xin Jin et al. “Feature Alignment and Restoration for Domain Generalization and Adaptation”. In: *CoRR* abs/2006.12009 (2020). arXiv: 2006.12009. URL: https://arxiv.org/abs/2006.12009.

[35] Chang Liu Jindong Wang Cuiling Lan and Yidong Ouyang. “Generalizing to Unseen Domains: A Survey on Domain Generalization”. In: *Victoria University of Wellington* (2022). URL: https://arxiv.org/pdf/2103.03097.pdf.

[36] Marimuthu Kalimuthu et al. “Fusion Models for Improved Visual Captioning”. In: *CoRR* abs/2010.15251 (2020). arXiv: 2010.15251. URL: https://arxiv.org/abs/2010.15251.

[37] Sergios Karagiannakos. *Vision Language models: towards multi-modal deep learning*. 2022-03-03. URL: https://theaisummer.com/vision-language-models/.

[38] Iot Lab KIIT. *Deep Neural Networks: First step towards thinking like Humans*. Jan 25, 2021. URL: https://becominghuman.ai/deep-neural-networks-first-step-towards-thinking-like-humans-18e6e332dbae.

[39] Alexander Kirdin, Sergey Sidorov, and Nikolai Zolotykh. “Rosenblatt’s First Theorem and Frugality of Deep Learning”. In: *Entropy* 24.11 (Nov. 2022), p. 1635. DOI: 10.3390/e24111635. URL: https://doi.org/10.3390%2Fe24111635.

[40] A. Krizhevsky. “Learning Multiple Layers of Features from Tiny Images”. In: *AAAI* (2009).

[41] Khushbu Kumari and Suniti Yadav. “Linear regression analysis study”. In: *Journal of the Practice of Cardiovascular Sciences* 4 (Jan. 2018), p. 33. DOI: 10.4103/jpcs.jpcs_8_18.

[42] Machine Learnia. *FORMATION DEEP LEARNING COMPLETE (2021)/*. Apr 25, 2021. URL: https://www.youtube.com/watch?v=XUFLq6dKQok.

[43] Yann LeCun, Y. Bengio, and Geoffrey Hinton. “Deep Learning”. In: *Nature* 521 (May 2015), pp. 436–44. DOI: 10.1038/nature14539.

[44] Haoliang Li et al. “Domain Generalization for Medical Imaging Classification with Linear-Dependency Regularization”. In: *CoRR* abs/2009.12829 (2020). arXiv: 2009.12829. URL: https://arxiv.org/abs/2009.12829.

[45] Haoliang Li et al. “Domain Generalization with Adversarial Feature Learning”. In: Apr. 2018. DOI: 10.1109/CVPR.2018.00566.

[46] Liunian Harold Li et al. “VisualBERT: A Simple and Performant Baseline for Vision and Language”. In: *CoRR* abs/1908.03557 (2019). arXiv: 1908.03557. URL: http://arxiv.org/abs/1908.03557.

[47] Nan Liu et al. *Compositional Visual Generation with Composable Diffusion Models*. 2022. DOI: 10.48550/ARXIV.2206.01714. URL: https://arxiv.org/abs/2206.01714.

[48] Zheyuan Liu et al. "Image Retrieval on Real-life Images with Pre-trained Vision-and-Language Models". In: *CoRR* abs/2108.04024 (2021). arXiv: 2108.04024. URL: https://arxiv.org/abs/2108.04024.

[49] Mingsheng Long et al. "Domain Adaptation with Randomized Multilinear Adversarial Networks". In: *CoRR* abs/1705.10667 (2017). arXiv: 1705.10667. URL: http://arxiv.org/abs/1705.10667.

[50] Jiasen Lu et al. "ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks". In: *CoRR* abs/1908.02265 (2019). arXiv: 1908.02265. URL: http://arxiv.org/abs/1908.02265.

[51] Ben Lutkevich. *BERT language model*. January, 2020. URL: https://www.techtarget.com/searchenterpriseai/definition/BERT-language-model.

[52] Divyat Mahajan, Shruti Tople, and Amit Sharma. "Domain Generalization using Causal Matching". In: *CoRR* abs/2006.07500 (2020). arXiv: 2006.07500. URL: https://arxiv.org/abs/2006.07500.

