UNIVERSITY OF TORINO

Ph.D. in Modeling and Data Science

Final dissertation

Discovering Latent Information from Noisy Sources

Supervisor: prof. Maria Luisa Sapino

Candidate: Fabrizio Scarrone

ACADEMIC YEAR 2022/2023

Summary

Today, there are many publicly available data sources. But the data is heterogeneous and complex (diverse, multi-modal, sparse, and noisy). In particular, the availability of information from social media, such as Twitter, has both advantages and disadvantages: social media messages can potentially provide information not available otherwise, but these messages are short, noisy, and not infrequently contain grammatical and linguistic errors.

The claim of this research is that the availability of publicly available information can be leveraged to fulfill various tasks. Taking as an example the cultural heritage domain, the context from which this research starts from, the fruition of the domain information can take advantage of the development of tools capable of signaling to the various classes of users (such as the public, local governments, researchers) the entities that make up the domain and the relationships existing among them.

This research aims at developing novel algorithms and tools for leveraging multi-modal, sparse, and noisy data available from multiple public sources. In doing so our models make use of multi-modal features extracted by existing deep neural models to improve the performance on various tasks.

There are a number of challenges that must be addressed:

- Data Volume: Social media generates vast amounts of data, including images, leading to challenges in storage, processing, and analysis.
- Data Quality: Textual data from social media are oftent very short and contains grammatical errors, images can be poorly correlated to the textual portion .
- Data Labeling: Manually labeling image data for supervised learning tasks such as classification can be time-consuming and resource-intensive.
- Multimodal Integration: Integrating textual and visual information for information extraction is a complex task.

- Computational costs: Multimodal neural models require significant computational and time resources to be implemented, trained and analyzed

Named Entity Recognition on Twitter messages is the first tasks to be explored with the aim of finding entities related to the cultural heritage domain. We developed models that integrate textual and visual information and use an approach inspired by factorization machines to analyze and leverage the interaction inside and between these modalities.

The exploration of this natural language processing task within a multimodal context has prompted examination of algorithms capable of indicating the most relevant information from the available data to achieve the objectives of a specific task.

This type of algorithm goes under the name of attention mechanism and, within the scope of sparse and noisy multi-modal data of this thesis,this has led to researching novel algorithms that can be used to address the problem of finding relevant information within the noisiness of the data. An attention mechanism, inspired again by factorization machines, has been devised, implemented and applied on a task where portions of captions must be linked to specific regions of related images.

Attention-like algorithms can also be used to address the issue of data sparsity by determining the most suitable information to utilize when employing data augmentation techniques. In this regard, three different algorithms are proposed in this thesis. All of them are applied on the embeddings created by existing neural network models with two goals in mind. The first goal is to be able to determine the diversity, with respect to the available data obtained by modifying the data. The second goal is to assess, on a classification task, the performance impact of data augmentation based on the different degrees of diversity added to the original dataset.

Data augmentation techniques can also play a role to address the problem of domain adaptation where a model trained on a domain is applied on a different domain, in this context data augmentation can help to reduce the difference between the domains. We study the results of the application of two different strategies of features level augmentation on three different benchmark multi-domain data sets in order to obtain indication on the most effective approach.

Acknowledgements

I would like to express my gratitude to Professors Maria Luisa Sapino and Kasim Selçuk Candan for their guidance and support throughout my doctoral studies. Their expertise and mentorship have been crucial in shaping this thesis.

I would like to express my appreciation to the management of CSI Piemonte for enabling my participation in this PhD program, which has been a significant opportunity for my growth.

I am grateful to Professor Laura Sacerdote, the coordinator of the PhD program.

Furthermore, I wish to thank Professor Luigi Portinale and Professor Huiping Cao for their revision and constructive criticism, which have significantly improved the quality of this work.

I am sincerely grateful to all of them for their contributions.

Contents

Chapter 1

Introduction

Nowadays, many public data sources are available on the most varied topics.

Along with the data availability, there is an increasing desire to use this data in order to improve the offering and the creation of these assets. One obstacle in achieving this goal is the heterogeneity and complexity of this data, they are diverse, multi-modal, sparse and noisy, each one of these characteristics brings a challenge.

This is particularly true for data coming from social media. They provide a huge amount of data so it comes natural to try to exploit them. Consider the case of microblogging platforms (such as X, the former Twitter [1]) the messages posted by the users can potentially provide information in great quantity and very up to date, not available otherwise, yet such messages are short, noisy, not rarely contain idiomatic expression and are not exempt from grammatical and linguistic errors.

The claim of this research is that availability of publicly available information can be leveraged to fulfill various tasks, such as detect particular fragment of information or discover properties of the information itself that can in turn be used for other purposes. For example, in the cultural heritage domain, the first domain of application of this research the information can be improved with tools capable of signaling to the various classes of users (such as the public, local governments, researchers) the entities that make up the domain and the relationships existing among them.

Given this context, one of the task that can be performed is the so called Named Entity Recognition (NER), a task of supervised learning where a model is trained on a corpus of texts where some portion of text are tagged with the name of a category belonging to a predefined set depending on the domain of interest. These tagged portions of text are called named entities.

Possible examples are of such categories are: Person, Organization, Location or Other. After the training, the model can detect such named entities on texts never seen before.

It is a task that has seen an evolution both in methodology and in types of data of application, going from solutions applying predefined rules on very well-structured texts to solutions using neural networks and large language models on very sparse and noisy text, eventually accompanied by visual information [2], [3].

Information obtained from social media allows this kind of approach offering the possibility of having the simultaneous availability of text and images, a setting termed as multi-modal, and this research uses the X (former Twitter) messages for finding Named Entities in the categories of Artist, Artworks, Artistic movements and Venues.

The experience in dealing with this task has lead to take into consideration a more general and underlying issue, that is, if it is possible to improve an algorithm by leveraging a new methodology to increase its ability to consider the relations between inputs of different modalities in order to give importance to the relations more important for the task at hand and decrease the contribution of the less important relations. A class of algorithm known in the literature as "attention mechanisms" [4, 2, 5].

Drawing inspiration from a comparison in the fields of biology and psychology, the attention mechanism is an attempt to mimic the action of selectively focusing on a few relevant parts, ignoring others, a behavior that human beings perform instinctively. Inside a machine learning algorithm, this translates into operations on the numerical representation of the data that somehow increase or decrease the data importance in relation to achieving a desired task.

The concept of attention appeared in the context of the machine translation task where a sequence of words must be processed to be translated in another language [6, 7]. This sequence processing involves a number of problems related to: the representation of the words, the length of the sequence, the position and relation of the words inside the sequence. The representation of a word as a real valued vector is called an embedding. A requirement for such embeddings to be meaningful is that words with similar meaning must be transformed in embeddings that are close in the target vector space and words with very different meaning must be transformed in vectors that are far away in the target vector space.

The evolution of the creation of these embeddings has today led to the

ability to create embeddings that also take into account the context within which an embedding is created, in order to distinguish the language changes that a word can undergo within different contexts. Just to give an example, consider the phrases “Bank of a river” and “Bank robbery” [8, 9] .

The length of sequences can be a problem for lengthy sequences because simple sequential processing can lead to a “forgetting” behavior where previous words lose importance as the phrase proceeds. Last but not least, relations between words inside a phrase can be quite complicated and related words can also be far away inside a phrase.

Attention in some way addresses these problems by giving a way to analyze the embeddings of an entire sequence in search for important relation and giving scores that highlight the important parts. This can also be used to address the problem of the noisiness and sparsity of the information [4].

Attention is not confined to natural language processing problems. Different types of attention have also been developed in the computer vision field. For example, in tasks involving the recognition of objects or trying to relate portions of texts to part of images [10, 5].

In this research we explore a mechanism that addresses this last type of setting, where texts and images are simultaneously available. This is called a multimodal setting. In a multimodal task, it is interesting to analyze the relations inside the modality separately, but is also important to analyze cross modal interactions and try to see how they contribute.

A further consideration about attention, beyond its use to overcome shortcomings in the information available, is to explore its ability to give insight into the available information with the aim of driving an augmentation schema during the training of a model.

Data augmentation is a technique used in machine learning to increase the performance of a model on some given task. The underlying belief is that the available information in the training dataset is somehow insufficient to give a model the ability to fully recognize the data it will face after the training [11]. It has sometimes been compared to giving to the model some sort of "imagination" [12] where the model is confronted with data that are not real but are plausible modifications of the real data. These modified data are added to the training set so that the model is trained on more diverse data and, subsequently, correctly interprets previously unseen data after training.

Obviously, some caution must be taken because, in a context of supervised learning, it must be assured that the label of the original image still

applies to the transformed data [13]. The transformations that can be applied to the data can be roughly categorized as follows [11, 12].

Geometric transformations. In computer vision, geometric transformations are applied to the full image in order to have different versions of the same image. Rotation, translation, mirroring along horizontal or vertical axes are examples of this types of transformation.

Photometric transformations. Given an image, these types of transformations are applied at pixel level. In this way, it is possible to modify properties such as contrast, luminosity or color balancing.

Feature level transformations. These are more general transformations because, given the fact that they are applied to the vectors generated by a neural network, they are somehow independent on the modality of the data (text, image, ...), addition of gaussian noise or interpolation of vectors are example of these types of transformations.

Independently of the transformations, however, a question is if there are portions of the training dataset that is more useful to add to the training set, after some modifications, to obtain a benefit in performance, or if it is possible to devise an augmentation schema particularly suitable to the type of data and model used.

Researching these topics has led to taking into consideration the technique of *transfer learning* where information obtained by large models trained on huge amount of data is used on different or more specific tasks.

Three approaches have been researched, each of them aimed at identifying a "measure" of diversity so that one can decide how much diversity is more useful to inject in the training dataset. The range in this way goes from minimum diversity (or more of the same) addition, to maximal amount of diversity.

- The first approach is based on factorization machines [14]. By analyzing the interactions between the samples features, this method seeks to quantify the level of diversity present in the dataset.

- The second approach leverages transfer learning and cosine similarity. Here, cosine similarity is utilized to compare data representations derived from pretrained models that have been trained on large datasets. This method offers a simple means to quantify diversity by measuring the angular difference between embeddings. Since it operates on vectors, it can be applied irrespective of the original modality of the

data. When used on the embeddings of both the original and transformed data, it enables the creation of distributions of diversity measures. These distributions can then be used to determine the extent of diversity being added to a new training dataset and to evaluate its impact on model performance.

- The third approach is based on Euclidean Distance in conjunction with a Support Vector Machine (SVM) classifier [15]: Euclidean distance is a measure of the straight-line distance between two points in Euclidean space. When used in conjunction with an SVM classifier, it allows for geometric considerations regarding the diversity of the data. By calculating the Euclidean distance between data points and examining their spatial distribution, insights into the diversity of the dataset can be gained.

Each of these approaches offers a different perspective on measuring diversity within a dataset. While factorization machines provide a more holistic understanding of diversity through feature interactions, cosine similarity and Euclidean distance offer more direct measures based on the similarity or dissimilarity of data points. The utilization of Euclidean distance with SVM classifier adds a geometric aspect, allowing for a deeper analysis of the dataset's diversity and informing augmentation strategies. By employing these approaches, one can better understand the diversity present in the training dataset and make informed decisions about injecting additional diversity as needed for improved model performance.

Finally, research on transfer learning and data augmentation led us to address the problem of domain adaptation [16, 17]. In this case a model trained on a domain is applied on a domain that is somewhat different from the training domain. Data augmentation in this context can help in reducing the shift between the domains. We study the impact of in-out and rotation-based augmentation strategies applied at feature level on a image classification task using a max-margin and a fully-connected classifiers as alternative learning models.

1.1 Main contributions of the thesis

The contributions of this research are about an approach inspired by factorization machines from two points of view: the first, in the analysis of the relationships that exist within and between the modalities integrated in

a model with application on a NER task and an image caption alignment task , the second shows the ability to highlight different degrees of diversity present within a dataset in order to identify the most interesting parts for data augmentation actions with application on classification task on dataset composed of text and images.

Remaining in the field of data augmentation, further contributions concern approaches that show the possibility of capturing the diversity present in the data. Cosine distance and Euclidean distance between representations of original data and modified data are used with the aim of selecting the most useful data for augmentation. The Euclidean distance based approach has a particular focus on the case in which, as a classifier, support vector machines are used.

Expanding our research in the field of data augmentation and transfer learning we have addressed the issue of their application in the context of domain adaptation. Our contribution is in the study of the impact of two augmentation strategies based on translation and rotation applied at feature level on different multi-domain data sets in order to obtain indication on the most effective augmentation approach to reduce the shift between different domain.

The thesis continues with the following organization.

Chapter 2 introduces the context surrounding the first part of this research, which is concerned with data related to the cultural heritage domain obtained from Twitter.

Chapter 3 focuses on the more specific task of Named Entity Recognition (NER). Evolution and previous works are presented. The research done in this area is described along with the experimental results.

Chapter 4 introduces the concept of attention and a review ot its evolution.

Chapter 5 deals with the research done on a novel type of attention inspired by factorization machines, applied on a text-images alignment task.

Chapter 6 is about the research done on data selection for data augmentation and domain adaptation applied on classification tasks.

Part I

The Cultural Heritage Domain

Chapter 2

Background

Today, there are many publicly available data sources, such as online museum, catalogues (e.g. [18]), Wikipedia (e.g. [19]) and social media (e.g. [1]), in the cultural heritage domain. The premise of the research is that the availability of publicly available information related to the cultural heritage domain can be significantly improved with tools capable of extracting useful information for the interested users (the public, local governments, researchers) such as the mentioning of entities that make up the domain and the relationships existing among them. However, the realization of this idea is not easy due to the complexity (multi-modality, sparsity, and noise) of the data available in this domain. See Figure 2.1.

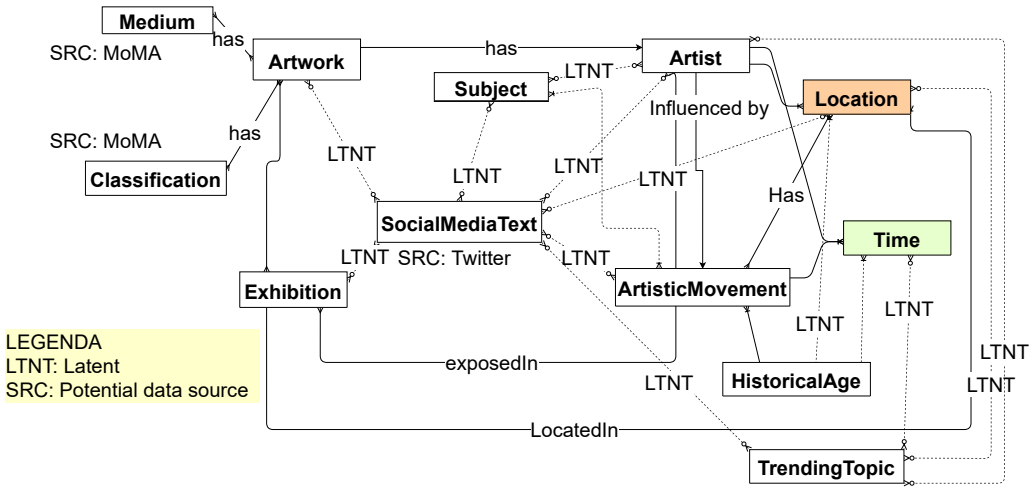


Figure 2.1: A simplified Semantic Net diagram outlining some of the data sources and their contributions to the information available in the cultural heritage domain – here any edge marked as "LTNT" represents a latent relationship that needs to be extracted

Given the above context, the focus of this research is on developing novel algorithms, techniques, and tools for leveraging multi-modal, sparse and noisy data available from multiple public sources to integrate and enrich useful information available to the public in the cultural heritage domain.

2.1 The Context

Figure 2.1 is the diagram of a semantic net outlining some of the data sources and their contributions to the information available in the cultural heritage domain. For example, one can observe that a number of entities (such as artwork, artist, exhibition, artistic movement) are available from various sources. Some of the relationships among these entities can be explicit in these sources (such as an artwork is being associated to an artist or an artist being included in an exhibition), while some other relationships, such as an artist being influenced by an artistic movement or an artist being interested on a particular subject may be latent. In Figure 2.1, such latent relationships are highlighted with edges marked as "LTNT" and they represent areas of interest for this research.

2.1.1 The Role of Social Media

Availability of *social media (such as Twitter messages) brings advantages and disadvantages* in this domain:

- *How can social media help in this domain?:* Social media posts have the potential to offer unique insights not found elsewhere. Numerous museums and collections frequently share updates on Twitter regarding artists, artworks, and exhibitions. Furthermore, online communities may engage in discussions about specific artistic movements. These posts not only offer exclusive information but also aid in understanding existing data and uncovering hidden connections within the field of interest.

- *Challenges in leveraging social media in this domain :* Using social media data for this purpose poses challenges, despite the opportunities it presents. Twitter messages, in particular, are brief, filled with noise, and frequently include acronyms, grammatical errors, linguistic mistakes, or slang expressions. Additionally, a significant portion of messages within this domain are generated automatically via APIs,

Berthe Morisot (France, 1841-1895), #impressionism #painting "Le berceau" , 1874 (Musée d'Orsay, Paris).
artist *movement* *artwork* *venue*

The Minimalist Art Movement Is Fundamentally "SIMPLICITY " DonaldJudd was a leading figure in his contributions. "Fabricated" 1981-82 @americanart
movement *artist* *artwork*

Happy Birthday Mark Rothko 1903 -1970 Visit The Abstract Expressionism show at the RA London - Read Review Here: https://goo.gl/8hD4QR
artist *movement* *venue*

Figure 2.2: Sample Twitter messages and associated entity types

lacking linguistic structure. Given their brevity, these messages often lack sufficient context for easy interpretation.

In fact, our experience with Mechanical Turk [20], in agreement with other experiences [21], has shown that even manually labeling portions of the messages for a supervised methodology is difficult due to the underlying ambiguities.

2.1.2 A Multi-Modal Approach

It is important to note that social media, in this context, is often multi-modal in that many messages in this domain are accompanied by visuals – our experience has shown that roughly 30% of the messages have one or more associated images.

In recent times, there has been a growing interest in research focusing on joint learning from sources with different modalities [22, 23, 24]. This interest has been further fueled by the advancement of deep learning techniques, leading to increased utilization of such approaches across a range of domains, including image understanding and annotation. [25], and natural language processing [3].

In theory, visuals can aid in providing context essential for enhancing the interpretation of social media messages for efficient information extraction and integration. However, in practice, these visuals vary greatly, encompassing images of artwork, images of artists, snapshots of exhibition venues, or promotional flyers, often lacking descriptive labels themselves. Consequently, harnessing such visual data to implement a multi-modal approach proves challenging. The subsequent section outlines some tasks feasible within this framework.

2.1.3 Possible Tasks

Given the context described so far, exploring social media content to find hidden information involves several tasks. These tasks include understanding and discovering connections between different pieces of information expressed in natural language.

Namely:

- Named Entity Recognition (NER): a task of information extraction that seeks to locate and classify named entities mentioned in unstructured text into pre-defined categories such as person names, organizations, locations, ... [26, 27].
- Entity Linking (EL): the task of relating a certain text to one or more entities [28, 29].
- Mention Detection (MD) : the task of identifying a portion of text where an interesting entity is present [30].
- Candidate generation (CG): The task of finding the best possible entities to be assigned to a found citation [31].
- Entity Disambiguation (ED): the task of assigning the best (possibly the right) entity (among the available ones) to the detected citation [32].
- Topic modeling: The task of detecting groups of words defining a number of topics discussed in a corpus of ducuments [33].
- Relationship detection: The task of detecting relations among entities in a domain [34].

Research on the listed tasks has been performed with a possible final target of recommendation, information retrieval, information augmentation, knowledge discovery for a better understanding of the available information from the entities owning the information and a better fruition from the general public.

All of these tasks have been explored in different scenarios characterized by an ever-increasing degree of uncertainty and ambiguity, from large corpora of documents about well-defined topics and entities and from well-defined entities up to collections of very short texts with little contextual information and a lot of ambiguities with respect to entities, arguments and scope.

An example of the first scenario is a collection of research articles and research authors to detect research topics and related researchers. An example of the second scenario is a collection of tweets used to detect events and participating people, e.g. concert-musician, conference-scientist.

The focus of the first part of the research is in the context of this second scenario, where ambiguity and uncertainty are quite high and will address a task of *Named Entity Recognition (NER)* in the context of microblogging messages related to *Cultural Heritage* information.

Chapter 3

Dealing With Noisy Information - Named Entity Recognition

In general, works on tasks listed in the previous chapter propose different methodologies to fulfill one or more of them, reducing the impact of the available information shortcomings.

In the following there is an overview of how NER task has been addressed in the past.

3.1 Named Entity Recognition

A Named Entity (NE) is a word or a group of words that clearly identifies one item among others in a text. In the general domain, typical examples of named entities are organization, person, and location names.

NER is the task of locating named entities in text and classifying them into predefined categories.

Formally, given a sequence of tokens $s = w_1, w_2, ..., w_N$, NER is to output a list of tuples I_s, I_e, t, each of which is a named entity mentioned in s.

$I_s \in [1, N]$ and $I_e \in [1, N]$ are the start and the end indexes of a named entity mention; t is the entity type from a predefined category set.

As an example, applying a NER system to the phrase “Mary said that The Ritz was a great hotel option to stay in London."” in search for the categories PERSON, LOCATION, and ORGANIZATION would produce: (Mary, PERSON) said that The (Ritz, ORGANIZATION) was a great hotel option to stay in (London, LOCATION)’.

In the following some areas that have an influence in the development of a NER system.

3.1.0.1 The Language, Domains and Textual Genre

The majority of the works are applied to English language documents. However, the works related to other languages or that are multilingual are increasing.

Along with the 'general domain" characterized by persons, organizations and locations, many other domains are an object of research, so other examples of categories and domain are gene, protein, drug and disease names in the biomedical domain or "protein", "DNA", "RNA", "cell line" and "cell type" (e.g., D. Shen et al. 2003, B. Settles 2004) in the bioinformatics domain. Artist, artwork, title are entities in the cultural heritage domain which is part of this research.

Sometimes, in addition to the specific categories, an "Other" or "Miscellaneous" category is added to gather entities related to the domain but not falling in the other well defined categories. As one can imagine, this brings a certain amount of specialization in a NER system, so that it is not immediate to move a NER system across different domains or create a multi-domain system.

Also the textual genre (scientific, journalism, emails) has an impact moving a NER system between corpora of different genres produces very different results.

3.1.1 Evolution of NER Systems

What is considered the first research paper on recognizing named entities in documents was presented by Lisa F. Rau (1991) [26] at the Seventh IEEE Conference on Artificial Intelligence Applications. It was based on heuristics and handcrafted rules to extract company names.

In general, earlier attempt relied on handcrafted rules, lexicons, orthographic features, ontologies and human experts. A following approach was to create NER systems based on features-engineering and machine learning (Nadeau and Sekine, 2007) [27]. At last, neural network NER with Collobert et al. (2011) [35] systems with minimal feature engineering, began to appear. Neural network models are appealing because they usually do not rely on domain specific resources like ontologies, and thus are somewhat less domain dependent.

In the following more details on the different approaches.

Rule-based Historically, the first rule-based NER systems rely on hand-crafted rules. Rules can be designed based on domain-specific documents corpora [26], syntactic-lexical patterns [36], reference dictionaries [37].

Rule-based systems work very well when rules, patterns and dictionaries are well thought of and complete, which is difficult to achieve, usually they cannot be transferred to different domains.

Unsupervised Learning Clustering can be leveraged for the creation of a NER system [38]. It can be used to extract named entities from clustered groups based on context similarity.

Feature-based Supervised Learning Supervised learning for NER is essentially a multi-class classification or sequence labeling task.

It requires annotated data samples and features must be designed to represent each training example. Machine learning algorithms are then utilized to learn/train a model to recognize similar patterns from unseen data.

Supervised NER systems require feature engineering. Feature vector representation where words are represented by numeric, boolean or nominal values.

Word-level features such as tagging, list lookup features have been used, feature design is in itself a research area.

Based on the chosen features, many algorithms have been applied: Support Vector Machines (SVM) [15], Decision Trees [39], Hidden Markov Models (HMM) [40], Conditional Random Fields (CRF) [41, 42], ...

3.1.2 Neural Network for NER

In recent years, Neural Network-based NER models achieved state-of-the-art results. Compared to the previous approaches, deep learning is beneficial in discovering hidden features automatically.

3.1.2.1 Neural Networks and Deep Learning

Neural Networks and Deep Learning

Neural networks are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. The typical layer's behavior consists of the forward pass and backward pass. The forward pass

computes a weighted sum of the layers' inputs from the previous layer and passes the result to the next layer, through a non-linear function. The backward pass computes the gradient of an objective function with respect to the weights of a multilayer stack of modules via the chain rule of derivatives.

The advantage of Neural Networks is the capability of representation learning and semantic composition using both vector representation and neural processing. This allows a machine to take in input the raw data and to automatically discover latent representations and processing needed for detection and/or classification.

A possible categorization of modern neural architectures for NER is based upon their representation of the words in a sentence. For example, representations may be based on words, characters, sub-word units or a combination of these. The next two paragraph introduce the first two cases.

3.1.2.2 Word Level Architecture

In a word level architecture (see Figure 3.1), the words of a sentence, represented by their word embedding, are given as input to a Recurrent Neural Network (RNN), a type of network where the hidden layers are used to "remember" previous inputs, which is a necessary requirement for some tasks, such as predicting the next word in a sentence.

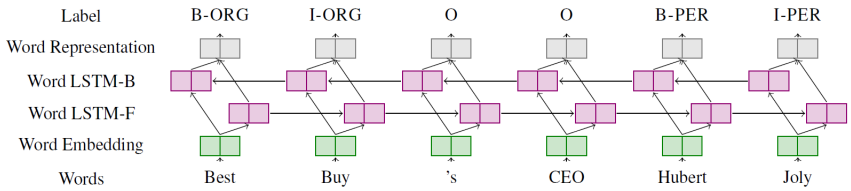


Figure 1: Word level NN architecture for NER

Figure 3.1: Word level NN architecture for NER [43]

3.1.2.3 Character Level Architecture

In this architecture, a sentence is considered to be a sequence of characters, see Figure 3.2.

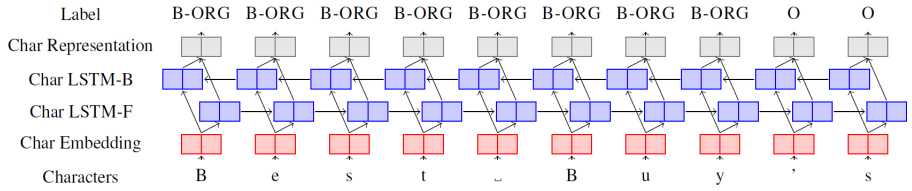


Figure 2: Character level NN architecture for NER

Figure 3.2: Character level NN architecture for NER [43]

This sequence is passed through an RNN, predicting labels for each character. Character labels are then transformed into word labels via post processing. The potential of character NER neural models was first highlighted by Kim et al. (2016) [44] using highway networks over convolution neural networks (CNN) on character sequences of words and then using another layer of Long Short-Term Memory network (LSTM) [45] + softmax for the final predictions.

3.1.2.4 About the Representations as Embeddings

Typically, these representations are pre-trained over large collections of text, as input. The pre-trained word embeddings can be either fixed or further fine-tuned during NER model training. Commonly used word embeddings include Google Word2Vec [46], [47], Stanford GloVe [48] .

In this case, the embedding of a word is always the same. There is no contextual information in the embedding itself.

Peters et al. [8] proposed ELMo ("Embeddings from Language Mode") representations, which are computed on top of two-layer bidirectional language models. ELMo is character-based in its input the learnt representations are at word level. ELMo looks at the entire sentence before assigning each word in it an embedding.

This new type of deep contextualized word representation is capable of modeling both complex characteristics of word usage (e.g., semantics and syntax), and usage variations across linguistic contexts (e.g., polysemy). For instance, for the phrase "He went to the prison cell with his cell phone to extract blood cell samples from inmates", ELMo would generate different vectors for the three vectors representing the word "cell".

In 2019 Devlin et al. [9] proposed a new language representation model called BERT, Bidirectional Encoder Representations from Transformers. BERT uses masked language models to enable pre-trained deep bidirectional representations. given a token (a sub-word), its input representation

is comprised by summing the corresponding position, segment and token embeddings. Pre-trained language model embeddings often require large-scale corpora for training, and intrinsically incorporate auxiliary embeddings such as position and segment embeddings, so it can be considered an hybrid architecture.

BERT represents input as subwords and learns embeddings for subwords. It has a vocabulary that is about 30,000 subwords. The model is trained on a corpus with a large number of words (millions). Representing input as subwords as opposed to words strikes a balance between character based and word based representations.

Character and sub-word level representation models like ELMO and BERT have the ability to output embeddings for out of vocabulary words (OOV), that is, words never seen before by the model.

3.1.2.5 Going Beyond the Text

Along with the improvements in the creation of embeddings for the text, there is, however, the problem that sometimes it is desirable to expand the contextual information for a NER task with other information that is associated with the text. A typical example are the images associated with the text messages on social networks. Not rarely the image is helpful to disambiguate or make explicit the meaning of the text.

There isn't a large amount of literature in this field, a previous work on NER applied to Twitter messages making use of the associated images is by Zhang, et al. [2]

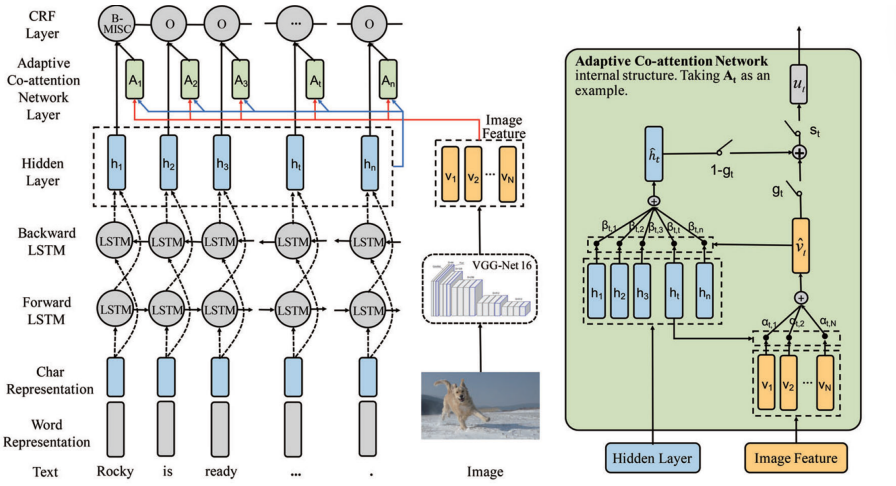


Figure 3.3: Multimodal architecture from Zhang et al. paper [2]

In Figure 3.3 the used architecture is displayed. Regarding the textual part, two LSTM are used in order to process the inputted text both in the forward and backward direction in order to extract the textual features. As for the visual part, the image features were extracted from 16-layer VGGNet [49].

A first remark is that depending on the task/methodology at hand, one can decide to extract features at different points from the VGGNet network, for example, to preserve spatial information regarding an image.

A second observation is that here we can see the rise of the issue that in a multimodal architecture there is the need to find a way to mix in a meaningful way the different modalities which will be the subject of the second part of this thesis.

3.1.3 The Dataset

The first part of this research is about researching a NER task on text characterized by noisiness and sparsity but associated with images that can potentially provide additional context. In order pursue this goal an English-language dataset was built using the Twitter API and collecting tweets with their associated images. The queries looked for messages related to artists, artworks, artistic movements and venues (museums, galleries, ...)

The entities in the text messages are manually labeled by Mechanical Turkers [20], using the IOB2 tagging scheme [50], [21]. the final dataset is composed of 2100 messages and associated images.

