This project has received funding from the European Union's Horizon 2020 research and innovation programme under **grant agreement No 780355**

FANDANGO DELIVERABLE

Deliverable No.:	D4.1
Deliverable Title:	Spatio-temporal analytics and out of context fakeness markers prototypes
Project Acronym:	Fandango
Project Full Title:	FAke News discovery and propagation from big Data and artificial inteliGence Operations
Grant Agreement No.:	780355
Work Package No.:	4
Work Package Name:	Fake news identifiers, machine learning and data analytics
Responsible Author(s):	CERTH
Date:	30.07.2019
Status:	v0.6 - Full Draft
Deliverable type:	REPORT
Distribution:	PUBLIC

REVISION HISTORY

VERSION	DATE	MODIFIED BY	COMMENTS
V0.1	07.05.2019	Theodoros Semertzidis	First draft
V0.2	30.05.2019	Panagiotis Stalidis	contributions section 4
V0.3.1	11.06.2019	Panagiotis Stalidis	contributions section 5
V0.3.2	19.06.2019	Gerasimos Palaiopanos	contributions section 5
V0.3.3	22.06.2019	Panagiotis Stalidis	contributions section 4
V0.4.1	12.07.2019	Panagiotis Stalidis	contributions section 6
V0.4.2	15.07.2019	Panagiotis Stalidis	contributions section 2
V0.4.3	19.07.2019	Gerasimos Palaiopanos	contributions section 3
V0.5	25.07.2019	Panagiotis Stalidis, Theodoros Semertzidis	corrections on format
V0.6	29.07.2019	Panagiotis Stalidis	Full Draft
V0.7	30.07.2019		Internal Review
V1.0			Quality check

TABLE OF CONTENTS

List of Figures

LIST OF TABLES

ABBREVIATIONS

ABBREVIATION	DESCRIPTION
H2020	Horizon 2020
EC	European Commission
WP	Work Package
CNN	Convolutional Neural Networks

1. Executive Summary

This report documents the work done regarding spatio-temporal analytics and out of context fakeness markers prototypes. This document describes the state–of–the–art methods for the extraction of markers about the spatial and temporal nature of textual and visual content of a news item and other information that can lead to an informed decision about the validity of an article.

The document starts with analysing the different methodologies that can be used to extract spatio-temporal and context information from text, as well as how these methods are employed in the Fandango project. Then the state–of–the–art for context and spatial information extraction from images and our implementation methodology are described. The functionalities used to place a media file in the correct temporal context by detecting when the media has appeared on the time axis is also presented. Several examples and explanations of the results are also presented within this document both for textual as well as visual modalities.

One of the most important components provided in this deliverable is the duplicate detection of images. The component is able to retrieve similar images even if they have been modified, cropped, or otherwise changed from the original image.

2. Introduction

The recent ubiquity of internet connection has significantly changed the way in which people acquire information. Nowadays, there are increasingly more people consuming news from online sources, which can provide timely and comprehensive multimedia information on the events taking place all over the world. Compared with traditional text news, the news with images and videos can offer a better storytelling and attract more attention from readers. Unfortunately, this is also taken advantage by fake news which usually contain misrepresented or even forged images, to mislead the readers and get rapid dissemination.

The question of fake news brings up the question of how to think about the nature of real news. News has been defined in a number of ways, ranging from being an account of a recent, interesting, and significant event, to a dramatic account of something novel or deviant. News is expected to provide "independent, reliable, accurate, and comprehensive information". At the same time, news is socially constructed, and journalists often exercise subjective judgment on which bits of information to include and which to exclude. Thus, news is vulnerable not only to journalists' own preferences, but also to external forces, such as the government, audiences, and advertisers.

According to the definition of what news is, we regard fake news as a news piece that is in some regard dependent, unreliable, inaccurate or partial, either deliberately or by mistake. In the world of electronic news where the norm is to include images and videos to corroborate the news story, even if the media that are used are authentic, they can be misused by mixing images from different events, or videos can be portional. Consequently, an article can only be characterized after all of its components are examined separately to be valid and the context and semantics of the components coincides.

So that later stages of the Fandango platform can evaluate an article, as much information as possible has to be extracted regarding the semantics of the text, images and videos included. The detection of what is discussed in the text has to be evaluated and the people that are discussed have to be identified. Similar identification has to be performed for the visual part of the article. The images that are included need to regard the same event and show the same people that are discussed in the article.

3. EXTRACTION OF TOPICS, ENTITIES AND KEYWORDS FROM THE ARTICLE TEXT

3.1. TOPIC EXTRACTION

In light of popular scientific advances in the domain of unsupervised machine learning—flexible components for modeling, scalable algorithms for posterior inference— and increased access to massive datasets, topic models promise to be a crucial component for summarizing and understanding the growing digitized archive of information. Topic modelling is a powerful natural language processing tool for the unsupervised inference of the latent topics of a collection of texts.

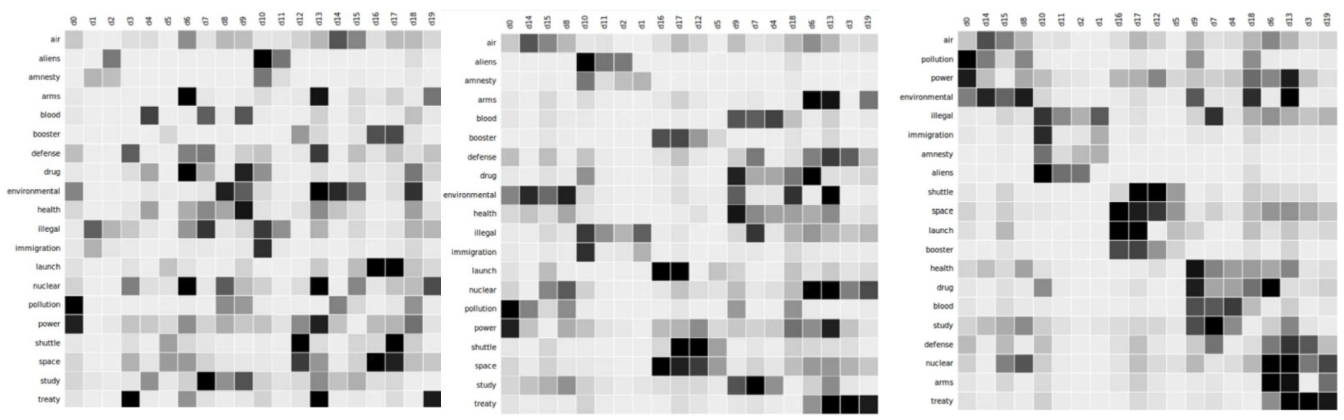


Figure 1 Every column corresponds to a document, every row to a word. A cell stores the frequency of a word in a document, higher word frequencies are indicated by darker cells. Moving from the left to the right, it is shown that topic models group gradually both documents, which use similar words, as well as words which occur in a similar set of documents. The resulting patterns in the right picture are called "topics" [35] displayed by the local concentration of the coloured cells.

The three pictures of *Figure 1,* outline the evolution of grouping the words in order to create sets of 'similar' words. In the final picture, there are groups of four words brought closely.

Probabilistic topic models such as Latent Dirichlet Allocation (LDA) are popular approaches for this task since they can discover latent topics from text collections.

3.1.1. State of the Art

The paper of Panda et al. [1] describes an implementation which relies on a generative model explained as a hierarchical graphical model (*Figure 2*) for the prediction of the underlying topics. The goal of the current Bayesian model is to leverage the latent high-dimensional features and binary label vectors of the data. It uses a likelihood model for each label instance and builds a deep hierarchy of low-dimensional embeddings for the label vector with several important advantages:

1. the possibility of using the EM algorithm (or Gibbs sampling), which makes the inference effective and allows it to scale up in the size of sparse label matrices. This property of the model also permits its use as a deep topic model for sets of documents accompanied by feature vectors.
2. the use of the model as a trainable nonlinear classifier with reference to each label (there is a geometric correspondence to the borders of the decision region). This induces nonlinear embeddings of the label representations and thus the deep generative model presents the possibility to learn such representations.
3. the possibility of tackling the problem as a clustering of the labels, based on viewing the labels as points of a separate space.

As in other suggested solutions involving a generative model, a formal description of the problem is:

given N training examples

$$(x_1, y_1), \ldots, (x_N, y_N), \text{ where } x_n \in R^D \text{ and } y_n \in \{0,1\}^L, n=1,2, \ldots, N$$

predict the label vector: $y_{n+1} \in \{0,1\}^L$ for a new test input $x_{n+1} \in R^D$.

We assume that each observation (x_N, y_N) is associated with a deep hierarchy of latent factors $u_n \in R^{K(t)}$, t= 1,...,T, which generate y_n. The β_i-s denote cumulatively the global parameters on which the u_n variables depend. The latent factors are conditioned on the input samples, as displayed in *Figure 2*. For the algorithm generating the latent variables and the count vectors m_{nl}, the details are given in [1].

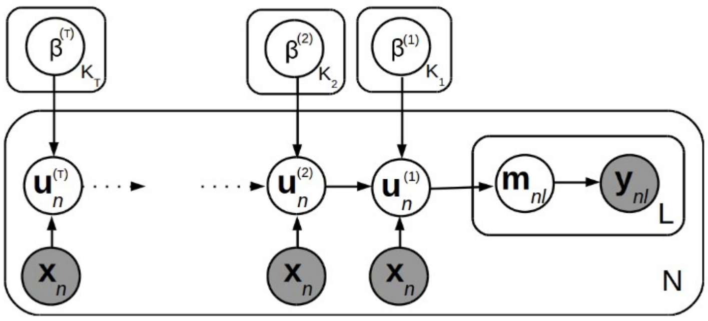


Figure 2: Model of the latent variables depending on β-variables.

Every example concerns a binary label vector and this is employed as input to the deep generative system which is trained to create latent representations depended on the features of the input samples. Under the lens of deep learning, each label vector stands for a bag-of-words article, with the input features being its meta-data. Apart from encoding the topological traits of the label space (for instance a near-low-rank label matrix), the geometric view is that the nonlinear classification boundaries captured by the training can be considered as the union of multiple convex polytopes. The algorithm was tested with 3 benchmark datasets, i.e. Bibtex, Eulerx-4k and Delicious [2], and an example of running on Eulerx dataset is shown in *Figure 3*.

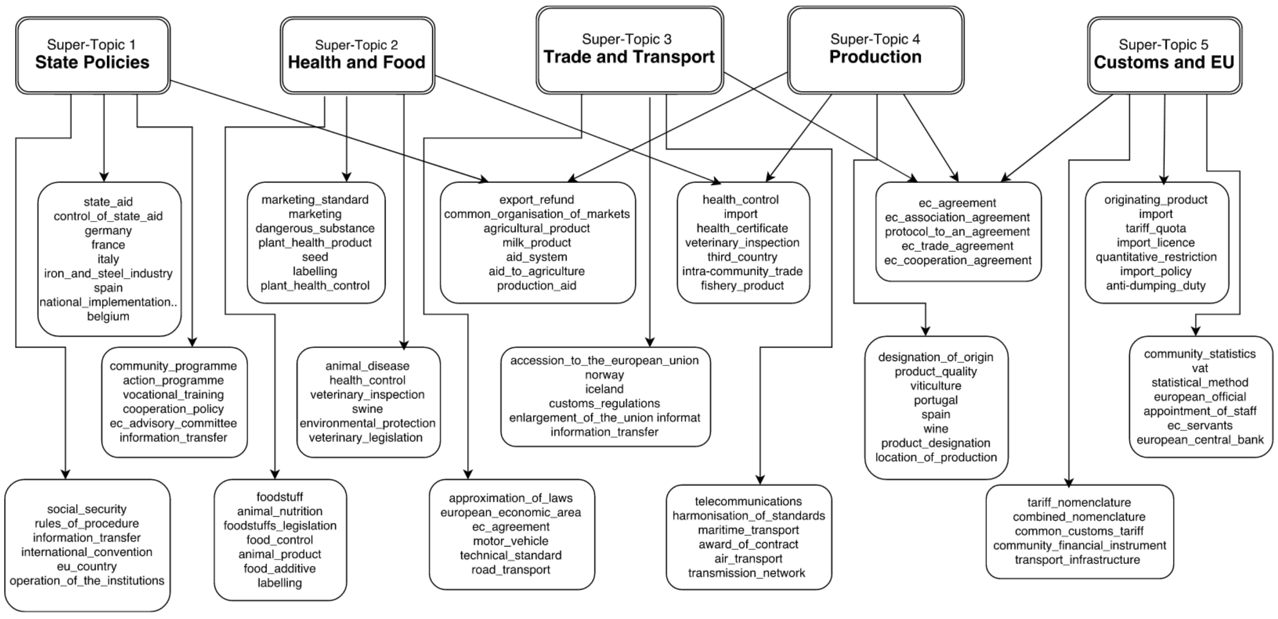

Super-Topic 1 State Policies
Super-Topic 2 Health and Food
Super-Topic 3 Trade and Transport
Super-Topic 4 Production
Super-Topic 5 Customs and EU
state_aid control_of_state_aid germany france italy iron_and_steel_industry spain national_implementation.. belgium
marketing_standard marketing dangerous_substance plant_health_product seed labelling plant_health_control
export_refund common_organisation_of_markets agricultural_product milk_product aid_system aid_to_agriculture production_aid
health_control import health_certificate veterinary_inspection third_country intra-community_trade fishery_product
ec_agreement ec_association_agreement protocol_to_an_agreement ec_trade_agreement ec_cooperation_agreement
originating_product import tariff_quota import_licence quantitative_restriction import_policy anti-dumping_duty
community_programme action_programme vocational_training cooperation_policy ec_advisory_committee information_transfer
animal_disease health_control veterinary_inspection swine environmental_protection veterinary_legislation
accession_to_the_european_union norway iceland customs_regulations enlargement_of_the_union informat information_transfer
designation_of_origin product_quality viticulture portugal spain wine product_designation location_of_production
community_statistics vat statistical_method european_official appointment_of_staff ec_servants european_central_bank
social_security rules_of_procedure information_transfer international_convention eu_country operation_of_the_institutions
foodstuff animal_nutrition foodstuffs_legislation food_control animal_product food_additive labelling
approximation_of_laws european_economic_area ec_agreement motor_vehicle technical_standard road_transport
telecommunications harmonisation_of_standards maritime_transport award_of_contract air_transport transmission_network
tariff_nomenclature combined_nomenclature common_customs_tariff community_financial_instrument transport_infrastructure

Figure 3:Topics and Super topics inferred from Eurlex dataset ([37])

In Jang and Hero [3] the authors introduce a geometric inference method (Minimum Volume Topic Modeling) leveraging the fact that the Latent Dirichlet Allocation (LDA) likelihood function approximates the volume of the topic simplex. Therefore, the problem is transformed to the equivalent version of finding the probability simplex that minimizes its volume 'β' and contains the documents (groups of samples) that are represented as distributions over words, under certain assumptions. The equivalence is about finding the global minimum. Based on the new problem setting, the paper introduces a convex relaxation of the minimization problem. The technique used is the Alternating Direction Method of Multipliers (ADMM) as an iterative Lagrangian algorithm and it is shown that it converges locally. In this work, a number of numerical instants demonstrate the qualities of this solution as for the complexity and the topic capturing accuracy.

Compared to previous research results using Gibbs sampling, Geometric Dirichlet Mean (GDM), RecoverKL and Vertex method [4] (VEM), the current geometric objective function is convex. For instance, *Figure 4b* shows that there is a kink in the optimization path, where MVTM is tracing the right orientation of the true simplex. Furthermore, there is a lack of loops in the optimization path, highlighting the identifiability of MVTM. The performance of MVTM is tested on real-world data, i.e. NIPS dataset. The raw data is preprocessed using a standard stopword list and the resulting data are filtered through a stemmer. The proposed algorithm MVTM is performing better than the vertex methods (Geometric Dirichlet Mean and RecoverKL) in terms of perplexity as it only requires the documents lie on the face of the topic simplex. GDM provides a similar performance to MVTM.