[53] Tomasz Malisiewicz, Abhinav Gupta, and Alexei Efros. "Ensemble of exemplar-SVMs for object detection and beyond". In: Nov. 2011, pp. 89–96. DOI: 10.1109/ICCV.2011.6126229.

[54] Anthony Marsland and Jonathan Schaeffer. *Computers, chess, and cognition.* 1990.

[55] Warren S. McCulloch and Walter Pitts. "A Logical Calculus of the Ideas Immanent in Nervous Activity". In: *The Bulletin of Mathematical Biophysics* 5.4 (1943), pp. 115–133. DOI: 10.1007/bf02478259.

[56] MLNotebook. *Convolutional Neural Networks - Basics*. URL: https://mlnotebook.github.io/post/CNN1/.

[57] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. *Domain Generalization via Invariant Feature Representation*. 2013. DOI: 10.48550/ARXIV.1301.2115. URL: https://arxiv.org/abs/1301.2115.

[58] W. Bastiaan Kleijn Muhammad Ghifary David Balduzzi and Mengjie Zhang. "Scatter Component Analysis: A Unified Framework for Domain Adaptation and Domain Generalization". In: *Victoria University of Wellington* (2016). URL: https://arxiv.org/abs/1510.04373.

[59] Jaideep Vitthal Murkute. "Domain Generalization and Adaptation with Generative Modeling and Representation Learning". In: *Rochester Institute of Technology* (2021). URL: https://scholarworks.rit.edu/cgi/viewcontent.cgi?article=11994&context=theses.

[60] Shi Na, Liu Xumin, and Guan Yong. "Research on k-means Clustering Algorithm: An Improved k-means Clustering Algorithm". In: *2010 Third International Symposium on Intelligent Information Technology and Security Informatics*. 2010, pp. 63–67. DOI: 10.1109/IITSI.2010.74.

[61] J. Orbach. "Principles of Neurodynamics. Perceptrons and the Theory of Brain Mechanisms." In: *Archives of General Psychiatry* 7.3 (Sept. 1962), pp. 218–219. ISSN: 0003-990X. DOI: 10.1001/archpsyc.1962.01720030064010. eprint: https://jamanetwork.com/journals/jamapsychiatry/articlepdf/488205/archpsyc_7_3_010.pdf. URL: https://doi.org/10.1001/archpsyc.1962.01720030064010.

[62] *PACS (Photo-Art-Cartoon-Sketch)*. 2018. URL: https://paperswithcode.com/dataset/pacs. (accessed: 20-04-2022).

[63] Abhishek Kumar Pandey. *Convolution, Padding, Stride, and Pooling in CNN*. Jun 25, 2020. URL: https://medium.com/analytics-vidhya/convolution-padding-stride-and-pooling-in-cnn-13dc1f3ada26.

[64] Di Qi et al. "ImageBERT: Cross-modal Pre-training with Large-scale Weak-supervised Image-Text Data". In: *CoRR* abs/2001.07966 (2020). arXiv: 2001.07966. URL: https://arxiv.org/abs/2001.07966.

[65] Fraicheur Quebec. *FRAISE D'ÉTÉ*. URL: https://www.fraicheurquebec.com/fraise/dete.

[66] Alec Radford et al. "Learning Transferable Visual Models From Natural Language Supervision". In: *CoRR* abs/2103.00020 (2021). arXiv: 2103.00020. URL: https://arxiv.org/abs/2103.00020.

[67] Aditya Ramesh et al. *Hierarchical Text-Conditional Image Generation with CLIP Latents*. 2022. DOI: 10.48550/ARXIV.2204.06125. URL: https://arxiv.org/abs/2204.06125.

[68] Aditya Ramesh et al. "Zero-Shot Text-to-Image Generation". In: *CoRR* abs/2102.12092 (2021). arXiv: 2102.12092. URL: https://arxiv.org/abs/2102.12092.

[69] Phani Ratan. *What is the Convolutional Neural Network Architecture?* October 28, 2020. URL: https://www.analyticsvidhya.com/blog/2020/10/what-is-the-convolutional-neural-network-architecture/.