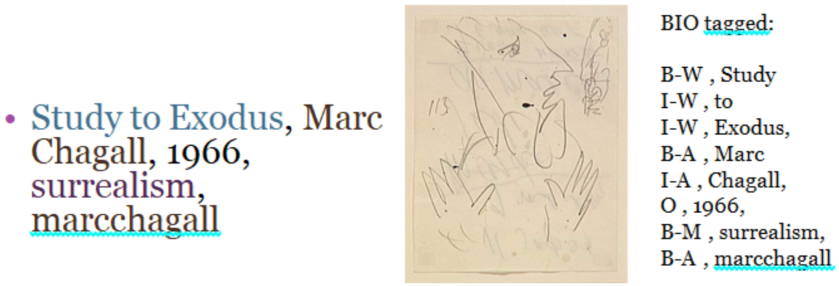


Figure 3.4: Example of BIO tagging

3.1.3.1 BIO Tagging

The IOB format (short for inside, outside, beginning), or BIO format, is a common tagging format for tagging tokens in a chunking task in computational linguistics (ex. named-entity recognition). It was presented by

Ramshaw and Marcus in their paper "Text Chunking using Transformation-Based Learning", 1995 [51]. The I- prefix before a tag indicates that the tag is inside a chunk. An O tag indicates that a token belongs to no chunk, the B- prefix before a tag indicates that the tag is the beginning of a chunk that immediately follows another chunk without O tags between them. It is used only in that case: when a chunk comes after an O tag, the first token of the chunk takes the I- prefix.

In this work, a similar format, the IOB2 format, is used. It is the same as the IOB format except that the B- tag is used at the beginning of every chunk (i.e. all chunks start with the B- tag) as in Figure [50]. In a NER task, the tags represent the classes used to classify the named entities.

In our case the BIO tagging of the entities is as follows:

- B-A: begin of an artist name.
- I-A: inside an artist name.
- B-M: begin of an artistic movement name.
- I-M: inside an artistic movement name.
- B-V: begin of a venue name.
- I-V; inside a venue name.
- B-W; begin of an artwork title.
- I-W; inside an artwork title.

3.1.3.2 Amazon Mechanical Turk

Classification tasks often need large amounts of labeled data, but obtaining such large dataset of annotated data is not easy. Traditionally, annotated training data has been provided by experts or by the researchers themselves, often at great time and money costs. Recently, attempts have been made to leverage non-expert annotations through the Amazon Mechanical Turk (AMT) service to create large training datasets.

Named entity recognition is a task where usually, in terms of available data, more is better, so an attempt to use this approach has been made and mechanical turk workers contributed to the creation of the dataset.

AMT workers were selected among english speakers and asked to label tweets according to the categories relevant for our research (artists, artwork, artistic movement, venues). Through the AMT interface, they were able to

see the tweets and they were asked to annotate them. For the sake of speed, they had to surround what they consider a relevant portion of text with the following annotations: [[A text]] for an artist, [[W text]] for an artwork title, [[V text]] for a venue, [[M text]] for an artistic movement.

In this way the message "Marc Chagall is a painter" would be changed in "[[A Marc Chagall]] is a painter " . Obviously the quality of the labeling strongly depends on the skill and dedication of the AMT worker. A subsequent script was used to bring the text to the desired IOB2 format.

3.1.4 BERT and Models

As mentioned in the previous chapter, BERT was introduced by Google in 2018 [9].

BERT, Bidirectional Encoder Representations from Transformers, is based on the transformers architecture where every output element is connected to every input element. In its name it highlights the bidirectionality of its working. Different from previous models, BERT can read its input in both left to right and right to left directions. This enables BERT to create embeddings capturing good contextual information, which allows for the disambiguation of polysemous words, or words with multiple meanings.

Bert uses a 30,000 WordPiece vocabulary on input. It was trained on a dataset made of Wikipedia data (2.5B words) and BookCorpus (800M words) and required 4 days for its training. BERT, in its research stages, achieved groundbreaking results in various natural language understanding tasks, including sentiment analysis, semantic role labeling, sentence classification.

This approach of using a network (pre) trained on a generic task and then applied (fine-tuned) for a specific task such a classification, can also be applied to a task on Named Entity Recognition. This can be done using the pre-trained BERT model made available by the Hugginface project.

In Figure 3.9 there is what can be considered the baseline model architecture for using BERT for a NER task.

BERT uses a specific representation of the language and does not work at word level but at sub-words (tokens) level. This implies that in order to use the model, the sentences and their labels (the BIO tags) must be preprocessed so the words in the sentences must be tokenized and the BIO-tags are distributed/re-assigned on the single tokens. Moreover, [CLS] token is inserted at the beginning of the first sentence and a [SEP] token is inserted at the end of each sentence.

This can be done using the tokenizer class made available by the Huggingface project. The results of this process is illustrated in the following examples where the ## sequence of characters means that the word has been split in tokens belonging the BERT vocabulary. In this way, BERT can handle (up to a point) never seen words or misspelled words.

The phrase:

```
There are many french painters.
```

is tokenized as

```
[CLS] /there/ are/ many/ french/ painters /[SEP]
```

As one can see, since the words are all in the dictionary there is no modification between words and tokens.

But if we look at the phrase

```
Monet was a french impressionist painter
```

the tokenization becomes.

```
[CLS] /mon /##et /was /a /french /impression /##ist /painter /[SEP]
```

Where it is possible to see the mechanism that allows BERT to handle previously unseen words by splitting them in known tokens.

As a consequence, the classification made by leveraging BERT is done at token level and, when finished, the original words must be reconstructed and a decision must be made on how to reassign the label in the presence of tokens (belonging to the same word) having different classification, which is one of the issues linked to the evaluation of a NER system results that will be addressed in a subsequent paragraph.

3.1.5 FAM for NER

The research on named entity recognition intends to explore the possibility of using the simultaneous availability of text and images present in the context of Twitter messages related to cultural heritage in the hypothesis that images can contribute to improving the information context that can be used to identify names of artists, titles of works of art, names of artistic movements and venues (exhibition places, museums, etc., ...).

We intend to use an approach inspired by factorization machines, introduced in the next paragraph, an algorithm capable of analyzing the correlations that occur within the inputs supplied to it.

3.1.5.1 Factorization Machines

Factorization machines (FMs) is a model introduced by Stephen Rendle in 2010 [14] related to factorization models. It is a approach that is well suited for settings characterized by sparsity in the data. It embeds features into a latent space and models the interactions between features via inner product of their embedding vectors for efficiently using the second-order feature interactions according to the following model.

$$\hat{y} := w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j \tag{3.1}$$

Where the model parameters that have to be estimated are

$$w_0 \in \mathbb{R}, \ \mathbf{w} \in \mathbb{R}^n, \ \mathbf{V} \in \mathbb{R}^{n \times k} \tag{3.2}$$

Where:

- w_0 is the global bias.
- w_i models the strength of the i-th variable.
- $\hat{w}_{i,j} :=< v_i, v_j > i$ models the interaction between the i-th and j-th variable

And $\langle \cdot, \cdot \rangle$ is the dot product of two vectors of size k:

$$\langle \mathbf{v}_i, \mathbf{v}_j \rangle := \sum_{f=1}^{k} v_{i,f} \cdot v_{j,f} \tag{3.3}$$

A row v_i within V describes the i-th variable with k factors. $k \in N_0^+$ is a hyperparameter the defines the number of latent factors that ones deems necessary to use for the problem at hand like in matrix factorization for recommender systems.

The last term in equation 3.1 can be expressed in the following way:

$$\sum_{i=1}^{n} \sum_{j=i+1} < \mathbf{v}_i \mathbf{v}_j > x_i x_j = \frac{1}{2} \sum_{f=1}^{k} ((\sum_{i=1}^{n} v_{i,f} x_i)^2 - \sum_{i=1}^{n} v_{i,f}^2 x_i^2) \tag{3.4}$$

Bringing the computational complexity from $\mathcal{O}(kn^2)$ to $\mathcal{O}(kn)$.

The choice of k is particularly critical in that the number of latent variables should be able to correctly capture the complexity of the information available in the data. In sparse contexts, a small k would be preferable,

because there is not enough data to estimate complex interactions. In such cases limiting k , and consequently, the expressiveness of the factorization machine, enhances generalization, resulting in improved interaction matrices when dealing with sparsity.

Factorization machines can be adapted to different tasks such as regression and classification. For regression $\hat{y}$ can be used directly by minimizing the mean squared error between the model prediction and target value, e.g. $\frac{1}{N}\sum^{N}(y-\hat{y})^2$.

It is possible to adapt them for a **multiclass classification task**, which is our case in the first part of the research, by adding a dimension, of size equal to the number of classes, to each of the parameters w_0, w, so they become vectors, and to the matrix V, that becomes a 3-D tensor as in figure 3.5. The computations of equations 3.1 are then applied individually to the slices, since there is no interference among them. The result will be a vector of size equals to the number of classes on which a $softmax$ can be applied to select the current prediction.

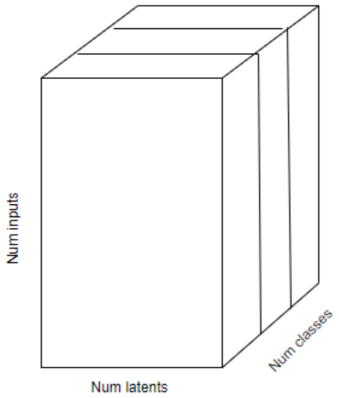


Figure 3.5: V matrix of a factorization machine extended as tensor for a multiclass classification task.

We propose a methodology inspired by Factorization Machine to analyze the interaction between textual and visual features in order to work as an attention mechanism layer for improving the performance of a neural network.

3.1.5.2 Ideal Features and Visual Dictionaries

Mixing textual and visual information is not an easy task, and it is not easy to assess the contribution of the modalities, especially for the visual part,

so, in order to have a term of comparison, in this respect two strategies were devised.

Both strategies creates syntehetic visual information that, since are generated from the labeled messages, perfectly match the textual portion of the information. The idea is to remove the ‘’noise” (light conditions, poorly related images,...) that a real image introduces

The first strategy creates vectors of “ideal features” (embeddings) starting from the textual part of the message, in this way the entire role of the visual network is removed. The following algorithm is used:

Ideal features creation algorithms

1: For a given labeled message and given that the number of entities are 8.
2: Create a vector of size of 4096 (the VGG19 embeddings size) and segment it in 8 portions of size 512. So portion 1 is from index 0 to index 511, portion 2 is from index 512 to 1023 and so on.
3: Map the entities to the portions with the following conventions.
- B-A portion 1
- I-A portion 2
- B-M portion 3
- I-M portion 4
- B-W portion 5
- I-W portion 6
- B-V portion 7
- I-V portion 8

4: For each entity in the message the corresponding portion is filled with all ones.

Repeated entities are not captured but in twitter messages this is rarely the case. Figure 3.6 shows an example for a message containing the name of an artist (of one word) and the name of an artwork (of one word).

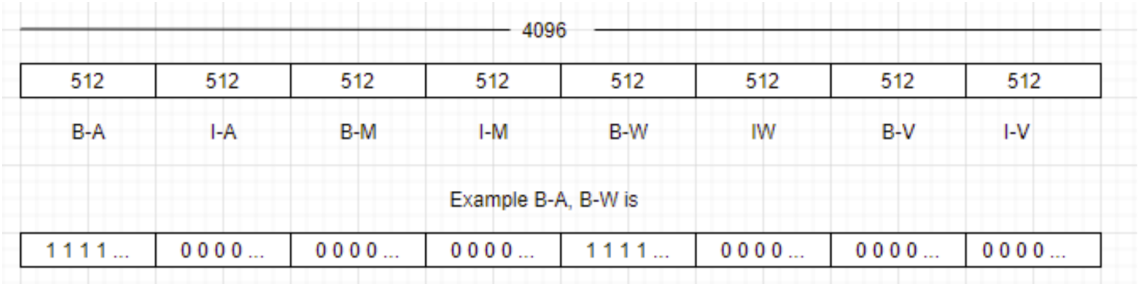


Figure 3.6: Ideal features example.

In this way it is possible to create a vector of the same size of the embeddings that can be obtained from VGG19 that can be interpreted as the embedding of an image that perfectly matches the text of the Twitter message. During some of the experiments this "ideal embeddings" will be injected in the models instead of the VGG19 embeddings.

The second strategy ceates a 'visual dictionaries" to obtain images that perfectly match the tagging of the textual part. The algorithm is as follow:

Ideal images creation algorithms

1: For a given labeled message, given that the number of entities are 8.
2: Create an image segmented in 8 different regions according to a chosen grid.
3: Map the 8 entities to the regions accroding to a chosen convention.
4: For each entity in the message fill the corresponding region with a color and/or a shape.

Examples of such dictionaries are depicted in Figure 3.7 and in Figure 3.8.

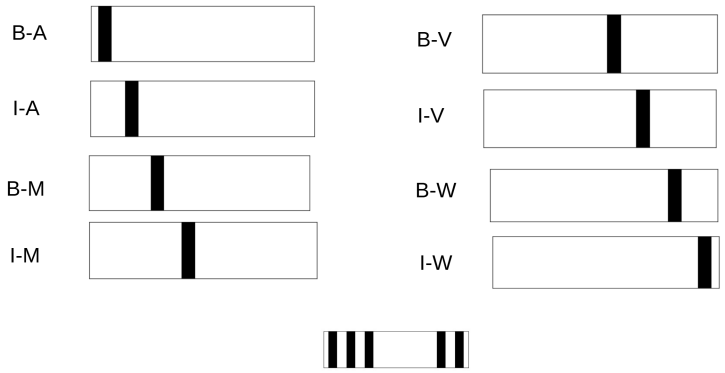


Figure 3.7: Visual dictionary example

In Figure 3.7 the image is segmented with a grid that results in a sort of bar code.

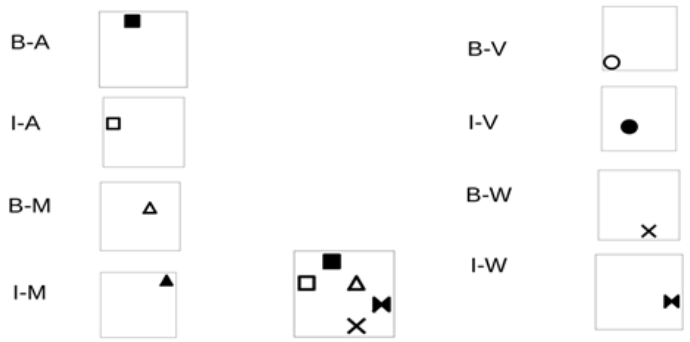


Figure 3.8: Visual dictionary example

Iin Figure 3.8 the image is segmented in four sectors, each one divided in two parts, so that the entities are differentiated by the position, the shape (square, circle triangle and cross), and the filling of the shape, white for the beginning and black for the inside.

Given the tagging definitions in both figures the central image represents a message containing the name of an artist made of two words, an artistic movement made of one word and the title of an artwork made of two words, The latter dictionary was the one used during the experiment having performed better in some trial.

These "perfect matching" images were given in input to VGG19 and used in the models. in order to have a term of comparison with the case where the real images were used.

3.1.6 NER Evaluation

Usually, NER systems are evaluated by comparing their outputs against human annotations. Exact-match or relaxed/partial match can be quantified for the comparison.

3.1.6.1 Exact-match Evaluation

NER essentially involves two subtasks: boundary detection and type identification. In "exact-match evaluation", an entity is correctly recognized if a system identifies both its boundary and type simultaneously.

The following quantities are generally used to quantify and analyze the performance: The numbers of True positives (TP), False positives (FP) and False negatives (FN) are used to compute Precision, Recall, and F-score.

- True Positive (TP): entity that is returned and also appears in the ground truth.

- False Positive (FP): entity that is returned but does not appear in the ground truth.
- False Negative (FN): entity that is not returned by a NER system but appears in the ground truth.

Precision refers to the percentage of results which are correctly recognized. Recall refers to the percentage of total entities correctly recognized by a system.

$$Precision = \frac{\#TP}{\#(TP + FP)}$$

$$Recall = \frac{\#TP}{\#(TP + FN)}$$

A measure that combines precision and recall is the harmonic mean of precision and recall, the traditional F-measure or balanced F-score:

$$F_{score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Two more metrics are the macro-averaged F-score and micro-averaged F-score. Both consider the performance across multiple entity types. Macro-averaged F-score independently calculates the F-score on different entity types, then takes the average of the F-scores. Micro-averaged F-score sums up the individual false negatives, false positives and true positives across all entity types, then applies them to get the statistics.

3.1.6.2 Relaxed-match Evaluation

Some studies [52] [53] use a relaxed-match evaluation. In this case an entity is considered correctly detected if its type is correctly assigned regardless to its boundaries as long as there is an overlap with ground truth boundaries.. However, complex evaluation methods are not intuitive and make error analysis difficult. Thus, these evaluation methods are less used in recent studies.

3.1.6.3 Models

The development of the models experimented during the research was done using Pytorch (https://pytorch.org/), a framework for machine learning applications. Originally developed by Meta AI now is part of the Linux

Foundation. It is used in research and industrial (Tesla, Uber) contexts. It supports the implementation and training of neural networks through the package torch.nn and allows the execution of computation on both the CPU and the graphics cards (GPU) hardware, which greatly accelerates the operations involving tensors.

The following images represents three types of models implemented. The pretrained BERT model is obtained from Huggingface platform (https://huggingface.co/) while the pretrained VGG16 is obtained from Pytorch project. The models were trained for 400 epochs with a batch size of 32 using crossentropy loss and adamW optimizer. During the training the weights of BERT are not frozen [1].

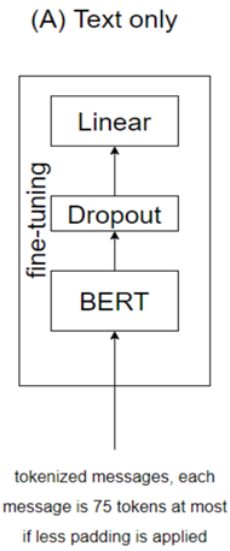


Figure 3.9: Baseline model (A) for unimodal NER task using BERT

Figure 3.9 shows a model for only text NER, the text embeddings are obtained using BERT. This is considered the baseline against which observe the behavior of the model without the contribution of the images (real and generated) or of the ideal features.

[1] In all the experiments conducted throughout the research, the number of epochs has been chosen in order to have a balance between performance improvement, time costs and computational costs.

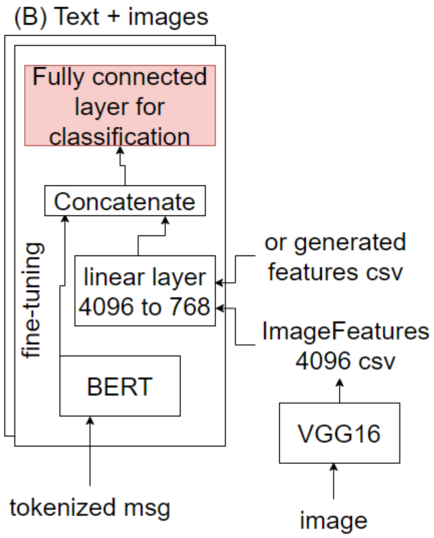


Figure 3.10: Model (B) for multi-modal NER task using BERT

Figure 3.10 shows a model for multimodal NER. In this case the text embeddings are obtained using BERT while the image embeddings are obtained from VGG16. In order to obtain embeddings from the image branch of the same size of the BERT embeddings, which are vectors of length 768 a linear layer is applied with an input dimension of 4096 (the size of VGG16 embeddings) and an output dimension of 768. The two embeddings are then concatenated and sent through a fully connected layer for the final classification in eight classes. Instead of the images this model can take, as input, the ideal features described in 3.1.5.2

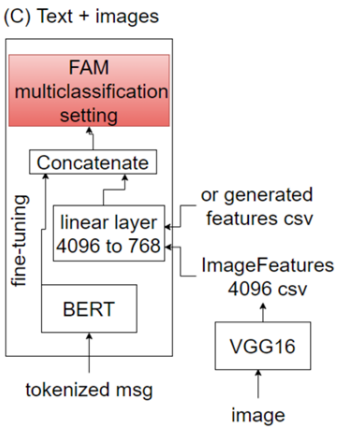


Figure 3.11: Model (C) for multi-modal NER with a FAM module

Figure 3.11 shows the model for multimodal NER where is present our factorization machines inspired approach. In this case, the text embeddings are obtained using BERT while the image embeddings are obtained from VGG16. In order to obtain embeddings from the image branch of the same size of the BERT embeddings, which are vectors of length 768. The obtained embeddings are then given in input the **Factorization Module (FAM)** implemented as a custom pytorch netwok layer where the factorization machine is adapted for a multiclassification setting, with the idea that the ability of the FAM to analyze the interaction within and between the vectors will contribute to the improvement of the performance. Instead of the images, this model can also take, as input, the ideal features described in 3.1.5.2

3.1.7 Experiments

In the following the experiments performed with results and considerations.

The experiments were conducted utilizing the Chameleon cloud computing platform [54], employing Nvidia P100, M40, and K80 graphics cards, as well as on the HPC4AI computing infrastructure at the University of Turin [55], utilizing Nvidia Tesla T4 graphics cards.

3.1.7.1 Experiment Using Model A and B

This experiment aims at verifying the behavior of a fully connected layer as classifier (model B) compared to the only text model (model A).

The parameters used are:

- epochs: 400.
- batch size: 32.
- dataset size: 1500 samples

	Text only (A)	text + ideal features	text + real images	text + generated images	text+ random images
Artist	0,675	0,723	0,744	0,7	0,667
Movement	0,776	0,8	0,758	0,776	0,745
Venue	0,5	0,667	0,545	0,6	0,667
Artwork	0,571	0,564	0,571	0,564	0,658

Figure 3.12: F1 scores results using model (B) with ideal features, real images, generated images and random images.

Observation:

- In this case, the use of ideal features shows a general improvement on all the classes with respect to the text-only model. The use of generated or real images improves on the artist class but not on the other. The case of text associated with random images is less damaging than one could expect and indeed improves on the Venue and Artwork classes.

3.1.7.2 Experiment Using Model A and C

.

This experiment aims at verifying the behavior of FAM layer as classifier (model C) compared to the only text model (model A).

The parameters used are:

- epochs: 400.
- batch szze: 32.
- dataset size: 2100 samples

	Text only (A)	text + ideal features	text + real images	text + generated images	text+ random images
Artist	0,693	0,745	0,45	0,467	0,699
Movement	0,861	0,843	0,812	0,61	0,875
Venue	0,541	0,811	0,305	0,206	0,378
Artwork	0,687	0,725	0,623	0,555	0,675

Figure 3.13: F1 scores results using model (C) with ideal features, real images, generated images and random images.

Observation:

- With this model, including ideal features along with the text shows a general improvement in all the classes with respect to the text-only model. Using generated or real images does seem to damage the performance. The case of text associated with random images is degrades the performance with the surprising result on the Artistic movement class.

3.1.8 Experiments on Another Dataset

In order to compare the methodology with a literature example, the model was applied to a dataset from the Zhang 2018 paper introduced in paragraph 3.1.2.5

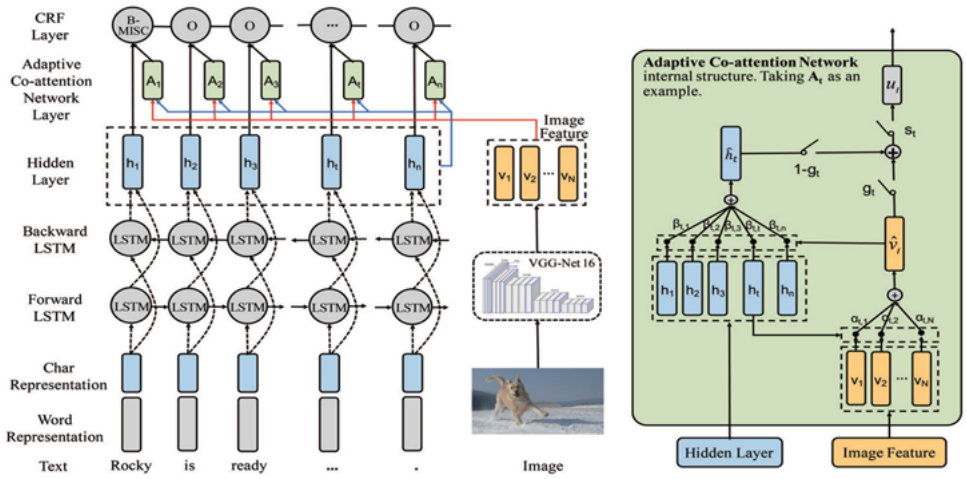


Figure 3.14: Multimodal NER architecture [2]

In this paper, the named entity recognition is done on the more conventional schema of Location, Organization, Persons and Others.

The dataset is composed of 8,257 tweets posted by 2116 users. The total number of entities is 12,784. The dataset is splitted into three parts: training set, development set, and testing set, which contain 4,000, 1,000, and 3,257 tweets, respectively

Table 3.1 reports the results between the Zhang's paper results the text only model (A) using BERT and model (C) using FAM

F1 Score

Entity	Zhang's paper	Text-only (model A)	Text and real images (model C)
LOC	0,789	0,802	0,796
ORG	0,530	0,570	0,568
OTHER	0,340	0,394	0,387
PER	0,819	0,831	0,840

Table 3.1: Comparison between F1 scores of Zhang's paper, text only model (A) and multi modal Model (C)

Observation: In this case the model text only model (A) based on BERT improves over the Zhang paper's results, also model (C) is slightly better than the Zhang's paper results but compared with the text only model only the Person category gets benefit from incorporating images.

3.1.9 Some Final Considerations

Here are some key observations drawn from this phase of the research:

- When available, including ideal features alongside text generally improves performance on the majority of classes compared to a text-only model.
- The presence of generated or real images can help depending on the model and classes, but can also deteriorate the performance, as can be seen for the model including the FAM module. A possible explanation being the very high variability of images associated to the text messages.
- Interestingly, there are instances where randomly pairing text and images leads to performance improvements for certain classes.

Part II

A Novel Attention Mechanism and Data Augmentation

Chapter 4

Previous Works on Attention

The attention has now become an important component in current neural networks solutions and has been researched within diverse application domains. In the following of this chapter an overview of the development of the attention mechanism is given togheter with its application in different tasks and settings.

4.1 Different Types of Attention

The concept of attention emerged and evolved, with different flavors, in two different contexts; neural machine translation and computer vision.

As with neural networks themselves a biological comparison can be used to give the intuition behind the concept of attention, where, for example, in the viewing process, the focus is not on the entire scene in front of an observer but only on some parts that the observer (perhaps instinctively) considers most relevant for understanding what is in front of him.

Regarding computer vision, in 2014 Mnih et al. [10] proposed a method to focus on important parts of an image in order to process at high resolution. The image, instead of being entirely processed at once, is processed sequentially, attending to different locations considered relevant to the task.

Always in 2014, in the field of neural machine translation, the most common framework was the sequence-to-sequence (seq2seq) [6] Sutskever et al. (2014). In seq2seq, there are two recurrent neural networks (RNNs) [56] arranged in an encoder-decoder architecture: the encoder side reads the input words one by one to obtain a vector representation of a fixed dimension, and, conditioned on these inputs, the decoder part extracts the output words one by one using another RNN.

The main problem with seq2seq is that the last encoder hidden state is

the only information that the decoder receives, a vector representation that is like a numerical summary of an input sequence. So, for a long input text, the decoder receives too little information to output a translation, in particular, the first word of a sequence leave too few traces in the last hidden state representation. A situation sometimes called catastrophic forgetting.

A human analogy would be like to start to translate a text only after having read all the text that must be translated.

The problem was addressed in a 2014 paper by Bahdanau et al. [7] The idea was to give the decoder not just the information from the last hidden state but a vector representation from every encoder time step, so that it could make more informed translations.

Going back to the human analogy, in this case, while reading the text the translator takes notes he will reuse during the actual translation. The mechanism by which this additional information from every encoder hidden state is passed to the decoder is called *attention*. In this way, the model doesn't forget useful parts of the input sequence and can use them (pay attention to them) for the translation.

Figure 4.1 gives a visualization of Bahdanau attention.

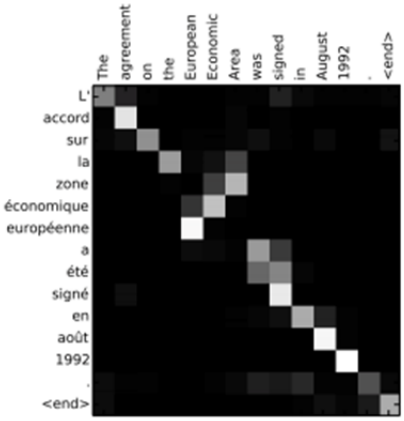


Figure 4.1: Bahdanau's attention [7], the grey cells highlight the attention put on the words of the starting language to produce the translation in the target language.

With Vaswani et al. "Attention is All You Need" paper [4] in 2017 a new transformer architecture was introduced which computed what the authors named "Scaled Dot-Product Attention" (4.1), realized through the flow depicted in Figure 4.2.

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d^k}})V \tag{4.1}$$

Here the notation recalls information retrieval systems and K/V/Q stand for key/value/query concepts. An intuitive example can be drawn from a search for videos on a website where the query (the text in the search bar) is mapped against a set of keys (video title, description, etc.) associated with videos in a database, the best matches are are then presented as results along with the associated videos (the values).

An important novelty introduced in the same paper is the so-called *self-attention*, which is realized when K and V are equal and enables the model to weigh the importance of different elements in an input sequence (a phrase), allowing to capture relations between distant part of the sequence.

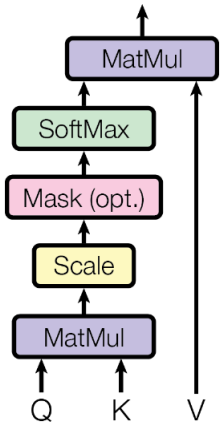


Figure 4.2: Scaled dot product attention as descried in Vaswani's paper [4]

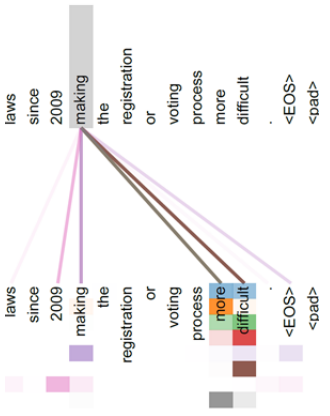


Figure 4.3: Vaswani's attention.

A further evolution came from computer vision, when in 2015 [5] Xu at

al. proposed the use of visual attention to the task of image captioning. They introduced the concept of *soft attention* and *hard attention*. Soft deterministic attention is smooth and differentiable, is trained by standard back propagation and focuses on smoother regions of the image, while hard stochastic attention is trained by maximizing an approximate variational lower bound and focuses on more circumscribed regions of the image. Soft attention is similar to Bahdanau et al.'s proposal.

Figure 4.4: An example of hard attention on the left aside an example of soft attention on the right. The difference is in the role of the surrounding part of the image that is excluded in the hard version and somehow kept into consideration in the soft one

Given that attention was successfully applied to language and vision problems separately, it was a natural evolution to apply this approach to tasks where text and images are naturally connected and more generally where inputs of different nature (textual, visual, auditory, ...) coexist.

A number of tasks belong to this category.

Named Entity Recognition, as adressed by Zhang 2018 paper [2] introduced in paragraph 3.1.2.5.

Image-caption alignment, like in the paper by by Liu, Mao et al. [57] where a portion of an image is linked to a portion of a caption introduced in section 5.1.

The architectures of the models by Zangh and Liu are reported in Figure 3.3 and 4.5, it is possible to see that, despite the difference of tasks, there are commonalities between these multi-multimodal approaches. In both cases two distinct neural networks, one for each modality, process the inputs and then the attention mechanism contributes to the different tasks of the models.