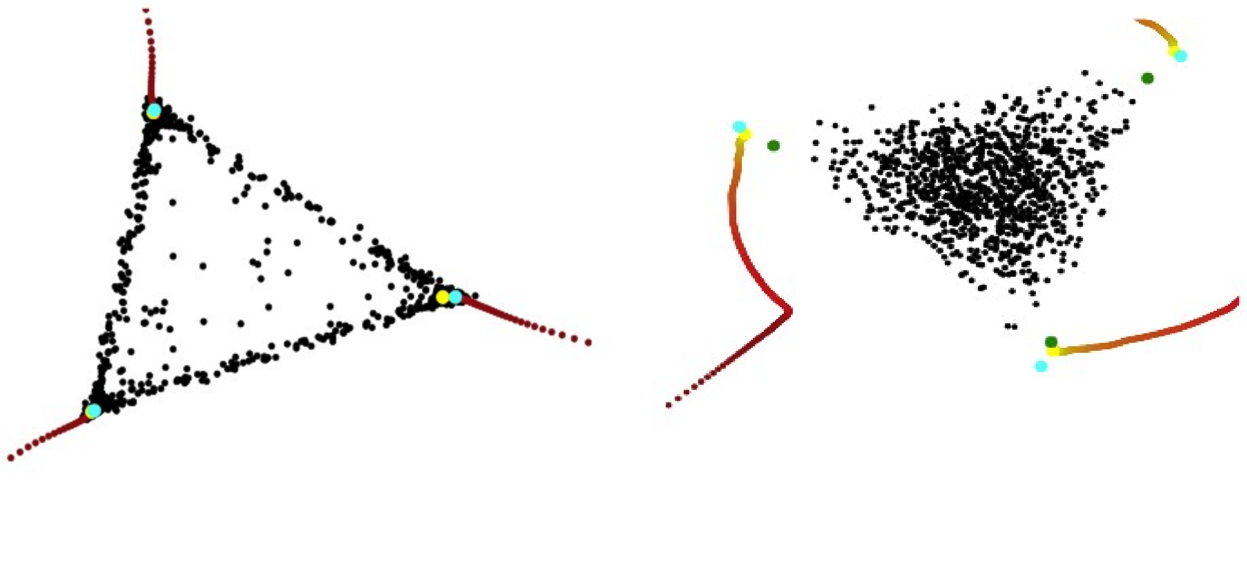


Figure 4a: Demonstration of Minimum Volume Topic Modeling (MVTM). The observed documents are the black dots, the optimization path of MVTM is shown in the gradient (dark red = beginning, light red = end), and the final estimate is in yellow. The ground-truth topic vertices are plotted in cyan. Experimental parametrization: the Dirichlet parameter for the topic proportion was set at a=0.1, and MVTM was initialized at the identity matrix.

Figure 4b: Demonstration for the Dirichlet parameter of the topic proportion: a=3. The final estimate of the compared method 'GDM' (Geometric Dirichlet Mean) is plotted in green for comparison. MVTM was initialized at the identity matrix.

In Jahnichen et al. [5], the researchers analyzed the fact that topics are subject to modification as time passes and provide a scalable approach involving dynamic models. While previous explorative techniques presume that topics are static, the current work is motivated by the following high level idea: in the case of training topic models on historical articles, let us focus on the extracted topic 'technology'. Confining the corpus to articles of the early years of the 20th century, one may come across words such as fire, ignition, engine, electricity to be most frequently linked to this topic. However, for modern articles the terms machine learning, cell phones, computers, laptops, cars and gates may be found among the top words.

In similar instances, the topic users need to be able to correlate documents with similar topic proportions over extensive time intervals. In addition to this, they need to allow topics to dynamically update their vocabulary. This is achieved in dynamic topic models (DTMs).

The paper introduces a two-fold generalization of dynamic topic models: it extends the class of tractable priors from previously used stochastic processes to the more comprehensive class of Gaussian processes using variational inference. This way it permits the capture of topics that have a long-term memory as time passes or present temporal concentration. Moreover, it develops an approximate Bayesian inference routine based on inducing points, variational inference and sparse processes. This allows to scale the model up to large text collections, a need which is prevalent in nowadays' ample collections of data.

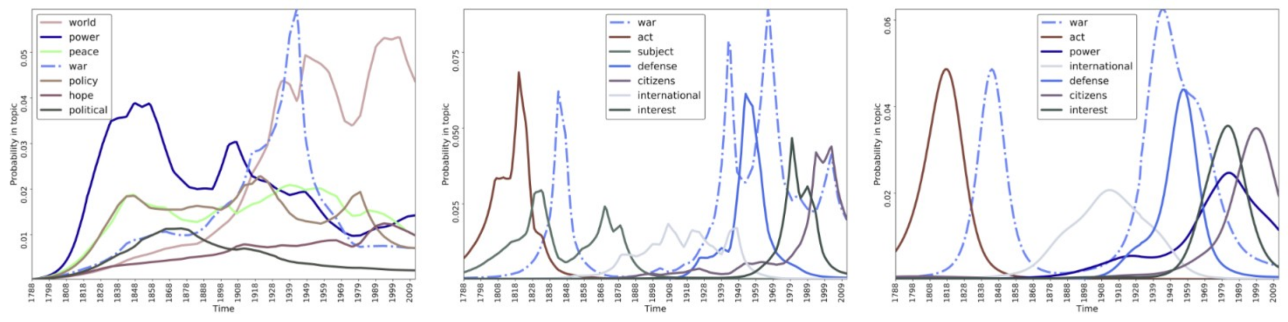


Figure 5: Learned word trajectories of the "war" topic using the Wiener (left), Ornstein-Uhlenbeck processes (middle) and Cauchy processes (right). The Cauchy kernel provides smoother trajectories yet the OU kernel is able to provide a better resolution in time.

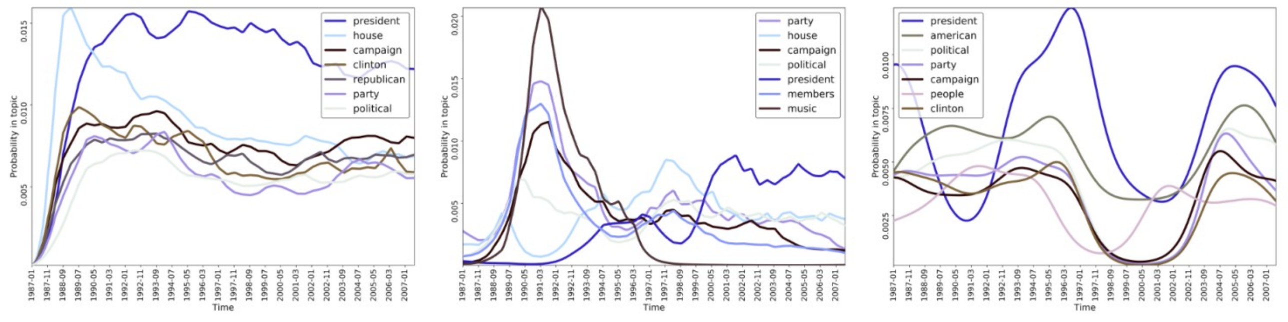


Figure 6: Learned word trajectories of the "election campaign" topic using the Wiener (left), Ornstein-Uhlenbeck processes (middle) and Cauchy processes (right), which results in the smoothest curves.

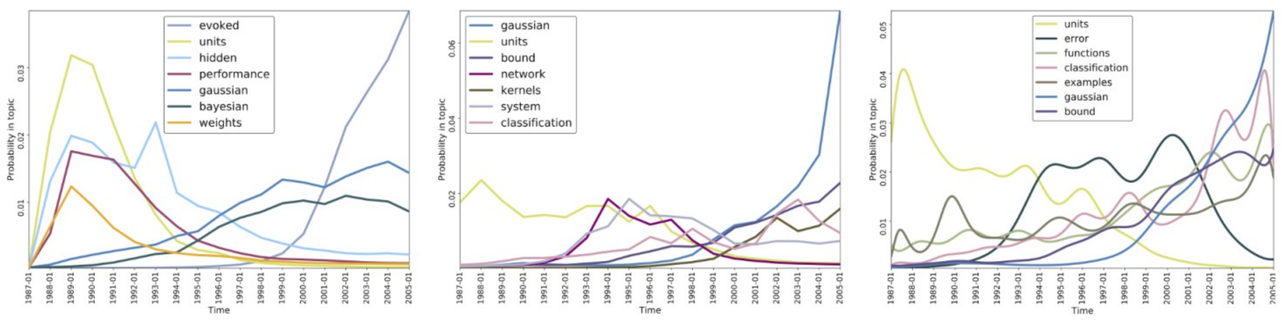


Figure 7:Learned word trajectories of the "function approximation" topic using the Wiener kernel (left), OU kernel (middle) and Cauchy kernel (right). All three approaches identify terms that gain or lose importance within the topic overtime.

The work in Hughes et al. [6] relies on the employment of supervisory signals in order to build a system that succeeds in two goals: it produces low-dimensional vectors of the data and performs predictions based on them, which are competitive to other methods. The authors introduce a framework for semi-supervised training which uses prediction-constrained latent variables as objective that trains the generative model.

Previous suggested systems in supervised topic modelling have been specialized in training with data and then predicting labels. Nevertheless, they prove insufficient in administering the inverse operation: using labels to predict data. The presented objective function incorporates supervisory signals also in the case when a percentage of the training examples is labeled.

The current topic model displays improved accuracy in the prediction task over past models, with applications in text analysis and electronic health records analysis while learning plausible topics. In the particular case of semi-supervised classification in healthcare and movie/yelp reviews, empirical evidence assures that the accuracy is impressive in small percentages of labeled data versus the standard training of supervised Latent Dirichlet allocation models.

In Huang et al. [7] the authors suggest a novel unsupervised method which leverages the emission data of a hidden Markov model (HMM) in order to determine the transition and emission probabilities. In the ordinary case of identifying large text records, the use Expectation-maximization rule (EM) presents high complexity restricting its applicability in learning the HMM. In contrast to EM, the current novel method does not grow with length of the observation sequence and thus it is scalable in large datasets.

Moreover, it can be used when the sample size is sufficient to estimate second-order output probabilities, but not higher-order ones. It is proven that if one is only able to obtain a reliable estimate of the emission probabilities that pertain to pairwise co-occurrence, it is still possible to uniquely recover the latent variables of the HMM under certain scattering conditions. Therefore, viewing the documents as realizations of HMMs (with common output probability), the coherence of the learned topics outranks the one from the algorithms which regard the documents as bag-of-words models. This is exhibited in the comparative results of the experiments with the employment of this theory.

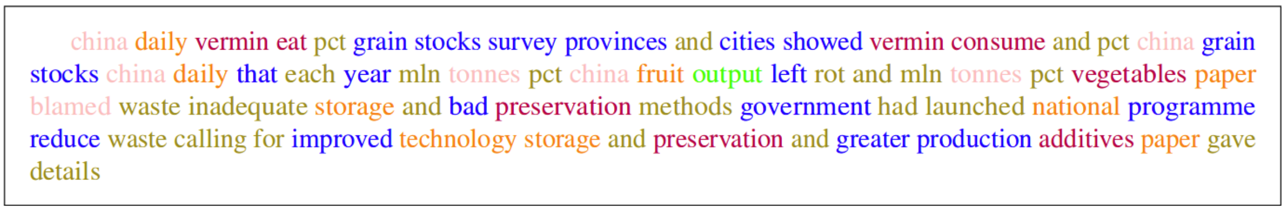

china daily vermin eat pct grain stocks survey provinces and cities showed vermin consume and pct china grain stocks china daily that each year mln tonnes pct china fruit output left rot and mln tonnes pct vegetables paper blamed waste inadequate storage and bad preservation methods government had launched national programme reduce waste calling for improved technology storage and preservation and greater production additives paper gave details

Figure 8: Inferred topics of the words shown in different colors given by probabilistic latent semantic analysis (pLSA). The dataset is Reuters-21578 [38].

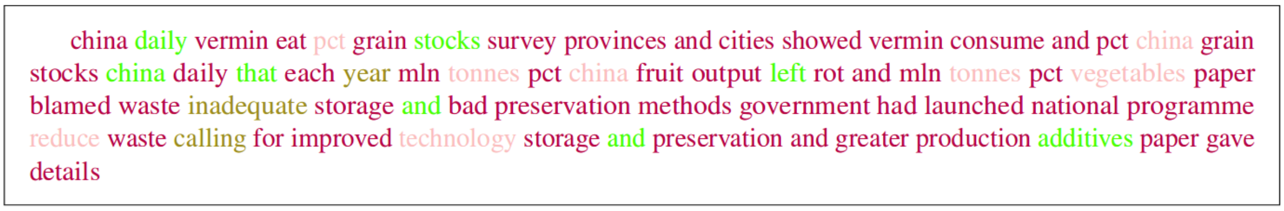

china daily vermin eat pct grain stocks survey provinces and cities showed vermin consume and pct china grain stocks china daily that each year mln tonnes pct china fruit output left rot and mln tonnes pct vegetables paper blamed waste inadequate storage and bad preservation methods government had launched national programme reduce waste calling for improved technology storage and preservation and greater production additives paper gave details

Figure 9: Topic inference given by Hidden Topics from Markov Models (HTMM). The extracted topics seems more concise and coherent, which is aligned with human comprehension

For the experiments, the raw document from was used to construct the word co-occurrence statistics, as well as bag-of-words representations for each document for the baseline algorithms. The dataset 'Reuters-21578'[1] was cleaned from the stopwords, an operation that eliminates any syntactic dependencies from the HTMM model, leaving only semantic dependencies. Its vocabulary size is around 200,000, making any method relying on triple-occurrences impossible to be applied on it. Hence, tensor-based methods are not compared with the current experiments. *Figure 10* displays the evaluation of the experiments: higher coherence means more meaningful topics (see definition in [8]). This study shows that for different number of topics on the entire dataset, HTMM coherence prevails over the rest of the algorithms.

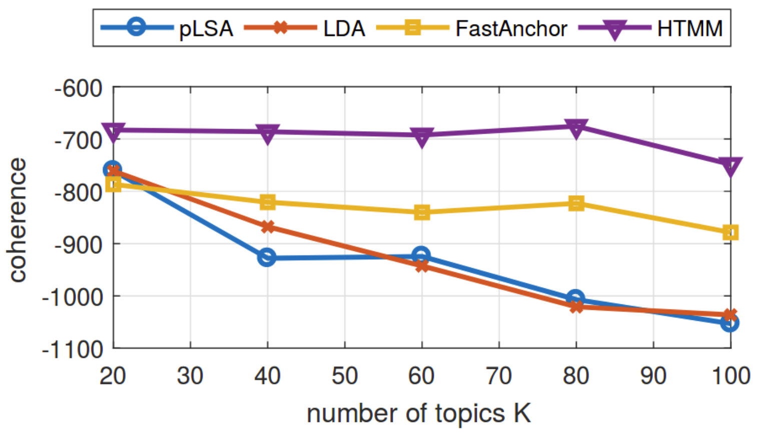


Figure 10: Comparative evaluation of the coherence of the topics. pLSA: Probabilistic latent semantic analysis, LDA: Latent Dirichlet Allocation, HTMM: Hidden Topics from Markov Models

The paper Zhao et al. [9] proposes 'Word Embeddings Deep Topic Model' (WEDTM), a novel topic model that exploits word embeddings to determine intertopic structures with topic hierarchies and intratopic structures (within every topic) with subcategories (sub-topics).

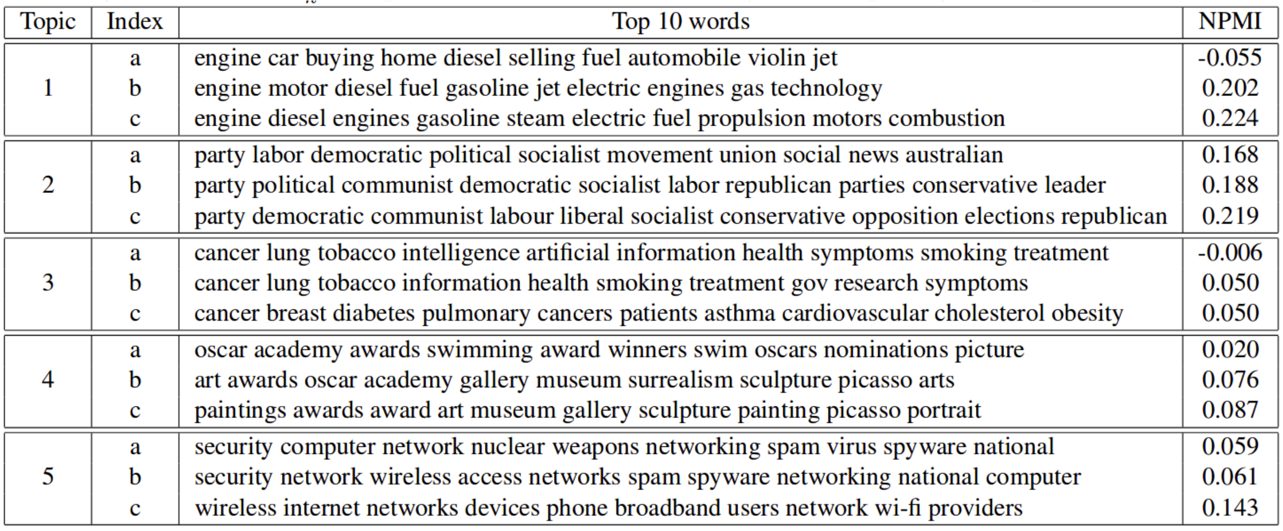
1 38. Mimaroglu, Selim.Some text datasets, 2007. url: https://www.cs.umb.edu/~smimarog/textmining/datasets/

In previous models, topics are detected locally from the word co-occurrences in a corpus; this kind of topics is defined as 'local'. The current work also focuses on the fine-grained thematic structure within each single topic. The restriction of the context of a target corpus is likely to lead in local topics difficult to comprehend and the reasons are that:

1. there is the possibility that the terms co-appearing locally in the target corpus are combined despite being semantically uncorrelated
2. local topics may be prevailed by specialized terms, which require either expertise to become explainable or even the conception of the global semantics of the terms.

Specifically, by applying subtopic embeddings, each subtopic can be enriched with the overall characteristics of word embeddings. This supplementary information faces the sparsity problem in topic models and leads to unveiling a possible fine-grained characteristic of a local topic. With topic embeddings, WEDTM provides different views to a topic, from global to local, which further improves the intelligibility of the model.