[70] Lior Rokach and Oded Maimon. “Decision Trees”. In: vol. 6. Jan. 2005, pp. 165–192. DOI: 10.1007/0-387-25465-X_9.

[71] Yusuf Sarıgöz. *What is a neural network?* Mar 24, 2022. URL: https://towardsdatascience.com/triplet-loss-advanced-intro-49a07b7d8905.

[72] Yashodeep Sengupta. *Deep Learning and the Human Brain: Inspiration, Not Imitation*. Mar. 04, 19. URL: https://dzone.com/articles/deep-learning-and-the-human-brain-inspiration-not.

[73] Alex Sherstinsky. “Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network”. In: *CoRR* abs/1808.03314 (2018). arXiv: 1808.03314. URL: http://arxiv.org/abs/1808.03314.

[74] Siddharth, Tzyy-Ping Jung, and Terrence J. Sejnowski. “Multi-modal Approach for Affective Computing”. In: *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. 2018, pp. 291–294. DOI: 10.1109/EMBC.2018.8512320.

[75] Steven W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. Available at www.dspguide.com. California Technical Publishing, 1997. URL: http://www.dspguide.com.

[76] Yash Srivastava et al. “Visual Question Answering using Deep Learning: A Survey and Performance Analysis”. In: *CoRR* abs/1909.01860 (2019). arXiv: 1909.01860. URL: http://arxiv.org/abs/1909.01860.

[77] Weijie Su et al. “VL-BERT: Pre-training of Generic Visual-Linguistic Representations”. In: *CoRR* abs/1908.08530 (2019). arXiv: 1908.08530. URL: http://arxiv.org/abs/1908.08530.

[78] Alane Suhr et al. “A Corpus for Reasoning About Natural Language Grounded in Photographs”. In: *CoRR* abs/1811.00491 (2018). arXiv: 1811.00491. URL: http://arxiv.org/abs/1811.00491.

[79] Alane Suhr et al. “Evaluating Visual Reasoning through Grounded Language Understanding”. In: *AI Magazine* 39 (July 2018), p. 45. DOI: 10.1609/aimag.v39i2.2796.

[80] Umut Sulubacak et al. “Multimodal Machine Translation through Visuals and Speech”. In: *CoRR* abs/1911.12798 (2019). arXiv: 1911.12798. URL: http://arxiv.org/abs/1911.12798.

[81] Xuejiao Tang et al. “Cognitive Visual Commonsense Reasoning Using Dynamic Working Memory”. In: *CoRR* abs/2107.01671 (2021). arXiv: 2107.01671. URL: https://arxiv.org/abs/2107.01671.

[82] Anil Tilbe. *How DALL · E 2 Understands Human Language*. Jul 5, 2022. URL: https://pub.towardsai.net/how-dall-e-2-understands-human-language-22cd2df5aad3.

[83] Shagun Uppal et al. "Multimodal research in vision and language: A review of current and emerging trends". In: *Information Fusion* 77 (2022), pp. 149–171. ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2021.07.009. URL: https://www.sciencedirect.com/science/article/pii/S1566253521001512.

[84] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[85] C. Villani. "Optimal transport: old and new". In: *Springer Science Business Media* (2008).

[86] Riccardo Volpi and Vittorio Murino. "Addressing Model Vulnerability to Distributional Shifts Over Image Transformation Sets". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2019.

[87] Min Wang, Ata Mahjoubfar, and Anupama Joshi. *FashionVQA: A Domain-Specific Visual Question Answering System*. 2022. DOI: 10.48550/ARXIV.2208.11253. URL: https://arxiv.org/abs/2208.11253.

[88] Yiyu Wang, Jungang Xu, and Yingfei Sun. *End-to-End Transformer Based Model for Image Captioning*. 2022. DOI: 10.48550/ARXIV.2203.15350. URL: https://arxiv.org/abs/2203.15350.

[89] Yue Wang et al. "VD-BERT: A Unified Vision and Dialog Transformer with BERT". In: *CoRR* abs/2004.13278 (2020). arXiv: 2004.13278. URL: https://arxiv.org/abs/2004.13278.