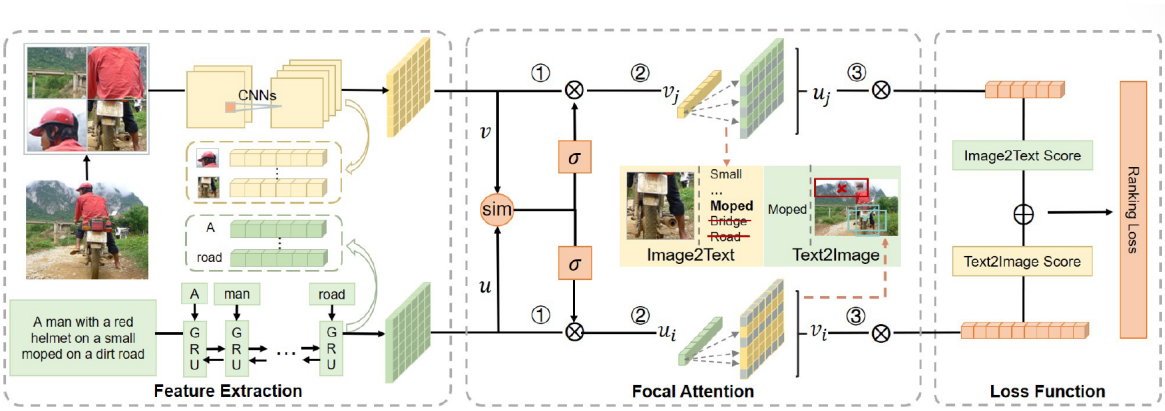


Figure 4.5: BFAN model overall architecture: consists of feature extraction, focal attention and loss function module. The focal attention module takes the extracted features as input, and then attends to regions and words interactively [57]

Multimedia Description, generates a natural language textual description of a multimedia input sequence which can be image and video [Cho et al. 2015] [58].

Even more sofisticated is the task of **Human Communication Comprehension** by Zadeh et al. 2018 [59] that addresses face-to-face communication which involves language, vision and speech modalities simultaneously. In this case attention is used to highlight interactions between different modalities.

Chapter 5

A Novel Type of Attention J-FAM

In this chapter a novel type of attention, J-FAM (Joint Factorization Module), potentially able to be applied in a multi-modal setting, characterized by data sparsity, is proposed with the results of experiments of its application.

The experiments were conducted utilizing the Chameleon cloud computing platform [54], employing Nvidia P100, M40, and K80 graphics cards, as well as on the HPC4AI computing infrastructure at the University of Turin [55], utilizing Nvidia Tesla T4 graphics cards.

5.1 The Task: Image Caption Alignment

Text image matching is an interesting field of research where an algorithm tries to find a semantic match between images and texts.

Among the existing works in the field, a paper by Liu et al. [57], "Focus Your Attention: A Bidirectional Focal Attention Network for Image-Text Matching", addresses the problem of finding associations between words of a caption and portions of a corresponding image, see Figure 5.1. The paper comes with the availability of the code and the dataset [60], making it a good term of comparison for the experiments where our approach is introduced.

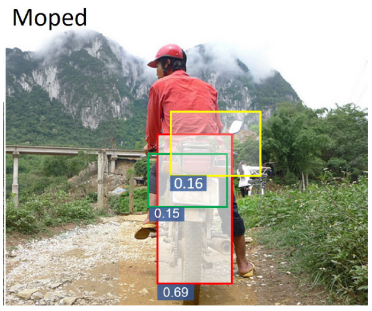


Figure 5.1: Correspondence of the word moped, a low-powered motorcycle, with the region of the image that contains it.

In Figure 5.2 the Bidirectional Focal Attention Network (BFAN) architecture of Liu's paper is presented.

The framework of BFAN consists of a neural network for extracting features for the images parts, a neural network for extracting features for the textual part, a - so called - focal attention module and a loss function module.

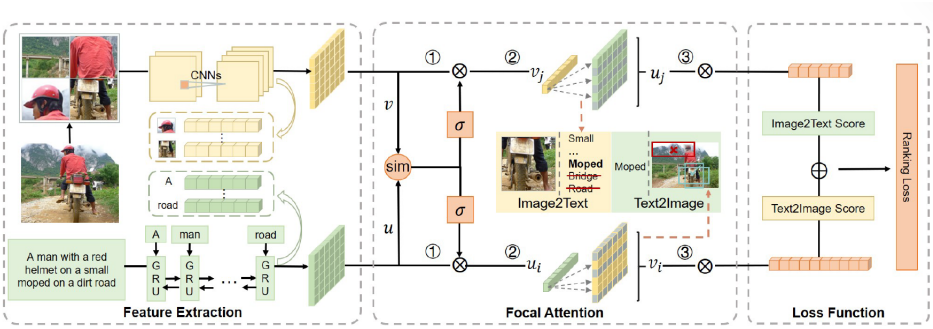


Figure 5.2: BFAN model overall architecture: it consists of feature extraction, focal attention and loss function module. The focal attention module takes the extracted features as input, and then attends to regions and words interactively [57].

BFAN methodology operates in the belief that irrelevant portions of text and images damage the text-image alignment process. To avoid this problem, after the features extraction is performed by the two initial neural networks, two steps are performed in both the text to image and the image to text directions. In a first step (point 1 in Figure 5.2), BFAN assigns, using cosine similarity, attention scores S to all the parts in one modality based on fragments from the other modality. In a second step (point 2 in Figure 5.2) the scores are compared with their average $< S >$.

$$F(x) = S(x) - < S(x) > \tag{5.1}$$

Scores above the average are considered related to relevant fragments while scores below the average related to irrelevant parts, in this way it is possible to build an **indicator function**:

$$H(X) = I(F(x) > 0) \tag{5.2}$$

used to multiply by 1 the representation of relevant parts and by 0 the representations of the irrelevant parts, removing them.

5.1.1 Introduction to J-FAM Attention

In Figure 5.3 two possible areas of insertion of an attention mechanism inside the BFAN architecture are highlighted. The blue rectangle would completely remove the focal attention module by substituting it with a factorization machine attention based approach. The green rectangle instead depicts a scenario where the factorization machine based attention is used as a pre-processing activity applied on the embeddings extracted for the image and the textual parts.

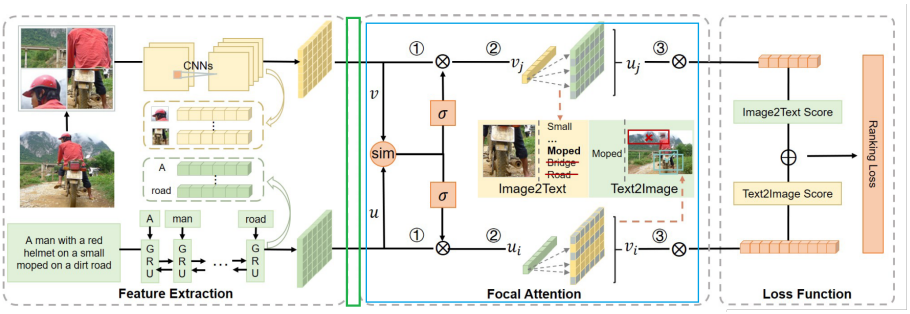


Figure 5.3: BFAN model [57], blue and green rectangles denote possible point of insertion of different types of attention.

This, ultimately, is the way J-FAM is applied, J-FAM is inserted after the features extraction neural networks but before the first step of computation of the attention scores described in the previous section. J-FAM acts as a pre-attention processing of the inputs before they are passed to the logic of relevance detection of the fragments, in the belief that the applied novel attention mechanism is able to give improved representations of parts of texts and images.

Figure 5.4 illustrates the architecture applied.

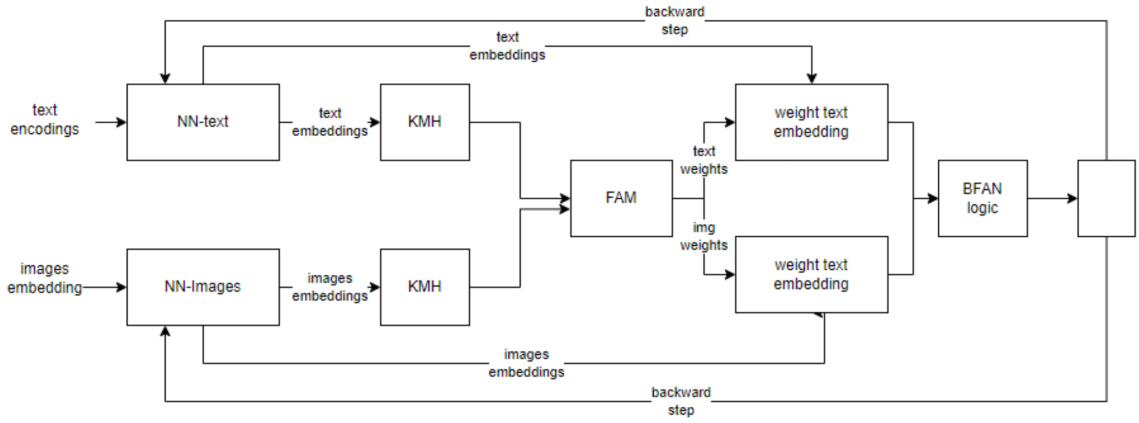


Figure 5.4: Proposed model for inserting FAM attention in the BFAN model.

5.1.2 The Methodology

In Liu's model, a sample, composed of parts of an image and the image caption, is represented in embedding form by the following two tensors.

- Caption tensor $C \in \mathbb{R}^{n \times m}$ with n equals to number of words in the caption and m dimension of the embedding (1024).

- Image tensor $I \in \mathbb{R}^{i \times m}$ with i equal to the number of relevant parts of the image (always 36) and m dimension of the embedding (1024).

This implies that the representation of a caption changes for each sample depending on the number of words in the caption.

A factorization machines inspired algorithm inherits the requirement of a tabular format for the training data, where the rows are of fixed size and each row represents a sample. See Figure 5.5.

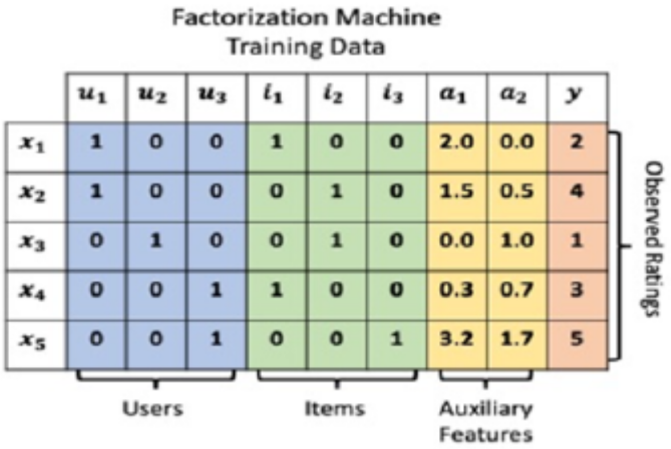


Figure 5.5: Input data shape for a factorization machine based algorithm [14], In our case, a row is a concatenation of a text related vector and an image related vector.

There is the need to cast the tensor $C \in \mathbb{R}^{n \times m}$ into a vector $C' \in \mathbb{R}^u$ and the tensor $I \in \mathbb{R}^{i \times m}$ into a vector $I' \in \mathbb{R}^v$ so that their concatenation $S \in \mathbb{R}^{u+v}$ will represent a single sample.

Recalling that the training of a neural network is usually done not one sample at a time, but using batches of samples, it is possible to achieve this desired result by creating histogram vectors using the result of a clusterization algorithm like k-means, applied at batch level as described in the following section.

5.1.2.1 The K-means and Histogram Algorithm (KMH)

Let first introduce the k-means clusterization algorithm and the associated elbow method.

The *k-means clustering algorithm* works as follows: Given a training set of vectors $x^{(1)}, ..., x^{(m)}$, we want to group the data into a few cohesive "clusters". Here, we are given feature vectors for each data point $x^{(i)} \in \mathbb{R}^n$; but no labels $y^{(i)}$ (so this is an unsupervised learning setting). Our goal is to predict k centroids and a label $c^{(i)}$ (belonging to a cluster) for each datapoint.

The aim is to cluster the embeddings that are more similar because we believe they are also close in meaning. The algorithm works iteratively. When the assignments no longer change, the algorithm has converged.

Algorithm 1 K-means clustering algorithm assuming euclidean distance

1: Initialize cluster centroids $\mu_1, \mu_2, ..., \mu_k \in \mathbb{R}^n$ randomly, These can be random points in the dataset.

2: Compute the membership degree of all feature vectors in all clusters C_i at iteration t.

$$C_i^{(t)} = \left\{ x_p : \left\| x_p - \mu_i^{(t)} \right\|^2 \leq \left\| x_p - \mu_j^{(t)} \right\|^2 \ \forall j, 1 \leq j \leq k \right\}, \tag{5.3}$$

3: Recalculate the means (the new centroids) given the assignements found in the previous step

$$\mu_i^{(t+1)} = \frac{1}{\left| C_i^{(t)} \right|} \sum_{x_j \in C_i^{(t)}} x_j \tag{5.4}$$

Given an element of the dataset, in addition to the assignment to a cluster, another output that can be obtained from the algorithm is its distance from the centroid of the class, which can give an idea of how strong its membership to a class is. This information will be used in conjunction with others during the experiments.

Obviously, in this process, one must decide the number k of clusters to be used. An indication for this choice can be obtained using an heuristic methodology called elbow method.

The *elbow method* is a method of cluster analysis used to justify the number of clusters to be used in a clustering algorithm (e,g, k-means).

During the elbow method execution, the number of hypothetical clusters k is progressively increased. For each value of k, a quantity called Within-Cluster Sum of Square (WCSS) is calculated. WCSS is the sum of the squared distance between each point and the centroid in a cluster. After plotting the WCSS for each k values, the plot can be analyzed (see Figure 5.6). WCSS value is largest when $k = 1$. As the number of clusters increases, the WCSS value decreases and it may show a transition point (the elbow) where the rate of decrease in WCSS slows down, suggesting that adding more clusters doesn't significantly improve the clustering quality.

The K corresponding to the transition point is the one used as the "optimal" number of clusters. Algorithmically the elbow method can be expressed as in Algorithm 2.

Algorithm 2 Elbow method

1: For $k = 1,2,3, ...$
2: For each k execute k-means
3: Compute the Within-Cluster Sum of Square (WCSS) defined as:

$$\text{WCSS} = \sum_{i=1}^{k} \sum_{x \in C_i} ||x - c_i||^2 \tag{5.5}$$

where

- k is the number of cluster
- C_i is the i-th cluster
- x is an element in C_i
- c_i is the centroids of C_i

4: Create a chart of WCSS for all the chosen k and evaluate the change in slope

As said, it is an heuristic approach, the transition point may not be so pronounced, resulting in a certain degree of uncertainity in the choice for the number of clusters.

Figures 5.6 and 5.7 show the result of the application of the elbow method on a batch of captions and corresponding visual embeddings. It is not so clear, but a change of slope can be located between 10 and 23 for the captions and 15 and 35 for the images. It must also be considered that in a batch there are less embeddings for words than for visual parts since the captions have around seven to ten words each while the embeddings for each image are always 36.

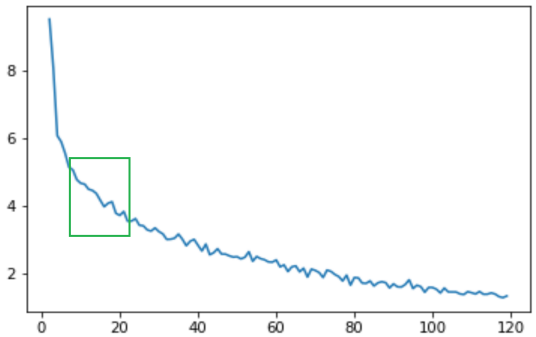


Figure 5.6: Elbow method for a batch of captions, with the "elbow area" approximately spanning from 10 to 23

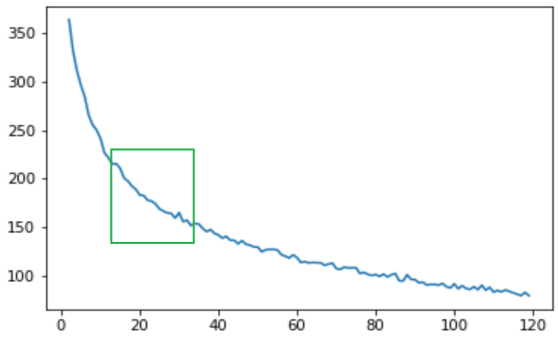


Figure 5.7: Elbow method on a batch of parts of images, with the "elbow area" approximately spanning from 15 to 35

Having introduced K-means and the elbow method, we can describe our **KMH algorithm**.

Let u be the number of clusters for text and v the numbers of clusters for part of images. After the execution of the clusterization, separately for the embeddings of the words and the embeddings of the parts of images, it will be possible to build for each caption and each parts of image a corresponding histogram vector as follows.

Recalling that **C'** $\in R^u$ and **I'** $\in R^v$ are the vectors introduced in the previous section, the single components of the vectors will be valorized in the following way:

$$\begin{aligned} C'_1 &= \text{number of caption's words in cluster 1} \\ C'_2 &= \text{number of caption's words in cluster 2} \\ &\dots \\ C'_u &= \text{number of caption's words in cluster u} \end{aligned} \quad . \tag{5.6}$$

so that that the vector becomes a histogram of the belonging to the clusters.

The same happens for the parts of image vector.

$$\begin{aligned} I'_1 &= \text{number of parts of image in cluster 1} \\ I'_2 &= \text{number of parts of image in cluster 2} \\ &\dots \\ I'_u &= \text{number of parts of image in cluster v} \end{aligned} \quad . \tag{5.7}$$

This will be repeated for each sample in the batch so that the final batch can be defined as $B \in \mathbb{R}^{b\times(u+v)}$ with b equal to the size of the batch.

This is a format that can be used to train a factorization machine.

Algorithm 3 gives an algorithmic description of the procedure.

Algorithm 3 KMH algorithm

1: Let the tensor $C \in \mathbb{R}^{n \times m}$ be the embedded representation of a caption of n words each one represented by a vector of size m.
2: Let the tensor $I \in \mathbb{R}^{i \times m}$ be the embedded representation of the parts of a corresponding image with i equal to the number parts and m the dimension of the embedding.
3: Consider a set of b couples of Cs and Is so that $B_C \in \mathbb{R}^{b \times n \times m}$ represents a set of embeddings of words in captions and $B_I \in \mathbb{R}^{b \times i \times m}$ represents a set of embeddings of parts of images
4: Perform k-means for u cluster on the embeddings of the words in B_C and for v clusters on the embeddings of the parts in B_I.
5: define a vector $\mathbf{C'} \in \mathbb{R}^u$ and a vector $\mathbf{I'} \in \mathbb{R}^v$
6: define w_i the number of words of a caption in cluster i and p_i the number of parts of an image in cluster i
7: Given the assignment $\mathbf{C'}_i = w_i$, $\mathbf{C'}$ will represent the histogram vector associated to a caption.
8: Given the assignment $\mathbf{I'}_i = p_i$, $\mathbf{I'}$ will represent the histogram vector associated to the visual part.

Also, for each word/part of image the inverse distance from the centroids is stored with the aim of using it as part of the overall re-weighting mechanism. The inverse is used in order to give more importance to the elements near to the centroids. This final tensor can be given as an input to the FAM module.

Figure 5.8 describes a toy example of KMH applied to a four word caption (word embeddings of size 5) and corresponding image composed of embeddings of the relevant image parts (size 6 in the sample) with clustering effects and vector histogram creation and concatenation .

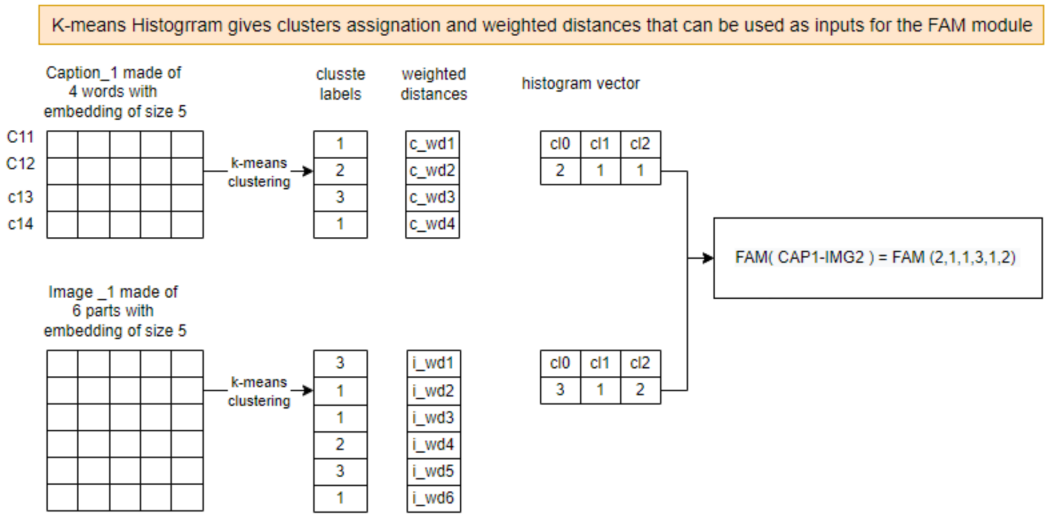


Figure 5.8: A toy example (in very low dimensions) of the effect of KMH application on a couple caption/image inside a batch made of embeddings for a caption of four words and embeddings for 6 parts of an image. For the caption: 2 words belong to cluster 1 and one word each to cluster 2 and 3, for the image: 3 words belong to cluster 1, 1 word to cluster 2 and 2 words to cluster 3.

In the following sections two more issues will be addressed:

- Training targets.
- How to use the outputs of the factorization machine training as attention weights.

5.1.2.2 Training Targets

Factorization machines are a supervised algorithm. In order to be trained, each input data needs a target value and the training aims to minimize a loss function, such as an MSE loss. In a recommendation system, the targets can be the marks that users give to items (movies, books, etc...).

The case at hand is different since predefined targets are not present, so the factorization machine part of the Algorithm is trained assuming a target value of 1 for each couple caption-image, meaning with this that the caption and the image are correctly related.

In Figure 5.9 the chart of the training of a FAM on a batch over 3000 epochs with an MSE loss based on L2 norm while in Figure 5.10 is the chart of the training of a FAM isung an MSE L1 norm . In the case of L2 norm the plateau is reached at around 400 epochs while for the L1 norm is reached at aornt the 50th epoch and this was the choice when used within a model.

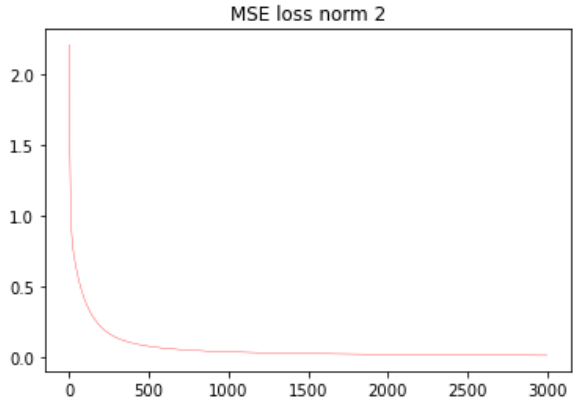


Figure 5.9: FAM training over 3000 epocs with MSE L2 norm

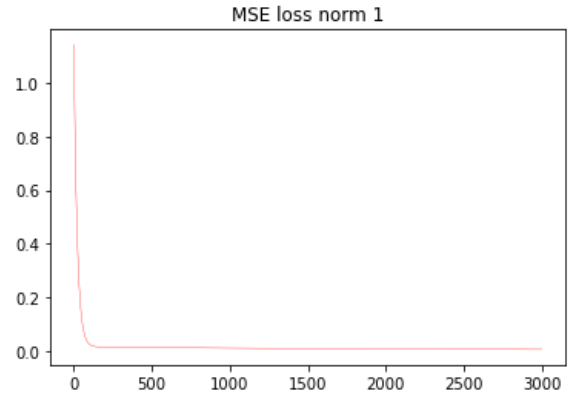


Figure 5.10: FAM training over 3000 epocs with MSE L1 norm

5.1.2.3 Extraction and Use of Weights

Recalling the factorization machines equation:

$$\hat{y} := w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j \tag{5.8}$$

With respect to equation 5.8 and given our construction of the clustering, after the training the linear coefficient w_i of the factorization machine are related to the clusters and interpreted as the relevance that a cluster has in the coupling text-images.

In order to extract weights from the cross part of the model, the matrix V of equation 3.2 and 3.3 of the weights is multiplied with its transpose to obtain a matrix

$$W = VV^T \tag{5.9}$$

The square root of the diagonal elements of W, $X_i = \sqrt{W_{ii}}$ are related to the clusters.

At the end of the process of clustering and factorization machine training, three quantities are available for reweighting the original embeddings of the words in the captions and the embeddings of the parts of the images.

- The inverse distance from the centroids I (size equal to the number embeddings in the batch.
- the trained FM linear coefficients w_i (size equal to the number of clusters)
- the trained FM cross coefficients X_i (size equal to the number of clusters)

With the weights obtained from the FAM we can build a tensor of weights F of length equal to the number of elements in the embeddings and values for each element based on the belonging to a particular cluster.

So, if, for example, embeddings e_i (e.g. the embedding of a word) belongs to cluster c_j, the corresponding J-FAM weight will be $F_i = F_i(w_j, X_j)$ e.g. $F_i = L_j + X_j$.

Taking into account the inverted distance D_i of embeddding e_i from its centroid, the total weight of the e_i embedding will have the form :

$$T_i = T_i(D_i, F_i) \tag{5.10}$$

There are a number of possible choices for the actual functions applied on D and F, some of them, with the i index dropped, are listed below. where Softmax and Normalization are applied to the whole set of distances and weights to avoid problems of scale.

- $T = norm(D) * norm(w + X)$
- $T = softmax(D) * norm(w + X)$
- $T = softmax(D) * softmax(w + X)$
- $T = softmax(norm(D) * norm(w + X))$

The final reweighted embedding will be of the form

$$e_w = e * T \tag{5.11}$$

e_w is then renormalized in order to be processed by the BFAN algorithm.

5.2 The Dataset and the Recall@k Metric

The datasets involved are Flickr30K [61] and MS-COCO [62]. Flickr30K is a standard dataset for image-text matching, it contains 31,000 images and 155,000 texts in total, each image relates to five texts. Flickr30K is split into 29K training images, 1K validation images and 1K testing images. MS-COCO is a large-scale benchmark dataset that contains 123,287 images with five texts each. The images used are 113,287 for training, 5,000 for validation and 5,000 for testing.

Liu's paper is built on top of another work by Lee et al. [63] which provides precomputed datasets of embeddings correspondings to the portions of the images from Flick30K and MS-COCO these datasets of embeddings are the ones actually used..

To compare the results between the Liu's code and the models experimented the metric used in Liu's paper is used which is the Recall@K [64] [57]. It is a common metric in information retrieval. It measures how many relevant items (true positives) are in the first K results against how many relevant items exist in the entire dataset, that is, the sum true positives and false negatives.

$$Recall@K = \frac{\text{truePositives}}{\text{truePositives} + \text{falseNegatives}} \tag{5.12}$$

By the definition, one can observe that with recall@K, the score improves as K increases and the scope of returned items increases, eventually reaching 1 when K is large enough to include all the relevant items, in the worst case, when K matches the cardinality of the test dataset.

Recall@K is an easily interpretable evaluation metric The lower the K the harder is the retrieval and a score equal to one means that all relevant items have been returned.

It is an order unaware metric. It does not indicate where the relevant items are among the K returned.

5.3 Experiments with Weights as Multipliers

In this section the results for some experiments using the weigths obtained from J-FAM as multipliers according to equation 5.11 are reported.

In the tables below there are the results, expressed as Recall@K, of the experiments.

The Recall@K is reported in both the text to image direction and the image to text direction. it also reported the sum of these two values.

The weights are obtained with the following function, chosen, among the various options, to have a distribution of weights where the various components have been normalized in order to avoid imbalance between the weights.

$$T = softmax(norm(D) * norm(w + X)) \tag{5.13}$$

In each case the J-FAM component was trained for 50 epochs and the overall model was trained for 15 epochs. The other parameters, number of J-FAM latents, number of text clusters and number of images clusters were changed individually after having chosen a default.

Table 5.1 reports the results obtained running Liu's paper code from github [60].

Liu's code Recall@K

	R@1	R@3	R@5	R@10
text_to_img	59.9	79.9	86.6	92.2
img_to_text	45.4	65.5	73.3	82.7
sum	105.3	145.3	159.9	174.9

Table 5.1: Liu's code Recall@K

The following values were considered the default values for the parameters cited above:

- Number of factorization machine latent: 32
- Number of text clusters: 12
- Numer of image clusters: 30

Recall@K wih default values

	R@1	R@3	R@5	R@10
text_to_img	60.0	79.8	86.8	92.9
img_to_text	44.8	65.4	73.6	82.2
sum	104.8	145.2	160.4	175.1

Table 5.2: Recall@K wih default values

In table 5.3 the results of an experiment with less text clusters, from 12 to 8 are reported.

Recall@K with number of text clusters reduced

	R@1	R@3	R@5	R@10
text_to_img	59.0	81.0	86.2	92.8
img_to_text	44.9	65.6	73.3	82.9
sum	103.9	146.6	159.5	175.7

Table 5.3: Experiment with number of text clusters reduced to 8

In table 5.4 the results of an experiment with more text clusters, from 12 to 20 are reported.

Recall@K with number of text clusters augmented

	R@1	R@3	R@5	R@10
text_to_img	58.1	79.4	85.6	92.1
img_to_text	44.6	64.3	72.9	82.2
sum	102.7	143.7	158.5	174.3

Table 5.4: Experiment with number of text clusters augmented to 20

In table 5.5 the results of an experiment with less image clusters, from 30 to 25 are reported.

Recall@K with number of image clusters reduced

	R@1	R@3	R@5	R@10
text_to_img	58.1	79.4	85.6	92.1
img_to_text	44.6	64.3	72.9	82.2
sum	102.7	143.7	158.5	174.3

Table 5.5: Experiment with number of image clusters reduced to 25

In table 5.6 the results of an experiment with more image clusters, from 30 to 45 are reported.

Recall@K with number of image clusters augmented

	R@1	R@3	R@5	R@10
text_to_img	56.1	79.5	87.2	93.1
img_to_text	44.6	65.2	73.2	82.3
sum	102.7	144.7	160.4	175.4

Table 5.6: Experiment with number of image clusters reduced to 45

In table 5.7 the results of an experiment with less J-FAM latent, from 32 to 27 are reported.

Recall@K with number of J-FAM latent reduced

	R@1	R@3	R@5	R@10
text_to_img	59.0	79.6	88.0	93.3
img_to_text	45.1	66.0	73.9	82.6
sum	104.1	145.6	161.9	175.9

Table 5.7: Experiment with number of J-FAM latent reduced to 27

In table 5.8 the results of an experiment with more J-FAM latent, from 32 to 40 are reported.

Recall@K with number of J-FAM latent augmented

	R@1	R@3	R@5	R@10
text_to_img	58.1	80.3	86.9	93.0
img_to_text	44.2	64.7	73.1	81.5
sum	102.3	145.0	160.9	174.5

Table 5.8: Experiment with number of J-FAM latent augmented to 40

Observations: Looking at the results, there is not an approach that is clearly better than the others. The best Recall@1 is the one from Liu's code. The best Recall@3 is obtained with the reduced number of text clusters, the best Recall@5 and Recall@10 are obtained reducing the number of J-FAM latent.

Conducting additional experiments to obtain a deeper understanding of these findings would have incurred significant costs in both computational resources and time. Therefore, we opted to explore an alternative approach as described in the following section.

5.4 Experiments with N% of Cluster and M% of Cluster Elements

As written earlier another possible way to use the weights generated by J-FAM is to use them to drive an augmentation schema at the batch level.

The approach, formalized in Algorithm 4, is to take a portion of the clusters and inside the chosen clusters, a portion of the samples.

These selected samples are modified, adding gaussian noise, and added to the batch.