Extensive experiments (where the inference tasks we carried out by Gibbs sampling) have shown that WEDTM outmatches previous algorithms in perplexity, document classification and topic coherence.



Topic	Index	Top 10 words	NPMI
1	a	engine car buying home diesel selling fuel automobile violin jet	-0.055
	b	engine motor diesel fuel gasoline jet electric engines gas technology	0.202
	c	engine diesel engines gasoline steam electric fuel propulsion motors combustion	0.224
2	a	party labor democratic political socialist movement union social news australian	0.168
	b	party political communist democratic socialist labor republican parties conservative leader	0.188
	c	party democratic communist labour liberal socialist conservative opposition elections republican	0.219
3	a	cancer lung tobacco intelligence artificial information health symptoms smoking treatment	-0.006
	b	cancer lung tobacco information health smoking treatment gov research symptoms	0.050
	c	cancer breast diabetes pulmonary cancers patients asthma cardiovascular cholesterol obesity	0.050
4	a	oscar academy awards swimming award winners swim oscars nominations picture	0.020
	b	art awards oscar academy gallery museum surrealism sculpture picasso arts	0.076
	c	paintings awards award art museum gallery sculpture painting picasso portrait	0.087
5	a	security computer network nuclear weapons networking spam virus spyware national	0.059
	b	security network wireless access networks spam spyware networking national computer	0.061
	c	wireless internet networks devices phone broadband users network wi-fi providers	0.143

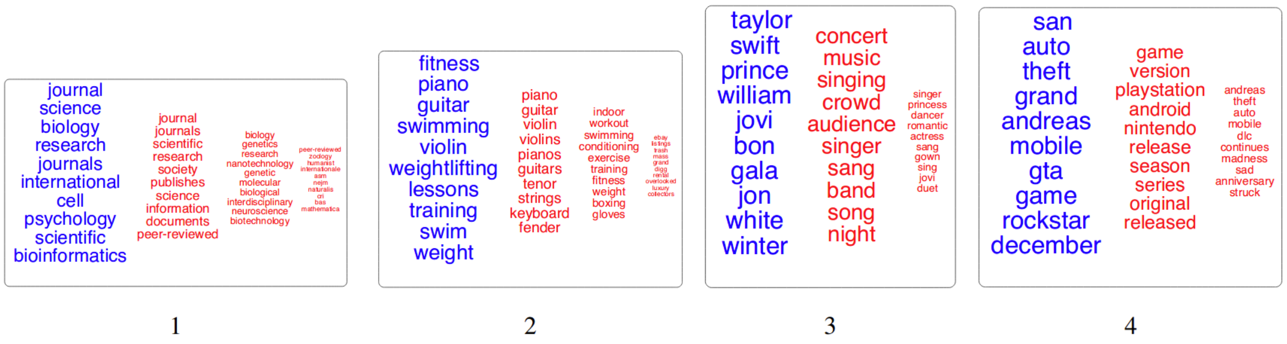
Figure 11: Top 10 words of five sets of example topics on the Web Snippets dataset (contains 12,237 web search snippets with 8 categories). Each set contains the top words of 3 topics: topic 'a' is generated by φ_{k1} in Gamma Belief Networks-3; topic 'b' is generated by φ_{k1} in WEDTM-3; topic 'c' is generated by $e^{F \cdot w(k1)}$ in WEDTM-3 ([17]). Topics 'a' and 'b' are matched by the Hellinger distance of $\varphi_{k1}^{(1)}$. Topic 'b' and 'c' are different ways of interpreting one topic in WEDTM.

Consequently, with topic hierarchies, subtopics and topic embeddings, 'WEDTM' can provide more decipherable topics, offering a concise summarization of the observed documents to the user and sets the base for the designing of scalable algorithms. even in large datasets.

The introduced model considers each document as a word count vector $x^{(1)}_j \in N^V_0$, where V is the size of the vocabulary; the pre-trained L-dimensional real-valued embeddings for each word $\upsilon \in \{1, \ldots, V\}$ are stored in a L-dimensional vector $f \in R^L$. The system WEDTM is comprised by T hidden layers, where the t-th layer contains K_t topics and k_t is the index of every topic.

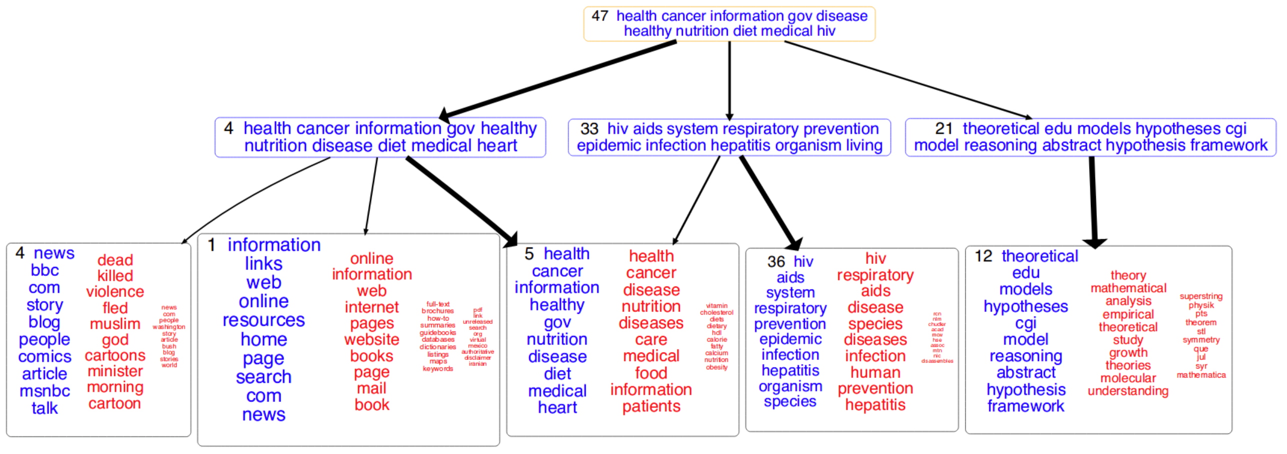
WEDTM models the word counts $x^{(1)}_j$ in a document by a Poisson (Pois) distribution and factorizes the Poisson parameters into a product of the factor loadings $\Phi^{(1)} \in R_+^{V\times K}{}_1$ and hidden units $\theta_j^{(1)}$. $\theta_j^{(1)}$ is the first-layer latent representation (unnormalized topic weights) of document j, each element of which is drawn from a gamma (Gam) distribution. The k_1-th column of $\Phi^{(1)}$, $\phi_{k1}^{(1)} \in R_+^V$, is the word distribution of topic k_1, drawn from a Dirichlet (Dir) distribution. The components for discovering inter topic hierarchies are similar to the structure of Gamma Belief Networks [10].

In Figure 13 there are the top 10 words of five sets of example topics on the Web Snippets dataset (it contains 12,237 web search snippets with 8 categories). Each set contains the top words of 3 topics: topic 'a' is generated by ϕ_{k1} in Gamma Belief Networks-3; topic 'b' is generated by ϕ_{k1} in WEDTM-3; topic 'c' is generated by $e^{F\cdot w(k1)}$ in WEDTM-3 [9]. Also, topics 'a' and 'b' are matched by the Hellinger distance of $\phi_{k1}^{(1)}$. In particular, topic 'b' and 'c' are different ways of interpreting one topic in WEDTM.



journal science biology research journals international cell psychology scientific bioinformatics
journal journals scientific research society publishes science information documents peer-reviewed
biology genetics research nanotechnology genetic molecular biological interdisciplinary neuroscience biotechnology
fitness piano guitar swimming violin weightlifting lessons training swim weight
piano guitar violin violins pianos guitars tenor strings keyboard fender
indoor workout swimming conditioning exercise training fitness weight boxing gloves
taylor swift prince william jovi bon gala jon white winter
concert music singing crowd audience singer sang band song night
singer princess dancer romantic actress sang gown sing jovi duet
san auto theft grand andreas mobile gta game rockstar december
game version playstation android nintendo release season series original released
andreas theft auto mobile dlc continues madness sad anniversary struck
1
2
3
4

Figure 12: The sub-topics (red) of the example topics (blue). Larger font size indicates larger weight of a sub-topic to the local topic. The sub-topics with extreme small weights are trimmed off.



47 health cancer information gov disease healthy nutrition diet medical hiv
4 health cancer information gov healthy nutrition disease diet medical heart
33 hiv aids system respiratory prevention epidemic infection hepatitis organism living
21 theoretical edu models hypotheses cgi model reasoning abstract hypothesis framework
4 news bbc com story blog people comics article msnbc talk
dead killed violence fled muslim god cartoons minister morning cartoon
1 information links web online resources home page search com news
online information web internet pages website books page mail book
5 health cancer information healthy gov nutrition disease diet medical heart
health cancer disease nutrition diseases care medical food information patients
36 hiv aids system respiratory prevention epidemic infection hepatitis organism species
hiv respiratory aids disease species diseases infection human prevention hepatitis
12 theoretical edu models hypotheses cgi model reasoning abstract hypothesis framework
theory mathematical analysis empirical theoretical study growth theories molecular understanding

Figure 13: One example sub-tree of the topic hierarchy discovered by WEDTM on the WS dataset with K_1=50. The tree is generated in the same way to Gamma Belief Networks (GBN, [10]). A line from node k_t at layer t to node k_{t-1} at layer t−1 indicates that $\varphi^{(1)}_{k\_(t-1)\text{-}kt} > 1.5/K_{t-1}$ and its width indicates the value of $\varphi^{(1)}_{k\_(t-1)\text{-}k\_t}$ (i.e. topic correlation strength). The outside border of the text box is colored as orange, blue, or black if the node is at layer three, two, or one, respectively. For the leaf nodes, sub-topics are shown in the same way as Figure 11a.

The work Esmaeili et al. [11] presents the 'Variational Aspect-based Latent Topic Allocation' (VALTA), a family of autoencoding topic models that learn aspect-based representations of reviews. VALTA defines a user-item encoder that maps bag-of-words vectors for combined reviews (associated with each paired user and item) onto structured embeddings, which in turn define per-aspect topic weights. The authors model individual reviews in a structured manner by inferring an aspect assignment for each sentence in a given review, where the per-aspect topic weights obtained by the user-item encoder serve to define a mixture over topics, conditioned on the aspect. The result is an auto-encoding neural topic model for reviews, which can be trained in a fully unsupervised manner to learn topics that are structured into aspects. Experimental evaluation on large number of datasets demonstrates that aspects are intelligible, yield higher coherence scores than non-structured autoencoding topic model versions and can be employed to perform aspect-based comparison and genre discovery.

In Rodrigues et al. [12] the researchers rely on the observation that, since documents are frequently associated with other related variables, such as labels or ratings, a supervised topic model approach can be applied. However, the nature of most annotation tasks, prone to ambiguity and noise, often with high volumes of documents, deem learning under a single-annotator assumption unrealistic or impractical for most real-world applications. In the current approach, two supervised topic models are proposed, one for classification and another for regression problems, which account for the heterogeneity and biases among different annotators that are encountered in practice when learning from crowds. n efficient stochastic variational inference algorithm is developed, which is able to scale to very large datasets. They empirical demonstration the advantages of the proposed model over state-of-the-art approaches provides considerable evaluation results and allows for the application of this variant of topic model in real-world scenarios.

3.1.2. Methodology

Topic model refers to a category of probabilistic models for uncovering the underlying semantic structure of a document collection and help us organize and browse large these collections [13]. These probabilistic models are based on a hierarchical Bayesian analysis of the original text. A topic is considered as a subset of the words appearing in a document (*principal words* or *keywords*) which characterize the abstract topic which pertains to the document. In other words, a topic can be viewed as a set for similar words or a distribution over a fixed vocabulary of terms. By inspecting these keywords, it is identifiable what the topic concerns. It is expected that a document can be assigned in several topics with different membership percentages.

The topic model formalizes this concept using statistical methods to process a number of documents in natural language (i.e. data in an unstructured form) and the contained words, then extracts a separate list of topics from all of them and finally provides the distribution of topics for each document.

In Fandango we are using the Gensim [2] python library for topic extraction. Gensim is a statistical tool that uses Latent Dirichlet Allocation (LDA) [13] under the hood to produce a user-defined number of topics. LDA is an unsupervised generative model and a generalization of probabilistic latent semantic analysis: it assumes that each document is a mixture of (latent) topics with a weight assigned to each topic, thus forming a topic probability distribution and all documents share a common Dirichlet prior. Each latent topic in the LDA model is also represented as a probabilistic distribution over words and the word distributions of topics share a common Dirichlet prior as well.

Before the raw text can be used for topic modeling, a number of preprocessing steps have to be conducted. Initially, a tokenization step has to be performed. Tokenization refers to the process of chopping the raw text input to a list of words. In a process like topic modeling, different forms of the same word do not contain different information. Therefore, a lemmatization step follows, that extracts the lemma of each word. A lemma is the root of a word, for example the lemma of “taxes” is “tax” and the lemma of “said” is “say”. This process affects the statistical distribution of tokens positively and leads to better topic creation. A third step is the removal of words that appear very often in a language, which have no statistical value in differentiating between topics; this step is called stopword removal. Finally, words that appear very rarely are also removed, since the appearance of a word 2 or 3 times in a corpus of millions of documents cannot constitute a topic. After the preprocessing steps, a bag of words vector is created to describe each document.

We trained a separate model for each language used in the Fandango platform (English, Spanish, French and Dutch). For each language the training corpus was built from the articles that were gathered in the crawling process. For each model the same set of parameters was used, choosing the desired number of topics to be 50 and the number of passes over the dataset to be 4. The remaining available parameters were set to the default value. Before deciding on the specific parameters a set of experiments was conducted with different number of topics and passes over the dataset, evaluating the produced topics. The standard measures which help evaluate the quality of the extracted topics are the extent to which the topics cover all articles, the need for the topics to be semantically meaningful and whether the topics are semantically and syntactically disjoint. For a more detailed description of the available parameters, the reader can refer to the Gensim library documentation for python.

The provided topics were afterwards annotated with a meaningful label, summarizing their context. As a topic has probabilities of generating various words, such as politicians, Brexit, European, market, stock, capitals these can be classified and interpreted by the viewer as ‘politics’ or ‘finance’ ‘. Naturally, the word ‘politics’ itself would have high probability given this topic. A lexical word may occur in several topics with a different probability, but with a different typical set of neighboring words in each topic. This concurrent appearance of terms in large collections of documents is crucial for the efficiency of topic models as is the estimate of pairwise co-occurrence probabilities [7].

2 Topic model using Gensim: documentation, url: https://radimrehurek.com/gensim/index.html

3.2. Named Entity Recognition

One of the most common practices in Natural Language Processing (NLP) in data science is the ‘Named Entity Recognition’ (NER, also referenced as ‘Entity Extraction’ or ‘Entity Chunking’) which recognizes named units in a text. These can include names of locations, persons, dates, organizations, goods etc or percentages, dates, currency quantities etc, either physical or intangible, and in general anything that can be named and categorized into preset classes. The goal of this task is to answer questions like:

- Were particular cities cited in the news posts? Are they related to the accompanying images?
- Which names of journalists were cited most frequently in the fake articles?
- Were particular organizations noted together with locations in several articles?

3.2.1. State of the Art

Among the state-of-the art algorithms is Zhang et al. [14] where the problem targeted is that of discovering entity occurrences that follow or mimic patterns identifiable by regular expressions (REs). In contrast to the majority of the previous work on this topic, the focus is not on learning/inferring REs that detect entities with the best accuracy. In particular, the authors aim to show that deep learning can exploit less accurate REs and achieve very high performance, keeping the human intervention to the lowest possible.

The authors state that it is a valid assertion that no RE can be constructed that is capable of retrieving entities from web documents with high precision and recall. Rather than abandoning REs as a go-to approach for entity detection, the current work explores ways to combine the expressive power of REs, ability of deep learning to learn from large data, and human-in-the loop approach into a new integrated framework for entity identification from web data. The framework starts by creating or collecting the existing REs for a particular type of an entity. Those REs are then used over a large document corpus to collect weak labels for the entity mentions and a neural network is trained to predict those RE-generated weak labels. Finally, a human expert is asked to label a small set of documents and the neural network is fine tuned on those documents. Figure 14 portrays details of the pipeline.

The formalization of the problem is the following: given a text string t and an entity type E, the task is to predict whether t contains an entity mention of type E. This task is treated as binary classification. To build the classifier, it is assumed that a large corpus of unlabeled text strings is given: $T = \{t_1, t_2, \ldots, t_n\}$. The challenge is to train the classifier with minimal human effort. A human expert is allowed to contribute in two ways:

1. construct a new RE or find a RE created by others, and
2. label an unlabeled string.

More specifically, a subset of m strings from T are sampled randomly and a human annotator labels each of the sampled strings. String t_i is labeled as $y_i= 1$ if the annotator recognizes an entity type E in t_i and as y_i= 0, if not. The resulting strongly labeled data set is denoted as Dhuman={(t_i, y_i)|i= 1,2, ..., m}, where m << n.