[90] Wikipedia. *Artificial intelligence*. URL: https://en.wikipedia.org/wiki/Artificial_intelligence.

[91] Wikipedia. *BERT (language model)*. URL: https://en.wikipedia.org/wiki/BERT_(language_model).

[92] Wikipedia. *Perceptron*. URL: https://en.wikipedia.org/wiki/Perceptron.

[93] Wikipedia. *Transformer (machine learning model)*. URL: https://en.wikipedia.org/wiki/Transformer_(machine_learning_model).

[94] Brian Williams. *Contrastive Loss Explained*. Mar 3, 2020. URL: https://towardsdatascience.com/contrastive-loss-explaned-159f2d4a87ec.

[95] Chenyun Wu. *Video : Describing textures using natural language*. 2020. URL: https://drive.google.com/file/d/1VjtH18EWsZmYvOK1y8oNYu5nPSGDj33h/view.

[96] Maji S. Wu C. Timm M. "Describing textures using natural language". In: *In European Conference on Computer Vision (pp. 52-70). Springer, Cham.* (2020, August). URL: https://arxiv.org/abs/2008.01180.

[97] Shaowei Yao and Xiaojun Wan. "Multimodal Transformer for Multimodal Machine Translation". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 4346–4350. DOI: 10.18653/v1/2020.acl-main.400. URL: https://aclanthology.org/2020.acl-main.400.

[98] Evgeniya Ustinova Hana Ajakan Pascal Germain Hugo Larochelle François Laviolette Mario Marchand Yaroslav Ganin and Victor Lempitsky. "Domain-Adversarial Training of Neural Networks". In: (2015). URL: https://arxiv.org/abs/1505.07818.

[99] M. Loog Z. Wang and J. van Gemert. "Respecting domain relations: Hypothesis invariance for domain generalization". In: *Delft University of Technology* (2020). URL: https://arxiv.org/abs/2010.07591.

[100] Rowan Zellers et al. "From Recognition to Cognition: Visual Commonsense Reasoning". In: *CoRR* abs/1811.10830 (2018). arXiv: 1811.10830. URL: http://arxiv.org/abs/1811.10830.

[101] Zangwei Zheng et al. "Prompt Vision Transformer for Domain Generalization". In: *National University of Singapore and Berkeley* (2022). URL: https://arxiv.org/pdf/2208.08914.pdf.

[102] Kaiyang Zhou et al. "Domain Adaptive Ensemble Learning". In: *CoRR* abs/2003.07325 (2020). arXiv: 2003.07325. URL: https://arxiv.org/abs/2003.07325.

[103] Kaiyang Zhou et al. "Domain Generalization: A Survey". In: *CoRR* (2022). URL: https://arxiv.org/pdf/2103.02503.pdf.

[104] Stephanie Zhou, Alane Suhr, and Yoav Artzi. "Visual Reasoning with Natural Language". In: *CoRR* abs/1710.00453 (2017). arXiv: 1710.00453. URL: http://arxiv.org/abs/1710.00453.

[105] Z.-H. Zhou. *Ensemble methods: foundations and algorithms*. 2012. URL: https://books.google.be/books?hl=en&lr=&id=BDB50Ev2ur4C&oi=fnd&pg=PP1&dq=Z.-H.+Zhou,+Ensemble+methods:+foundations+and+algorithms.+man+and+Hall/CRC,+2012.&ots=OyHEFngSOJ&sig=U6nUFC0Wft1kGyzHEbOEruKrSwA#v=onepage&q&f=false.

[106] Fengda Zhu et al. "Vision-Language Navigation with Self-Supervised Auxiliary Reasoning Tasks". In: *CoRR* abs/1911.07883 (2019). arXiv: 1911.07883. URL: http://arxiv.org/abs/1911.07883.