Algorithm 4 Augmentation with w% of clusters and k% of elements of clusters

1: Let W be the vector of J-FAM weights associated to the clusters,
2: Sort W in descending order according to the weights and Let $C_{w\%}$ be the set of clusters corresponding the first (the highest) w% weights in the sorted W.
3: For each $C' \in C_{w\%}$:
4: Let $D_{k\%}$ be the subset of C' containing the $k\%$ elements which are closest to the cluster's centroid.
5: Let S be a training sample $\in C'$.
6: If $S \in D_{k\%}$ then modify it adding gaussian noise with mean 0 and standad deviation 10^{-5} and add S to the batch.

In the following we presents and discuss the outcomes of the experiments with this approach.

For these experiments the following settings were used.

- Number of clusters for the captions: 12
- number of clusters for the images: 30
- number of latent features for the factorization machines: 36
- Percentages of clusters with highest values as given by J-FAM linear weights: 10%, 30%, 90%
- Percentages of elements as ranked by their distance from their cluster centroid: 10%, 30%, 90%

In the following tables there are the results, expressed as Recall@K, of the experiments.The models have been trained for 18 epochs each.

Image-to-Text denotes retrieved texts using image query, and Text-to-Image denotes retrieved images using text query. The Recall@K is reported in both the text to image direction and the image to text direction. it also reported the sum of these two values.

Observation. As shown in the tables below, in general, the results the Liu's code and the modified version are too close to declare that there is

a true improvement, but it is possible to observe that the best results are obtained with the lower values (10% cluster, 10% elements) while adding more and more samples to the batches seems to damage the performance.

5.4.1 10% High Value Clusters

Liu's code Recall@K

	R@1	R@3	R@5	R@10
text_to_img	63.2	81.6	88.1	94.5
img_to_text	45.1	66.4	74.6	83.1
sum	108.3	148	162.7	177.6

Table 5.9: Liu's code Recall@K

10% cluster 10% elements Recall@K

	R@1	R@3	R@5	R@10
text_to_img	63.8	81.4	87.5	94.0
img_to_text	46.0	66.4	74.5	82.9
sum	109.8	147.8	162.0	176.9

Table 5.10: Recall@K for 10% high value clusters 10% nearest elements

10% cluster 30% elements Recall@K

	R@1	R@3	R@5	R@10
text_to_img	60.7	82.0	88.5	94.3
img_to_text	45.3	66.6	74.8	82.5
sum	106.0	148.6	162.9	176.8

Table 5.11: Recall@K for 10% high value clusters 30% nearest elements

10% cluster 90% elements Recall@K

	R@1	R@3	R@5	R@10
text_to_img	63.3	82.2	87.7	94.8
img_to_text	46.2	66.7	74.1	83.0
sum	109.5	148.9	161.9	177.8

Table 5.12: Recall@K for 10% high value clusters 90% nearest elements

5.4.2 30% High Value Clusters

Liu's code Recall@K

	R@1	R@3	R@5	R@10
text_to_img	63.2	81.6	88.1	94.5
img_to_text	45.1	66.4	74.6	83.1
sum	108.3	148	162.7	177.6

Table 5.13: Liu's code Recall@K

30% cluster 10% elements Recall@K

	R@1	R@3	R@5	R@10
text_to_img	62.1	82.8	87.8	94.1
img_to_text	44.8	65.9	73.7	83.0
sum	106.9	148.7	161.5	177.1

Table 5.14: Recall@K for 30% high value clusters 10% nearest elements

30% cluster 30% elements Recall@K

	R@1	R@3	R@5	R@10
text_to_img	63.4	82.3	87.6	94.1
img_to_text	44.6	65.7	73.9	82.8
sum	108.0	148.0	161.5	176.9

Table 5.15: Recall@K for 30% high value clusters 30% nearest elements

30% cluster 90% elements Recall@K

	R@1	R@3	R@5	R@10
text_to_img	62.2	80.3	86.5	93.7
img_to_text	44.5	65.3	73.3	82.6
sum	106.7	145.6	159.8	176.3

Table 5.16: Recall@K for 30% high value clusters 90% nearest elements

5.4.3 90% High Value Clusters

Liu's code Recall@K

	R@1	R@3	R@5	R@10
text_to_img	63.2	81.6	88.1	94.5
img_to_text	45.1	66.4	74.6	83.1
sum	108.3	148	162.7	177.6

Table 5.17: Liu's code Recall@K

90% cluster 10% elements Recall@K

	R@1	R@3	R@5	R@10
text_to_img	62.5	80.9	87.0	93.5
img_to_text	44.4	65.5	73.5	82.2
sum	106.9	146.4	160.5	175.7

Table 5.18: Recall@K for 90% high value clusters 10% nearest elements

90% cluster 30% elements Recall@K

	R@1	R@3	R@5	R@10
text_to_img	62.4	82.0	88.3	94.6
img_to_text	44.8	65.6	74.4	82.9
sum	107.0	147.6	162.7	177.5

Table 5.19: Recall@K for 90% high value clusters 30% nearest elements

90% cluster 90% elements Recall@K

	R@1	R@3	R@5	R@10
text_to_img	61.8	81.9	88.6	94.1
img_to_text	44.8	65.0	73.1	82.5
sum	106.6	146.9	161.7	176.6

Table 5.20: Recall@K for 90% high value clusters 90% nearest elements

5.5 Experiments Removing the Indicator Function

The results of the experiments described in the previous section don't give a clear signal of deviation from the results obtained with Liu's code. A set of experiments was conducted switching off the indicator function 5.2 in section 5.1 that is keeping all the fragments, while keeping the J-FAM augmentation schema on. Given the previous results, the experiments were limited to the case of addition of the 10% best clusters and 10% best elements.

Tables 5.21 and 5.22 show the results for the modified Liu's code and for the model with the pre-attention for the experiments conducted on the Flickr30K dataset.

Tables 5.23 and 5.24 show the results for the modified Liu's code and for the model with the pre-attention for the experiments conducted on the MS-COCO dataset.

[Flickr30K] Recall@K
attention mask switched off

	R@1	R@3	R@5	R@10
text_to_img	65.2	81.6	88.0	92.5
img_to_text	46.8	67.9	76.1	83.3
sum	112	149.5	164.1	175.8

Table 5.21: Recall@K of Liu's code (indicator function switched off) applied to the flickr30k dataset.

[Flickr30K] Recall@K

	R@1	R@3	R@5	R@10
text_to_img	64.0	83.5	89.4.0	95.2
img_to_text	47.5	68.3	76.0	84.2
sum	111.5	**151.8**	**165.4**	**179.4**

Table 5.22: Recall@K for J-FAM with 10% high value clusters 10% nearest elements and indicator function swithed off.

Observations:

- A comparison between tables 5.21 and 5.22 shows that J-FAM models prevails on R@3, R@5 and R@10 but again by a small margin.
- Comparison with the results with the indicator function active, tables 5.17 and 5.22, shows that the versions with the indicator function switched off off have better performance.

[MS-COCO] Recall@K,
indicator function switched off

	R@1	R@3	R@5	R@10
text_to_img	67.3	87.1	92.8	97.5
img_to_text	55.1	78.5	86.2	93.7
sum	122.4	165.6	179.0	191.2

Table 5.23: Recall@K of Liu's code (indicator function switched off) applied to the MS COCO dataset.

[MS-COCO] Recall@K for
10% high value clusters 10% nearest elements

	R@1	R@3	R@5	R@10
text_to_img	68.5	87.3	93.2	97.7
img_to_text	55.3	78.6	86.76	94.0
sum	**123.8**	**165.9**	**179.9**	**191.7**

Table 5.24: Recall@K for J-FAM with 10% high value clusters 10% nearest elements and indicator function switched off.

Observations:

- A comparison between tables 5.23 and 5.24 shows again a general improvement but with very small differences.
- We encountered two challenges. Firstly, despite utilizing the code provided on GitHub, replicating the performance documented in Liu's paper proved unattainable. Secondly, disabling a significant segment of Liu's paper algorithm actually resulted in performance improvement. Further exploration of these findings would have required significant investments in both time and computational resources. Therefore, we opted to explore the application of a factorization machine inspired methodology in a different domain: data augmentation.

Chapter 6

Data Augmentation

In this part of the research we want to explore methodologies able to give a measure of diversity between the samples already present in a training dataset and modified versions of these samples. The goal is to find criteria able to suggest which portion of a training dataset is better to modify and use for expanding the number of training samples in order to improve the performance of a model.

In this chapter, a review of the data augmentation landscape is followed by the description of the approaches explored during the research activity: a factorization machine-based approach, a transfer learning approach utilizing cosine similarity, and an augmentation approach customized for an SVM classifier employing Euclidean distance.

The experiments were conducted utilizing the Chameleon cloud computing platform [54], employing Nvidia P100, M40, and K80 graphics cards, as well as on the HPC4AI computing infrastructure at the University of Turin [55], utilizing Nvidia Tesla T4 graphics cards.

6.1 Introduction

Data are a very important component for obtaining good performance from a machine learning algorithm. Given a domain, training an algorithm on large and representative data almost always results in good performance. Conversely, training on small and poorly representative data leads to poor generalization performance no matter how sophisticated an algorithm is. Unfortunately, in many practical applications, obtaining data in large quantity and of high quality, in terms of labeling and diversity is challenging if not impossible. Another challenge is that, depending on the task, even very

large datasets [11] lack the necessary diversity for obtaining good performance.

Data augmentation aims to increase the available training data both in quantity and in diversity, in order to increase the accuracy and the generalization performance of an algorithm. This is done by using the available data to generate new training samples, that can increase the overall quality of the training data [11].

6.2 Previous Works

For a given dataset D made of training samples S and corresponding labels L, augmentation finds transformations T that can be applied to the original samples S to create additional training data S' in such a way that the transformed sample

$$S' = T(S)$$

can still be described with the labels L. These transformations, known as label-preserving transformation operations in image processing and computer vision literature [13], can vary in their preservation of labels depending on the dataset.

For instance, mirroring an image in a task involving the classification of numbers does not preserve labels, whereas it might in a task focused on classifying fashion items. However, human evaluation still plays a role in determining the extent of label preservation.

Augmentation approaches can be organized in a few categories as done in some surveys [11] [12]. In the following each category is taken into consideration.

6.2.1 Input Space Augmentation

The input space approach directly modifies the input images, completely or partially, in order to increase variability and improve the models' generalization performance. This is an intuitive approach and can be human-monitored.

Input space modifications can be subdivided in groups that we illustrate in the following sections.

6.2.1.1 Geometric Transformations

These transformations change the pixels position without modifying the pixels values. The idea is to expose the network to image variations not

present in the original training set, but that can be encountered in real scenarios. Simple transformations are rotation, shearing, translation, resizing, reflections, flipping [65] [66].

More complex transformations involve perspective or projective transformations, which are utilized to generate new images presenting different viewpoints for an observer [67].

6.2.1.2 Photometric Transformations

This type of transformations modify the values of the pixels of the images. This is done to take into account the fact that an image can be affected by the camera used to take it and by the shooting conditions. The modification of the RGB channels of an image is a typical example of a photometric transformation.

In this way, it is possible to modify visual properties such as brightness, color, contrast, sharpness and saturation levels. In turn, this can be related to different shooting conditions such as weather conditions, time of the day or lighting conditions.

6.2.1.3 Region Level Augmentation

Previously described techniques are applied to the entire image. In time new techniques appeared that considered manipulation of only portions of the images, this is done in order to force the models not to focus on specific portions of an image.

A number of strategies have been devised.

- Region Deletion: during the training process some portions of the images are deleted in order to introduce diversity. Regions deletion can also be interpreted as a form of occlusion which can be useful for object tracking tasks where occlusion can occur as a consequence of a dynamic environment. It is a form of information dropping that can help the model to concentrate on the more relevant information. In this respect, it is similar to dropout but done at the input level and not at feature level.

  An example of this technique is Cutout [68] where random patches are deleted by replacing them with zero valued pixels. Random Erasing by Zhong et al. [69] applies a similar approach but with a mask of random size and values of pixels in the range 0-255.

Gong et al. [70] tackle the problem of avoiding the deletion of useful portions of the image. Informative portions of the image are identified during the training, and only non-informative parts are then deleted.

- Region replacement: instead of deleting portions of the training images, in this approach, portions of the images are cropped, and then pasted in random positions on the original image or in different images of the training set. Possibly these cropped images can be transformed before being pasted.

  This approach aims at avoiding information loss problems due to erasing. An example is [71] where a thumbnail of the same or another training image is placed at a random place in a training image.

  In the case of mixing of different images, the labels also must be modified accordingly.

- Region combination: In this approach the patches from the same or different images are added instead than replaced.

6.2.2 Feature Space Augmentation

This approach aims to increase variability by manipulating vectors extracted from the intermediate layers of deep neural networks.

This type of strategies have interesting characteristics. In theory they allow to learn representations invariant to input space image transformations and manipulating vectors is computationally cheaper than manipulating or generating images. Moreover, they can be combined with other strategies and can be independent of the nature of the input.

Some drawbacks, however, exist. The vectors can be difficult (or impossible) to relate to the original images and modifications not based on domain knowledge may potentially miss important domain attributes.

A few approaches are reported below.

- Transformation of deep features: Transforming feature vectors extracted from the internal layers of a model is an attempt to enrich representation by increasing the diversity in data and improving the performance of a model by preventing it from learning specific configuration of the data.

  Shen [72] proposes to perform affine transformation (translation, rotation and scaling) while Li [73] injects Gaussian noise into the layers

of a Convolutional Neural Network (CNN) to increase diversity and prevent overfitting.

- Mixing of features: this approach aims at enriching learned features by augmenting them with artificially generated feature vectors or combining features extracted from different input samples [74].

- Feature elimination: This technique consists in dropping neural connections in order to introduce sparsity in the network. Connections typically are dropped with a certain probability and result in training different versions of a base model. The final result averages the effects of this different version. This prevents overfitting of the data by encouraging diverse features representation. In a sense, this is the features level equivalent of the region deletion approach at the input level.

  Aside from suppressing features, it is also possible to enhance features [75] or avoiding their suppression [76] at least for some random unit.

  Some recent works [77], [78] aim at improving on the random deletion approach by identifying features that can be considered less important and then proceeding to their elimination.

6.2.3 Image Synthesis

When it's challenging to obtain real data with distributions that match the need to work in a setup with controlled parameter distributions, they can be generated artificially. This approach also gives the possibility to construct data tailored to the specific task.

6.2.4 Meta-learning

Meta-learning is a relatively new approach that finds application in data augmentation.

Meta-learning approaches can be divided in two broad classes.

- Deep learning models leverage meta-data that can perform well on different tasks.

  In this case, data augmentation uses few-shot learning methods to increase the sample efficiency of machine learning methods.

- Methods devoted to automatically find the best hyperparameters.

Applied to augmentation, they search for the augmentation schema that maximizes the performance of a model on a given task.

Our contributions, described in following sections, aim to determine whether it's possible to identify not just the most convenient augmentation scheme, but which parts of a dataset are most suitable for a particular augmentation scheme.

6.3 First Approach: FM for Augmentation

This approach explores the hypothesis of exploiting a factorization machines inspired methodology in a scenario of data augmentation.

As previously stated, our objective is to explore whether the utilization of factorization machine (FM) can effectively identify the subset of a dataset that would offer the greatest potential for enriching diversity within a training dataset via data augmentation techniques.

The idea is to train a FM on the unmodified images/caption and then make the trained FM predict the set of modified images.

A few samples of the images used is shown in Figure 6.5, they are take from the dataset detailed in section 6.4.

Figure 6.1: jacket

Figure 6.2: t-shirt

Figure 6.3: trousers

Figure 6.4: vest

Figure 6.5: fashion items examples

The images are modified using the following transformations:

- rotation
- flip up-down
- flip left-right
- images with added gaussian noise

The modified images are associated with the original caption.

Similarly to what described in section 5.1.2.2 the FM is trained using the model in figure 6.6, on the whole training set this time, again, with the target 1 we designates the correctly associated images/captions in the training set.

The text embeddings are derived using SBERT, a model introduced by Reimers et al. in 2019 [79]. SBERT, short for "Sentence-BERT" is a model designed to produce vector representations of sentences or text fragments. It extends the popular BERT (Bidirectional Encoder Representations from Transformers) model to generate embeddings specifically tailored for sentences rather than individual words or tokens.

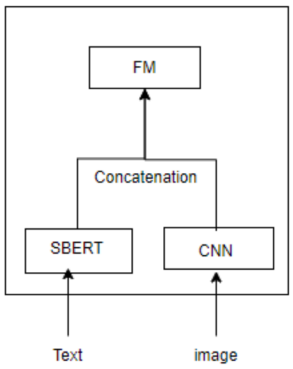


Figure 6.6: Model used to train the factorization machine. SBERT (for the text) and the custom CCN neural network (for the images) are used to obtain the embeddings needed as input for the factorization machine.

In Figure 6.7 an histogram of the results of the training is shown, as expected the predictions on data used for the training are close to 1.

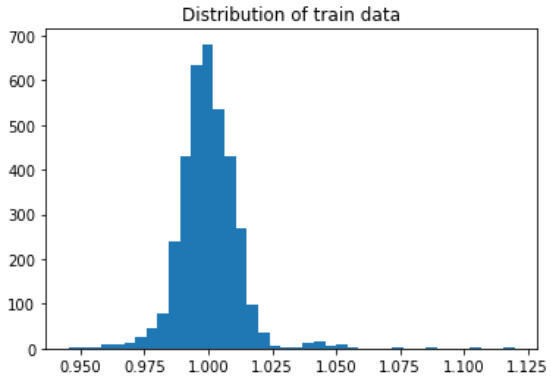


Figure 6.7: Distribution of FAM prediction for all the unmodified training data. After the completion of training, the data are tightly grouped around the value of 1 (the target) as expected.

After the training, the FM is applied to the modified images, the results of this operation can be categorized as follows:

- Images/captions having FM prediction near to 1. In this case the modified samples are not very diverse from the samples already seen during the training,

- Images/captions FM prediction more and more distant from 1. In this case the modification has created more diverse samples.

From these considerations stems the idea of applying the following steps:

Steps for distribution creation

1: For each of the above transformations
2: Apply the transformation to all the images of the dataset.
3: Obtain the distribution of FM predictions for the transformed dataset.
4: Use the distribution to take decision about what portion of the transformed dataset use to augment the training dataset.

So, given a set of modified images, the distribution is segmented in five regions: A, B, C, D, E see Figure 6.8 where region A is the less diverse from the original training dataset and regions E and D are the most diverse from the unmodified training dataset.

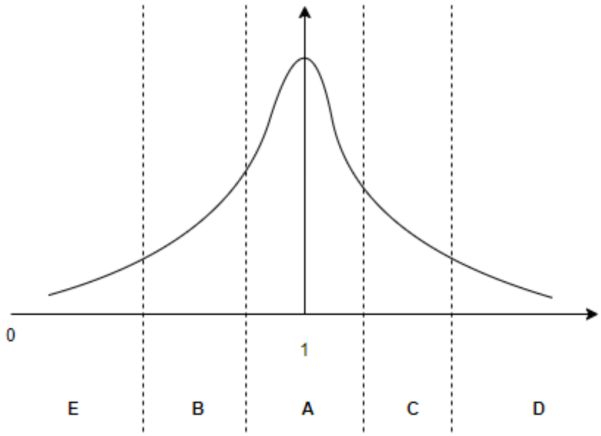


Figure 6.8: Qualitative diagram of FM predictions distribution with the inidication of the regions

An annotation: the experiments are performed on many different splits of the dataset in train and test subsets, in order to take average values of the performance. What has been observed is that, the distributions, depending on the split of the dataset (in training and test set), can be quite different from each other as shown in the following figures (6.9, 6.10, 6.11, 6.12). They all refer to the distribution of the predictions for images obtained applyng a 30 degree rotation to the images in the training dataset. What is different is that the train datasets are from different splits of the full dataset (indicated by the random seed, used for replicability, in the function that perform the split).

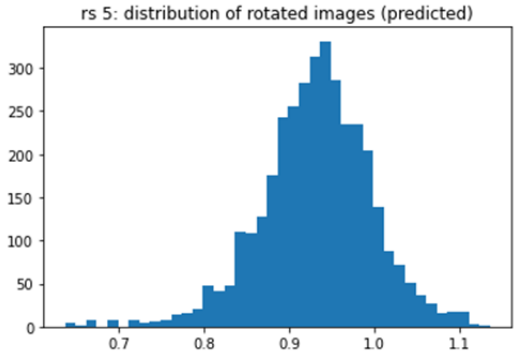


Figure 6.9: Distribution of FM predictions for images obtained rotating the images of the training dataset (random seed 5).

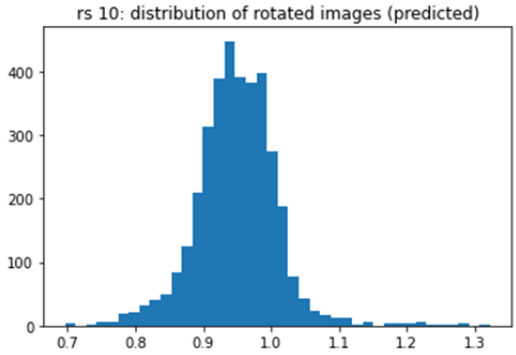


Figure 6.10: Distribution of FM predictions for images obtained rotating the images of the training dataset (random seed 10).

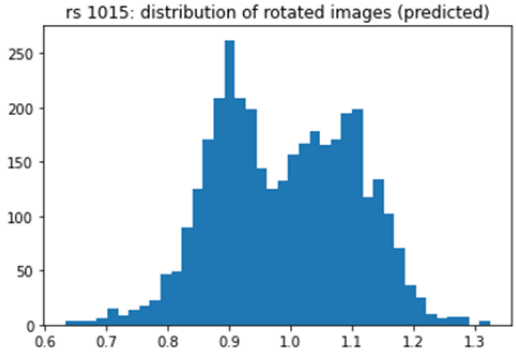


Figure 6.11: Distribution of FM predictions for images obtained rotating the images of the training dataset (random seed 1015).

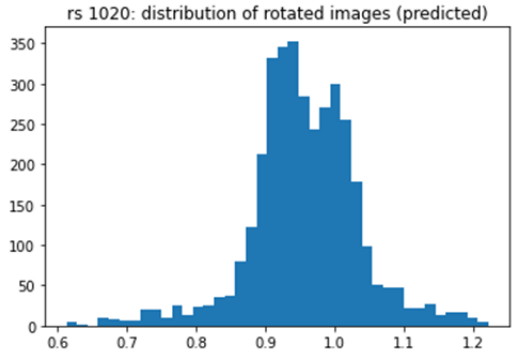


Figure 6.12: Distribution of FM predictions for images obtained rotating the images of the training dataset (random seed 1020).

The sizes of the regions have been chosen in order to have at least 500 images to add to the training set. The following section illustrates how this

organization of the data has been used in the experiments.

6.4 Dataset and Experiments

The dataset is derived from the Fashion Product Images (Small) dataset [80], composed of 44000 fashion products containing images of the products, a classification (160 classes) of the types of items (shirt, trousers, ...), a description. Figure 6.13 displays an histogram of the classes showing that is a quite unbalanced dataset.

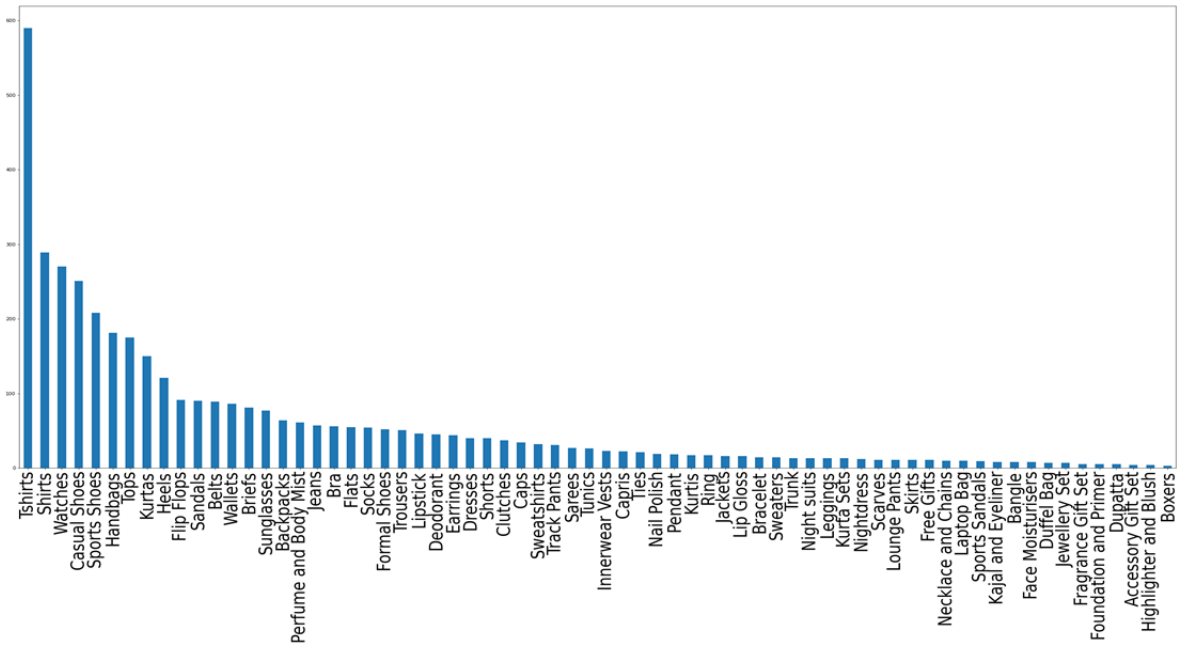


Figure 6.13: Small clothes dataset histogram of the classes

To avoid problems due to the unbalancing, it has been reduced to 23 classes each one containing 174 images for a total of 4002 images. 3600 images are used as training set 400 images are used as test set.

The experiments involved images taken from the single regions and from various unions of the single regions (e.g. BA). In the case of union of regions, the total number of images added to the training set does not change. The total number of scenarios is 15. N (not augmented), A, B, C, D, E, P, R (RANDOM), ABC, AC, ACD, EA, BA, CD, EBA. The R scenario takes samples without regard for the position of the predictions.

The P scenario takes images from the union of A, B, C: 30% of samples come from B, 50% of samples come from A and 20% of samples are from C.

Each of the scenarios corresponds to a different mix of diversity that is added to the training dataset. For example, taking samples having prediction in region A means adding the less diverse possible samples to the training set while taking images having prediction in region E or D means adding the most possible diverse sample to the training set. But discovering the most convenient scenario needs experiments.

In Figure 6.14 the model used for the experiments is presented. The image part is processed by a custom CNN while the textual part is processed by a pretrained SBERT model from huggingface. The outputs are concatenated and sent to a linear layer for the classification.

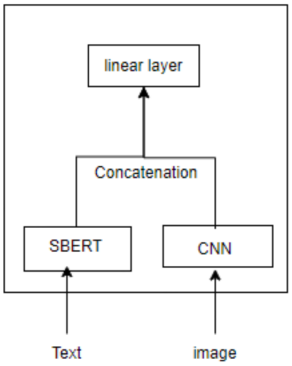


Figure 6.14: Model used for the experiments on the different regions scenarios.

In the following results and considerations from the experiments are presented.

	N	A	B	C	P	R	ABC	AC	ACD	D	E	EA	BA	CD	EBA
Avg	0,84228	0,84694	0,84615	0,84650	0,84607	0,84694	0,84743	0,84693	0,84708	0,84469	0,84776	0,84734	0,84810	0,84635	0,84717

Figure 6.15: Average F1 score

	N	A	B	C	P	R	ABC	AC	ACD	D	E	EA	BA	CD	EBA
Avg	9,73797	7,83422	8,07487	7,91979	8,11230	7,79144	7,67380	7,91979	7,65775	8,67380	7,48663	7,74866	7,37968	8,05348	7,93583

Figure 6.16: Average F1 score ranking

		N	A	B	C	P	R	ABC	AC	ACD	D	E	EA	BA	CD	EBA
Best solution	1	4	12	15	12	10	10	22	12	15	13	15	12	14	7	14
	2	9	20	10	12	9	15	10	12	14	5	13	14	11	17	16
	3	8	10	14	17	14	17	12	12	11	13	10	10	9	18	12
	4	7	8	12	11	12	16	14	10	18	15	16	12	20	6	10
	5	10	13	16	14	10	6	14	15	13	16	16	9	14	12	9
	6	7	14	10	11	19	11	11	8	12	6	11	23	19	11	14
	7	9	15	12	11	13	15	11	18	10	11	13	7	14	13	15
	8	8	14	11	14	10	10	13	14	14	9	18	16	11	16	9
	9	19	8	6	13	15	14	7	10	12	11	13	19	12	15	13
	10	13	14	12	10	12	15	12	16	12	13	14	11	12	10	11
	11	16	11	10	13	10	16	12	21	8	9	7	14	13	14	13
	12	16	15	14	10	17	9	8	8	11	14	13	9	17	12	14
	13	19	6	20	14	14	9	16	12	12	11	9	14	8	10	13
	14	18	11	14	12	17	16	10	10	14	19	8	9	7	11	11
Worst solutior	15	24	16	11	13	5	8	15	9	11	22	11	8	6	15	13

Figure 6.17: Average F1 overall ranking

	N	A	B	C	P	R	ABC	AC	ACD	D	E	AE	AB	CD	EBA
Good at returning best results	102,454868	126,376422	126,771448	124,450793	125,534856	117,974574	129,637957	121,651141	129,853764	123,955637	125,829249	123,401783	129,896112	128,918579	125,658267
Good at avoiding worst results	60,5639681	83,7835451	82,3253352	78,8840443	81,5136628	87,9297651	87,9297651	80,7394403	90,4484656	83,1178167	85,4181139	80,4302954	82,9703697	84,4073757	81,0010062
Good at returning best results	15	6	5	10	9	14	3	13	2	11	7	12	1	4	8
Good at avoiding worst results	15	6	9	14	10	2	2	12	1	7	4	13	8	5	11

	N	R	A	B	C	D	E	P	AB	AC	AE	CD	ABC	ACD	EBA
mean effectiveness	15	6	7	12	9,5	8	2,5	12,5	1,5	9	5,5	12,5	4,5	7	9,5

Figure 6.18: Summariy of ranking

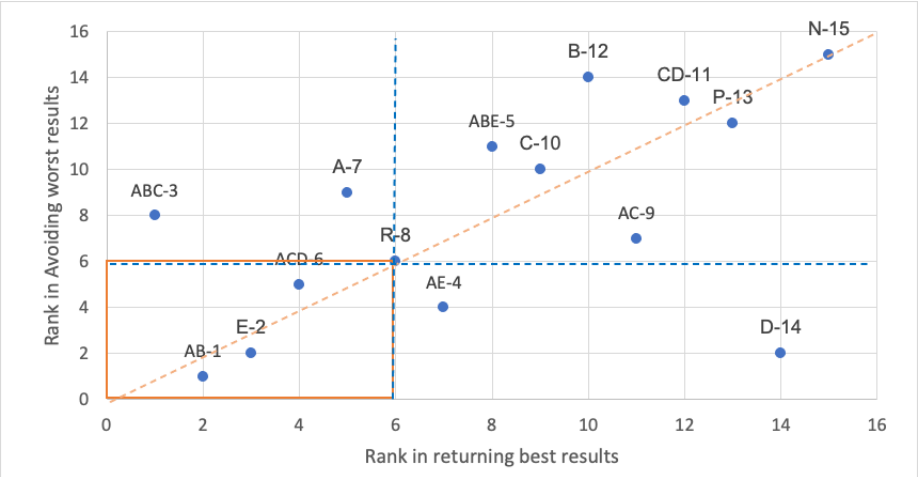


Figure 6.19: Chart of rankings of the various scenarios with respect to their abillity to return the best results and avoid the worst results

A first approach for looking at the results of these experiments is from the perspective of the series of the metrics (e.g. the F1 score) of the different runs for each random state and look at the difference of the mean between the series using a a paired t-test, a test where the null hypothesis is that the average of the differences between the series of paired observations is zero. On the other hand given a certain mean difference between the series it is possible to compute [81] the sample size needed to say that the difference satisfies a paired t-test having a certain type-I error probability of rejecting the null hypothesis when true (e.g 0.05) and a certain type-II error probability of accepting the null hypothesis when false (e.g 0.2) .