The scores are displayed in Figure 15. The experimental evaluation on several entity identification problems shows that the proposed framework achieves impressive accuracy, while requiring very modest human effort.

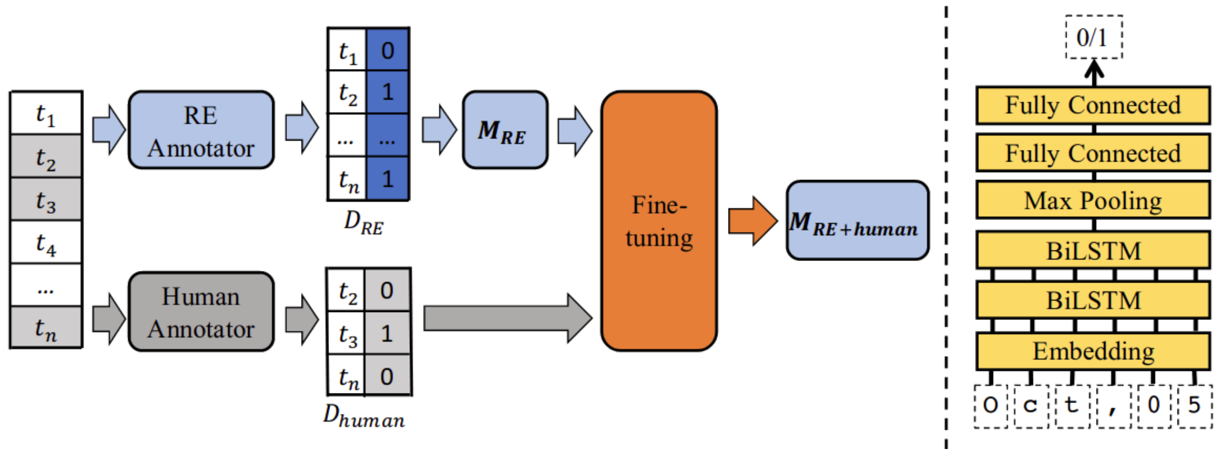


Figure 14: Overview of the solution (left) and deep learning architecture (right) used to train the current model MRE.

Model Name	Date Time (%)				Course Number (%)			
	ACC	F1	Precision	Recall	ACC	F1	Precision	Recall
Naive	77.34	0.00	0.00	0.00	68.47	0.00	0.00	0.00
RE	90.65	76.77	88.00	68.33	71.64	59.91	54.01	67.28
M_{RE}	91.45	79.09	**89.01**	71.39	72.64	61.52	55.21	69.47
M_{human}	74.78	40.75	37.25	46.38	62.81	30.74	37.63	28.45
$M_{RE+human}$	**93.50**	**85.44**	87.03	**84.95**	**82.86**	**74.31**	**72.45**	**77.01**
	Bill Date (%)				Email Address (%)			
Naive	92.83	0.00	0.00	0.00	88.75	0.00	0.00	0.00
RE	**94.56**	38.23	**97.92**	24.01	**98.72**	**94.61**	**89.79**	**100.00**
M_{RE}	94.53	38.10	96.30	23.96	98.64	94.28	89.19	**100.00**
M_{human}	92.56	2.15	6.25	1.30	83.44	42.30	35.99	53.03
$M_{RE+human}$	94.36	**39.95**	87.88	**27.59**	97.75	90.95	83.60	**100.00**

Figure 15: The comparisons in the presence of very few human annotations:|Dhuman|= 20

In Xie et al. [15] the authors address the problem of bilingual embeddings for the NER model: they attempt to tackle the scenario of unsupervised transfer of NER models from resource-rich languages, where no labeled data is available in a target language with few or no labels. The challenges in this investigation are obvious: firstly discrepancies in the vocabulary and, secondly, term sequencing between the two languages prevent an effective mapping of the NER model.

The success of previous methods is based on a reasonably large amount of annotated training data. In the case of limited amounts of labeled data, cross-lingual NER tries to address this by transferring knowledge of the trained models to a new language.

The introduced method consists of the next stages:

1. *Learning Monolingual Embeddings*: Train separate word embeddings on mono-lingual corpora (training data) by standard embedding training methods.

2. *Learning Bilingual Embeddings*: Project word embeddings of the two languages into a shared embedding space. For this, optimize the word embedding alignment using the given dictionary.

3. *Learning Word Translations*: For each word in the source language training data, translate it by finding its nearest neighbor in the shared embedding space.

4. *Training the NER Model*: Train a NER model using the translated words along with the named entity tags from the English corpus.

Figure 16 displays the architecture of the pipeline which uses components from [16], such as the Bidirectional Long Short-term memory neural network with a sequential conditional random layer (CRF) above it.

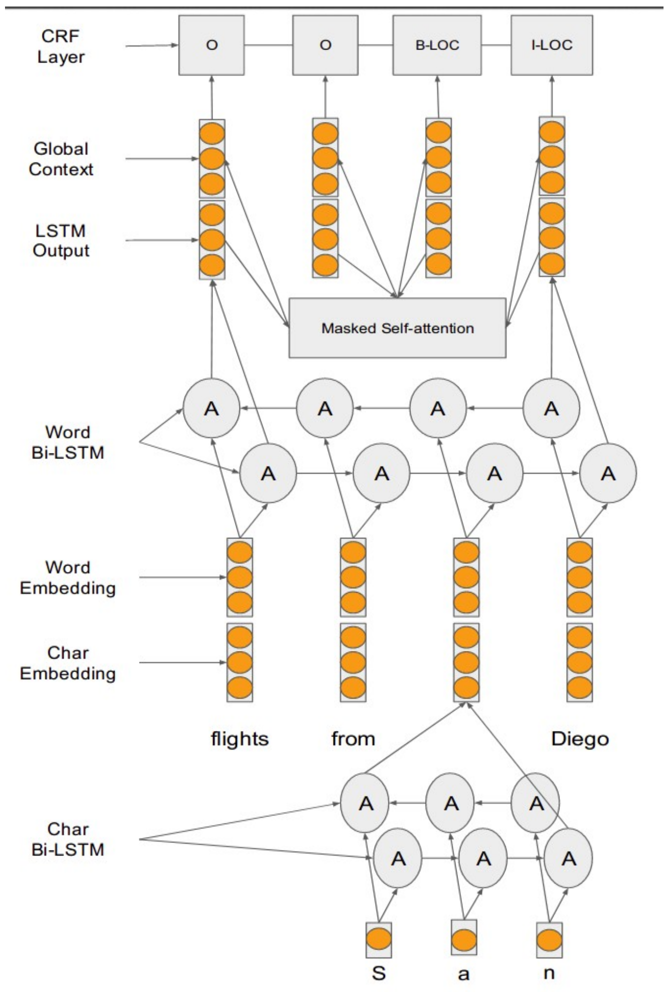
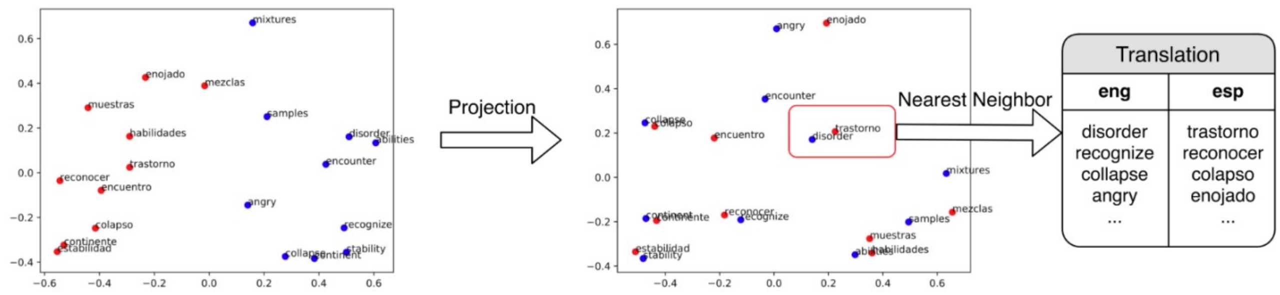


Figure 16: Architecture of the BiLSTM model



eng	esp
disorder	trastorno
recognize	reconocer
collapse	colapso
angry	enojado
...	...

Figure 17: Example of the result of model application on Spanish-English words not included in the dictionary (embeddings are reduced to 2 dimensions for visual clarity)

Figure 17 shows an example of this system applied in NER model-transfer from English to Spanish. As illustrated, the word embeddings of a word pair present a divergence in different languages, which is expected as different languages have discrete characteristics and dissimilar mono-lingual data. Consequently it is inherently difficult to learn a flawless alignment. This indicates that models trained directly on data using the source embeddings may not generalize well to the slightly disparate embeddings of the target language.

The authors also evaluate the challenges of applying these methods to Uyghur, a low-resource language and present the considerable performance of the model.

In Shang et al. [17] the goal of the paper is to learn a named entity tagger using dictionaries. Each dictionary entry consists of 1) the surface names of the entity, including a canonical name and a list of synonyms; and 2) the entity type. Considering the limited coverage of dictionaries, the authors extend existing dictionaries by adding high-quality phrases as potential entities with unknown type.

An outline of the introduced idea is that, given a raw corpus and a dictionary, entity labels are firstly generated (including unknown labels) by exact string matching, where conflicted matches are resolved by maximizing the total number of matched tokens. Based on the result of dictionary matching, each token falls into one of three categories: 1) it belongs to an entity mention with one or more known types; 2) it belongs to an entity mention with unknown type; and 3) It is marked as non-entity. Accordingly, the paper designs and explores two neural models, Fuzzy-LSTM-CRF and AutoNER with the 'Tie or Break' scheme, to learn named entity taggers based on labels with unknown and multiple types.

The outcomes of the current study are:

- The novel neural model AutoNER with the new 'Tie or Break' scheme for the distantly supervised NER task.
- The traditional NER model is revised to the Fuzzy-LSTM-CRF model, which serves as a strong distantly supervised baseline.
- Distant supervision is refined for better NER performance, such as incorporating high quality phrases to reduce false-negative labels and experiments are conducted to verify the effectiveness.

Finally experiments on three benchmark datasets demonstrate that AutoNER achieves the best performance when using dictionaries with no additional human effort and is even competitive with the supervised benchmarks.

3.2.2. METHODOLOGY

3.2.2.1. EXTRACTION OF ENTITIES

Named Entity Recognition (NER) is the task of locating, extracting and classifying contiguous parts of strings, which refer to specific entities such as monetary values, people, organizations, locations, companies, expressions of time and emails and belongs to the family of problems of Information Extraction or Information Mining from text. The main characteristic of named entities is that they often follow an underlying syntactic pattern [14].

For the purpose of developing a Named Entity Recognition system, we employ the Spacy python library. The Spacy library is developed as a neural network: it contains several convolutional layers with residual connections where layer normalization and maxout non-linearity are used, giving much better efficiency than the standard BiLSTM [18] systems. It was trained in a supervised way for each language: it is shown each unlabelled example (text) and it performs a prediction of the entities. Given feedback in the form of an error gradient of the loss function that calculates the difference between the predicted label and the expected output, it updates its parametres. The higher the difference, the higher the gradient and the updates to this network's weights. During this process it uses linguistic grammar-based techniques. As a statistical model, it requires a large amount of manually annotated training data.

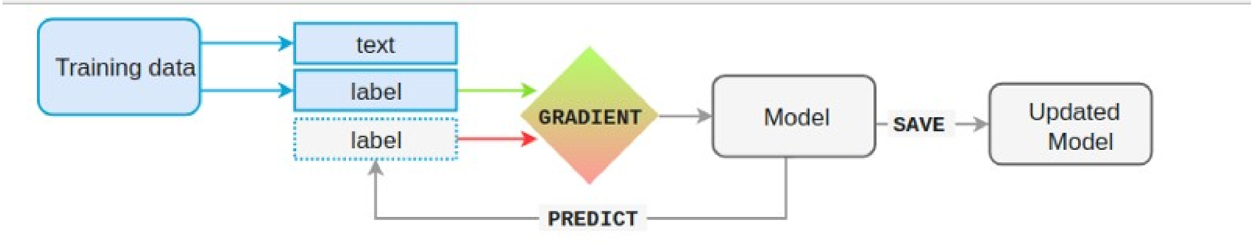


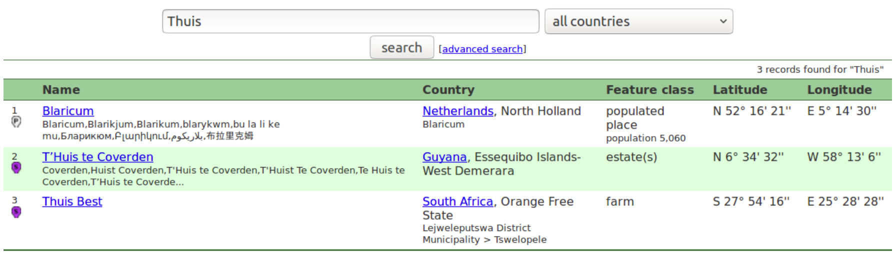
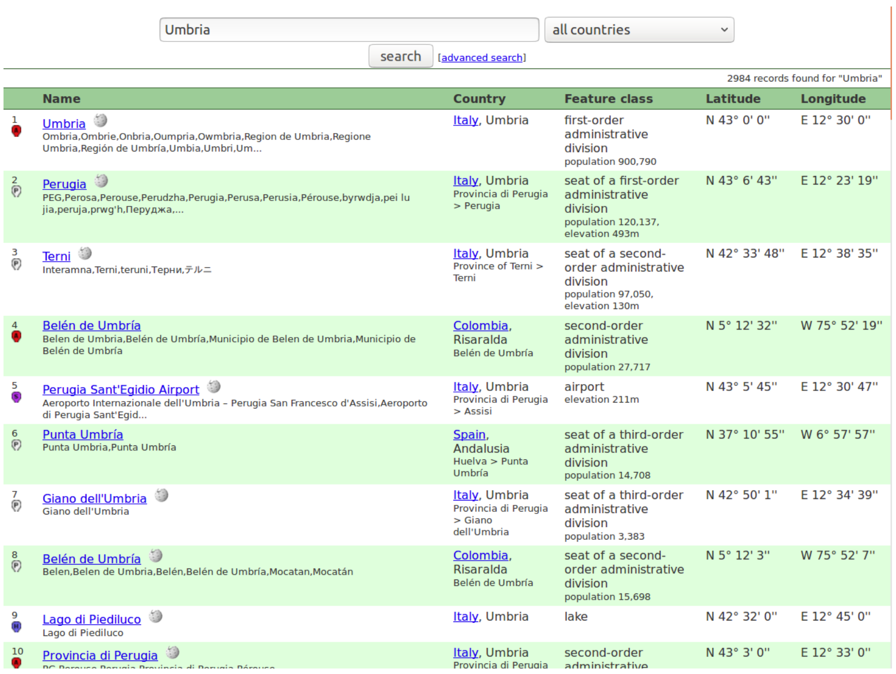
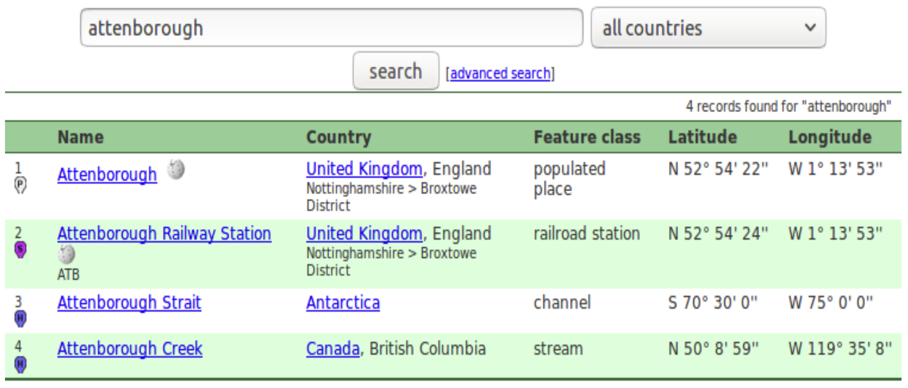
Figure 18: Spacy library architecture

The Spacy library provides trained NER models in multiple languages, including English, Spanish, Italian and Dutch, which are the languages used in the Fandango platform.

Firstly the language is determined and the relevant package is loaded. Then, providing the text input, every sentence is parsed, cleared from punctuation characters, broken into its words (tokens) and then converted into its associated root word (lemmatization, e.g. 'went' is converted to 'go', 'visited' to 'visit' etc).

3.2.2.2. *FOCUS ON SPATIAL INFORMATION*

The entities that pertain to geographic location are listed separately in the file 'Geographical Entities' for each language, e.g. a part of the list of the English entities is in the first column of Table 5.