Chapter 10

Appendix

10.1 Cleaning Dataset

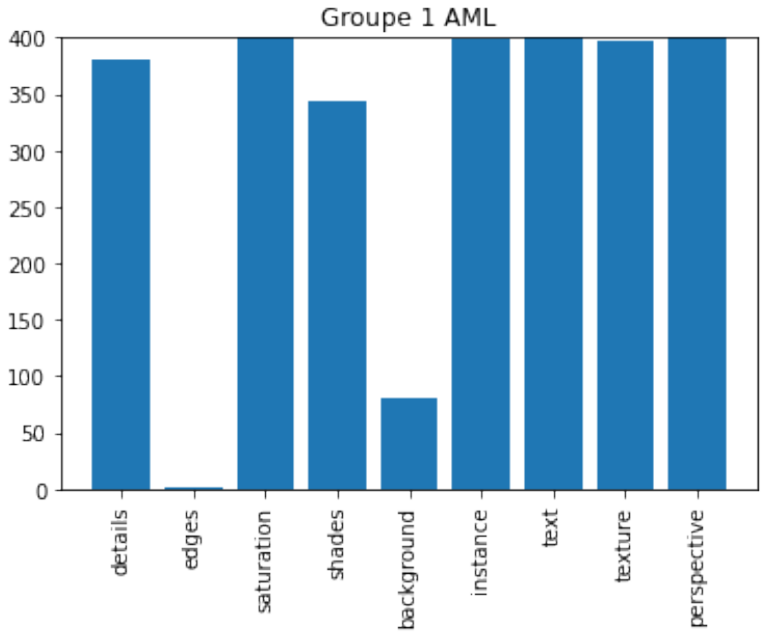


Figure 10.1: Number of differences between student of Groupe 1 AML and actual used descriptions (63,07%)

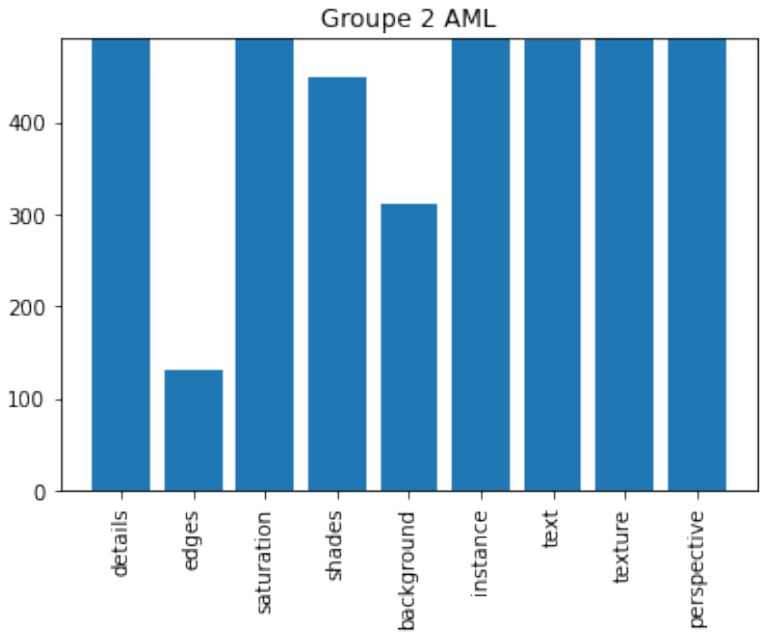


Figure 10.2: Number of differences between student of Groupe 2 AML and actual used descriptions (86,82%)