In our case, to give an example, for the F1 score series between the Random setting and the B sector after 190 runs we have a mean of difference of -0.00092 and a standard deviation of 0.01650 which implies a sample size of 2527, beyond our computational and time resources.

A different approach to look at the results is described below.

Figure 6.15 displays the average F1-score of the various scenarios after 160 runs, each one with a different random state for the dataset splitting, the higher the better. the BA and E scenarios are the best ones and the N (not augmented scenario) is the one with the lowest mean F1 score.

Figure 6.16 displays the average ranking of the scenarios after 160 run. Also thiis view confirms that E and BA regions are the best choices and the N scenario, ranking in mean at around the 10th position (9.73) is the one with the worst performance, confirming that augmentatation helps.

Figure 6.17 displays how many times the scenarios appear at a certain

ranking.

Figure 6.18 and chart 6.19 display an elaboration on the data of Figure 6.17 aimed at ranking the strategies according to their ability to get the best results and avoid the worst results. using two scores S_b and S_w obtained in the following way.

For each strategy (N, R, A, ...), let x_i with $i = 1,2,...,15$ be the number of times such strategy obtains the i-th position.

The ability of the model of getting the best results is expressed by equation 6.1.

$$s_b = \sqrt{\sum_{i=1}^{15} x_i(16-i)^2} \tag{6.1}$$

While the ability of the model of avoiding the worst results is expressed by equation 6.2.

$$s_w = \sqrt{15^2 \sum_{i=1}^{15} x_i} - \sqrt{\sum_{i=1}^{15} x_i i^2} \tag{6.2}$$

Observations.

A first observation is that, for this dataset, augmentation seems to help, the N scenario is the worst one.

Looking at other scenarios, there are some results that are not really intuitive. The two best performing scenarios are obtained by taking images for the regions AB an E.

AB can be interpreted as adding images with a relatively small variety with respect to the couples already in the train dataset.

On the other hand, the E scenario adds the most diverse images from the train dataset since samples in region E have FM predictions well below 1.

Other aspects of these results need more investigations, for example the low performance of augmentation using samples from region D which are the other most diverse set of samples, this time with prediction above 1.

Also, adding samples taken from the union of BA and E, the EBA scenario, does not match the performance of AB and E scenarios, In this case a possible explanation could be that the presence of very dissimilar samples "cancels" the contribution of very similar ones and vice versa.

6.5 Second Approach: Transfer Learning

There are other possible way to characterize diversity inside a set of data, different from using a factorization machine. In the following approach a methodology named transfer learning associated with cosine similarity is applied.

Transfer Learning (TL) is a machine learning technique where a model developed and trained on some domain (the source domain) for some task (the source task) is somehow reused for another related task on a related domain (the target task ad domain) taking advantage of the knowledge acquired during the learning on the sources domain and task [82].

Intuition can be obtained from human activities like learning a second musical instruments that can be made faster by taking advantage from the experience done on the first one. On the other hand if there is little in common between domains, knowledge transfer could be unsuccessful. For example, learning to ride a bicycle cannot help in learning a musical instrument faster.

It has become an adopted and convenient method in deep learning, enabling the utilization of pre-trained models as a foundation for various tasks, e.g. in computer vision and natural language processing. This approach offers the advantage of leveraging pre-trained models, which are typically trained on extensive datasets using substantial computational resources that may not be accessible to everyone [83].

Based on conditions between the source domain, target domain, and the tasks, Transfer Learning can be categorized as: inductive TL, transductive TL and unsupervised TL [84].

Inductive TL. In this case, the target task differs from the source task, despite the source and target domains are similar.

Transductive TL. Here, both the tasks (source and target) are the identical in this case. However, the domains are distinct.

Unsupervised TL. In this TL scenario, the target and the source task are different but somehow related, similar to the inductive TL. Unsupervised TL, on the other hand, focuses more on completing unsupervised tasks, such as clustering and dimension reduction [85], [86]. In this situation, both the domains, i.e., source and target, have no labeled data.

Another use of a pre-trained model is for extracting the output from some layers, which goes under the name of features extraction. In computer vision this technique can be used to extract representation of geometric features,

statistical features and color features [87].

The idea is that a neural network is able to detect the most important features for solving a problem avoiding the sometimes very expensive activity of feature engineering made by humans. If the extracted features are sufficiently general, they can be reused for other tasks. Sometimes this also reduces the dimension of the dataset.

In our case given a pre-trained ResNet-50 model, for each image in the training set we get the embeddings from the last layer before classification layer and we measure the cosine similarity between the embeddings of the unmodified image and the embeddings of a modified version of that image according to four different geometric modifications. The goal is having a measure of diversity and then using this information to verify which portion of the dataset is better to use for improving the performance through augmentation.

The transformations applied are:

- 45 degree rotation.
- Flipping left - right.
- Flipping up - down.
- Addition of gaussian noise to the image.

A threshold is introduced so that for each experiment only images above or below that similarity threshold are added to the training set. As said above, the threshold acts as a measure of the diversity we are introducing in the training dataset, and allows to observe the impact the diversity has on the training of the neural network. Introducing diversity can be beneficial for the generalization capabilities of a model but too much diversity in a training set can result in samples too difficult to be learned and damage the performance [11]. At each cosine similarity threshold, a maximum of 1500 images is added to the training set.

6.6 Experiments

For the experiments, four different strategies are taken are into consideration regarding the training dataset.

- Not augmented training dataset.
- Training dataset augmented with random images.

- Training dataset augmented with images having cosine similarity with the original image above a given threshold (H strategy).

- Training dataset augmented with images having cosine similarity with the original image below a given threshold (L strategy).

Algorithm 5 formalizes how transformations, similarities and a given strategy are applied.

Algorithm 5 Steps for cosine similarity based augmentation

1: For each of the above transformations and each image:
2: Obtain the embeddings from the last layer before classification of the unmodified image and of the transformed image;
3: Compute the cosine similarity between the obtained embeddings;
4: Add the transformed image if the strategy under testing is satisfied.

Two different models have been used. A custom CNN network and a pre-trained ResNet-50 obtained from torchvision [88] a package in the PyTorch library [89] specifically designed for computer vision tasks.

The custom CNN model has been fully trained on a single random split of the dataset and then for the subsequent experiments, in order to speed up the training, only the last layer has been substitued with a new layer with trainable weights while all the other weights were frozen.

The ResNet (Residual Network) is a type of convolutional neural network (CNN) used in computer vision applications introduced in 2015 by He Kaiming et al [90].

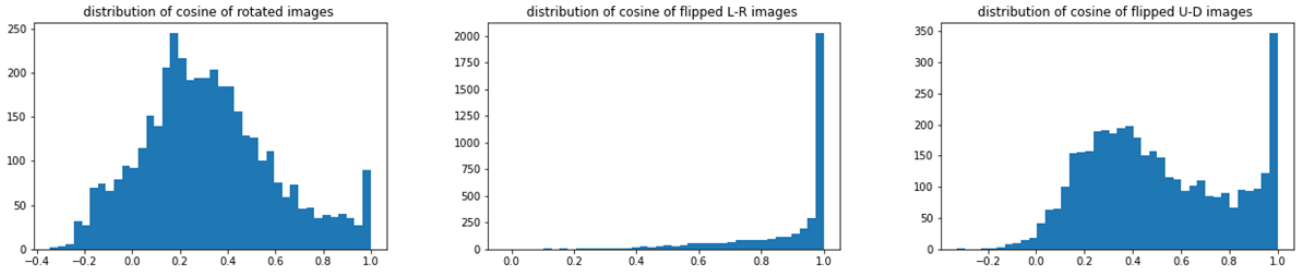
Among the set of ResNet networks, ResNet-50 is a 50-layer convolutional neural network with 48 convolutional layers, one MaxPool layer, and one average pool layer. The network is trained on the ImageNet dataset [91] having 1.28 million images in 1000 classes.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 6.20: ResNet architectures for ImageNet [90]

For the ResNet-50 experiments on the obtained, we adopt a transfer learning approach since we modifiy original model substituing thel last layer with a linear layer of suitable dimension for the 23 class classification task at hand. The only trainable weights are the weights of the last linear layer.

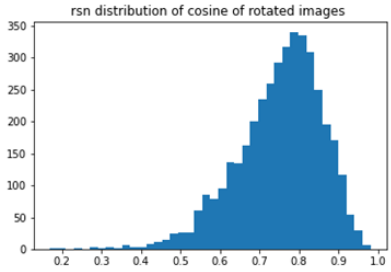
Figure 6.21 shows the distributions of cosine similarity between the unmodified images and the modification applied to the custom CNN.



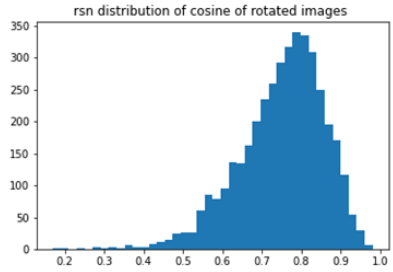
(a) Rotation (b) Flip left rignt (c) Flip up down

Figure 6.21: Distributions of cosine similarity between unmodified and modified images of the train dataset for each transformation. The embeddings for the unmodified image and the modified one are obtained from the last layer before the classification of a custom CNN. Quite a number of images after flipping and noise addition keep a very high similarity with the corresponding unmodified image, probabiy due to the simmetry.

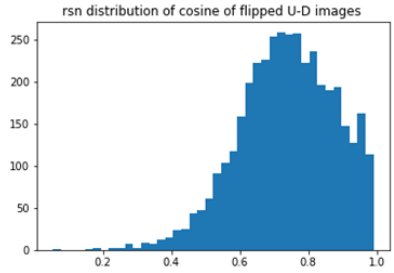
Figure 6.22 shows the distribution of cosine similarity between the unmodified images and the modification obtained from the ResNet-50 model.



(a) Rotation



(b) Flip left rignt



(c) Flip up down

Figure 6.22: Distributions of cosine similarity between unmodified and modified images of the train dataset for each transformation. The embeddings for the unmodified image and the modified one are obtained from the last layer before the classification of a ResNet-50 model. In this case the distributions are skewed toward 1 but the maximum is not at the maximal similarity, the noise generates a quite divese set of images having similarity more distant from 1 than the other transformations.

6.6.1 Results and Comments

Table 6.1 lists the four strategies applied to verify the impact of the selection of the images based on the cosine similarity.

Strategies	Descrition
N	Not augmented
R	Randomly augmented
H	Augmented with images with cosine similarity above threshold
L	Augmented with images with cosine similarity below threshold

Table 6.1: Augmentation strategies

Using the ResNet-50 network the experiments were conducted using fifty different splits of the dataset and for 30 epochs. The average results are reported in Figures 6.23 and 6.24. 1500 images were added.

The results are represented under two different points of view. In Figure 6.23 the average accuracies obtained are represented. In Figure 6.24 the same results are presented as rankings so lower values are better.

Observation:

- In both cases at lower values of the threshold the L strategies seems the best one while at the highest threshold tested the H strategy prevails.

in fact we observe that giving very similar or very different images improves the performance.

Threshold	N	R	H	L
0.95	0,8037	0,8026	0,8106	0,7973
0.8	0,8037	0,8026	0,8016	0,7984
0.5	0,8037	0,8026	0,8002	0,8046
0.05	0,8037	0,8026	0,7970	0,8069

Figure 6.23: ResNet average accuracies for the four strategies

Threshold	N	R	H	L
0.95	2,4510	2,4706	2,1569	2,9216
0.8	2,1961	2,3137	2,5098	2,9804
0.5	2,4706	2,5882	2,6667	2,2745
0.05	2,3137	2,4706	3,0588	2,1569

Figure 6.24: ResNet average rankings for the four strategies

For the custom CNN network, which is computationally less expensive, the experiments were conducted using seventy different splits of the dataset for seventy epochs. Also, this allowed us to test the approach on a higher number of thresholds. In all the experiments 1500 images are added.

The average results are reported in Figures 6.25 and 6.26.

In Figure 6.25 the results are represented, also in this case under two different points of view. The average accuracies obtained are represented, despite the very high values and the small difference between the results, the H strategy results to be the best one for the majority of the random splits.

In Figure 6.26 the same results are presented as rankings, so lower values are better., also in this view the prevalence of the H strategy can be observed.

Threshold	N	R	H	L
0.95	0,9547	0,9515	0,9612	0,9465
0.90	0,9547	0,9515	0,9604	0,9470
0.85	0,9547	0,9515	0,9602	0,9469
0.80	0,9547	0,9515	0,9604	0,9472
0.7	0,9547	0,9515	0,9599	0,9468
0.6	0,9547	0,9515	0,9579	0,9484
0.5	0,9547	0,9515	0,9583	0,9468
0.4	0,9547	0,9515	0,9561	0,9479
0.3	0,9547	0,9515	0,9554	0,9473
0.2	0,9547	0,9515	0,9556	0,9494
0.15	0,9547	0,9515	0,9530	0,9475
0.1	0,9547	0,9515	0,9534	0,9465
0.05	0,9547	0,9515	0,9539	0,9475

Figure 6.25: Custom CNN average accuracies for the four strategies.

Threshold	N	R	H	L
0.95	2,3778	2,6889	1,6111	3,3222
0.90	2,3556	2,7333	1,5889	3,3222
0.85	2,3667	2,7778	1,5889	3,2556
0.80	2,3667	2,6889	1,6778	3,2667
0.7	2,3556	2,7111	1,6556	3,2778
0.6	2,3444	2,7778	1,8000	3,0667
0.5	2,2778	2,6556	1,8667	3,2000
0.4	2,1778	2,6333	2,0333	3,1556
0.3	2,2333	2,5556	2,1111	3,1000
0.2	2,2889	2,6444	2,0556	3,0000
0.15	2,1444	2,4556	2,3111	3,0889
0.1	2,1000	2,4333	2,2333	3,2222
0.05	2,1444	2,5222	2,2333	3,1000

Figure 6.26: Custom CNN average rankings for the four strategies.

About the results in Figures 6.25 and 6.26 the following observations can be made.

- The strategy H emerges as the best performing.
- There is no indication about an optimal threshold to be used.
- The computational cost of the result obtained in Figures 6.25 (seventy different splits of the dataset for seventy epochs 15 days of runs) is high.

Following these considerations emerges the question if it is possible to obtain an indication about an optimal threshold using just a subset of the runs.

We applied two strategies using only the runs related to the thresholds 0.05, 0.5, 0.95; polynomial fitting and linear interpolation.

Polynomial fitting of degree 2: in this case, Algorithm 6 has been applied.

Algorithm 6 Polynomial fitting

1: Let t be a threshold with $t \in \{0.05, 0.5, 0.95\}$
2: Let y_t be the accuracy at threshold t
3: Let r_j be the run associated to a dataset split j with $j = 1,2,...,70$
4: **for all** r_j **do** :
5: obtain the polynomial fitting (of degree 2) for the three points $(0.05, y_{0.05}), (0.5, y_{0.5})$ and $(0.95, y_{0.95})$, which is the solution with respect to A, B and C of the system

$$\begin{cases} y_{0.05} &= A * 0.05^2 + B * 0.05 + C \\ y_{0.5} &= A * 0.5^2 + B * 0.5 + C \\ y_{0.95} &= A * 0.95^2 + B * 0.95 + C \end{cases}$$

6: **end for**

At the and of Algorithm 6 we have 70 curves, one for each test.

In Figure 6.27, there is a representation including points calculated on the curves at intermediate points between the chosen thresholds. A coloring has been applied to highlight (in red) areas with higher values.

The average position of the peaks is located at 0.833, which we can interpret as the threshold to aim for to achieve good performance with an H-type strategy.

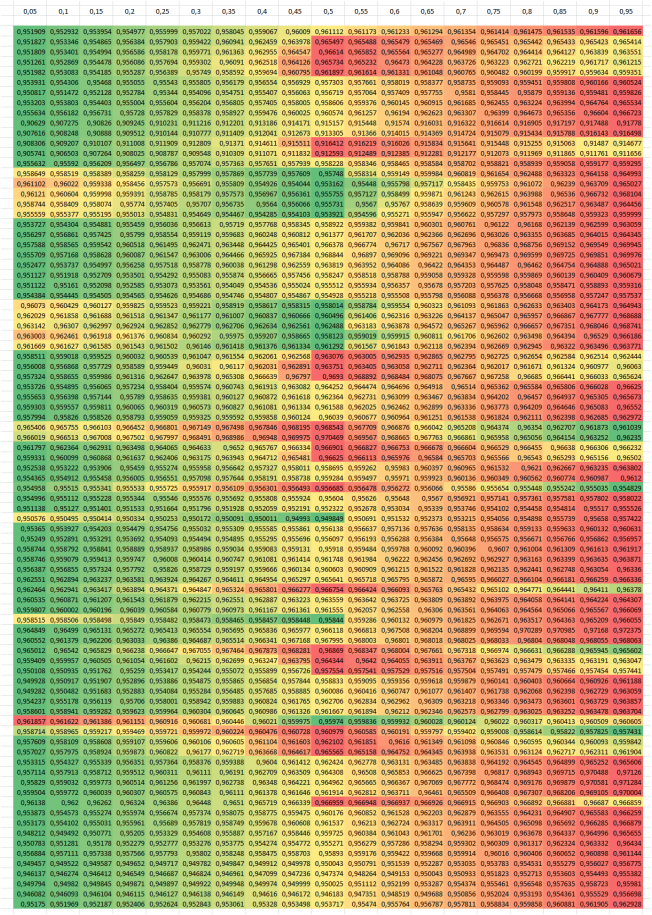


Figure 6.27: Gradient map for three samples quadratic fitting of H strategy results

Linear interpolation: in this case Algorithm 7 is adopted

Algorithm 7 Linear interpolation

1: Let t be a threshold with $t \in \{0.05, 0.5, 0.95\}$
2: Let y_t be the accuracy at threshold t
3: Let r_j be the run associated to a dataset split j with $j = 1,2,...,70$
4: **for all** r_j **do**:
5: obtain the linear interpolation for the segments $[(0.05), y_{0.05}), (0.5, y_{0.5})]$ and $[(0.5, y_{0.5}), (0.95, y_{0.95})]$, that is, respectively:

$$\frac{y - y_{0.05}}{x - 0.05} = \frac{y_{0.5} - y_{0.05}}{0.5 - 0.05}$$

and

$$\frac{y - y_{0.5}}{x - 0.5} = \frac{y_{0.95} - y_{0.5}}{0.95 - 0.5}$$

6: **end for**

After Algorithm 7, we obtain 70 curves.

In Figure 6.28, there is a representation including points computed on the segments at intermediate points between the chosen thresholds. A coloring has been applied to highlight (in red) areas with higher values.

The average position of the peaks is located at 0.827, which we can interpret as the threshold to aim for to achieve good performance with an H-type strategy.

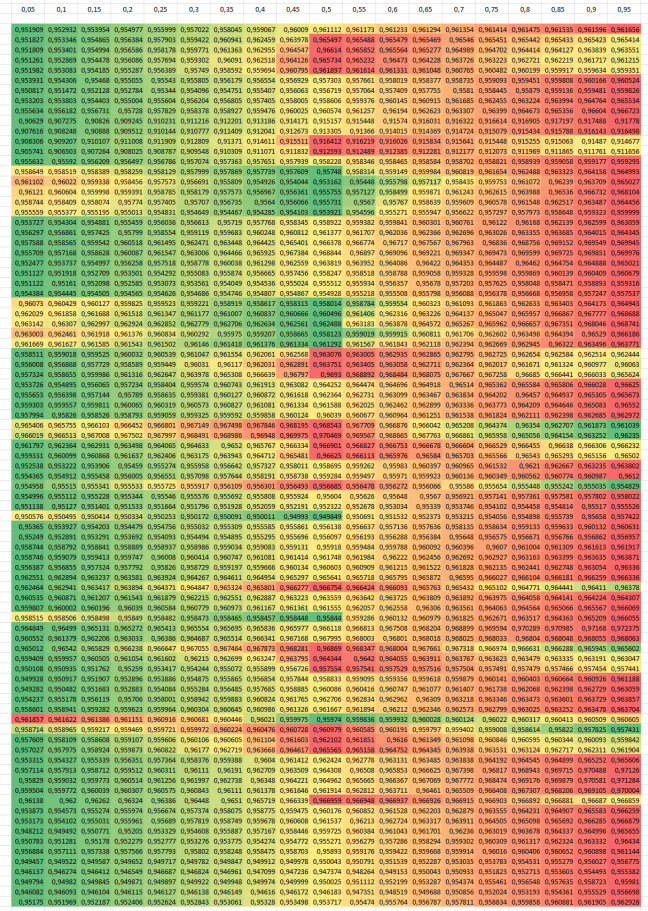


Figure 6.28: Gradient map for three samples interpolation of H strategy results

In summary, our investigation using cosine similarity between embeddings derived from pre-trained models, applied within the Fashion Product Images (Small) dataset, indicates that the H strategy is the best performing one. We utilized two methods to determine the optimal threshold for cosine similarity, converging at approximately 0.83. This suggests that in this scenario, the classification model derives greater benefit from incorporating images that resemble those already present in the training dataset.

6.7 Third Approach: Data Driven Augmentation for SVM

The experience with the experiments described in the previous sections, triggered the interest in experiments conducted in a more controllable environment.

In the following sections, a set of classification experiments executed using different generated datasets are described. As classifier, Support Vector Machines are used, given the possibility of a geometric interpretation of their behavior that even allows (n two and three dimensions) to visualize their decision boundaries.

The strategies applied are still based on a concept of distance but in this case an Euclidean distance is used.

6.7.1 Support Vector Machines

Support Vector Machines (SVM) are supervised learning methods that can be applied to classification and regression problems. SVM were first proposed by V. N. Vapnik and A. Ya. Chervonenkis (Institute of Control Sciences of the Russian Academy of Sciences, Moscow, Russia) in the framework of the "Generalised Portrait Method" for computer learning and pattern recognition. The development of these ideas started in 1962 and they were first published in 1964 [15].

Below is an overview of how SVMs work.

In a binary classification setting, suppose some points are given with their class label, the aim is to predict which class a new, previously unseen, point belongs to.

In SVM a point is an N-dimensional vector and the algorithm searches for a (N-1)-dimensional hyiperplane that a) separates the two classes b) maximizes the distance with the nearest point of each classes. This is called a linear classifier.

The training dataset is composed of n points

$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n),$$

where the y_i are either 1 or -1, each indicating the class to which the point $\mathbf{x}_i$ belongs.

Each $\mathbf{x}_i$ is a p-dimensional real vector. We want to find the "maximum-margin hyperplane" that divides the group of points $\mathbf{x}_i$ for which $y_i = 1$

from the group of points for which $y_i = -1$, which is defined so that the distance between the hyperplane and the nearest point $\mathbf{x}_i$ from either group is maximized.

A hyperplane can be defined as the set of points $\mathbf{x}$ satisfying.

$$\mathbf{w}^\mathsf{T}\mathbf{x} - b = 0, \tag{6.3}$$

where $\mathbf{w}$ is the (not necessarily normalized) normal vector for the hyperplane. The parameter $\frac{b}{\|\mathbf{w}\|}$ is the offset of the hyperplane from the origin along the normal vector $\mathbf{w}$.

In almost all real-world applications, however, usually data are not linearly separable, see Figure 6.29, but it is possible to adapt SVMs in order to deal with this situation with the introduction of *soft margins* and the so-called *kernel trick*.

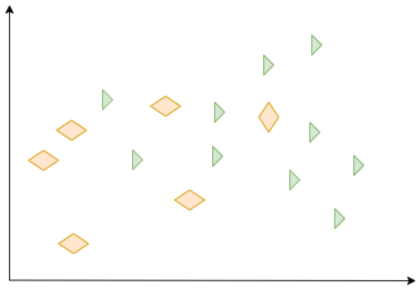


Figure 6.29: Data not linearly separable

Soft margins are related to the idea of allowing a certain number of mistakes while keeping the margin as wide as possible so that other points can still be classified correctly. This can be obtained by modifying the objective function of SVM.

Notice that this may be desirable even in (rare) cases where the data is linearly separable. A decision boundary too narrow that perfectly separates the data can be a form of overfitting.

As an example, looking at Figure 6.30, the green boundary somehow seems more correct in separating the two classes, with respect to the red boundary that is influenced by what can be considered as an outlier.

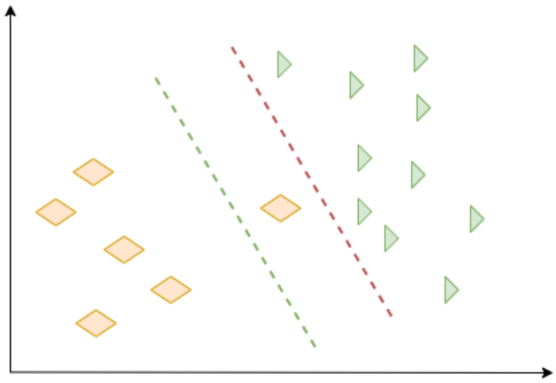


Figure 6.30: Two possible decision boundaries, the red dotted line is influenced by the "outlier"

Mathematically a fist step is to modifiy the objective function adding a term related to errors

$$\mathbf{L} = \frac{1}{2}||w||^2 + C \text{ (number of errors)} \tag{6.4}$$

In equation (6.4) C is a hyperparameter that decides the trade-off between maximizing the margin and minimizing the mistakes. A small C implies that classification mistakes are given less importance and the focus is more on maximizing the margin, whereas a large C implies that the focus is more on avoiding misclassification at the expense of keeping the margin small.

However, it is useful to take into account that there are small errors and large errors in terms of distance from the decision boundary. Data points that are far away on the wrong side of the decision boundary should incur more penalty as compared to the ones that are closer.

For every data point x_i, a so called "slack variable" ξ_i is introduced. The value of ξ_i is the distance of x_i from the corresponding class's margin if x_i is on the wrong side of the margin, otherwise zero. Thus, the points that are more distant from the margin on the wrong side would get more penalty.

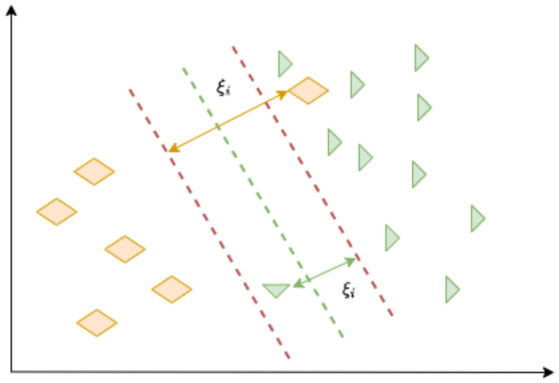


Figure 6.31: Soft margin allows for misclassified points with a penalty measured by the slack variable ξ_i for being in the wrong side of the decision boundary

Each data point x_i needs to satisfy the following constraint:

$$y_i(\mathbf{w} \cdot \mathbf{x_i} + b) \geq 1 - \xi_i \tag{6.5}$$

The left-hand side of the inequality could be considered as the confidence of classification. Confidence ≥ 1 suggests that the classifier has classified the point correctly. If confidence < 1, it means that the classifier did not classify the point correctly and a linear penalty of ξ_i is assigned.

Given these constraints, the objective function to minimize is:

$$\mathbf{L} = \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i + \sum_i \lambda_i(y_i(\mathbf{w} \cdot \mathbf{x_i} + b) - 1 + \xi_i) \tag{6.6}$$

where we have used the concepts of Lagrange Multiplier for optimizing loss function under constraints.

Comparing it with the objective function that deals with linearly separable data, it is possible to notice that the only difference is the presence of the ξ_i slack variables.

$$\mathbf{L} = \frac{1}{2}||\mathbf{w}||^2 + \sum_i \lambda_i(y_i(\mathbf{w} \cdot \mathbf{x_i} + b) - 1 + \xi_i) \tag{6.7}$$

The Kernel Trick is an approach that addresses the problem of non-linearly separable data using the so-called kernel functions.

Kernel functions are generalized functions that take in inputs two vectors (of any dimension) and output a score that denotes how similar the input vectors are. A first example of Kernel function is the dot product function: a

small dot product implies that vectors are different while, if the dot product is large, the vectors are more similar.

Differentiating the equation 6.7 with respect to $\mathbf{w}$ and b and equating it to 0 the optimal value of $\mathbf{w}$ and b are obtained, substituting them in 6.7 we obtain

$$\mathbf{L} = \sum_i \lambda_i - \frac{1}{2} \sum_i \sum_j \lambda_i \lambda_j y_i y_j \mathbf{x}_i \mathbf{x}_j \tag{6.8}$$

So the objective function depends on the dot product $\mathbf{x}_i \mathbf{x}_j$.

It is worth noting that, depending on the problem at hand, it is possible to choose functions different from the dot product the so called "kernel functions".

This freedom of choice of the kernel functions is known as the "Kernel Trick". A generic kernel function can be written as:

$$K(x, y) = \psi(x) \cdot \psi(y) \tag{6.9}$$

The dot product of the vector functions $\psi(x)$ and $\psi(y)$ (so in the case of the dot product $\psi(.)$ is the identity), or, in other terms, the kernel functions are the dot product of transfomeed input vectors.

An example of application of this method can be given considering data that are disposed in a concentric way as in Figure 6.32 with an inner class surrounded by points of an outer class, which is clearly a non linearly separable situation because a circular decision boundary is needed.

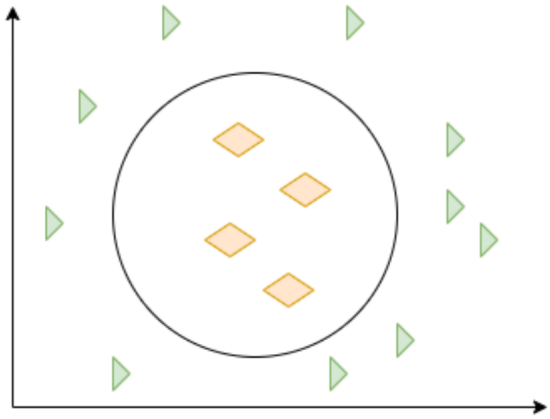


Figure 6.32: Data disposed in a concentric way

In the 2-dimensional setting of Figure 6.32 one could consider a transforming function of the point $P = (x_1, x_2)$ like $\psi(P) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$ so that equation (6.9) becomes

$$K(x, y) = \psi(x_1, x_2) \cdot \psi(x_1', x_2') \tag{6.10}$$

$$K(x, y) = x_1^2 x_1'^2 + x_2^2 x_2'^2 + 2x_1x_2x_1'x_2' \tag{6.11}$$

So the final form of the kernel function is a circle. Now the "similarity function" K' measures whether points are within a circle.

What has been done is the following:

1. Each point P is represented by coordinates (x,y) in 2D space.
2. The points are projected to 3D space by transforming their coordinates to $(x^2, y^2, \sqrt{2}xy)$
3. Points which have high value of x.y would move upwards along the z-axis (in this case, the yellow diamonds).
4. It is possible to find a hyperplane in 3D space that correctly separates the classes.

By embedding the data in a higher-dimensional feature space, we can keep using a linear classifier because the induced transformation moves the problem in a higher dimensional setting where a linear boundary, a plane in this case, can still be found (see Figure 6.33).

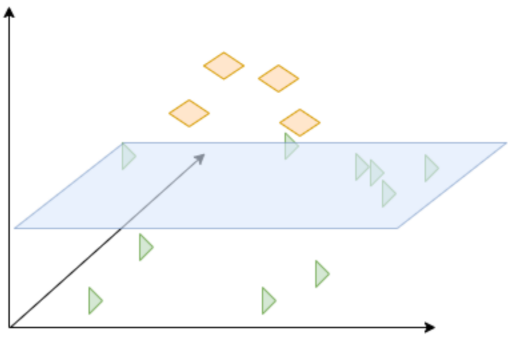


Figure 6.33: Transformation induced by the kernel function

SVM methods can also be extended to multiclass problems employing the "one-vs-one" reduction method. That is, for K classes we train $\binom{K}{2}$ binary classifiers. Each classifier will be trained on samples from a pair of classes and learn to distinguish between them. In an image classification

setting for example, when making a prediction on an unlabeled image, each of the $K(K-1)/2$ classifiers will be given the image, and then "vote" on which class they think the image belongs to, i.e. the class that gets the highest number of positive predictions among all classifiers' predictions will be taken as the "winner".