ABBREVIATION	DESCRIPTION
Thuis	
Umbria	
Attenborough	

Thuis | all countries

search [advanced search]

3 records found for "Thuis"

	Name	Country	Feature class	Latitude	Longitude
1	Blaricum Blaricum,Blarikjum,Blarikum,blarykwm,bu la li ke mu,Бларикюм,Բլարիկում,بلاريكوم,布拉里克姆	Netherlands, North Holland Blaricum	populated place population 5,060	N 52° 16' 21''	E 5° 14' 30''
2	T'Huis te Coverden Coverden,Huist Coverden,T'Huis te Coverden,T'Huist Te Coverden,Te Huis te Coverden,T'Huis te Coverde...	Guyana, Essequibo Islands-West Demerara	estate(s)	N 6° 34' 32''	W 58° 13' 6''
3	Thuis Best	South Africa, Orange Free State Lejweleputswa District Municipality > Tswelopele	farm	S 27° 54' 16''	E 25° 28' 28''

Umbria | all countries

search [advanced search]

2984 records found for "Umbria"

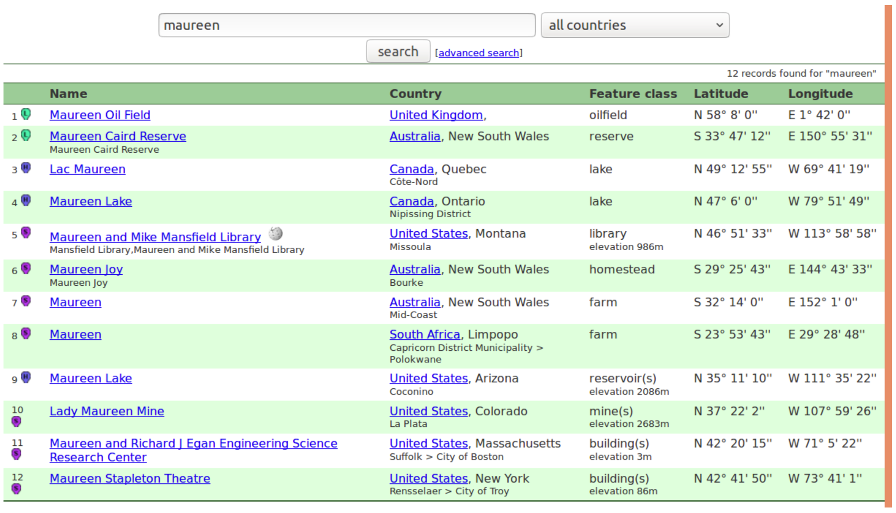
	Name	Country	Feature class	Latitude	Longitude
1	Umbria Ombria,Ombrie,Onbria,Oumpria,Owmbria,Region de Umbria,Regione Umbria,Región de Umbría,Umbia,Umbri,Um...	Italy, Umbria	first-order administrative division population 900,790	N 43° 0' 0''	E 12° 30' 0''
2	Perugia PEG,Perosa,Perouse,Perudzha,Perugia,Perusa,Perusia,Pérouse,byrwdja,pei lu jia,peruja,prwg'h,Перуджа,...	Italy, Umbria Provincia di Perugia > Perugia	seat of a first-order administrative division population 120,137, elevation 493m	N 43° 6' 43''	E 12° 23' 19''
3	Terni Interamna,Terni,teruni,Терни,テルニ	Italy, Umbria Province of Terni > Terni	seat of a second-order administrative division population 97,050, elevation 130m	N 42° 33' 48''	E 12° 38' 35''
4	Belén de Umbría Belen de Umbria,Belén de Umbría,Municipio de Belen de Umbria,Municipio de Belén de Umbría	Colombia, Risaralda Belén de Umbría	second-order administrative division population 27,717	N 5° 12' 32''	W 75° 52' 19''
5	Perugia Sant'Egidio Airport Aeroporto Internazionale dell'Umbria – Perugia San Francesco d'Assisi,Aeroporto di Perugia Sant'Egid...	Italy, Umbria Provincia di Perugia > Assisi	airport elevation 211m	N 43° 5' 45''	E 12° 30' 47''
6	Punta Umbría Punta Umbria,Punta Umbría	Spain, Andalusia Huelva > Punta Umbría	seat of a third-order administrative division population 14,708	N 37° 10' 55''	W 6° 57' 57''
7	Giano dell'Umbria Giano dell'Umbria	Italy, Umbria Provincia di Perugia > Giano dell'Umbria	seat of a third-order administrative division population 3,383	N 42° 50' 1''	E 12° 34' 39''
8	Belén de Umbría Belen,Belen de Umbria,Belén,Belén de Umbría,Mocatan,Mocatán	Colombia, Risaralda Belén de Umbría	seat of a second-order administrative division population 15,698	N 5° 12' 3''	W 75° 52' 7''
9	Lago di Piediluco Lago di Piediluco	Italy, Umbria	lake	N 42° 32' 0''	E 12° 45' 0''
10	Provincia di Perugia PG,Perouse,Perugia,Provincia di Perugia,Pérouse	Italy, Umbria Provincia di Perugia	second-order administrative	N 43° 3' 0''	E 12° 33' 0''

attenborough | all countries

search [advanced search]

4 records found for "attenborough"

	Name	Country	Feature class	Latitude	Longitude
1	Attenborough	United Kingdom, England Nottinghamshire > Broxtowe District	populated place	N 52° 54' 22''	W 1° 13' 53''
2	Attenborough Railway Station ATB	United Kingdom, England Nottinghamshire > Broxtowe District	railroad station	N 52° 54' 24''	W 1° 13' 53''
3	Attenborough Strait	Antarctica	channel	S 70° 30' 0''	W 75° 0' 0''
4	Attenborough Creek	Canada, British Columbia	stream	N 50° 8' 59''	W 119° 35' 8''

Maureen	

maureen | all countries

search [advanced search]

12 records found for "maureen"

	Name	Country	Feature class	Latitude	Longitude
1	Maureen Oil Field	United Kingdom,	oilfield	N 58° 8' 0''	E 1° 42' 0''
2	Maureen Caird Reserve Maureen Caird Reserve	Australia, New South Wales	reserve	S 33° 47' 12''	E 150° 55' 31''
3	Lac Maureen	Canada, Quebec Côte-Nord	lake	N 49° 12' 55''	W 69° 41' 19''
4	Maureen Lake	Canada, Ontario Nipissing District	lake	N 47° 6' 0''	W 79° 51' 49''
5	Maureen and Mike Mansfield Library Mansfield Library,Maureen and Mike Mansfield Library	United States, Montana Missoula	library elevation 986m	N 46° 51' 33''	W 113° 58' 58''
6	Maureen Joy Maureen Joy	Australia, New South Wales Bourke	homestead	S 29° 25' 43''	E 144° 43' 33''
7	Maureen	Australia, New South Wales Mid-Coast	farm	S 32° 14' 0''	E 152° 1' 0''
8	Maureen	South Africa, Limpopo Capricorn District Municipality > Polokwane	farm	S 23° 53' 43''	E 29° 28' 48''
9	Maureen Lake	United States, Arizona Coconino	reservoir(s) elevation 2086m	N 35° 11' 10''	W 111° 35' 22''
10	Lady Maureen Mine	United States, Colorado La Plata	mine(s) elevation 2683m	N 37° 22' 2''	W 107° 59' 26''
11	Maureen and Richard J Egan Engineering Science Research Center	United States, Massachusetts Suffolk > City of Boston	building(s) elevation 3m	N 42° 20' 15''	W 71° 5' 22''
12	Maureen Stapleton Theatre	United States, New York Rensselaer > City of Troy	building(s) elevation 86m	N 42° 41' 50''	W 73° 41' 1''

Table 1: Geolocation online tool

Each such entity can be sent as a query to the 'Geonames' database tool[3] which will return all existing locations with this name along with their geographical coordinates (Latitude, Longitude), an example of which is in the second column of Table 1.

3 GeoNames geographical database. url: https://www.geonames.org/advanced-search.html.

4. EXTRACTION OF VISION-BASED OBJECT MARKERS

One of the key modalities of the articles coming into the Fandango system are the visual content i.e. images and videos. Humans glance at an image and instantly know what objects are in the image, where they are, and how they interact. In order for an informed decision to be made for the validity of an article, a full understanding of the visual content is necessary so that it can be compared with the other modalities. In order to provide with such information in the Fandango system, we use state of the art deep learning approaches that detect, classify and localize objects in an image. This information is then passed on to later stages of the system in the form of markers i.e. information vectors.

4.1. INTRODUCTION

Object detection is a technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos. Semantic segmentation is an approach to the object detection problem where the goal is to assign a label (from a pool of possible labels) to each pixel, therefore producing a mask for each object, which is an even more difficult problem to solve. Taking it one step further, the problem of instance segmentation not only asks for a per pixel classification but also for different instances of each class to be distinguished.

Current object detection systems repurpose classifiers to perform detection. To detect an object, these systems take a classifier for that object and evaluate it at various locations and scales in a test image. Older systems, like deformable parts models (DPM) [19] use a sliding windows approach where evenly spaced patches are extracted from the image and passed through the classifier. More recent approaches (R-CNN) use region proposal networks to first detect potential areas where objects might appear [20]. Then the proposed areas are evaluated with the classifier for a final object detection. By incorporating the region proposal network inside the feature extraction model, Faster R-CNN [21] offers a much faster setup that manages the same performance. An even faster alternative, which unfortunately sacrifices some of the predictive performance, is the YOLO [22] (You Only Look Once) model. This speed boost is gained by unifying all the separate components into a single neural network. By replacing the top classification layers, of any of the above models, with an appropriate per pixel classification layer, one can modify the problem from object detection to image segmentation.

4.2. STATE OF THE ART

4.2.1. FULLY CONVOLUTIONAL INSTANCE-AWARE SEMANTIC SEGMENTATION (FCIS)

A fully convolutional approach for instance aware semantic segmentation is first proposed in *Li et. al.* [23] as a complete end-to-end model. In a fully convolutional network, a classifier is trained to predict each pixel's likelihood of belonging to some object category. This process is translation invariant and unaware of individual object instances. But a single score map per category is insufficient for solving the instance segmentation problem. In [24][25] the authors use k^2 position sensitive score maps that correspond to $k \times k$ evenly partitioned cells of objects. Each score map is a lower resolution representation of the likelihood a pixel belongs to some object instance at a relative position. For example the first map represents the likelihood of each pixel to belong in the top left position of an instance of some object. Consequently each pixel can take k^2 different values for each of the relative positions.

Li et. al. extend the approach and build upon this idea to perform the tasks of object detection and object segmentation jointly and simultaneously by sharing the same set of score maps.

Given a region-of-interest (ROI), its pixel-wise score maps are produced by the assembling operation within the ROI. For each pixel in a ROI, there are two tasks:

1. detection: whether it belongs to an object bounding box at a relative position or not,
2. segmentation: whether it is inside an object instance's boundary or not.

The solution adopted in the baseline model is to train two separate classifiers each using one task's supervision; in this case using a 1 × 1 convolution layer for each. The output, then, fuzes the two answers into two scores: inside and outside. There are three cases:

1. high inside score and low outside score,
2. low inside score and high outside score,
3. both scores are low.

The two scores answer the two questions jointly via softmax and max operations. For detection, they use a max operation to differentiate cases 1 and 2 from case 3. The detection score of the whole ROI is then obtained via average pooling over all pixels' likelihoods (followed by a softmax operator across all the categories). For segmentation, they use softmax to differentiate case 1 from 2 at each pixel. The foreground mask (in probabilities) of the ROI is the union of the per-pixel segmentation scores (for each category). Similarly, the two sets of scores are from two 1×1 convolution layer. The inside/outside classifiers are trained jointly as they receive the back-propagated gradients from both segmentation and detection loses.

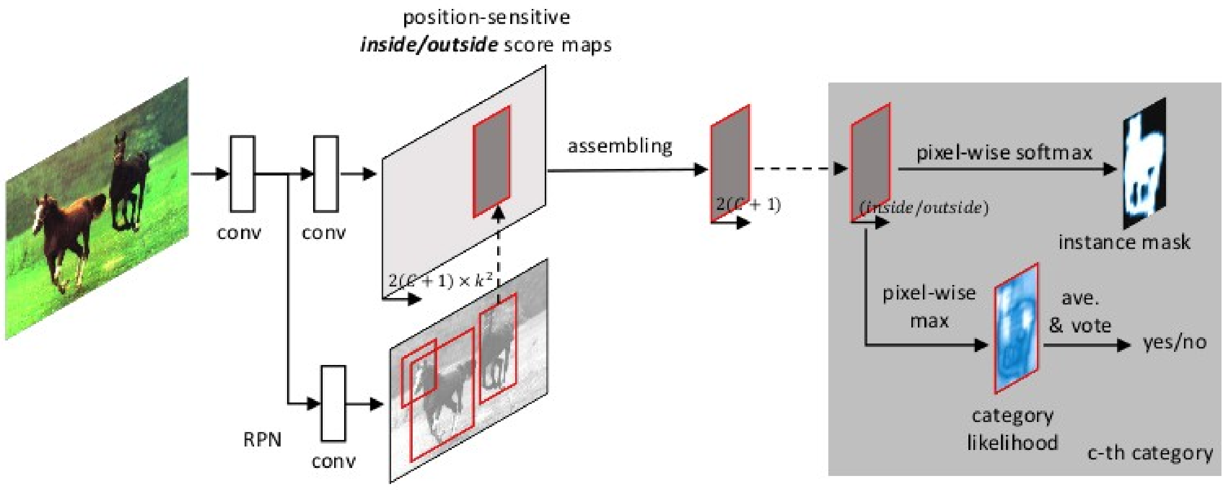


Figure 19: Overall architecture of FCIS. A region proposal network (RPN) shares the convolutional feature maps with FCIS. The proposed ROIs are applied on the score maps for joint object segmentation and detection. The learnable weight layers are fully convolutional and computed on the whole image. The per-ROI computation cost is negligible.

While any convolutional network architecture can be used, in this work they adopt the ResNet model [26]. ResNet's last fully-connected layer for 1000−way classification is discarded. Only the previous convolutional layers are retained. The resulting feature maps have 2048 channels. On top of it, a 1 × 1 convolutional layer is added to reduce the dimension to 1024.

In the original ResNet, the effective feature stride (the decrease in feature map resolution) at the top of the network is 32. This is too coarse for instance-aware semantic segmentation. To reduce the feature stride and maintain the field of view, *à trous* [27] is applied on the convolution layers. The stride in the first block of conv5 convolutional layers is decreased from 2 to 1. The effective feature stride is thus reduced to 16. To maintain the field of view, *à trous* is applied on all the convolutional layers of conv5 by setting the dilation as 2.

To generate ROIs, a region proposal network (RPN) is used on top of the conv4 layers. Note that RPN is also fully convolutional. From the conv5 feature maps, $2k^2 \times (C + 1)$ score maps are produced (C object categories, one background category, two sets of k 2 score maps per category, k = 7 by default in experiments) using a 1×1 convolutional layer. Over the score maps, each ROI is projected into a 16× smaller region.

Finally, the initial input ROIs are refined with the use of a sibling 1×1 convolutional layer with $4k^2$ channels. This layer is added on the *conv5* feature maps to estimate the bounding box shift in location and size.

4.2.2. Mask R-CNN

Conceptually, Mask R-CNN [28] is an extension of the Faster-RCNN [21] model, with the addition of a third branch to predict the segmentation mask of each detected object instance. Like Faster R-CNN, Mask-RCNN consists of two stages. The first stage, called a Region Proposal Network (RPN), proposes candidate object bounding boxes. The second stage, extracts features using RoIPool from each candidate box and performs classification and bounding-box regression, for

Faster-RCNN, and additionally a binary mask for Mask R-CNN. The features used by both stages are shared for faster inference. The binary mask prediction is done in parallel to class and bounding box predictions, in contrast to other systems where classification depends on the mask prediction.

A mask encodes an input object's spatial layout. Thus, unlike class labels or box offsets, that are inevitably collapsed into short output vectors by fully connected layers, extracting the spatial structure of masks can be addressed naturally by the pixel-to-pixel correspondence provided by convolutional layers. In order to avoid collapsing the object's dimensions into a flat vector representation, an $m \times m$ mask is predicted from each region of interest (RoI) using a fully convolutional network. This convolutional representation requires fewer parameters and, as shown in experiments, is more accurate. This pixel-to-pixel behaviour demands the RoI features to be well aligned in order to faithfully preserve the explicit spatial correspondence.

To address this, the authors propose a *RoIAlign* layer that promises to remove the harsh quantization introduced by the feature extraction method (RoIPool) for each RoI. RoIPool first quantizes a floating-number RoI to the discrete granularity of the feature map. This quantized RoI is then subdivided to bins before being quantized again. Finally, the features are aggregated by a pooling operation in each bin. Instead, the authors avoid any quantization of the RoI boundaries by using bi-linear interpolation to compute the exact values of the input features at four regularly sampled locations in each bin and aggregate the result using max or average.

The architecture of the Mask R-CNN uses two distinct fully convolutional parts: the backbone network which is used for feature extraction over an entire image and the head network for bounding box recognition and mask prediction that is applied separately to each RoI.