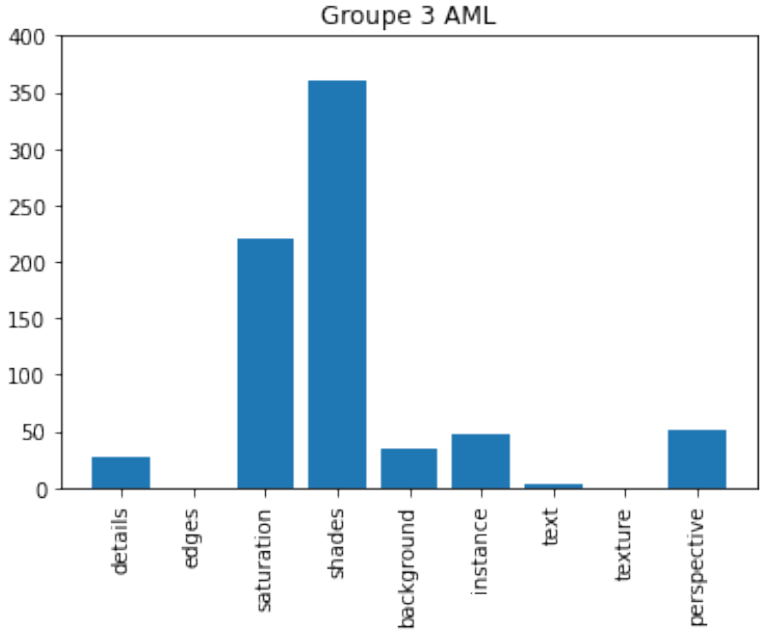


Figure 10.3: Number of differences between student of Groupe 3 AML and actual used descriptions (20,63%)

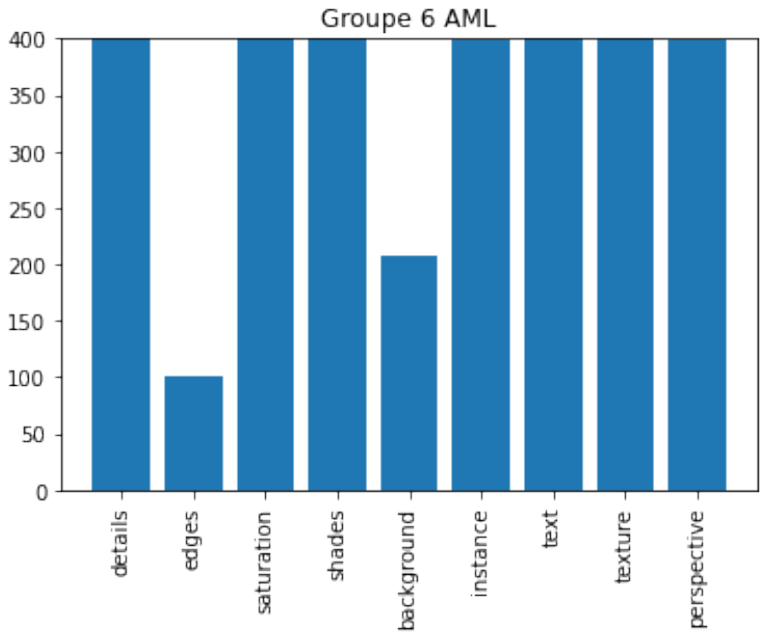


Figure 10.4: Number of differences between student of Groupe 6 AML and actual used descriptions (86,25%)

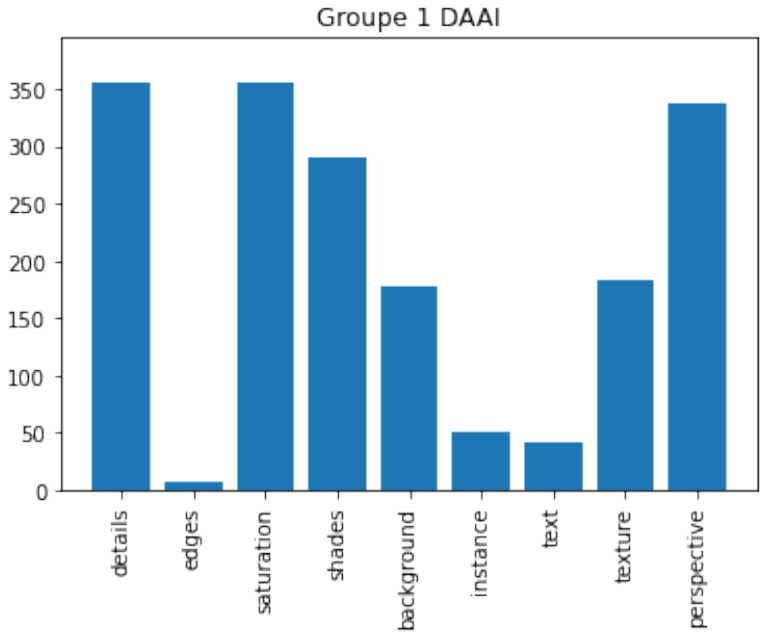


Figure 10.5: Number of differences between student of Groupe 1 DAAI and actual used descriptions (50,60%)

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**