In the experiments described below, a linear SVM classifier is used in a multiclass setting.

6.7.2 Data Augmentation Strategies and SVM

Machine learning models are usually trained on specific data but, after the training, they are exposed to data that can be quite diverse from the train data. The experiments described in the following aim to test the behavior of a SVM classifier model in presence of a train dataset that is made quite diverse, through elimination of a percentage of samples, from the test dataset. As written earlier, SVM were chosen because of the geometric nature of the decision boundaries that facilitate intuition and can even be plotted in 2-dimensional and 3-dimensional settings.

In the following experiments the coordinates of the points of the training dataset are perturbed with gaussian noise, then a portion of the perturbed points is added to the training set following the strategies listed below:

- Random Augmentation (RN): a portion of the modified data is randomly added to the training set
- Near Augmentation (NR): of all the modified points, a part of the points that are closest to the respective unperturbed point is added to the training set.
- Far Augmentation (FR): of all the modified points, a portion of the points that are further away from the respective unperturbed point is added to the training set.

In addition to the listed strategies a No Augmentation strategy (NA) is always performed.

In Figure 6.34 the strategies listed above are depicted.

It is worth noting that the proposed augmentation strategies only use the notion of distance from the original unmodified points but do not consider the direction of the expansion, so not all the added points of a Far strategy (FR) augment the radius of the training set and not all the points of a Near strategy (NR) augment the training set "density". Nevertheless the

FR strategy should augment the diversity of the train dataset while the NR strategy should add "more of the same" to the train dataset reinforcing the "identity" of the classes.

So none of the listed strategies make use of the notion of centroids. This is because in real data it may not be immediately available.

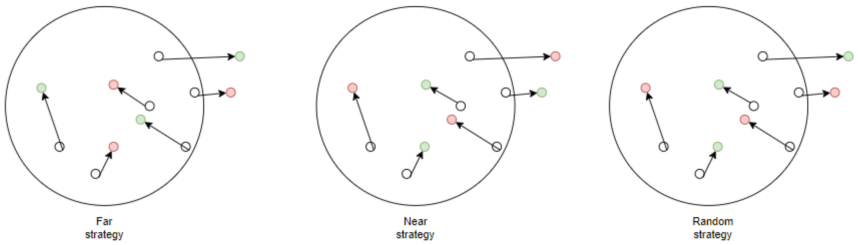


Figure 6.34: Far (FR), Near (NR) and Random (RN) strategies applied to the train dataset (white circle). The green point are added to the train dataset.

The number of points added is the same for all the strategies. The results of these strategies are compared with the case in which no new points are added to the training set.

The generated datasets are characterized using the Fisher ratio defined for two classes a and b as:

$$f = \frac{(\mu_a - \mu_b)^2}{\sigma_a^2 + \sigma_b^2} \tag{6.12}$$

where μ_i and σ_i^2 are the mean and variance of class i.

In our case where there are n classes the average of the Fisher ratios between each pair of classes is taken.

$$f_{avg} = \frac{1}{n(n-1)/2} \sum_{i}^{n(n-1)/2} f_i \tag{6.13}$$

Higher f_{avg} implies more separate classes, and we expect augmentation strategies to be less effective, but there are conditions where one can expect that a particular strategy is better than the others.

6.7.2.1 Same Noise for All Classes on a 2-dimensional Dataset

The first experiment was conducted in a 2-dimensional environment.

Aim of the experiment:

- Evaluate the impact on accuracy and decision boundary geometry resulting from the NA, RN, NR, FR strategies outlined in 6.7.2 following augmentation using Gaussian noise with a constant standard deviation.

The parameters used are:

- 4 classes.
- 1000 points.
- noise applied: gaussion with mean 0 and standard deviation 0.5.

For the setting of this experiment, four points are randomly chosen in a 2-dimensional square box of edge 10 with the constraint of having a reciprocal distance greater or equal than 5.

These points are used as the centroids of four classes generated with a gaussian distribution having standard deviations of 0.5, 1, 1.5 and 2, the final result is a dataset of 1000 points that is then splitted in train and test (with proportion 85% and 15%) as shown in Figure 6.35

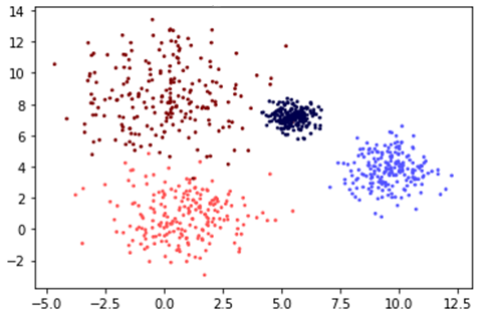


Figure 6.35: Train dataset: 4 classes, 1000 points

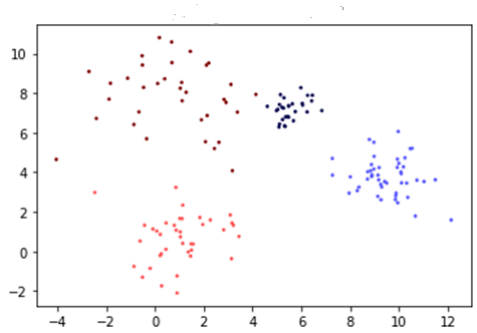


Figure 6.36: Test dataset: 4 classes, 1000 points

This leads to a Fisher ratio of 18.728

To test the strategies listed above, the train dataset is further restricted, keeping, in this 2-dimensional setting, only the points at a distance lower than a standard deviation as in Figure 6.38.

An example of the effects of the augmentations is shown in Figures 6.39 6.40 6.41 along with the boundaries found by the SVM classifier.

The accuracies obtained, with their standard deviation, are reported in the following Figure 6.37.

	1x					5x			
	NA	RN	NR	FR		NA	RN	NR	FR
avg	0,9856	0,9820	0,9847	0,9802	avg	0,9856	0,9824	0,9840	0,9833
stdev	0,0094	0,0111	0,0099	0,0127	stdev	0,0094	0,0118	0,0100	0,0105
	10x					15x			
	NA	RN	NR	FR		NA	RN	NR	FR
avg	0,9856	0,9818	0,9840	0,9816	avg	0,9856	0,9824	0,9831	0,9816
stdev	0,0094	0,0105	0,0100	0,0110	stdev	0,0094	0,0127	0,0106	0,0129
	20x								
	NA	RN	NR	FR					
avg	0,9856	0,9822	0,9838	0,9824					
stdev	0,0094	0,0110	0,0094	0,0103					

Figure 6.37: Accuracies (the higher the better) for augmentation done with fixed noise for all the classes, the notation 1x, 5x, ... represents how many times the augmentation process has been cumulatively repeated, green values are the best ones

Observation:

- It seems that there is not preferred strategy. Indeed, not augmenting in this case seems the best strategy. A possible reason can be found by looking at the boundaries found by the classifier in Figure 6.38 where the boundaries found for the augmentation done with ten iterations are depicted. As one can observe, the boundaries essentially do not move from the ones found in the not augmented case. This is probably due to the type of noise applied that is equal for all the classes.

The above observation have led to the experiment described in the next paragraph

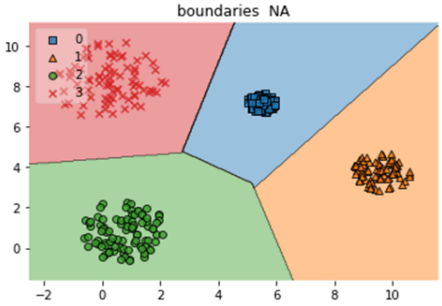


Figure 6.38: Train dataset: 4 classes, narrowed to points at less than a standard deviation from the centroid of the class, with the boundaries found by a classification performed with an SVM.

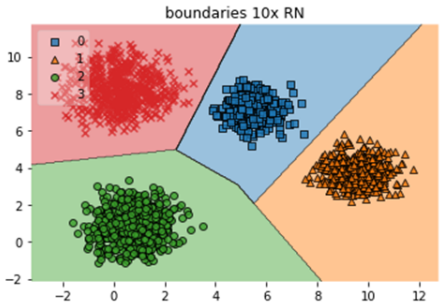


Figure 6.39: Train dataset after the RN strategy has been applied 10 times, the boundaries are essentially unchanged with respect to Figure 6.38.

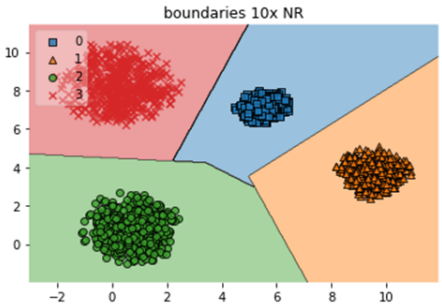


Figure 6.40: Train dataset after the NR strategy has been applied 10 times, the boundaries are essentially unchanged with respect to Figure 6.38.

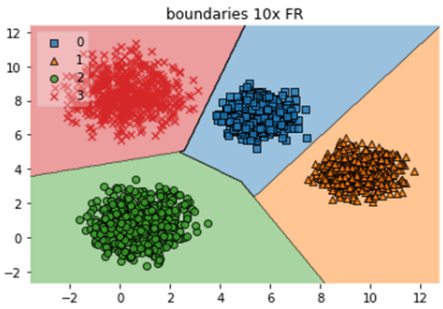


Figure 6.41: Train dataset after the FR strategy has been applied 10 times, the boundaries are essentially unchanged with respect to Figure 6.38.

6.7.2.2 Per Class Noise on a 2-dimensional Dataset

Aim of the experiment:

- Test the effect on accuracy and decision boundaries geometry of the strategies NA, RN, NR, FR described in 6.7.2 after an augmentation done using a gaussian noise related to the standard deviation of the classes. The expectation is to observe an impact on the geometry of the decision boundaries.

The parameters used are:

- 4 classes.
- 1000 points.
- noise applied: gaussion with mean 0 and standard deviation 75% of the class standard deviation.

The settings of this experiment are identical to the previous one. In this case however, the noise is applied based on the class, and has a standard deviation equal to 75% of the standard deviation of class.

The results are visible in figure 6.42

	1X					5X			
	acc_NA	acc_RN	acc_NR	acc_FR		acc_NA	acc_RN	acc_NR	acc_FR
avg	0,9856	0,9833	0,9856	0,9856	avg	0,9856	0,9882	0,9876	0,9891
stdev	0,0094	0,0100	0,0096	0,0107	stdev	0,0094	0,0095	0,0095	0,0081

	10X					15X			
	acc_NA	acc_RN	acc_NR	acc_FR		acc_NA	acc_RN	acc_NR	acc_FR
avg	0,9856	0,9909	0,9902	0,9911	avg	0,9856	0,9891	0,9900	0,9898
stdev	0,0094	0,0085	0,0083	0,0071	stdev	0,0094	0,0079	0,0082	0,0082

	20X			
	acc_NA	acc_RN	acc_NR	acc_FR
avg	0,9856	0,9900	0,9878	0,9907
stdev	0,0094	0,0084	0,0093	0,0073

Figure 6.42: Results with per class noise

Observation:

- In this case, the FR strategy starts to emerge as an interesting one. Looking at the boundaries, it is clear that this scenario implies a movement of the boundaries that can imply the ability of the model to capture more diversity in the data.

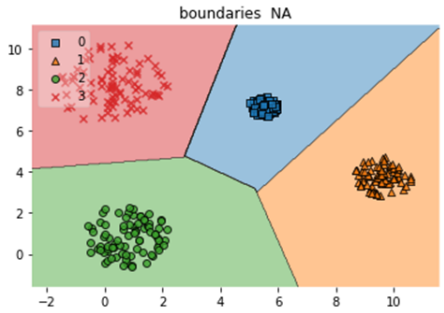


Figure 6.43: Filtered train dataset no augmentation: 4 classes, 1000 points

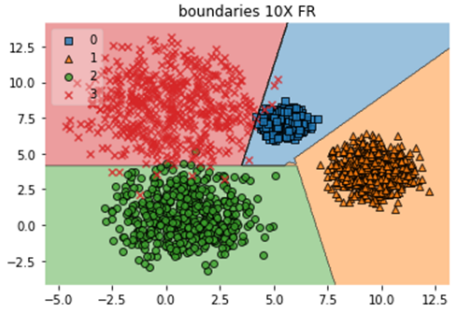


Figure 6.44: Filtered train dataset after 10 times FR augmentation, it is visible the movement of the boundaries

6.7.2.3 Per Class Noise on a 2-dimensional Dataset with 5 Classes

Aim of the experiment:

- Explore the scalability of the approach on a slightly more complex dataset having 5 classes.
- The expectation is to observe if the FR strategy still emerges as interesting.

The parameters used are:

- 45 classes.
- 1200 points.
- noise applied: gaussion with mean 0 and standard deviation 75% of the class standard deviation.

Five points are randomly chosen in a 2-dimensional square box of edge 10 with the constraint of having a reciprocal distance greater than 4.

This leads to a Fisher ratio of 14.233

Around these centroids, five classes are generated having respectively standard deviations 0.5, 1, 1.5, 2 , 2.5.

The final result is a dataset of 1200 points that is then split into train and test (with proportions of 85% and 15%) as shown if figure 6.46.

The results are in Figure 6.45.

	1X					5X			
	acc_NA	**acc_RN**	**acc_NR**	**acc_FR**		**acc_NA**	**acc_RN**	**acc_NR**	**acc_FR**
avg	0,9239	0,9193	0,9219	0,9224	avg	0,9239	0,9246	0,9207	0,9274
stdev	0,0191	0,0180	0,0158	0,0194	stdev	0,0191	0,0183	0,0164	0,0218
	10X					15X			
	acc_NA	**acc_RN**	**acc_NR**	**acc_FR**		**acc_NA**	**acc_RN**	**acc_NR**	**acc_FR**
avg	0,9239	0,9257	0,9228	0,9270	avg	0,9239	0,9246	0,9217	0,9274
stdev	0,0191	0,0208	0,0176	0,0204	stdev	0,0191	0,0200	0,0181	0,0201
	20X								
	acc_NA	**acc_RN**	**acc_NR**	**acc_FR**					
avg	0,9239	0,9274	0,9219	0,9274					
stdev	0,0191	0,0199	0,0191	0,0222					

Figure 6.45: Train dataset: 5 classes, 1200 points, , the annotation 1x, 5x, ... represents how many times the augmentation process has been repeated, green values are the best one

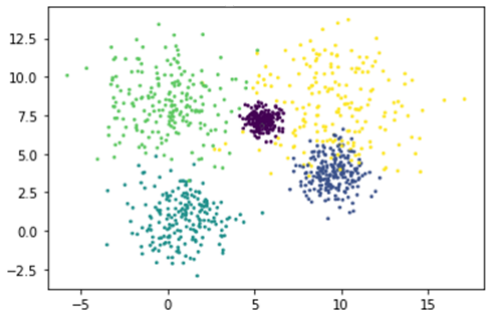


Figure 6.46: Train dataset: 5 classes, 1200 points

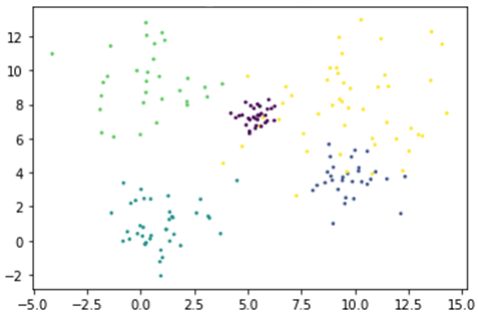


Figure 6.47: Test dataset: 5 classes, 1200 points

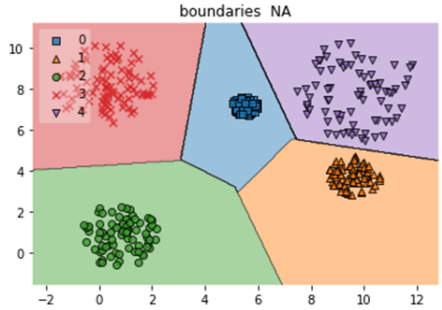


Figure 6.48: Filtered train dataset no augmentation: 5 classes, 1200 points

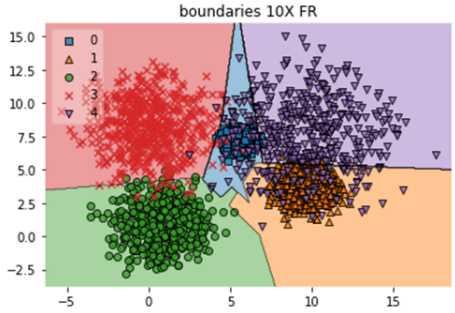


Figure 6.49: Filtered train dataset after 10 times FR augmentation: 5 classes, 1200 points

Observation:

- This scenario favors the FR strategy even more than the previous one and looking at the plots, the movement of the decision boundaries is even clearer than before.

However, usually classification tasks are conducted in spaces with high dimensionality, so there is the need for experiments in higher dimensions, from 3D and above.

6.7.2.4 Experiment on a 3-dimensional Dataset

Aim of the experiment:

- Evaluate, with a 3-dimensional dataset, the effect on accuracy and decision boundaries geometry of the NA, RN, NR, FR strategies described in 6.7.2 after an augmentation done using a gaussian noise related to the standard deviation of the classes. Test if it is possible to find a setting where the results of the previous experiments are observed.

Paramenters used:

- 5 classes.
- 1200 points.
- noise applied: gaussion with mean 0 and standard deviation 75% of the class standard deviation.

The Fisher ratio value in this case is 22.556 The procedure for creating the 3-D dataset is essentially the same as in the previous experiments where a dataset is generated around centroids and the training set is restricted to reduce the diversity, while the test set is kept unmodified. The following figures are a visualization of the scenario.

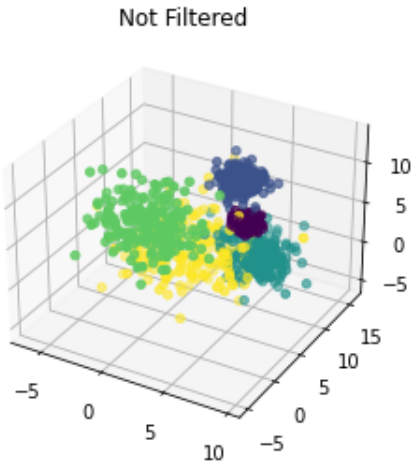


Figure 6.50: 3D Train dataset: 5 classes, 1200 points

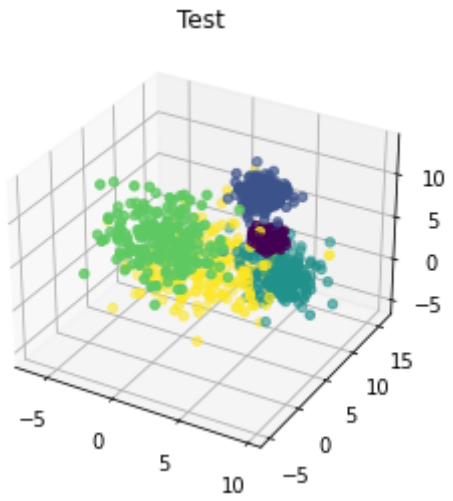


Figure 6.51: 3D Test dataset: 5 classes, 1200 points

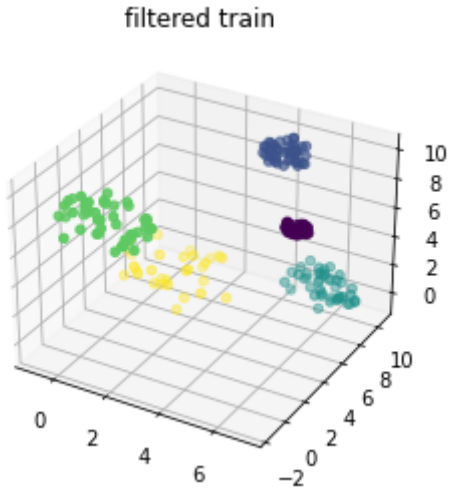


Figure 6.52: 3D Filtered train dataset no augmentation: 5 classes, 1200 points

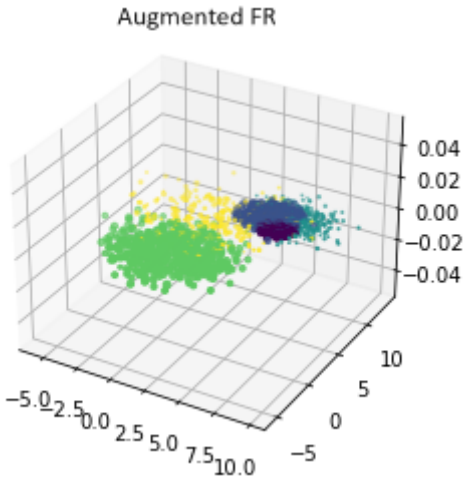


Figure 6.53: 3D train dataset after FR augmentation strategy has been applied 10 times

1X

	acc_NA	acc_RN	acc_NR	acc_FR
avg	0,9530	0,9457	0,9454	0,9452
stdev	0,0206	0,0210	0,0217	0,0183

5X

	acc_NA	acc_RN	acc_NR	acc_FR
avg	0,9530	0,9513	0,9524	0,9544
stdev	0,0118	0,0039	0,0236	0,0157

10X

	acc_NA	acc_RN	acc_NR	acc_FR
avg	0,9530	0,9524	0,9519	0,9581
stdev	0,0118	0,0236	0,0275	0,0118

15X

	acc_NA	acc_RN	acc_NR	acc_FR
avg	0,9530	0,9556	0,9531	0,9596
stdev	0,0118	0,0196	0,0079	0,0079

20X

	acc_NA	acc_RN	acc_NR	acc_FR
avg	0,9530	0,9576	0,9546	0,9593
stdev	0,0118	0,0189	0,0118	0,0039

Figure 6.54: 3D accuracies at various iteration of augmentation, , the annotation 1x, 5x, ... represents how many times the augmentation process has been repeated, green values are the best one

Observation:

- Looking at Figure 6.54 it seems that also in this configuration the best option is the FR strategy as a way to capture the diversity not present in the train dataset.

Exploring this approach in higher dimension brings an issue regarding the criterion for restricting the dataset. Taking the samples inside a standard deviation from the centroids is no more feasible because the probability of having a point in that region drops quite sharply as the number of dimensions increases.

In a 10 dimensional space, the probability of finding a point at less than one standard deviation is 0.0002.

So, instead of taking the points at less than one standard deviation the dataset is restricted to the 50% points nearest to the centroid of their class. Obviously it is no more possible to plot the dataset and the decision boundaries.

In the following sections are described the experiments conducted with a number of dimensions greater than 3.

6.7.2.5 Experiment on a 5-dimensional Space

Aim of the experiment:

- Test in a 5-dimensional space the effect on accuracy of the NA, RN, NR, FR strategies described in 6.7.2 after an augmentation done using a gaussian noise related to the standard deviation of the classes.

The parameters used are:

- 5 classes.
- 3000 points.
- noise applied: gaussion with mean 0 and standard deviation 75% of the class standard deviation.

5 classes with standard deviation (0.5, 1, 1.5, 2, 2.5) are generated around 5 centroids.

The Fisher ratio value is 13.591

A graphic representation is no more possible but also in this case it seem possible to find a configuration that favors a strategy (FR in this case) as shown in the accuracies reported below at variuous level of iteration over the augmentation

1x

	acc_NA	acc_RN	acc_NR	acc_FR
avg	0,9410	0,9442	0,9424	0,9459
stdev	0,0089	0,0088	0,0080	0,0102

5X

	acc_NA	acc_RN	acc_NR	acc_FR
avg	0,9410	0,9475	0,9463	0,9488
stdev	0,0089	0,0087	0,0086	0,0083

10X

	acc_NA	acc_RN	acc_NR	acc_FR
avg	0,9410	0,9491	0,9479	0,9499
stdev	0,0089	0,0074	0,0076	0,0089

15X

	acc_NA	acc_RN	acc_NR	acc_FR
avg	0,9410	0,9499	0,9463	0,9496
stdev	0,0089	0,0071	0,0075	0,0083

20X

	acc_NA	acc_RN	acc_NR	acc_FR
avg	0,9410	0,9496	0,9469	0,9500
stdev	0,0089	0,0071	0,0071	0,0079

Figure 6.55: 5-dimensional dataset accuracies, the notation 1x, 5x, ... represents how many times the augmentation process has been repeated, green values are the best one

Observation:

- Also in this case, there is a strategy (FR) that emerges as the best one.

Real datasets often have embeddings having thousands of dimensions so it is insteresting observe what happens in such a setting.

6.7.2.6 Experiment on a 1000-dimensional Dataset

Aim of the experiment:

- Test the effect on accuracy of the NA, RN, NR, FR strategies described in 6.7.2 after an augmentation done using a gaussian noise related to the standard deviation of the classes. in a setting with 1000 dimension.

The parameters used are:

- 15 classes.
- 6000 points.
- noise applied: gaussion with mean 0 and standard deviation 75% of the class standard deviation.

The classes are created around centroids with respective standard deviations (0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4,5, 5, 5.5, 6, 6.5, 7, 7.5)

Fisher Ratio value is 35.438

1X

	acc_NA	acc_RN	acc_NR	acc_FR
avg	0,8122	0,8092	0,8088	0,8084
stdev	0,0120	0,0139	0,0144	0,0122

5X

	acc_NA	acc_RN	acc_NR	acc_FR
avg	0,8122	0,8057	0,8057	0,8060
stdev	0,0120	0,0119	0,0126	0,0119

10X

	acc_NA	acc_RN	acc_NR	acc_FR
avg	0,8122	0,8099	0,8127	0,8074
stdev	0,0120	0,0139	0,0137	0,0143

15X

	acc_NA	acc_RN	acc_NR	acc_FR
avg	0,8122	0,8158	0,8140	0,8172
stdev	0,0120	0,0133	0,0132	0,0134

20X 30 runs

	acc_NA	acc_RN	acc_NR	acc_FR
avg	0,8122	0,8178	0,8202	0,8194
stdev	0,0120	0,0154	0,0146	0,0127

Figure 6.56: 1000-dimensional dataset, the notation 1x, 5x, ... represents how many times the augmentation process has been cumulatively repeated, green values are the best one

Observation:

- In this case, it seems that no strategy emerges as preferred. Having 15 classes with centroids in a hyperellipse of radius 3 and standard deviation with a progression with a 0.5 gap (0.5, 1, 1.5, ...) results in a dataset that is not so superimposed, the Fisher ratio is the highest computed among the experiments.

6.7.2.7 Experiment on a 1000-dimensional Dataset with a More Superimposed Dataset

Aim of the experiment:

- Test the effect on accuracy of the NA, RN, NR, FR strategies described in 6.7.2 after an augmentation done using a gaussian noise related to the standard deviation of the classes. in a setting with 1000 dimension.

The parameters used are:

- 15 classes.
- 6000 points.
- noise applied: gaussion with mean 0 and standard deviation 75% of the class standard deviation.

In this case the dataset the standard deviation assigned are bigger (0.5, 2, 3.5, 5, 6,5, 8, 9.5, 11, 12.5, 14, 15.5, 17, 18.5, 20, 21.5) so that there is quite a strong superimposition between the classes.

The Fisher ratio value is 6,081

1X

	acc_NA	acc_RN	acc_NR	acc_FR
avg	0,3987	0,3970	0,3983	0,3957
stdev	0,0146	0,0153	0,0142	0,0127

5X

	acc_NA	acc_RN	acc_NR	acc_FR
avg	0,3987	0,3967	0,3963	0,3943
stdev	0,0146	0,0147	0,0150	0,0165

10X

	acc_NA	acc_RN	acc_NR	acc_FR
avg	0,3987	0,3984	0,3994	0,4002
stdev	0,0146	0,0142	0,0153	0,0144

15x

	acc_NA	acc_RN	acc_NR	acc_FR
avg	0,3987	0,4021	0,4021	0,4019
stdev	0,0146	0,0114	0,0144	0,0139

20x

	acc_NA	acc_RN	acc_NR	acc_FR
avg	0,3987	0,4081	0,4089	0,4108
stdev	0,0146	0,0151	0,0142	0,0129

Figure 6.57: 1000-dimensional dataset, accuracies, the notation 1x, 5x, ...represents how many times the augmentation process has been cumulatively repeated, green values are the best one

Observation:

- For low augmentation iterations (1, 5) it seems that there is no benefit in adding points. Above 15x iteration of augmentation the Near and/or Far strategy seems to be beneficial.

6.7.2.8 Experiment on Oxford Flowers 102 Dataset

It is interesting to experiment this approach on a dataset with real data, not a generated one. The Oxford Flowers 102 dataset [92] is a 102 category dataset, consisting of 102 flower categories. The flowers chosen are flowers commonly occurring in the United Kingdom. Each class consists of between 40 and 258 images. The images have large scale, pose and light variations. In addition, there are categories that have very diverse images within the category and several categories containing similar images so low diversity.

The Fisher ration value for this dataset is 0.580.

A transfer learning approach has been used.

Experiments setting:

- 24 different random split of the dataset
- Dataset composed of embeddding extracted from the last layer of a VGG19 network
- Narrowing down of the train dataset to the 50% points closer to the class centroids.

Observation:

- Looking at Figure 6.58 it is possible to see that the NR strategy appears to be interesting. This can be explained by the fact that the classes of the dataset are quite superimposed, and the NR strategy reinforces the "identity" of the classes.

	1x			
	acc_NA	**acc_RN**	**acc_NR**	**acc_FR**
average	0,766382	0,766127	0,766891	0,766127
stdev	0,014074	0,013858	0,013472	0,013168
	5x			
	acc_NA	**acc_RN**	**acc_NR**	**acc_FR**
average	0,766382	0,76572	0,766586	0,765873
stdev	0,014074	0,01422	0,013695	0,01351

Figure 6.58: Flowers 102 dataset: average accuracies for the four strategies

Experiments were done to see if it is possible to obtain scenarios where the FR strategy could be interesting.

So the train dataset was further narrowed down at 40%, 30% and 25% points closer to the class centroids.

Indeed to avoid the uncertainty on the expansion of the dataset intrinsic in the FR strategy, in this case, a new strategy is adopted that makes use of the notion of class centroid.

Outward expansion strategy. For each one of these restricted dataset an expansion strategy is applied where, for each point in the train dataset a new point is added at X% further distance from the class centroid with X equal to 10%, 20%, 30%, 40%, 50%, 60%.

These actions are repeated on 24 different splits and in Figure 6.59 the average accuracies are reported.

	50% points closer to the centroid						
	acc_NA	**acc_10%**	**acc_20%**	**acc_30%**	**acc_40%**	**acc_50%**	**acc_60%**
average	0,766382	0,765313	0,765466	0,764601	0,763684	0,762258	0,761851
stdev	0,014074	0,013229	0,013754	0,012979	0,013195	0,013751	0,014143
	40% points closer to the centroid						
	acc_NA	**acc_10%**	**acc_20%**	**acc_30%**	**acc_40%**	**acc_50%**	**acc_60%**
average	0,742349	0,741432	0,74179	0,740974	0,740719	0,741228	0,740413
stdev	0,012924	0,014641	0,013997	0,013197	0,012946	0,013169	0,013143
	30% points closer to the centroid						
	acc_NA	**acc_10%**	**acc_20%**	**acc_30%**	**acc_40%**	**acc_50%**	**acc_60%**
average	0,720352	0,721269	0,721167	0,720912	0,720454	0,720403	0,719995
stdev	0,015189	0,014524	0,015057	0,01485	0,015051	0,0151	0,015
	25% points closer to the centroid						
	acc_NA	**acc_10%**	**acc_20%**	**acc_30%**	**acc_40%**	**acc_50%**	**acc_60%**
average	0,702684	0,703346	0,703192	0,704058	0,704108	0,703853	0,703904
stdev	0,012316	0,013649	0,012632	0,013115	0,01336	0,013652	0,013663

Figure 6.59: Flowers 102 Expansion strategy applied on different narrowing of the train dataset

Observation:

- It is possible to observe that applying the expansion strategy at the 50% and 40% narrowings is not beneficial, while at 30% and 25% narrowings it is possible to see improvement. It is also possible to see the presence of a threshold, after which the performance starts to decline, so adding diversity is beneficial but only up to a point. After that point, the augmentation starts to damage the model's performance.