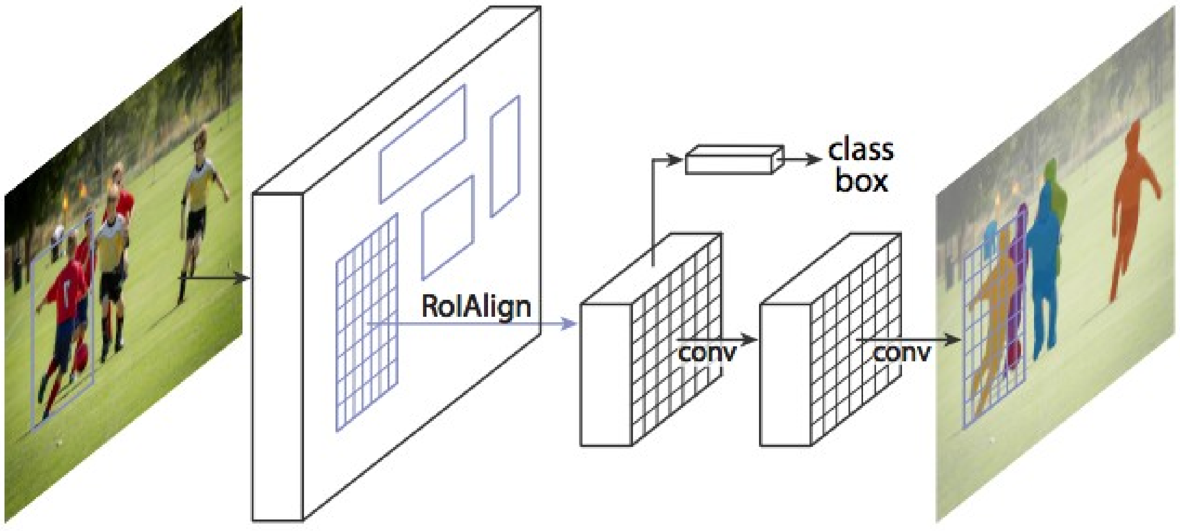


Figure 20: Overall architecture of Mask R-CNN.

As is natural, for both the head and backbone, any number of networks can be used. The authors evaluated the backbone with ResNet and ResNeXt networks of 50 and 101 layers and with FPN (Feature Pyramid Network). Using a ResNet-FPN backbone for feature extraction gave excellent gains in both accuracy and speed. In the head network they tested the top convolutional layers of both ResNet and FPN networks.

PATH AGGREGATION NETWORK (PANET)

An extension to the Mask R-CNN model is proposed by Liu et al. [29]. They claim that the way that information propagates in neural networks is of great importance and that by boosting information flow in a proposal-based architecture like Mask R-CNN the prediction masks can be more accurate. Specifically they enhance the entire feature hierarchy with accurate localization signal in lower layers using a bottom-up path augmentation. This shortens the information path between lower layers and top most feature layers. They also propose an adaptive feature pooling method which links feature grid and all feature levels to allow useful information from each feature level to propagate directly to the proposal subnetworks that follow. A complementary branch capturing different views for each proposal is created to further improve mask prediction. These improvements are simple to implement, with subtle extra computational overhead.

4.2.3. SIAMESE MASK R-CNN

The key idea of one-shot instance segmentation is to detect and segment object instances based on a single visual example of some object category. The proposal of such a system comes with [30]. The Siamese Mask R-CNN model has to deal with arbitrary, potentially previously unknown object categories which are defined only through a single reference image, rather than with a fixed set of categories for which extensive labeled data was provided during training. To solve this problem, the authors take a metric-learning approach: the model learns a similarity metric between the reference and image regions in the scene. Based on this similarity metric, it can then generate object proposals and classify them into matches and non-matches. The key advantage of this approach is that it can be directly applied to objects of novel categories without the need to retrain or fine-tune the learned model.

To compute the similarity metric they use Siamese networks, a classic metric learning approach. They combine this form of similarity judgment with the domain knowledge built into current state-of-the-art object detection and instance segmentation systems by integrating it into Mask R-CNN [28]. To integrate the reference information into Mask R-CNN, the same backbone, ResNet50 with Feature Pyramid Networks (FPN), is used with shared weights to extract features from both the reference and the scene.

To obtain a measure of similarity between the reference and different regions of the query image, they treat each (x,y) location of the encoded features of the query image as an embedding vector and compare it to the embedding of the reference image. This procedure can be viewed as a non-linear template matching in the embedding space instead of the pixel space. The matching procedure works as shown in *Figure 21*:

1. Average pool the features of the reference image to an embedding vector. In the few-shot case (more than one reference image) compute the average of the reference features as in prototypical networks [31].
2. Compute the absolute difference between the reference embedding and that of the scene ateach (x,y) position.

3. Concatenate this difference to the scene representation.
4. Reduce the number of features with a 1×1 convolution.

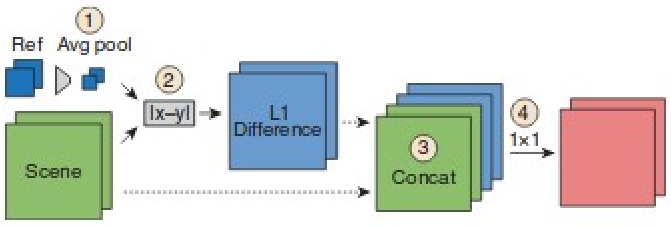


Figure 21: Sketch of the matching procedure

The resulting features are then used as a drop-in replacement for the original Mask R-CNN features. The key difference is that they do not only encode the content of the scene image, but also its similarity to the reference image, which forms the basis for the subsequent heads to generate object proposals, classify matches vs. non-matches and generate instance masks.

Because the computed features can be used as a drop-in replacement for the original features, the same region proposal network and ROI pooling operations as MaskR-CNN can be used. The same classification and bounding box regression head as Mask R-CNN are also used, the classification is changed from an 80-way category discrimination to a binary match/non-match discrimination and only a single, class-agnostic set of bounding box coordinates is generated. Similarly, for the mask branch only a single instance mask is predicted instead of one per potential category.

4.2.4. YOLACT

Over the past few years, the vision community has made great strides in instance segmentation, in part by drawing on powerful parallels from the well-established domain of object detection. State-of-the-art approaches to instance segmentation like FCIS and Mask R-CNN directly build off of advances in object detection like R-FCN and Faster R-CNN. Yet, these methods focus primarily on performance over speed, leaving the scene devoid of instance segmentation parallels to real-time object detectors like SSD [32] and YOLO [33]. Yolact [34] is such a method trying to fill this gap with a fast, one-stage instance segmentation model in the same way that SSD and YOLO fill that gap for object detection.

The approach of this model is to add a mask branch to the existing YOLO one-stage object detection model but without an existing localization step. To do this they break up the complex task of instance segmentation into two simpler, parallel tasks that can be assembled to form the final masks. The first branch uses an FCN to produce a set of image-sized “prototype masks” that do not depend on any single instance. The second, adds an extra head to the object detection branch to predict a vector of “mask coefficients” for each anchor that encode an instance’s representation in the prototype space. For each instance that survives the duplicate detection process (NMS) they construct a mask for that instance by linearly combining the work of these two branches.

Object detector systems produce class and box coefficients for each anchor as an output of a fully connected layer. This poses a problem because masks are spatially coherent, a fact that fully connected layers cannot take advantage of. Two stage approaches like Mask-RCNN get around this problem by using a localization step (i.e. RoIAlign) which preserves the spatial coherence of the features while also allowing the mask to be the output of a convolution layer. However, doing so requires for the second stage to wait for the first stage RPN to propose localization candidates, inducing a significant speed penalty.

Yolact breaks the problem into two branches that can be used in parallel. The first branch makes use of fully connected layers to produce semantic vectors of the mask coefficient. The second branch makes use of an all convolution layer architecture to compute prototype masks. The assembly step can then be implemented as a single matrix multiplication thus minimizing the computational overhead over the backbone detector.

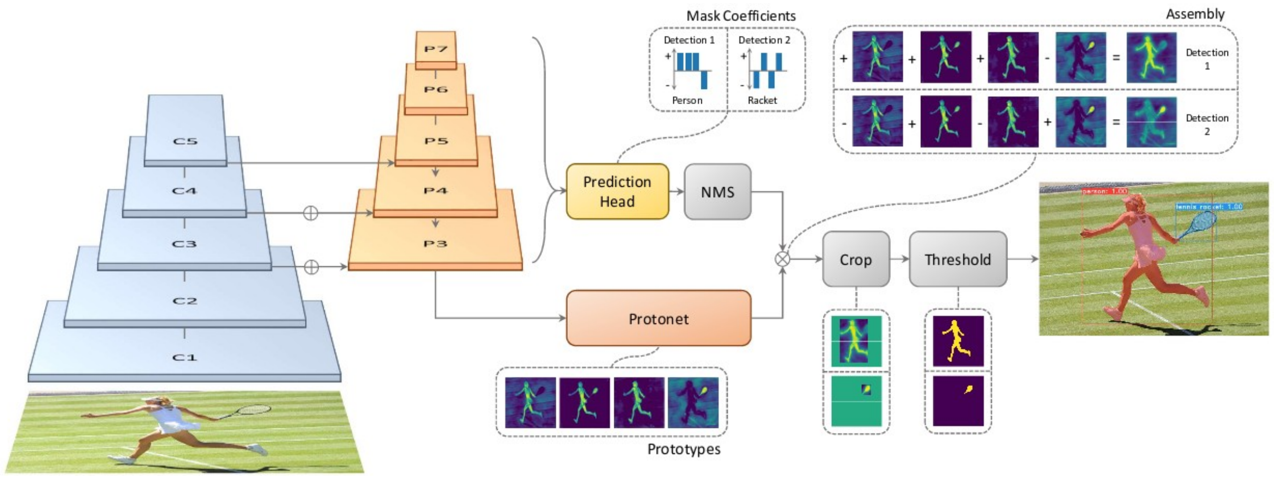


Figure 22: YOLACT Architecture. Blue/yellow indicates low/high values in the prototypes, gray nodes indicate functions that are not trained, and k= 4 in this example. We base this architecture off of RetinaNet [ref] using ResNet-101 + FPN.

The prototype generation branch (Protonet) predicts a set of k prototype masks for the entire image. The implementation of protonet is based on FCN whose last layer has k channels, one for each prototype and is attached to a backbone feature layer. While this formulation is similar to standard semantic segmentation, the supervision comes from the final mask loss, after assembly, instead of using an explicit loss on the prototypes.

Typical anchor-based object detectors have two branches in their prediction heads, one to predict class confidences and one to perform bounding box regression. In addition to these two branches, Yolact uses a third branch for mask coefficient prediction. This branch predicts a mask coefficient for each prototype, thus the whole system produces $4+c+k$ coefficients for the 4 bounding box coordinates, c possible classes and $k$prototypes. In order to tackle nonlinearity in mask construction they apply tanh activation to the mask coefficients.

The mask assembly uses a linear combination of the prototype and mask coefficient branches, followed by a sigmoid activation operation. The operation used is the dot matrix multiplication of the $h \times w \times k$ output of the prototype branch with the transposed $n \times k$ output of the mask coefficients branch (where n is the number of instances).

During training they apply a classification loss on the predicted class of each instance, a loss on the regression of the predicted bounding box of each instance and a pixel-wise binary cross entropy loss between assembled masks and the ground truth masks. In order to limit mask loss to logical values, they crop the final masks to the predicted bounding box limits and normalize with the ground truth bounding box area.

In their paper, the authors use ResNet-101 with FPN as the default feature backbone but any convolutional network can be used instead, limited by the resources and the difficulty of the prediction task.

4.3. Extending Yolact for Fandango

Our choice of model in the Fandango project is factor by two main conditions: the volume of media that have to be processed and the accuracy of the masks that is demanded. On the one hand the number of images that is expected to be analyzed is in the order of tens of millions. In order for our system to be able to analyze such a huge amount of data in logical time, the various components used have to be as fast as possible. On the other hand, based on user requirements, the final decision in any analysis is left to the human reviewer. This allows for the instance segmentation task to have somewhat more lax thresholds in the accuracy of the prediction. Taking these two factors into account, we selected to develop our research based on the Yolact model, that despite being a little less accurate than the current state-of-the-art Mask-RCNN, is a lot faster both in training and inference.

MS-COCO [35] is a dataset by Microsoft that offers labels for multiple tasks, from classification to object detection, captioning and keypoint detection. It is a superset of Pascal VOC that contains more than 300K images, millions of segmentation annotations and 5 captions per image and it recognizes 80 categories of objects. It is the most extensive instance segmentation dataset in the time of writing and the de facto dataset for algorithm comparison.

While this datasets categories are not the ideal from a reporters point of view, the majority of common life objects depicted in a news article’s image are included. Pending to find, or create, a more detailed annotated dataset in a specific domain of interest, we are deploying the object detection service using the pretrained Yolact model distributed by the authors, using the ResNet101-FPN backend. This model, with the specific backend can process more than 20 images per second and achieves 31.2 mAP on the MS-COCO dataset.

As is the case in all image analysis microservices provided in the Fandango project, this analyzer is deployed as a RESTful service, expecting either a base64 encoded image or a valid url where it can be retrieved. The response of the service is a vector of the same size as the number of classes that are detected with the counts of the instances of each category that are detected. In addition, we are returning a base64 encoding of the original image after we have drawn the predicted masks of each instance detected by the model.

Further research on this module will be conducted on two directions: trying different backbone networks in order to get better accuracy in the predictions without compromising the speed and in collaboration with the Fandango end users find more specialized datasets on objects that are of more interest to an investigative reporter.

5. EXTRACTION OF VISION-BASED SPATIAL MARKERS

5.1. A MODEL ON THE TYPE OF LOCATION

In 2014, Zhou et al [36], introduced a new database, called Places, where they gathered 7,076,580 million images of scenes and places, divided into 476 different categories. The dataset is provided in 2 subsets, the Places205 and the Places88, where the first contains the images in the 205 categories which contain more than 5000 images, while the later contains the 88 categories which contain more than 1000 images in the ImageNet dataset[4]. In their paper, they also demonstrate the adequacy of Places database in scene-based visual tasks by using CNNs for deep features extraction. The latest update of the places dataset is Places365 [37] where 365 categories of images are provided, with more than 5000 samples each, from a new collection of more than 10 million images. The categories that the dataset provides include abstract categorization such as indoors, outdoors, room, park. A tag cloud of these abstract categories is depicted in *Figure 23*. But additionally they provide with semantic categories of places which are defined by their function: the labels represent the entry-level of an environment. To illustrate, the dataset has different categories of bedrooms, or streets, etc, as one does not act the same way, and does not make the same predictions of what can happen next, in a home bedroom, a hotel bedroom or a nursery.

4 Imagenet project, 1k Images dataset. url: http://image-net.org/about-overview

Figure 23: A tag cloud of Places dataset abstract categories

By using the pre-trained models of the Places dataset, we can provide a very accurate and helpful labelling of the images that exist in the analyzed articles so that the localization of the included media can be performed easier for the end user. As a first approach we are delivering a porting of the original kaffe model into the Keras/Tensorflow framework for deep learning, using the ResNet50 [26] model. This prototype permits us to retrain and fine-tune the model to more detailed localization labels based for each pilot if this is necessary.

5.2. NARROWING DOWN THE LOCATION BASED ON KNOWN LANDMARKS

While the type of a location depicted in an image can be very useful for the end user to detect spatial context, there are situations where a very specific landmark is depicted in the image, which if detected by the Fandango platform, can be an even bigger boost.

5.2.1. DATASETS

Designing systems capable of accurate instance-level landmark recognition (i.e., distinguishing Niagara Falls from just any waterfall) and retrieving images (matching objects in an image to other instances of that object in a catalog) is a longstanding pursuit of Google's AI research division. Last year, it released Google-Landmarks, a landmarks data set they claimed at the time was the world's largest, and hosted two competitions (Landmark Recognition 2018 and Landmark Retrieval 2018) in which more than 500 machine learning researchers participated.

In support of this goal, this year they released Google-Landmarks-v2 [38], a completely new, even larger landmark recognition dataset that includes over 5 million images (2x that of the first release) of more than 200 thousand different landmarks (an increase of 7x). Due to the difference in scale, this dataset is much more diverse and creates even greater challenges for state-of-the-art instance recognition approaches. The training dataset was constructed by mining web landmark images with permissive licenses, leveraging crowdsourced labels. For this reason, each category may contain quite diverse data: for example, images from a museum may contain outdoor images showing the building and indoor images depicting a statue located in the museum. To avoid bias, no computer vision algorithms were used for ground truth generation.

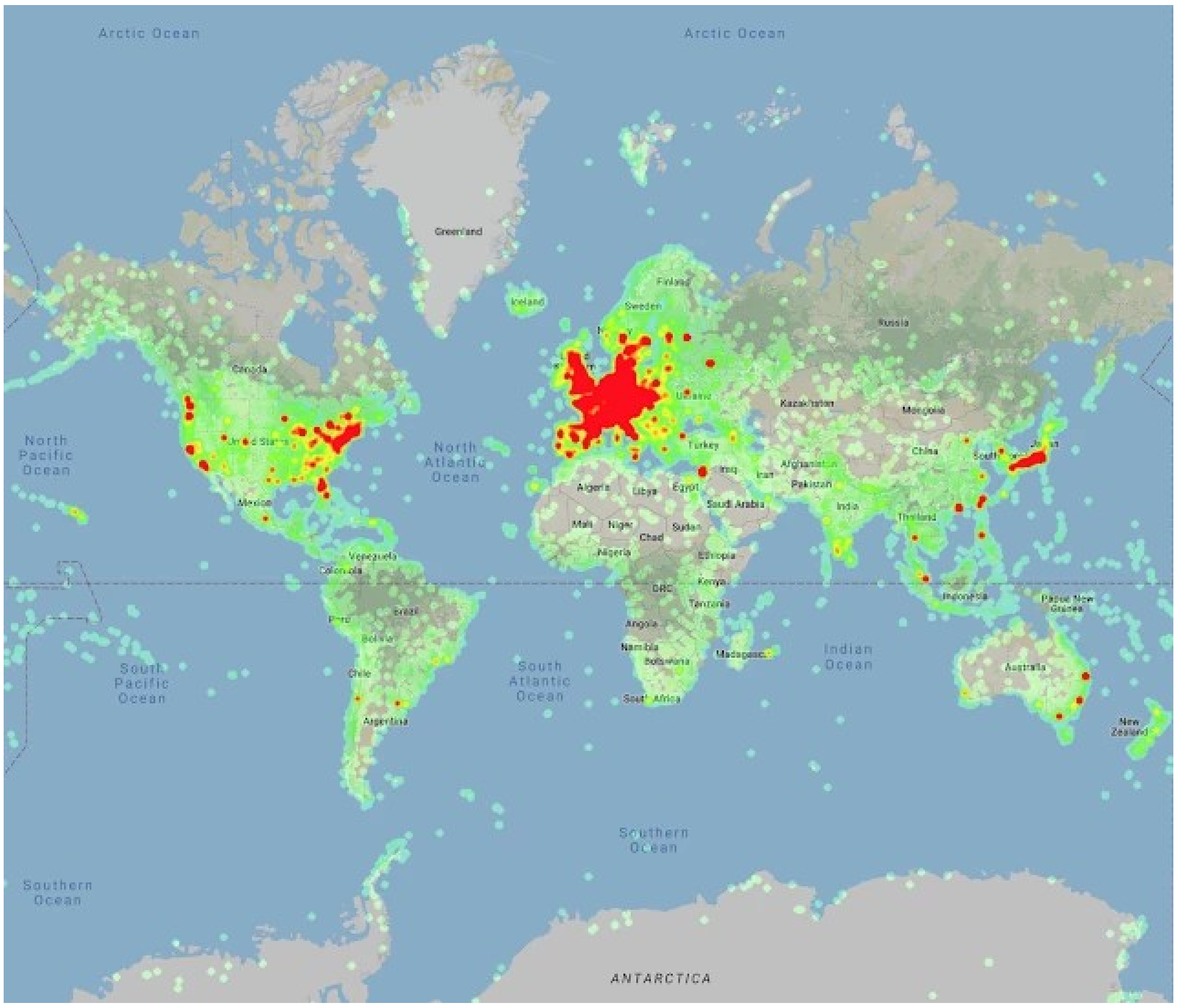


Figure 24: Landmark images distribution on the world.

Based on this new dataset, they also announced two new Kaggle challenges—Landmark Recognition 2019[5] and Landmark Retrieval 2019[6] and released the source code and model for Detect-to-Retrieve [39], a novel image representation suitable for retrieval of specific object instances. This method leverages bounding boxes from an object detection model to give "extra weight" to image regions containing items of interest, significantly improving accuracy.

5 'Google Landmark Recognition 2019: Label famous (and not-so-famous) landmarks in images', https://www.kaggle.com/c/landmark-recognition-2019/overview.

6 'Google Landmark Retrieval 2019: Given an image, can you find all of the same landmarks in a dataset?', https://www.kaggle.com/c/landmark-retrieval-2019

5.2.2. State-of-the-art

The two challenges announced by Google were conducted simultaneously and cross participation was encouraged. It is logical to expect top participations in the contest to be common for the two challenges. The Landmark Recognition challenge[5] was about correctly identifying if a landmark exists in a query image and which landmark it is. On the other hand, the Landmark Retrieval challenge[6] asked the participants to retrieve all images in a world database that depict the same landmark as the one contained in a query image.

The evaluation metric for the competition is Google Average Precision ('GAP') [40]: the winning teams make sure that distractors, if predicted, should have lower confidence scores than real landmark images, i.e. this precision penalizes if non-landmark images (distractors) are predicted with higher confidence score than landmark images.

Large-scale Landmark Retrieval/Recognition under a Noisy and Diverse Dataset

The winning solution in the retrieval challenge by 'Smlyaka' team is described in [41]. The contestants' solution was also one of the top-performing in the recognition challenge, achieving third place. They believe that while the provided dataset for the challenges is huge, it is also very noisy and heterogenous. In order to tackle the noisiness and heterogeneity of the data they employed a local feature mapping and a discriminative reranking technique (*Figure 25*).

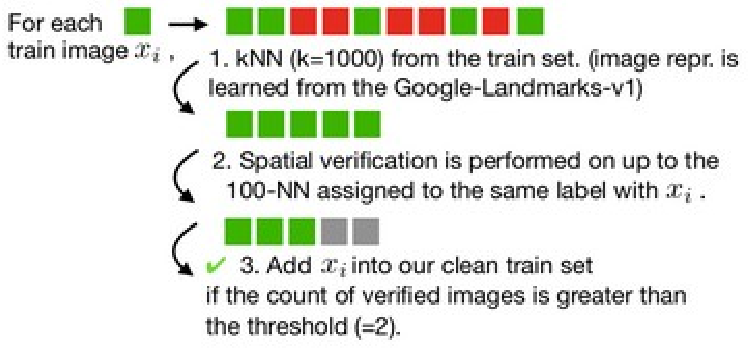

For each train image x_i,
1. kNN (k=1000) from the train set. (image repr. is learned from the Google-Landmarks-v1)
2. Spatial verification is performed on up to the 100-NN assigned to the same label with x_i.
✓ 3. Add x_i into our clean train set if the count of verified images is greater than the threshold (=2).

Figure 25:Automated data cleaning algorithm

For cleaning the dataset, they apply spatial verification to filtered images by k-nearest neighbor search. Given a representation I_k learned from image (Google landmarks version 1 dataset[7]), the developed algorithm yields the k-closest neighbors from the training data. Then, from the 100 nearest neighbors from the same classifying category as I_k, spatial verification is carried out, by means of RANSAC [42] and Deep local attentive features [38].

Over the previous pipeline, an ensemble model (*Figure 26a*) is used taking as input, descriptors from 6 backbone models. The derived descriptors are combined by concatenation and produce a new vector of 3072 dimensions. The train-image is rescaled at factors of [0.75, 1.0, 1.25]. The final descriptor is the mean value of the different scales.

The backbone neural networks used are CNNs, FishNet-150 [43], ResNet-101 [26] and SE-ResNeXt-101 [44] trained with cosine-softmax based losses. They are initially pre-trained on

7 Landmarks dataset: https://github.com/cvdfoundation/google-landmark

ImageNet and with the first version (v1) of the dataset, then fine tuned on the cleaned dataset. The losses were cosine-softmax based: Arc-Face and CosFace with a margin of 0.3. As a pooling method, generalized mean-pooling was preferred to other methods (with fixed parameter equal to 3.0).

The training procedure for the combined model that was chosen contained a 5-epoch training with "soft" data augmentation (random cropping and scaling), and 7-epoch training with the generalized version of the previous way, which is "hard" data augmentation (also involves random brightness shift and random sheer translation). To make the image processing uniform for all samples, mini-batch samples of similar aspect ratios are resized before being used for training.

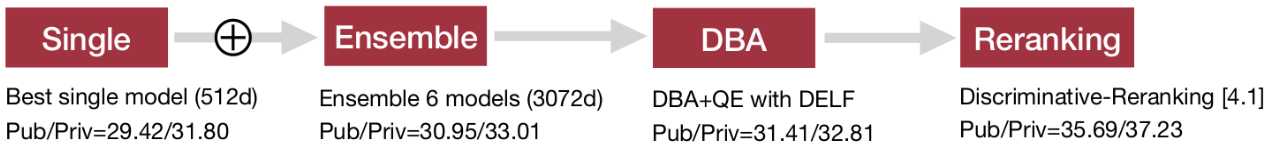


Figure 26 (a) Ensemble architecture of [60]

For the retrieval operation, they employ a similarity search in the whole index set using a euclidean metric (and L2-normalized descriptors). The data include a variety of image labels, such as landscapes and interior spaces of apartments. If these data pertain to the same landmark, they can hardly be differentiated. More specifically, the lack of visual similarity does not allow to a common distance metric to detect any 'proximity' of the images in the space of descriptors if there is not a relative context provided. In order to face this issue, they propose a procedure dubbed Discriminative Re-ranking, which leverages the classes of the training data in order to list the retrieved images.

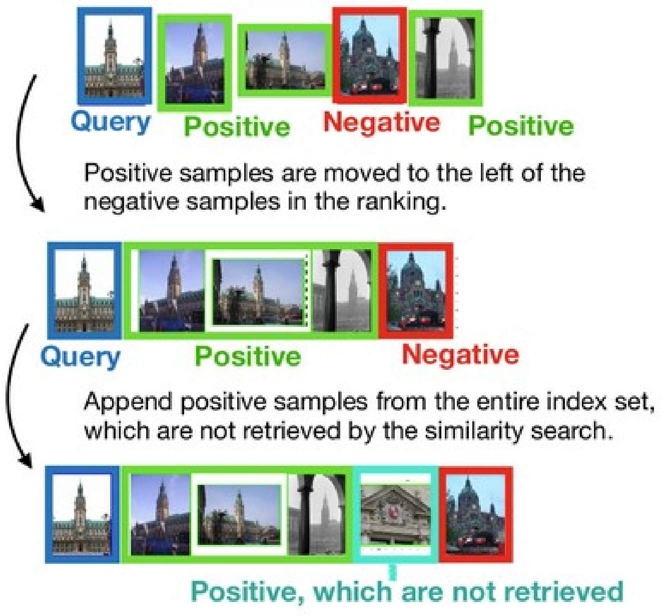

Query
Positive
Negative
Positive
Positive samples are moved to the left of the negative samples in the ranking.
Query
Positive
Negative
Append positive samples from the entire index set, which are not retrieved by the similarity search.
Positive, which are not retrieved

Figure 26 (b) Discriminative reranking

In the beginning of the testing stage, a landmark-id is predicted for each image. The images from the index set that are assigned the same landmark prediction as the test image are regarded as 'positive samples'. On the other hand, the index set images that are assigned a prediction different from each test image are called 'negative'. During the re-ranking process, given a query-image (test data), samples are retrieved from the index set and are coloured as positive or negative (*Figure 26b*). Afterwards, the algorithm shifts the positive samples on the left of the negative ones and then attaches non-retrieved positive ones from the entire index pool (because it uses their labels, not a similarity criterion), which results in sorting the images by relevance to the test image, even despite being visually dissimilar. This system does not need training of any discriminative model in any of its components.

GLRunner Team

The second place winners of both challenges, 'GLRunner' team, present their work in [45]. It describes a retrieval based system with five parts, including feature extraction and matching, to get a candidates queue; database augmentation and query extension searching; reranking from recognition results and local feature matching.

The first part that the retrieval method involves is the global feature method. The authors initially train backbone systems for the purpose of feature extraction which are then fine-tuned on the challenge training data. The arc-margin loss is used for these systems, a function which comes from face recognition tasks but also results in meaningful descriptors for the current data. The quality of these descriptors improves if the last fully connected layer is discarded and two layers are added after the pooling layer. The sample descriptor is chosen to be the output of the first fully connected layer. The training image size is 448 and the optimizer is SGD. The constructed neural network provides six descriptors per image which are concatenated and then downscaled with PCA (*Figure 27*). They are pre-trained on Imagenet dataset[8] and are used in combination with nearest neighbor search to assist the retrieval task.

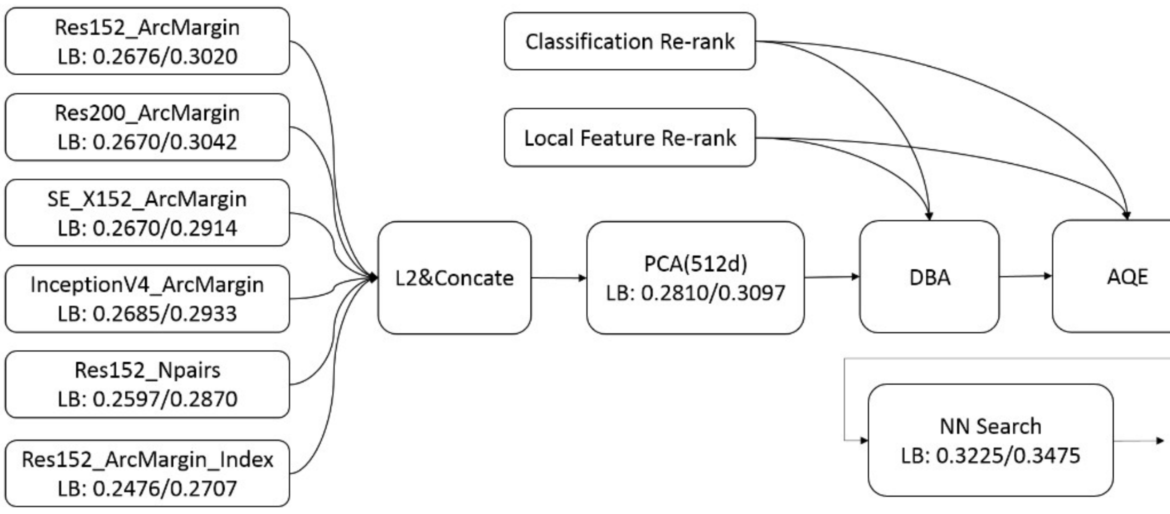


Figure 27: 'Paddle Paddle' framework for retrieval.

8 Imagenet project, 1k Images dataset. url: http://image-net.org/about-overview

Furthermore, the retrieval method, focuses also on local features. Despite the fact that the corresponding modern methods (convolutional neural networks) for this purpose have prevailed over the past methods, the past ones behave with higher accuracy for retrieval objectives when the images contain geometric transformations (scale, angle) as is the case with landmarks. The respective retrieval result is shown in *Figure 28*. The selected technique is called 'Speed Up Robust Features' [46] and is combined with object retrieval solutions for corner detection, curve representation [47] and affine region detection [48]. For the last operation of the system, k-means clustering is carried out on the query images. From the derived clusters, the top 20 clustering centers are used to create an inverted index for the Nearest Neighbor search, making it a faster retrieval procedure.

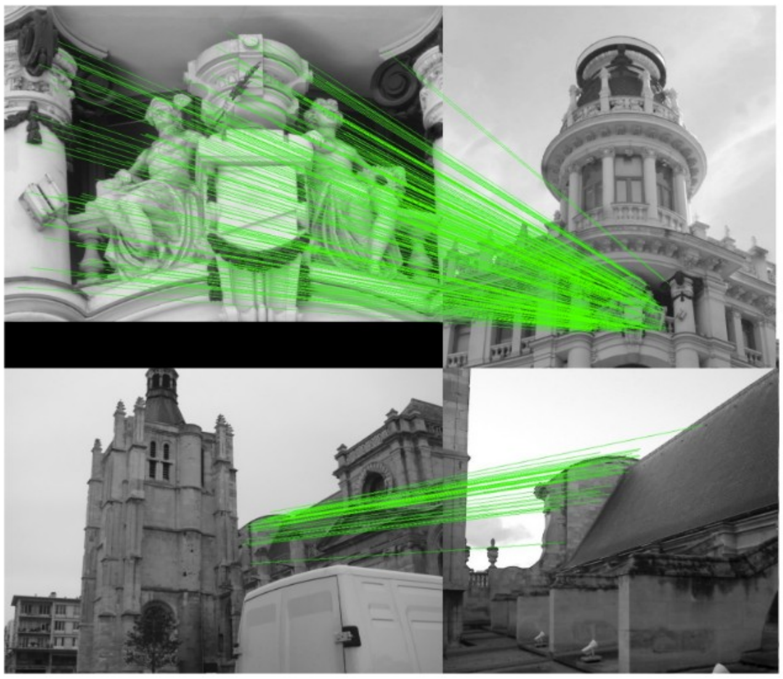


Figure 28: The use of local features succeeds in matching the current samples, which is a task of considerable difficulty. ('Paddle Paddle' framework is used for pretraining the backbone models).

The ensemble global feature extraction network uses Inception V4 and multiple variations of the ResNet-152 and SE ResNeXt152 ([26], [44], [49], [50])

As far as the previous neural network is concerned, apart from the prediction task on the images data pool, the authors use this network on the test and index dataset. In the case that the top 5 ensemble convolutional models return only two labels and the maximum label probability is over 85%, the sample is classified in the label of the highest vote.

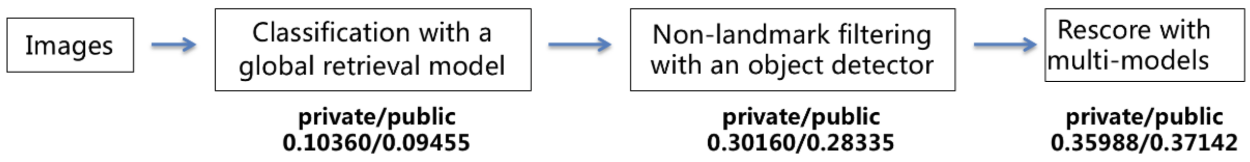
For the reranking stage they propose a two stage methodology. Initially, Nearest Neighbor search is performed in order to find the closest 300 neighbors to the query image. The neighbors are then passed to the classification model and the local feature system in order to obtain M same landmark samples. During this process, each image descriptor is replaced with a weighted version of itself and the top N neighbors (M and N are parameters of the system). The M images are placed first in the ranking along with the 300 adjacent images. This operation is involved in the database-aside feature augmentation (DBA) on the query and index datasets.