6.7.2.9 Experiment on Oxford Flowers 102 Dataset with a Fully Connected Layer as Classifier

Aim of the experiment:

- Test the behavior of a model made of a single fully connected layer as classifier instead of an SVM with this same expansion strategy on the same dataset composed of embeddings extracted using a VGG19 network and the same processing.

Figure 6.60 shows the overall results with both outward and inward augmentations.

50% points closer to the centroid

	acc_NA	acc_2.5%	acc_5%	acc_7.5%	acc_10%	acc_20%	acc_30%	acc_40%	acc_50%	acc_60%
average	0,768521	0,776157	0,775088	0,773663	0,771168	0,758541	0,744742	0,733948	0,726564	0,718621
stdev	0,013393	0,012295	0,012789	0,011046	0,011688	0,011771	0,011939	0,011928	0,01265	0,011674

40% points closer to the centroid

	acc_NA	acc_2.5%	acc_5%	acc_7.5%	acc_10%	acc_20%	acc_30%	acc_40%	acc_50%	acc_60%
average	0,745048	0,752379	0,751106	0,750393	0,747847	0,737205	0,724527	0,714496	0,70813	0,70258
stdev	0,013569	0,012711	0,014343	0,013446	0,013	0,012799	0,01267	0,014399	0,014777	0,014093

30% points closer to the centroid

	acc_NA	acc_2.5%	acc_5%	acc_7.5%	acc_10%	acc_20%	acc_30%	acc_40%	acc_50%	acc_60%
average	0,726564	0,731654	0,730076	0,728396	0,727988	0,722081	0,710828	0,700849	0,695706	0,690767
stdev	0,01681	0,016469	0,016353	0,015409	0,01583	0,015384	0,014586	0,012275	0,013824	0,014832

25% points closer to the centroid

	acc_NA	acc_2.5%	acc_5%	acc_7.5%	acc_10%	acc_20%	acc_30%	acc_40%	acc_50%	acc_60%
average	0,712664	0,717399	0,714954	0,714598	0,712255	0,706857	0,698406	0,691989	0,686948	0,680939
stdev	0,013978	0,015454	0,016224	0,014252	0,014715	0,01593	0,01525	0,014656	0,015308	0,015386

Figure 6.60: Flowers 102 Expansion strategy accuracies applied on different narrowing of the train dataset with a fully connected layer as classifier.

Observation:

- The behaviour is quite different from the SVM classifier behaviour, augmentation improves the performance but the best results are always obtained with the smallest expansion tested, that is, adding points 2.5% further away from the class centroid, so at much smaller scale.

6.7.2.10 Experiment on Oxford Flowers 102 Dataset with a Fully Connected Layer as classifier with Inward Augmentations

Aim of the experiment:

- Triggered by the results of the previous experiement this experiments explores the behavior of an inward augmentation. For each point in the training dataset a new point is added at percentage distances (2,5%, 5%, 7.5%) toward the centroid of the class of the unmodified point.

	50% points closer to the centroid											
	-7.5%	**-5%**	**-2.5%**	**NA**	**2.5%**	**5%**	**7.5%**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**
average	0,78298	0,782318	0,780587	0,768521	0,776157	0,775088	0,773663	0,771168	0,758541	0,744742	0,733948	0,726564	0,718621
stdev	0,01223	0,012673	0,011786	0,013393	0,012295	0,012789	0,011046	0,011688	0,011771	0,011939	0,011928	0,01265	0,011674

	40% points closer to the centroid											
	-7.5%	**-5%**	**-2.5%**	**NA**	**2.5%**	**5%**	**7.5%**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**
average	0,761392	0,759966	0,757726	0,745048	0,752379	0,751106	0,750393	0,747847	0,737205	0,724527	0,714496	0,70813	0,70258
stdev	0,012252	0,01286	0,01359	0,013569	0,012711	0,014343	0,013446	0,013	0,012799	0,01267	0,014399	0,014777	0,014093

	30% points closer to the centroid											
	-7.5%	**-5%**	**-2.5%**	**NA**	**2.5%**	**5%**	**7.5%**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**
average	0,744028	0,741431	0,737664	0,726564	0,731654	0,730076	0,728396	0,727988	0,722081	0,710828	0,700849	0,695706	0,690767
stdev	0,015177	0,01652	0,015952	0,01681	0,016469	0,016353	0,015409	0,01583	0,015384	0,014586	0,012275	0,013824	0,014832

	25% points closer to the centroid											
	-7.5%	**-5%**	**-2.5%**	**NA**	**2.5%**	**5%**	**7.5%**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**
average	0,727431	0,725393	0,722745	0,712664	0,717399	0,714954	0,714598	0,712255	0,706857	0,698406	0,691989	0,686948	0,680939
stdev	0,01467	0,013875	0,015159	0,013978	0,015454	0,016224	0,014252	0,014715	0,01593	0,01525	0,014656	0,015308	0,015386

Figure 6.61: Flowers 102 Expansion strategy accuracies applied on different narrowing of the train dataset with a fully connected layer as classifier. The inward augmentations (negative percentages) are included.

Observation:

- The results in Figure 6.61 seems to support the idea, already suggested by Figure 6.58, that, with this dataset, "reinforcing" the class identity helps the performance.

Figure 6.62 displays the overall behavior of the strategies applied using a fully connected layer as classifier.

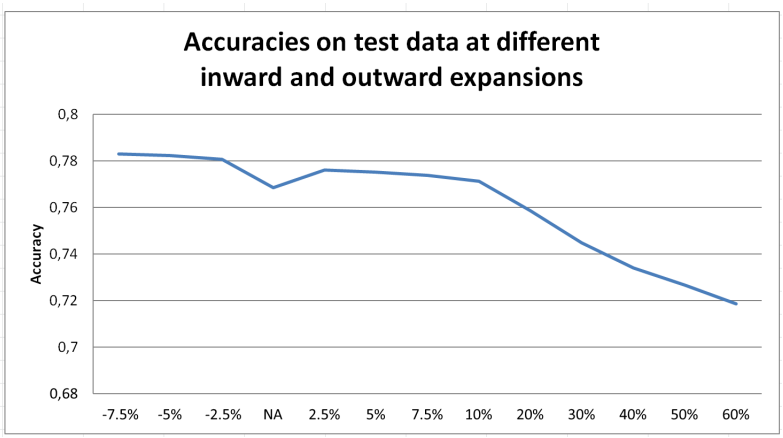


Figure 6.62: Flowers 102 Expansion strategis accuracies applied with augmentation in both the inward and outward directions.

It is possible to notice that there are two regions that improve on the NA (Not augmented) setting, the inward region (represented by the negative percentages) and a segment of the outward region from 2.5% to 10% expansions. The inward augmentation is the one with the best overall performance.

6.7.2.11 Experiment on Oxford Flowers 102 Dataset an SVM with centroids computed from samples

Aim of the experiment:

- Test the behavior of an SVM classifier with expansion strategy on the same dataset composed of embeddings extracted using a VGG19 network in this case the centroids are recomputed aftet the narrowing down of the dataset.

Centroids from samples

50% samples

	acc_NA	acc_10%	acc_20%	acc_30%	acc_40%	acc_50%	acc_60%
average	0,766382	0,768877	0,770914	0,772391	0,774072	0,775141	0,776311455
stdev	0,014074	0,012376	0,013089	0,013095	0,014376	0,014520	0,013461688

40% samples

	acc_NA	acc_10%	acc_20%	acc_30%	acc_40%	acc_50%	acc_60%
average	0,742349	0,744386	0,746117	0,746627	0,747492	0,748102	0,748662043
stdev	0,012924	0,013822	0,013233	0,012924	0,012715	0,012806	0,0130876

30% samples

	acc_NA	acc_10%	acc_20%	acc_30%	acc_40%	acc_50%	acc_60%
average	0,720352	0,723000	0,724375	0,724986	0,725902	0,726462	0,725953667
stdev	0,015189	0,015094	0,016240	0,016403	0,014493	0,014417	0,013626766

25% samples

	acc_NA	acc_10%	acc_20%	acc_30%	acc_40%	acc_50%	acc_60%
average	0,702684	0,702837	0,704110	0,704568	0,705127	0,705840	0,705840668
stdev	0,012316	0,012767	0,012337	0,012079	0,012053	0,011193	0,010656759

Figure 6.63: Flowers 102 Expansion strategy accuracies applied on different narrowing of the train dataset with an SVM as classifier. The inward augmentations (negative percentages) are included. In this case the centroids are recomputed after the narrowing.

Observation:

- We observe that recomputing the centroids brings a higher separation between the classes because all the experiments obtain the best results at the maximum expansion tested, without regard with the initial narrowing.

6.7.2.12 Experiment on Oxford Flowers 102 Dataset a Fully Connected Layer with centroids computed from samples

Aim of the experiment:

- Test the behavior o a Fully Connected Layer classifier with expansion strategy on the same dataset composed of embeddings extracted using a VGG19 network in this case the centroids are recomputed aftet the narrowing down of the dataset.

Centroids from samples

50% samples

	acc_ -7.5%	acc_-5%	acc_ -2.5%	acc_NA	acc_2.5%	acc_5%	acc_7.5%	acc_10%	acc_20%	acc_30%	acc_40%	acc_50%	acc_60%
average	0,776870	0,777787	0,777634	0,768011	0,777837	0,779722	0,779823	0,781249	0,784152	0,787767	0,788939	0,789600	0,787259
stdev	0,013158	0,012758	0,012642	0,012946	0,013378	0,011606	0,012572	0,011659	0,010911	0,011042	0,010327	0,010581	0,009643

40% samples

	acc_ -7.5%	acc_-5%	acc_ -2.5%	acc_NA	acc_2.5%	acc_5%	acc_7.5%	acc_10%	acc_20%	acc_30%	acc_40%	acc_50%	acc_60%
average	0,752532	0,753906	0,753398	0,745150	0,755383	0,756097	0,757217	0,759049	0,764243	0,769590	0,772899	0,773255	0,775547
stdev	0,012738	0,013095	0,013404	0,012660	0,013027	0,012727	0,013152	0,012566	0,013480	0,012785	0,013151	0,013959	0,012556

30% samples

	acc_ -7.5%	acc_-5%	acc_ -2.5%	acc_NA	acc_2.5%	acc_5%	acc_7.5%	acc_10%	acc_20%	acc_30%	acc_40%	acc_50%	acc_60%
average	0,733182	0,733386	0,734506	0,726564	0,735981	0,735524	0,736898	0,737561	0,742195	0,747745	0,749679	0,750749	0,753193
stdev	0,017392	0,016426	0,015382	0,016507	0,016894	0,015799	0,016444	0,014995	0,015248	0,016090	0,015614	0,015219	0,015840

25% samples

	acc_ -7.5%	acc_-5%	acc_ -2.5%	acc_NA	acc_2.5%	acc_5%	acc_7.5%	acc_10%	acc_20%	acc_30%	acc_40%	acc_50%	acc_60%
average	0,718366	0,719079	0,718672	0,713122	0,720811	0,720709	0,720963	0,720912	0,724935	0,728805	0,731757	0,735016	0,736596
stdev	0,014312	0,015269	0,015488	0,013925	0,015625	0,016365	0,014722	0,014725	0,015727	0,015372	0,014964	0,013636	0,015542

Figure 6.64: Flowers 102 Expansion strategy accuracies applied on different narrowing of the train dataset with a Fully Connected Layer as classifier. The inward augmentations (negative percentages) are included. In this case the centroids are recomputed after the narrowing.

Observation:

- We observe that recomputing the centroids brings an higher separation between the classes because all the experiments obtain the best results at the maximum expansion tested, without regard with the initial narrowing.

6.8 Domain Adaptation

The narrowing down of the training data sets applied in the experiments of the previous section can be viewed as a way to simulate a phenomenon called concept shift (or drift). The shift can be due to difference inherent in the training and target domain or can be the consequence of changes in the statistical properties of the training domain itself over time. This means that the relationship between the input features X and the output labels Y changes so that the condition probability $P(Y|X)$ evolves , making the model's initial assumptions and learned patterns potentially outdated or incorrect. In any case the model learned on a domain is confronted with a different domain. This leads to take into consideration the problem of domain adaptation.

Domain adaptation is a subfield of machine learning that focuses on adapting models trained on one domain (the source domain) to perform well on a different but related domain (the target domain). This is especially useful when there is a scarcity of labeled data in the target domain

but an abundance in the source domain [16].

More specifically,

The Domain consists of a feature space and a marginal probability distribution. For example, for a image classification task, images of objects represented with different techniques such as drawings, photos, computer generated images can constitute different domains.

The Source Domain is the domain on which the model is initially trained. It has a large amount of labeled data.

Target Domain is the domain where the model needs to be deployed. It typically has limited or no labeled data.

The differences between the source and target domains is termed as Domain Shift. These differences can be in terms of data distribution, feature space, or both.

There are a few types of Domain Adaptation [16]:

Supervised Domain Adaptation . Involves some labeled data in the target domain, but not as much as in the source domain. The model is fine-tuned using this limited labeled data from the target domain [93].

Unsupervised Domain Adaptation . No labeled data is available in the target domain. Techniques involve learning invariant features that perform well across both domains [94].

Semi-Supervised Domain Adaptation . Involves a combination of labeled and unlabeled data in the target domain, utilizing both to adapt the model [94].

In the following there is a list of some of the techniques developed in the field

- Feature Alignment: Aligning the feature distributions of the source and target domains [95].
- Instance Reweighting: Adjusting the importance of training samples from the source domain to better match the target domain distribution [96].
- Domain Adaptation Networks: Architectures like Domain-Adversarial Neural Networks (DANN) specifically designed for domain adaptation tasks, often incorporating adversarial loss to encourage domain invariance [97].

- Self-Training: Using pseudo-labels for the target domain data, iteratively refining the model as if these pseudo-labels were actual labels [98, 99].

Domain Adaptations technique find application in different fields such as:

- Natural Language Processing : Adapting models trained on one type of text (e.g., news articles) to work on another (e.g., social media posts) [100].
- Computer Vision: Adapting models trained on images from one source (e.g., synthetic images) to work on real-world images [101].
- Speech Recognition: Adapting models trained on one type of speech (e.g., adult speakers) to perform well on another type (e.g., children's speech) [102, 103].

A number of challenges must be addressed such as:

- Data Distribution Differences: Significant differences in data distributions can make adaptation challenging.
- Feature Misalignment: Incorrect alignment of features can lead to poor model performance.
- Limited Target Domain Data: Scarcity of labeled data in the target domain makes it difficult to fine-tune models effectively.

6.8.1 Datasets and Experiments

For a task of image classification we want to observe and analyze the effect of two strategies on the training domain and on the target domains in search for patterns that can be beneficial in term of performance. We are operating in the embedding space so the results should be less strictly connected to the initial nature of the data.

The experiments reported in this section aim to validate or invalidate the following hypotheses:

- The type of the domain shift (e.g., from more abstract to more complex) influences the type of feature-level augmentation required for effective classification in the target domain.

- The type of machine learning algorithm (e.g., max-margin vs. neural net) also affects the kind of feature-level augmentation needed for effective classification in the target domain.

Like in the data augmentation experiments two classifiers are used, an SVM classifier and a network formed by a single fully connected layer as classifier (FC). the SVM is implemented using the SVC class provided by the scikit-learn Python library for multi-class classification problems. A softmargin SVM implementation is used, meaning that some misclassification is allowed; the degree of tolerance is controlled by the regularization parameter C, which is set to 1. The kernel used is a linear kernel so the decision boundaries are hyper-planes. The FC is implemented as a single linear layer implemented with pytorch that takes in input the VGG19 embeddings and the generated embeddings and learn a classification accordingly to the number of classes of the considered dataset. We use cross entropy loss and the Adam optimizer. The batch size has been set to 32 and the number of epochs has been set to 30 which allowed to reach stability in the accuracy of prediction

The strategies used during the experiments are::

- Inward-Outward (In-Out) strategy: for each point in the train dataset a new point is added at a certain percentage further or nearer distance from the class centroid. The steps chosen are 2,5%, 5%, 7.5%, 10%, 20% 30% 40% 50%, 60% in both directions. (Figure 6.66). .

- Rotation strategy: for each point in the train dataset a new point is added after a rotation of α degrees around the class centroid in a rotation plane including the vector between the point and the class centroid (Figure 6.67). . A rotation in n dimensions is described as being in an $n-1$ dimensional hyper plane in a direction from one unit vector towards another. The rotation matrix is computed according to the following formula [104].

$$R = I + (\mathbf{n_2 n_1^T} - \mathbf{n_1 n_2^T}) \sin\alpha + (\mathbf{n_1 n_1^T} + \mathbf{n_2 n_2^T})(\cos\alpha - 1)$$

  Where $\mathbf{n_1}$ and $\mathbf{n_2}$ are two n-dimensional orthogonal unit vectors with $\mathbf{n_1}$ pointing from the centroid to the starting point. The $\mathbf{n_2}$ vector can be obtained swapping the values of two non zero components of $\mathbf{n_1}$ changing the sign of one of them, setting all the other components to 0 and normalizing.

The models are trained on a domain and then applied on a test subset of the same domain and on the other domains.

The following datasets are used for the experiments.

- Office Home dataset [105]. It is composed of 30475 images arranged in four different domains: Clip Art, Artistic images, Product images and Real-World images. For each domain, the dataset contains images of 65 object categories found typically in Office and Home settings. (Figure 6.65(a)).

- Modern Office-31 dataset [106]. It's a modified version of the Office-31 dataset. Modern Office-31 corrects annotation errors and low quality images in the Amazon domain of the original Office-31 dataset. Additionally, this dataset adds another synthetic domain based on the Adaptiope dataset. It contains 6712 images from four domains, Amazon, DSLR, Webcam and synthetic, each domains contains 31 categories (Figure 6.65(b)).

- Adaptiope dataset [106]. It contains 36900 images arranged in 3 domains of 123 classes: (a) product, (b) reallife, and synthetic, with different complexities (Figure 6.65(c)).

(a) Office Home (b) Modern Office-31 (c) Adaptiope

Figure 6.65: Sample images from the three benchmark data sets considered in this study: (a) Office home [105], (b) Modern Office-31 [106], and (c) Adaptiope [106] – all three data sets include subsets of images that are of different levels of abstraction.

For theses data sets the VGG19 network is used to extract the embeddings of the images.

The models are trained on a domain and the applied on the other domains.

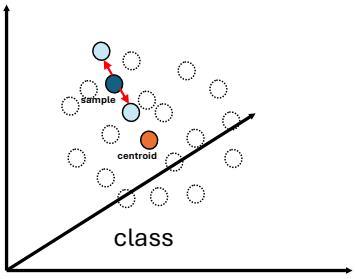


Figure 6.66: In-out augmentation: for each selected sample data vector, a new vector pointing in the same direction, but of a different distance from the class centroid, is created

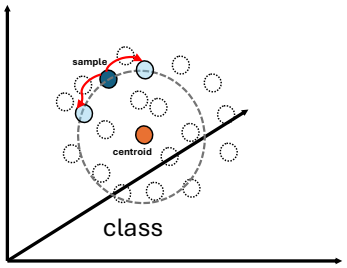


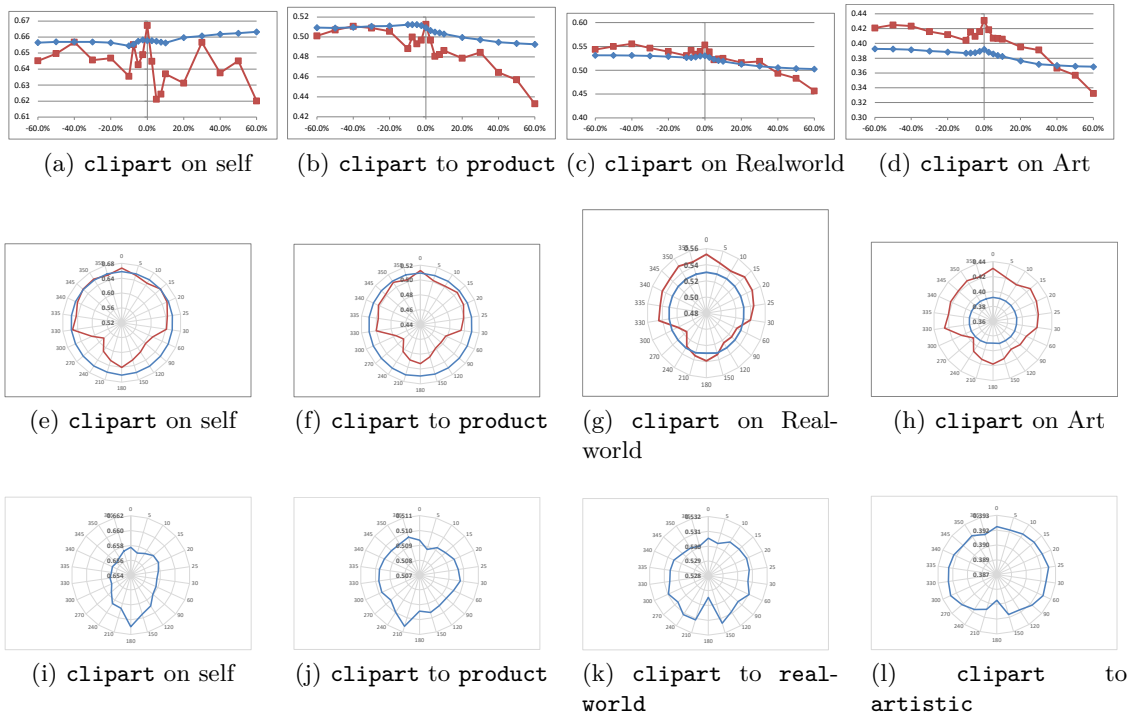
Figure 6.67: Rotation-based augmentation: for each selected sample data vector, a new vector at the same distance, but at a different angle from the corresponding class centroid, is created.

6.8.2 Experiments with the Office Home Data Set

6.8.2.1 Simple-to-Complex Domain Shift

In Figure 6.68, we present the results for the Office home data set, where a model is trained on a relatively simple/abstract domain (`clipart`) and the learned model (SVM or FC) is applied on a test subset of the same domain as well as other three domains under in-out and rotation-based augmentation strategies: More specifically, Figures 6.68(a) through (d) present in-out augmentation results for SVM and FC, whereas Figures 6.68(e) through (l) present rotation-based augmentation results – since, under rotation-based augmentation, SVM results are more stable than the FC results, in Figures 6.68 (i) through (l), we separately plot the SVM results to better see the effects of the degree of rotation:

- The first thing one can notice from Figure 6.68 is that, as we mentioned above, the SVM model is less sensitive to augmentation than the FC strategy.
- We also see that, as one would expect, the model trained on a simple/abstract domain becomes less and less effective as we test on more



(a) `clipart` on self (b) `clipart` to `product` (c) `clipart` on Realworld (d) `clipart` on Art

(e) `clipart` on self (f) `clipart` to `product` (g) `clipart` on Real-world (h) `clipart` on Art

(i) `clipart` on self (j) `clipart` to `product` (k) `clipart` to `real-world` (l) `clipart` to `artistic`

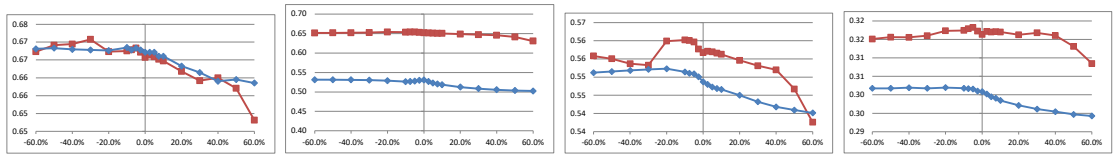
Figure 6.68: Office Home dataset. Application of model trained on the simple/abstract `clipart` domain – blue lines indicate SVM; red lines indicate FC based model: (a)-(d) in-out augmentation results; (e)-(l) rotation-based augmentation results (since SVM results are significantly more stable under rotation-based augmentation than the FC results, in (i)-(l) threre are the plots of the SVM results alone)

complex domains, with the lowest performance obtained for the most complex `artistic` domain.

- While the SVM-based classification appears to be more effective when the model trained on the relatively simple/abstract domain, `clipart`, is tested against itself or the other simple model, `product`, FC-based model appears to have a better generalization performance when the test domains are complex, such as `real-world` or `artistic`.

- Studying Figures 6.68(a) through (d), which present the in-out augmentation results for FC and SVM models, we observe the following:

  – For the FC model, plotted in red, the best results are obtained without augmentation (NA). While inwards augmentation strategy (negative values) does not appear to have significant impact on the results, the outward strategy (positive values) significantly hurt



(a) `artistic` on self (b) `artistic` to `real-world` (c) `artistic` to `product` (d) `artistic` to `clipart`

(e) `artistic` on self (f) `artistic` to `real-world` (g) `artistic` to `product` (h) `artistic` to `clipart`

(i) `artistic` on self (j) `artistic` to `real-world` (k) `artistic` to `product` (l) `artistic` to `clipart`

Figure 6.69: Office Home dataset. Application of model trained on the complex `artistic` domain – blue lines indicate SVM; red lines indicate FC based model: (a)-(d) in-out augmentation results; (e)-(l) rotation-based augmentation results (since SVM results are significantly more stable under rotation-based augmentation than the FC results, in (i)-(l) there are the SVM results alone)

performance, especially for large values ($\sim 60\%$). This appears to indicate that, in these scenarios, if augmentation is unavoidable, it is better to augment the data with samples closer to the class representative, rather than samples that are away from the centroid.

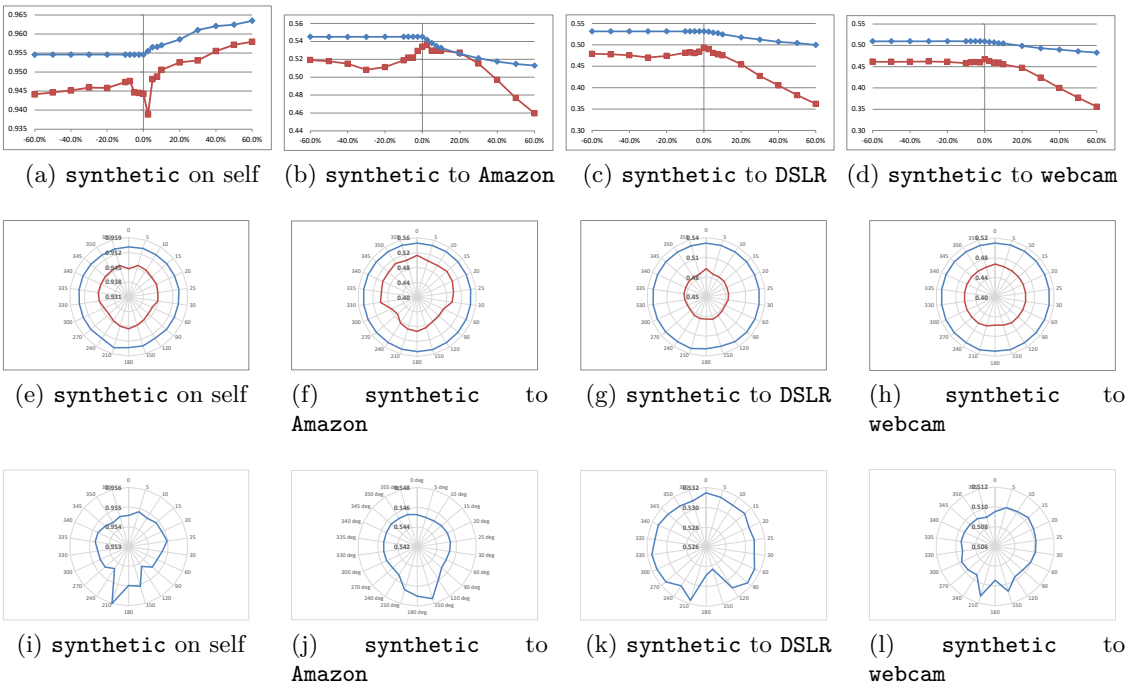
– For the SVM model, plotted in blue, however the results are somewhat different: while outwards augmentation still hurts performance when the model trained on the relatively simple/abstract domain, `clipart`, is tested against test domains that are more complex than itself, we see a generally more robust benefit from inwards augmentation in this case. Moreover, differently from the case of the the FC model, when the model is tested against itself, we see a notable benefit to outwards augmentation: this appears to imply that, while the outwards augmentation is not beneficial for abstract-to-complex domain shifts, it does help better define the max-margin boundary between the classes when the domain stays the same.

- Studying Figures 6.68(e) through (l), which present the rotation-based augmentation results for FC and SVM models, we observe the following:
  - As we mentioned earlier, we see in Figures 6.68(e) through (h) that the SVM model is less affected from rotation-based augmentation than FC model. We further see from these figures that, for FC, when the source domain is simple/abstract, "no augmentation" appears to be the best strategy also under rotation-based augmentation.
  - Looking more closely at the SVM behavior, in Figures 6.68(i) through (k), we observe that SVM does benefit from rotation based augmentation, with the largest benefits occurring at (for self) or around (for domain shifts) 180 degrees. This interesting behavior is due to the fact that, in rotation-based augmentation, each selected sample follows a rotation path on a different plane, which dilutes the effect of the augmentation samples on the max-margin border, except for rotations $\sim$ 180 degrees where all the augmentation samples reconverge, irrespective of the selected rotation plane. The fact that exactly 180 does not provide a sufficient benefit (and may in fact hurt performance when the target domain is more complex) is due to the fact that precisely 180 degree rotations fail to add any major diversity to the data with respect to the original sample selected for augmentation.

6.8.2.2 Complex-to-Simple Domain Shift

In Figure 6.69, are plotted the results for the Office home data set, where a model is trained on a complex domain (`artistic`) and the learned model (SVM or FC) is applied to itself as well as other three domains under in-out and rotation-based augmentation:

- We again observe that the SVM model is less sensitive to augmentation than the FC strategy.
- The model trained on the complex domain becomes less and less effective as we test on a much simpler domain, the lowest performance is obtained for the most abstract `clipart` domain.
- Once again the SVM-based classification is most effective when the model trained on the `artistic` domain is tested against itself, whereas



(a) `synthetic` on self (b) `synthetic` to `Amazon` (c) `synthetic` to `DSLR` (d) `synthetic` to `webcam`

(e) `synthetic` on self (f) `synthetic` to `Amazon` (g) `synthetic` to `DSLR` (h) `synthetic` to `webcam`

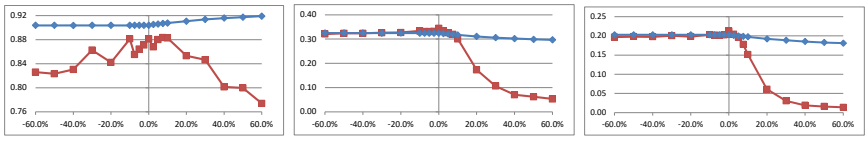
(i) `synthetic` on self (j) `synthetic` to `Amazon` (k) `synthetic` to `DSLR` (l) `synthetic` to `webcam`

Figure 6.70: Modern Office 31. Application of model trained on the `synthetic` domain — blue lines indicate SVM; red lines indicate FC based model: (a)-(d) in-out augmentation results; (e)-(l) rotation-based augmentation results (Ssince SVM results are significantly more stable under rotation-based augmentation than the FC results, in (i)-(l) the SVM results are plotted separately from the FC results)

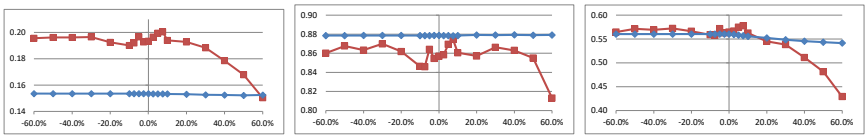
the FC-based model has a better generalization performance to simpler domains.

- Studying Figures 6.69(a) through (d), which present the in-out augmentation results for FC and SVM models, we observe that both models appear to benefit from some inwards augmentation, indicating that samples closer to the class representative helps reduce the impact of the noisy features in the complex source data.

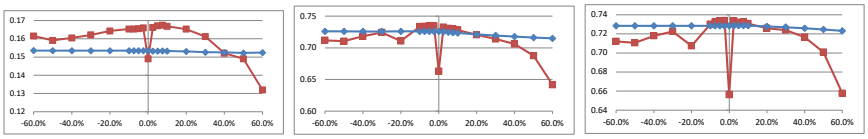
- Studying Figures 6.69(e) through (l), which present the rotation-based augmentation results for FC and SVM models, we observe that in this scenario, the impact of the rotation based augmentation is more subdued for both SVM and FC models and the largest benefits comes when the test domain is complex (`artistic`), whereas when the test domain is very simple (`clipart`) ther are nearly no benefits.



(a) `synthetic` on self (b) `synthetic` to `Product` (c) `synthetic` on Realife (`synthetic`)



(d) `Product` on Synthetic (e) `Product` on self (f) `Product` on `reallife` (`Product`)



(g) `reallife` on Synthetic (h) `reallife` on `Product` (i) `reallife` on self (`reallife`)

Figure 6.71: Adaptiope dataset. Application of models trained with in-out augmentation – blue lines indicate SVM; red lines indicate FC based model

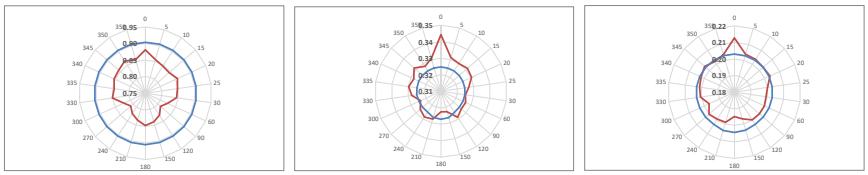
6.8.3 Experiments with the Modern Office 31 Dataset

In Figure 6.70, we present the experiments results for the Modern Office 31 data set, where a model is trained on a relatively simple/abstract domain (`synthetic`) and the learned model (SVM or FC) is applied to itself as well as other three domains (`Amazon`, `DSLR`, and `webcam`) under in-out and rotation-based augmentation strategies[1]: Figures 6.70 (a) through (d) present in-out augmentation results for SVM and FC, whereas Figures 6.70 (e) through (l) present rotation-based augmentation results. Also as in the experiments with the Office Home data set, Figures 6.70 (i) through (l) separately plot the SVM results to help better see the effects of the degree of rotation in rotation-based augmentation.

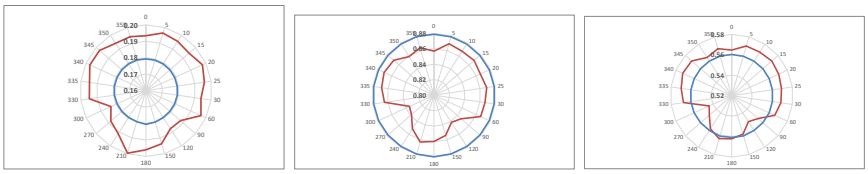
Results for the Modern Office 31 data set, reported in Figure 6.70, largely confirm the observation for the Office Home data set, reported in Figure 6.68:

- Once again, the SVM model is less sensitive to augmentation than the

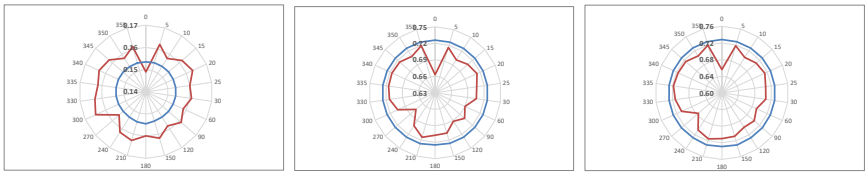
[1]Due to space limitations, for this data set, we omit results where the model is trained in a complex domain.



(a) `synthetic` on self (b) `synthetic` to `Product` (c) `synthetic` to `reallife` (`synthetic`)



(d) `Product` on Synthetic (e) `Product` on self (`Product`) (f) `Product` on `reallife`



(g) `reallife` on Synthetic (h) `reallife` on `Product` (i) `reallife` on self (`reallife`)

Figure 6.72: Adaptiope dataset. Application of models with rotation-based augmentation (blue lines indicate SVM; red lines indicate FC based model)

FC strategy.

- As in the Office Home data set, the model trained on the simple domain becomes less effective as we test one more complex models.
- Unlike the experiments with the Office Home data set (where FC showed a better generalization performance for complex domains), however, in this case, we see that SVM performs better than the FC for all domains. This is likely because, this data set does not include very complex domains, such as `artistic` images, and images in most of the domains have relatively clean backgrounds (see Figure 6.65).
- Studying Figures 6.70(a) through (d), which present the in-out augmentation results for FC and SVM models, we see once again different patterns when the learned model is tested on self and when it is used on other more complex models:
  - When the model is tested on self, adding more diversity to the data

by outwards augmentation provides significant gains for both SVM and FC models. Considering Figure 6.66, this appears to indicate that, in this data set, `synthetic` images are well separated, but with insufficiently well-defined max-margin boundaries.

  - When the model is tested on more complex data, however, we do not see much benefit from in-out augmentation (except for FC where we see slight boost to accuracy when +5% outwards augmentation is applied for model training on `synthetic` domain when being applied to the `Amazon` domain). In fact, in almost all cases, adding points further and further away from the centroid hurts performance, whereas inwards augmentation leads to similar or worse performance.

- Studying Figures 6.70(e) through (l), which present the rotation-based augmentation results for FC and SVM models, we also observe different patterns when the learned model is tested on self and when it is used on other more complex models:

  - As we mentioned earlier, the SVM model is less affected from rotation-based augmentation than FC-based model and performs better than FC based models – the latter is likely because the "complex" models are still relatively simple as they lack complicating factors like complex backgrounds. We further see from these figures that, for FC, "no augmentation" appears to be the best strategy also under rotation-based augmentation – except when the model is tested against itself, in which case augmentations with small (± 5 degrees) or very large ($\sim$ 180 degrees) rotations appear to provide performance boost.
  - Looking more closely at the SVM behavior, in Figures 6.70(i) through (k), we observe in all cases that SVM does benefit from rotation-based augmentation with 150 or 210 degree rotations (which as discussed earlier, all the augmentation samples, even if rotated through different planes, re-converge albeit on the almost opposite side, of the original samples).

6.8.4 Experiments with the Adaptiope Data Set

In Figures 6.71 and 6.72, we present the results of the experiments on the Adaptiope data set, with in-out and rotation-based augmentations respectively. Like for the other data sets, we consider both SVM and FC-based

models. However, in these figures, we consider all pairs of training/testing configurations for the three domains, `synthetic`, `product`, and `reallife`. Therefore, in these figures, we not only observe the effects of simple-to-complex domain shifts (such as `synthetic` to `reallife`), but also complex-to-simple domain shifts (such as `reallife` to synthetic).

Below, we summarize the observations from the in-out augmentation strategy reported in Figure 6.71:

- *Simpler source Domain (`synthetic`):* Figures 6.71(a) through (c) show the in-out augmentation results for SVM and FC-based models, where the models are trained on synthetic; i.e., the most simple domain in the data set.
  - Observations from these charts are similar to the corresponding observations for the Office Home and Modern Office data sets, confirming that in-out augmentation is largely ineffective when the source domain is simpler than the target domain – we, once again, see benefit in outwards-augmentation when the models trained on `synthetic` data are tested on data from the `synthetic` domain.
- *Simple source Domain :* Figures 6.71(d) through (f) show the in-out augmentation results for SVM and FC-based models, where the models are trained on the `product` domain with medium-level of complexity:
  - One interesting observation in this case that the more difficult domain shift involves applying the model to a simpler target domain, `synthetic` - while both domain shifts see a drop in accuracy, the most significant drop occurs in the case of the shift to the `synthetic` domain.
  - FC provides a better accuracy when the shift is to the simpler domain, whereas SVM performs better when the model is applied to itself. We see a similar performance when the model is applied to the more complex domain.
  - Most importantly, while the SVM is not much affected from in-out augmentation, we see that FC sees some gains with both inwards and outwards augmentations, indicating that, when the source is not overly abstracted, in-out augmentation strategy provides help in eliminating the effects of the noise in the data
- *Complex Source Domain (`reallife`):* Figures 6.71(g) through (i) present the in-out augmentation results for SVM and FC-based models, where

the models are trained on the most complex domain, `reallife`, in the data set.

  - Here, we once again see that the more difficult domain shift involves applying the model to a simpler target domain, synthetic.
  - Moreover, we see that the FC model performs very poorly without augmentation when the source data is complex and both inwards and outwards augmentations help eliminate the effects of noisy features in the data thereby significantly boosting the accuracies of the FC-based models.

Next, we summarize the observations from the rotation-based augmentation strategy reported in Figures 6.72:

- *Simpler Source Domain (*`synthetic`*):* Figures 6.72(a) through (c) show the rotation-based augmentation results for SVM and FC-based models, where the models are trained on synthetic; i.e., the most simple/abstract domain in the data set.
  - These results mirror the results for the in-out augmentation strategy in that augmentation is largely ineffective when the source domain is much simpler than the target domain.
- *Simple/Abstract Source Domain (*`product`*):* Figures 6.72(d) through (f) show the rotation-based augmentation results for SVM and FC-based models, where the models are trained on the `product` domain with medium-level of complexity and tested on simpler and more complex domains.
  - Once again, also in this case, the more difficult domain shift involves applying the model to a simpler target domain, `synthetic` - while both domain shifts see a drop in accuracy, the most significant drop occurs in the case of the shift to the `synthetic` domain.
  - Also in this case, FC provides a better accuracy when the shift is to the simpler domain, whereas SVM performs better when the model is applied to itself. We see more comparable performances when the models are applied to the more complex. `reallife` domain.
  - We also see that FC observes performance gains with rotation - based augmentation. When the target is the simplest domain, `synthetic`, the biggest gains are observed with 210 degree rotations; in other scenarios, smaller rotations within ± 30 degrees,

but larger than ±5 degees, appear to provide largest performance boosts.

- *Complex Source Domain (*`reallife`*):* Figures 6.72(g) through (i) present the rotation-based augmentation results for SVM and FC-based models, where the models are trained on the most complex domain, `reallife`, in the data set.
  - Once again, the more difficult domain shift involves applying the model to a simpler target domain, synthetic.
  - As in the case with in-out augmentation results, the FC model performs very poorly without augmentation when the source data is complex. In this case, augmentations with ±5 degree rotations as well as rotations ∼ 210 degrees appear to help eliminate the effects of noisy features and boost the accuracies of the FC-based models.

6.8.5 Summary

From the results presented in Figures 6.68 through and 6.72, across different datasets (Office Home, Modern Office 31, and Adaptiope), we observe several general patterns regarding the impact of the augmentation techniques on the performance of max-margin (SVM) and full-conectected layer (FC) based models under different augmentation strategies:

- *Robustness of SVM vs. FC Models:* SVM models perform more stably across various domain shifts and are consistently less sensitive to both in-out and rotation-based augmentation strategies compared to FC models. FC on the other hand is more sensitive to augmentation, showing significant performance variations depending on the augmentation strategy and the complexity of the domain shifts. However, they sometimes outperform SVM when generalizing to complex domains (as seen, for example, in the Office Home dataset).
- *Impact of Domain Complexity:* Models trained on simpler or more abstract domains (e.g., `synthetic` or `clipart`) generally perform worse when applied to more complex target domains (e.g., `real-world` or `artistic`). Interestingly, models trained on complex domains also tend to perform poorly when tested on much simpler domains, requiring augmentation to mitigate the drop in performance.
- *Effectiveness of In-Out Augmentation:* FC models perform best without augmentation or with inward augmentation when trained on simpler

domains; significant outward augmentation generally degrades performance, especially for larger outward shifts. When the model is trained on a complex model (such as `artistic` or `reallife`), however augmentation (especially inwards) helps FC-based models.

In some scenarios, SVM models may benefit from inward augmentation. Outward augmentation is helpful only when a simple/abstract model is tested on itself, aiding in defining max-margin boundaries.

- *Effectiveness of Rotation-based Augmentation:* FC models benefit from rotation-based augmentation, especially when the source domain is complex. In this case, either very small rotations or very large rotations (150-210 degrees) provide benefits. SVM models show relative stability across various rotation angles, with slight benefits seen around rotations of 150 to 210 degrees – these angles help realign the samples along the max-margin border.

- *Generalization Performance:* SVM models tend to generalize well across domains that are similar or only moderately different in complexity. FC models appear to show better generalization with major shifts in complexity, but this comes with higher sensitivity to augmentation.

- *Augmentation and Domain Shift:* For domain shifts from simpler to more complex domains, both in-out and rotation-based augmentations are generally ineffective. For domain shifts from complex to simple domains, on the other hand, both augmentation strategies may help improve performance by addressing noisy features.

6.8.6 Conclusions

Domain shift can be a major challenge, as a model learned using a given data set may not be applicable if the characteristics of the data changes. We considered augmentation-based solutions to the domain shift problem, especially focusing on simple-to-complex or complex-to-simple domain shifts. Based on the argument that depending on the nature of the shift and the learning models used, one may need to leverage different augmentation techniques, we have investigated the effectiveness of different feature-based training data augmentation strategies (in-out, rotation-based) for different types of domain shifts (simple-to-complex or complex-to-simple), and machine learning approaches (max-margin and fully-connected layer based classifiers). Experiments with several benchmark data sets that include

data of varying levels of abstraction (e.g. `clipart`, `artistic`, `product`, and `real-world`) confirmed that the nature of the domain shift and the characteristics of the learning strategy strongly impact the choice of the augmentation technique.

Chapter 7

Conclusions and Future Work

In this research we have presented different approaches for dealing with noisy and sparse information in different scenarios and tasks.

We investigated a Named Entity Recognition (NER) task in a multimodal setting, a text-image alignment task and different scenarios of data augmentation.

The Named Entity Recognition task was researched using Twitter messages with associated images characterized by noisiness and sparsity. This experience motivated us to explore a novel attention mechanism that we applied on a task of text and fragments of images alignment. In both cases there are motivation for further analysis and investigations especially looking at the results obtained with "ideal features" on the NER task.

Reflections on the attention mechanism have led us to consider possible applications in the field of data augmentation.

We considered a setting where test data are quite diverse from training data.

We explored the possibility of finding what portion of a training data set to modify and add to the training set in order to improve the performance on the test data.

We investigated approaches based on factorization machines, cosine similarity and Euclidean distance, finding that it is possible to obtain indications of strategies for finding and using portions of a training dataset for data augmentation. Further work in this area should investigate other techniques and application on more "real datasets".

The same applies about the research on domain adaptation where different techniques can be explored always remaining in the context of features

level augmentation in order to preserve as much as possible the independence from the original nature of the data.

List of Figures

List of Tables

Bibliography

[1] Twitter, "Moma the museum of modern art [@@museummodernart].tweets [twitter profile]," 2021. https://twitter.com/museummodernart, *AccessedJuly*21, 2021.

[2] Q. Zhang, J. Fu, X. Liu, and X. Huang, "Adaptive co-attention network for named entity recognition in tweets," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, Apr. 2018.

[3] M. Asgari-Chenaghlu, M. R. Feizi-Derakhshi, L. Farzinvash, M. A. Balafar, and C. Motamed, "CWI: A multimodal deep learning approach for named entity recognition from social media using character, word and image features," *Neural Computing and Applications*, vol. 34, pp. 1905–1922, sep 2021.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," NIPS'17, (Red Hook, NY, USA), p. 6000–6010, Curran Associates Inc., 2017.

[5] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," 2015.

[6] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," 2014.

[7] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014.

[8] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," 2018.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *ArXiv*, vol. abs/1810.04805, 2019.

[10] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu, "Recurrent models of visual attention," 2014.

[11] A. Mumuni and F. Mumuni, "Data augmentation: A comprehensive survey of modern approaches," *Array*, vol. 16, p. 100258, 2022.

[12] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, p. 60, 2019.

[13] J. Walsh, N. O' Mahony, S. Campbell, A. Carvalho, L. Krpalkova, G. Velasco-Hernandez, S. Harapanahalli, and D. Riordan, "Deep learning vs. traditional computer vision," 04 2019.

[14] S. Rendle, "Factorization machines," *2010 IEEE International Conference on Data Mining*, pp. 995–1000, 2010.

[15] A. Y. Chervonenkis, "Early history of support vector machines," *Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik*, pp. 13–20, 2013.

[16] M. Wang and W. Deng, "Deep visual domain adaptation: A survey," *Neurocomputing*, vol. 312, pp. 135–153, 2018.

[17] A. Farahani, S. Voghoei, K. Rasheed, and H. R. Arabnia, "A brief review of domain adaptation," in *Advances in Data Science and Information Engineering* (R. Stahlbock, G. M. Weiss, M. Abou-Nasr, C.-Y. Yang, H. R. Arabnia, and L. Deligiannidis, eds.), (Cham), pp. 877–894, Springer International Publishing, 2021.

[18] MoMa, "Github repository," 2021. https://github.com/MuseumofModernArt/collection.

[19] Wikipedia, "The free encyclopedia, list of art movements," 2021. https://en.wikipedia.org/wiki/List_of_art_movements, $accessed July 21, 2021$.

[20] Amazon, "Amazon mechanical turk," 2021. https://www.mturk.com/, (Accessed July 21, 2021).

[21] N. Lawson, K. Eustice, M. Perkowitz, and M. Yetisgen-Yildiz, "Annotating large email datasets for named entity recognition with Mechanical Turk," in *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, (Los Angeles), pp. 71–79, Association for Computational Linguistics, June 2010.

[22] B. P. Yuhas, M. H. Goldstein, and T. J. Sejnowski, "Integration of acoustic and visual speech signals using neural networks," *IEEE Communications Magazine*, vol. 27, pp. 65–71, 1989.

[23] M. Gurban, J.-P. Thiran, T. Drugman, and T. Dutoit, "Dynamic

modality weighting for multi-stream hmms inaudio-visual speech recognition," in *International Conference on Multimodal Interaction*, 2008.

[24] T. Baltruaitis, C. Ahuja, and L.-P. Morency, "Multimodal machine learning: A survey and taxonomy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, pp. 423–443, 2017.

[25] X. Wei, T. Zhang, Y. Li, Y. Zhang, and F. Wu, "Multi-modality cross attention network for image and sentence matching," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10941–10950, June 2020.

[26] L. Rau, "Extracting company names from text," in *[1991] Proceedings. The Seventh IEEE Conference on Artificial Intelligence Application*, vol. i, pp. 29–32, 1991.

[27] D. Nadeau and S. Sekine, "A survey of named entity recognition and classification," *Lingvisticae Investigationes*, vol. 30, 08 2007.

[28] M. A. Khalid, V. Jijkoun, and M. De Rijke, "The impact of named entity normalization on information retrieval for question answering," in *Proceedings of the IR Research, 30th European Conference on Advances in Information Retrieval*, ECIR'08, (Berlin, Heidelberg), p. 705–710, Springer-Verlag, 2008.

[29] X. Han, L. Sun, and J. Zhao, "Collective entity linking in web text: a graph-based method," in *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, (New York, NY, USA), p. 765–774, Association for Computing Machinery, 2011.

[30] K. Lata, P. Singh, and K. Dutta, "Mention detection in coreference resolution: survey," *Applied Intelligence*, vol. 52, 07 2022.

[31] W. Bouarroudj and B. Zizette, *A Candidate Generation Algorithm for Named Entities Disambiguation Using DBpedia*, pp. 712–721. 03 2018.

[32] G. Zhu and C. A. Iglesias, "Exploiting semantic similarity for named entity disambiguation in knowledge graphs," *Expert Systems with Applications*, vol. 101, pp. 8–24, 2018.

[33] A. Abdelrazek, Y. Eid, E. Gawish, W. Medhat, and A. Hassan, "Topic modeling algorithms and applications: A survey," *Information Systems*, vol. 112, p. 102131, 2023.

[34] K. Detroja, C. Bhensdadia, and B. S. Bhatt, "A survey on relation extraction," *Intelligent Systems with Applications*, vol. 19, p. 200244,

2023.

[35] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, 02 2011.

[36] E. Riloff and R. Jones, "Learning dictionaries for information extraction by multi-level bootstrapping," pp. 474–479, 01 1999.

[37] W. W. Cohen and S. Sarawagi, "Exploiting dictionaries in named entity extraction: combining semi-markov extraction processes and data integration methods," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, (New York, NY, USA), p. 89–98, Association for Computing Machinery, 2004.

[38] E. Alfonseca and S. Manandhar, "An unsupervised method for general named entity recognition and automated concept discovery," 2004.

[39] J. Barr, M. Littman, and M. desJardins, "Decision trees," *ACM Inroads*, vol. 10, p. 56, aug 2019.

[40] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state markov chains," *The annals of mathematical statistics*, vol. 37, no. 6, pp. 1554–1563, 1966.

[41] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, (San Francisco, CA, USA), p. 282–289, Morgan Kaufmann Publishers Inc., 2001.

[42] A. McCallum, "Efficiently inducing features of conditional random fields," in *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, UAI'03, (San Francisco, CA, USA), p. 403–410, Morgan Kaufmann Publishers Inc., 2002.

[43] V. Yadav and S. Bethard, "A survey on recent advances in named entity recognition from deep learning models," in *Proceedings of the 27th International Conference on Computational Linguistics*, (Santa Fe, New Mexico, USA), pp. 2145–2158, Association for Computational Linguistics, Aug. 2018.

[44] Y. Kim, Y. Jernite, D. A. Sontag, and A. M. Rush, "Character-aware neural language models," in *AAAI Conference on Artificial Intelligence*, 2015.

[45] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, 1997.

[46] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *International Conference on Learning Representations*, 2013.

[47] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," 2013.

[48] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (A. Moschitti, B. Pang, and W. Daelemans, eds.), (Doha, Qatar), pp. 1532–1543, Association for Computational Linguistics, Oct. 2014.

[49] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014.

[50] Wikipedia, "Bio tagging," 2021. https://en.wikipedia.org/wiki/Inside outside beginning (tagging), (Accessed July 21, 2021).

[51] L. A. Ramshaw and M. P. Marcus, "Text chunking using transformation-based learning," 1995.

[52] R. Grishman and B. Sundheim, "Message Understanding Conference-6: A brief history," in *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*, 1996.

[53] G. R. Doddington, A. Mitchell, M. A. Przybocki, L. A. Ramshaw, S. Strassel, and R. M. Weischedel, "The automatic content extraction (ace) program - tasks, data, and evaluation," in *International Conference on Language Resources and Evaluation*, 2004.

[54] K. Keahey, J. Anderson, Z. Zhen, P. Riteau, P. Ruth, D. Stanzione, M. Cevik, J. Colleran, H. S. Gunawi, C. Hammock, J. Mambretti, A. Barnes, F. Halbach, A. Rocha, and J. Stubbs, "Lessons learned from the chameleon testbed," in *Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC '20)*, USENIX Association, July 2020.

[55] M. Aldinucci, S. Rabellino, M. Pironti, F. Spiga, P. Viviani, M. Drocco, M. Guerzoni, G. Boella, M. Mellia, P. Margara, I. Drago, R. Marturano, G. Marchetto, E. Piccolo, S. Bagnasco, S. Lusso, S. Vallero, G. Attardi, A. Barchiesi, A. Colla, and F. Galeazzi, "Hpc4ai: an ai-on-demand federated platform endeavour," in *Proceedings of the 15th ACM International Conference on Computing Frontiers*, CF '18, (New York, NY, USA), p. 279–286, Association for Computing Machinery, 2018.

[56] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning internal representations by error propagation*, p. 318–362. Cambridge, MA, USA: MIT Press, 1986.

[57] C. Liu, Z. Mao, A.-A. Liu, T. Zhang, B. Wang, and Y. Zhang, "Focus your attention: A bidirectional focal attention network for image-text matching," 2019.

[58] K. Cho, A. Courville, and Y. Bengio, "Describing multimedia content using attention-based encoder-decoder networks," *IEEE Transactions on Multimedia*, vol. 17, p. 1875–1886, Nov. 2015.

[59] A. Zadeh, P. P. Liang, S. Poria, P. Vij, E. Cambria, and L.-P. Morency, "Multi-attention recurrent network for human communication comprehension," 2018.

[60] C. Liu, Z. Mao, A.-A. Liu, T. Zhang, B. Wang, and Y. Zhang, "Bfan github repository," 2019. https://github.com/CrossmodalGroup/BFAN.

[61] B. A. Plummer, L. Wang, C. M. Cervantes, J. C. Caicedo, J. Hockenmaier, and S. Lazebnik, "Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 2641–2649, Dec. 2015.

[62] T.-Y. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context.," *CoRR*, vol. abs/1405.0312, 2014.

[63] K.-H. Lee, X. Chen, G. Hua, H. Hu, and X. He, "Stacked cross attention for image-text matching," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 201–216, 2018.

[64] K. Candan and M. Sapino, *Data Management for Multimedia Retrieval*. Cambridge University Press, Jan. 2011. Publisher Copyright: © K. Selçuk Candan and Maria Luisa Sapino 2010.

[65] Y. Xu, R. Jia, L. Mou, G. Li, Y. Chen, Y. Lu, and Z. Jin, "Improved relation classification by deep recurrent neural networks with data augmentation," 2016.

[66] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, "Understanding data augmentation for classification: when to warp?," 2016.

[67] K. Wang, B. Fang, J. Qian, S. Yang, X. Zhou, and J. Zhou, "Perspective transformation data augmentation for object detection," *IEEE Access*, vol. PP, pp. 1–1, 12 2019.

[68] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," 2017.

[69] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 08 2017.

[70] C. Gong, D. Wang, M. Li, V. Chandra, and Q. Liu, "Keepaugment: A simple information-preserving data augmentation approach," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1055–1064, June 2021.

[71] T. Xie, X. Cheng, X. Wang, M. Liu, J. Deng, T. Zhou, and M. Liu, "Cut-thumbnail: A novel data augmentation for convolutional neural network," *Proceedings of the 29th ACM International Conference on Multimedia*, 2021.

[72] X. Shen, X. Tian, A. He, S. Sun, and D. Tao, "Transform-invariant convolutional neural networks for image classification and search," in *Proceedings of the 24th ACM International Conference on Multimedia*, MM '16, (New York, NY, USA), p. 1345–1354, Association for Computing Machinery, 2016.

[73] Y. Li and F. Liu, *Adaptive Gaussian Noise Injection Regularization for Neural Networks*, p. 176–189. Springer International Publishing, 2020.

[74] A. Dabouei, S. Soleymani, F. Taherkhani, and N. M. Nasrabadi, "Supermix: Supervising the mixing data augmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13794–13803, June 2021.

[75] G. Kang, J. Li, and D. Tao, "Shakeout: A new regularized deep neural network training scheme," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, Feb. 2016.

[76] D. Krueger, T. Maharaj, J. Kramár, M. Pezeshki, N. Ballas, N. R. Ke, A. Goyal, Y. Bengio, A. Courville, and C. Pal, "Zoneout: Regularizing rnns by randomly preserving hidden activations," 2017.

[77] Z. Dai, M. Chen, X. Gu, S. Zhu, and P. Tan, "Batch dropblock network for person re-identification and beyond," pp. 3690–3700, 10 2019.

[78] J. Choe and H. Shim, "Attention-based dropout layer for weakly supervised object localization," 2019.

[79] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, Association for

Computational Linguistics, 11 2019.

[80] P. Aggarwal, "Kaggle dataset," 2021. https://www.kaggle.com/datasets/paramaggarwal/fashion-product-images-small.

[81] W. Dupont and W. Plummer, "Power and sample size calculations. a review and computer program," *Controlled clinical trials*, vol. 11, pp. 116–28, 05 1990.

[82] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," 2020.

[83] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big Data*, vol. 3, no. 1, pp. 1–40, 2016.

[84] A. Hosna, E. Merry, J. Gyalmo, Z. Alom, Z. Aung, and M. Azim, "Transfer learning: a friendly introduction," *Journal of Big Data*, vol. 9, 10 2022.

[85] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu, "Self-taught clustering," in *International Conference on Machine Learning*, 2008.

[86] Y. Song and C. Zhang, "Transferred dimensionality reduction," pp. 550–565, 09 2008.

[87] W. K. Mutlag, S. K. Ali, Z. M. Aydam, and B. H. Taher, "Feature Extraction Methods: A Review," in *Journal of Physics Conference Series*, vol. 1591 of *Journal of Physics Conference Series*, p. 012028, July 2020.

[88] T. maintainers and contributors, "Torchvision: Pytorch's computer vision library." https://github.com/pytorch/vision, 2016.

[89] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, pp. 8024–8035, Curran Associates, Inc., 2019.

[90] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[91] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," 2015.

[92] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pp. 722–729, 2008.

[93] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu, "Transfer learning with joint distribution adaptation," in *Proceedings of the IEEE international conference on computer vision*, pp. 2200–2207, 2013.

[94] J. Choi, G. Sharma, M. Chandraker, and J.-B. Huang, "Unsupervised and semi-supervised domain adaptation for action recognition from drones," in *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1706–1715, 2020.

[95] C. Chen, W. Xie, W. Huang, Y. Rong, X. Ding, Y. Huang, T. Xu, and J. Huang, "Progressive feature alignment for unsupervised domain adaptation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 627–636, 2019.

[96] J. Huang, A. Gretton, K. Borgwardt, B. Schölkopf, and A. Smola, "Correcting sample selection bias by unlabeled data," in *Advances in Neural Information Processing Systems* (B. Schölkopf, J. Platt, and T. Hoffman, eds.), vol. 19, MIT Press, 2006.

[97] J. Li, M. Jing, H. Su, K. Lu, L. Zhu, and H. T. Shen, "Faster domain adaptation networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 12, pp. 5770–5783, 2021.

[98] H. Liu, J. Wang, and M. Long, "Cycle self-training for domain adaptation," 2021.

[99] Y. Zheng, L. He, X. Wu, and C. Pan, "Self-training and multi-level adversarial network for domain adaptive remote sensing image segmentation," *Neural Processing Letters*, vol. 55, pp. 1–26, 08 2023.

[100] A. Ramponi and B. Plank, "Neural unsupervised domain adaptation in NLP—A survey," in *Proceedings of the 28th International Conference on Computational Linguistics* (D. Scott, N. Bel, and C. Zong, eds.), (Barcelona, Spain (Online)), pp. 6838–6855, International Committee on Computational Linguistics, Dec. 2020.

[101] G. French, M. Mackiewicz, and M. Fisher, "Self-ensembling for visual domain adaptation," 2018.

[102] L. Mai and J. Carson-Berndsen, "Unsupervised domain adaptation for speech recognition with unsupervised error correction," 2022.

[103] B. Li, D. Hwang, Z. Huo, J. Bai, G. Prakash, T. N. Sainath, K. C. Sim, Y. Zhang, W. Han, T. Strohman, and F. Beaufays, "Efficient domain adaptation for speech foundation models," 2023.

[104] P. Masson, "Rotations in higher dimensions," 2017. https://analyticphysics.com/Higher Dimensions/Rotations in Higher Dimensions.htm.

[105] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan, "Deep hashing network for unsupervised domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5018–5027, 2017.

[106] T. Ringwald and R. Stiefelhagen, "Adaptiope: A modern benchmark for unsupervised domain adaptation," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 101–110, January 2021.