The same operation is repeated a second time but with a different weighting function. The final retrieval scores (Google Average Precision, 'GAP') are highest when using both stages of this methodology.

Similar are the pipeline components for the solution suggested in the recognition challenge. A cumbersome example of the query and corresponding fetched images is portrayed in *29b*.

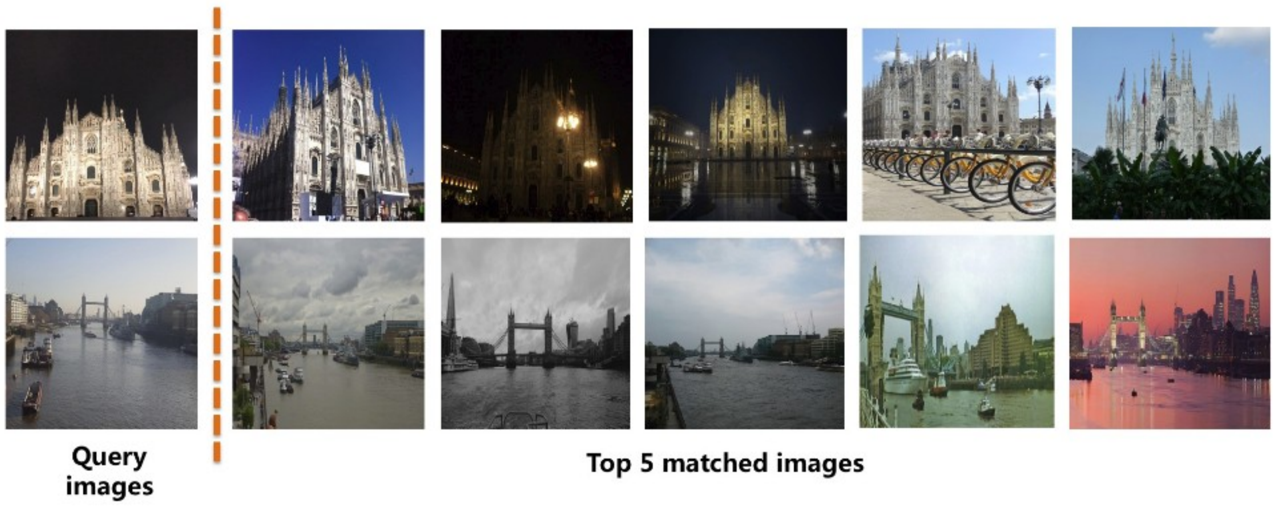
The first stage of the implemented model is the application of k-nearest neighbors for the classification of the image. Next, the RESNET152 network is employed for the global features extraction and all test images are associated with the train images. Given a test sample, from its highest 5 associated images, the majority vote is derived. Then this is the classification label that is assigned to the test sample and the highest score of the majority label will be used as the predicted score.

Following this, the non-landmark images are filtered out and the tool for that task is a single object detector based on Faster RCNN [21], trained in the open images dataset[9] for object detection. For the differentiation of the images, the 600 object classes are separated into three categories: landmark label (the object classes are: Building, Tower, Castle, Sculpture and Skyscraper), uncertain label (the object classes are: House, Tree, Palm tree, Watercraft, Aircraft, Swim-ming pool and Fountain) and non-landmark label. For a test image, if there exists at least one object labelled as landmark, it is regarded as a landmark image. If there exists one object labelled as non-landmark, it becomes a candidate non-landmark image. The initial criterion is that the detector score of the object must be greater than 0.3. Afterwards, the area ratio between the object bounding box and the whole image must be greater than 0.6. Hence, about 28k images from the test images (120k) are classified as non-landmark images.


Images
Classification with a global retrieval model
Non-landmark filtering with an object detector
Rescore with multi-models
private/public 0.10360/0.09455
private/public 0.30160/0.28335
private/public 0.35988/0.37142

a: Match results for the Recognition challenge of the GLRunner team in hard query samples along with the reported GAP scores

9 Open images dataset, url: https://storage.googleapis.com/openimages/web/factsfigures_v4.html



b: Match results of the GLRunner team in hard query samples

Figure 29: GLRunner team solution

Lastly, since the GAP is based on the rank of all predictions, promoting the credible landmark scores boosts the performance. So this stage grades and rescores the test images with multi-models, the aforementioned global retrieval model and a classification model (ResNet152 trained with 3M images filtered from the 4.13M train images). Scores of all components are shown in Figure 29a.

TEAM JL SOLUTION TO GOOGLE LANDMARK RECOGNITION 2019

The 1st place winners of the recognition challenge [51] use both global and local CNN retrieval approaches, thus leveraging the large number of classes, noisy data, imbalanced class sizes and the presence of a significant amount of distractors in the dataset.

This solution resembles the previous in that backbone models were used but the higher GAP is due to the re-ranking step: its purpose is to discriminate the authentic landmark samples from the distracting ones by raising their confidence metrics. After the predictions are sorted in terms of confidence, the highest twenty thousand are shortlisted and each one is matched to all of the rest images with the Detect-to-Retrieve model [39], a model that can detect local features. The images with inlier scores lower than a threshold are excluded, the derived pool of images is resorted with respect to the inlier scores and enqueued to the top ranked image (having the predicted labels attached). The full pipeline, after ensembling the models and applying several steps of re-ranking strategies, scored 0.37606 Google Average Precision (GAP), achieving the 1st place in the competition.

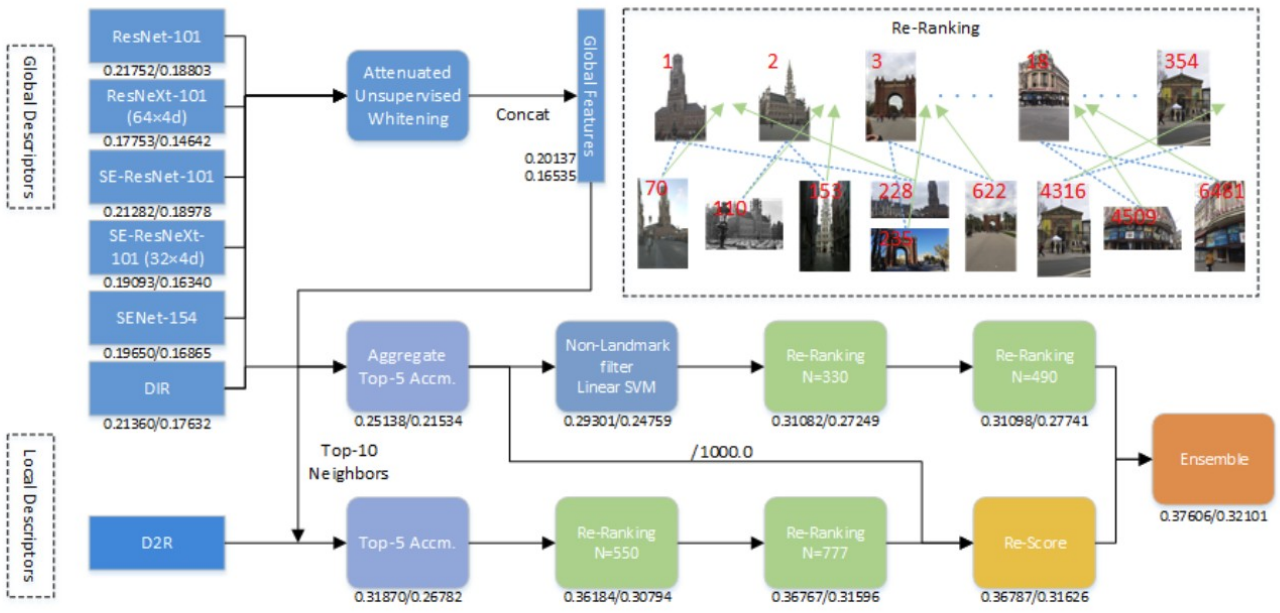


Figure 30: Architecture of team JL's system

Explore-Exploit Graph Traversal for Image Retrieval

An innovative graph-based approach is presented in [52]. The global descriptor model generates a nearest neighbor graph (the input to the system) and the main algorithm of the model moves across this by switching between 'exploit' and 'explore' operations. The 'exploit' operation handles the one-hop neighbors of every node (the nodes connected by one edge with the current node), whereas the 'explore' process examines distant vertices in the descriptor space. The two procedures are joined so that the learning process embeds the image manifold and the constructed system correctly fetches images that may be disparate to the query and global descriptors fail to retrieve. This way, it is possible to fetch images that are far from the query image in the descriptor space.

The used technique uses elementary data structures, has a low number of parameters and the steps it performs are not complex. Therefore real-time inference is possible for newly arrived queries with low memory cost. The goal is to perform spatial validation into the constructed graph by re-assigning values on the edges. The retrieval process outmatches other state-of-the-art approaches on several large-scale benchmarks in terms of both accuracy and speed, i.e. revisited Oxford, revisited Paris and the Google Landmark Retrieval challenge dataset. This work along with its notable results has been published in 2019 in the Computer Vision and Pattern Recognition conference. A challenge for follow-up study is to examine other graph traversal techniques and determine whether they outrank the current one.

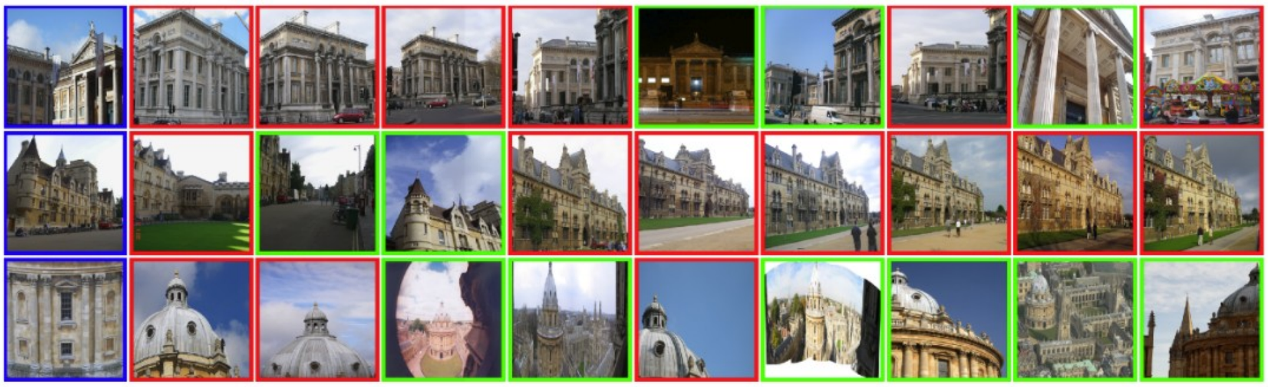


Figure 31: Image retrieval using diffusion. Query images are in blue, correctly retrieved images are in green and incorrect ones are in red.

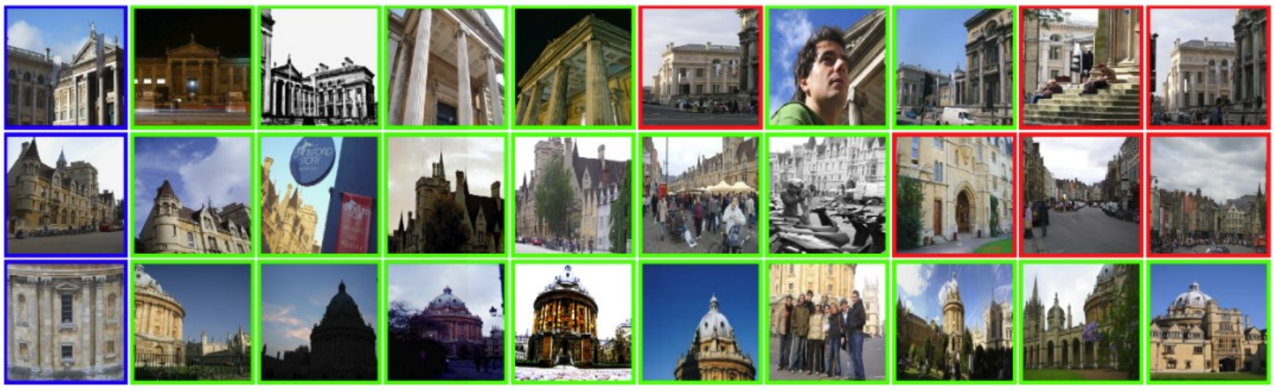


Figure 32: Image retrieval using EGT/rEGT. Query images are in blue, correctly retrieved images are in green and incorrect ones are in red. The task is performed with higher number of correct retrievals in comparison to using diffusion (figure 31).

There are three queries from R-Oxford shown in *Figure 31* and *Figure 32*, along with the top nine retrieved results using the following two algorithms: 'Diffusion' ([53], *Figure 31*) and the variant introduced by the current work 'Exploration-Exploitation Graph Traversal' of the constructed k-NN graph G_k with the application of RANSAC scoring (called 'rEGT', *Figure 32*). By comparing the two approaches, the first conclusion is that the Diffusion algorithm retrieves images with similar viewpoints which leads to incorrect fetches. On the contrary, images retrieved by rETG approach are much more diverse and include multiple viewpoints and conditioning (zoom, lighting etc.) variations.

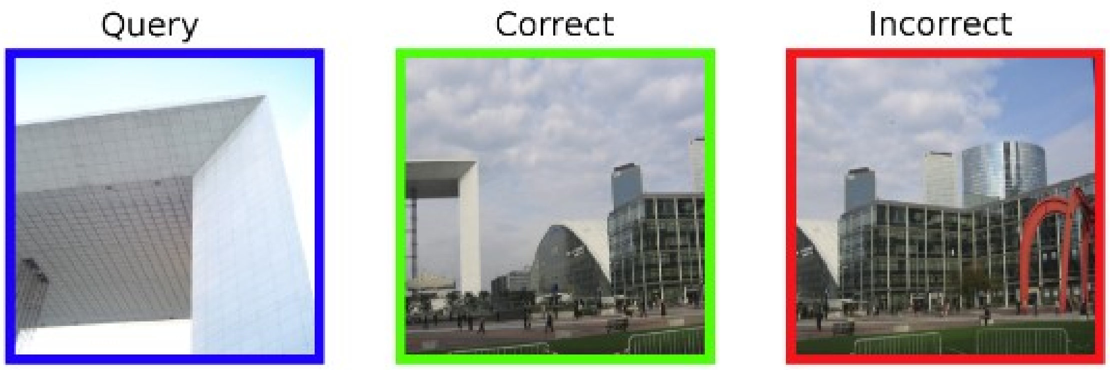


Figure 33: Example of an incorrect retrieval (which is due to the phenomenon of topic drift)

An observation on the accuracy of the explore step is that images which include e.g. additional buildings can lead to topic drift: in *Figure 33* given a query image in blue, EGT first retrieves the correct image in green and then an incorrect image with similar buildings. Cluttered scenes increase the likelihood of topic drift, and a large distractor set is likely to contain more images with similar structures. The weaker performance of EGT on RParis+R1M which is reported[10] so far can be partially attributed to the combination of these factors.

5.2.3. Methodology

The four methods that were presented cover most of the ideas used in the two kaggle challenges. All of these methodologies show that a preprocessing stage to remove noise from the dataset should be used. Additionally, all of the methodologies show that a reranking stage can benefit the system.

Most of the noise in the dataset comes from the fact that it was built using public images that were labeled by the annotators' activity rather than by describing the landmark that was shown. For example, pictures taken in a field trip around the Eiffel tower are all labeled the same irrespective of if the actual landmark is shown in the image. The second major source of noise is the fact that in many landmarks there can be both an interior and an exterior view. This is the case in almost all of the museums where images of the exterior of the museum share the same label with pictures of the exhibits. Additionally, many of the top ranking entries decided to remove landmarks with few samples.

The success of a system, in regard to the GAP metric used in the competition, greatly benefited from reranking the retrieved images. Almost all of the top ranking teams in the competition used some kind of tweaking in the order that their system returned the relevant images in the retrieval challenge. The most common methodology was some kind of self similarity score in the returned results. Another major conclusion is that not all pictures in the world have a landmark in the background. Besides the fact that, during the competitions, the score was greatly penalized when a landmark label was attached to a non-landmark image, in a news item scenario most of images analyzed come from sources where a landmark is not depicted.

In Fandango, the system should be able to detect the location where an image was taken. While most modern cameras will include the geolocation information, this information can be tampered with, making it unreliable. By detecting landmarks that might appear in an image, a very good estimation on the geolocation can be made. A major drawback of using the Google Landmarks datasets is that the dataset does not include the names of the landmarks but only an id. While for a competition this is enough, for a news reporter it is not useful. Having a system report that a protest took place in front of landmark 73565 has no meaning.

In order to solve this problem we are taking 2 steps. The first step is to use the urls of the images that are included in the dataset in order to locate them in the web. A major part of the images are hosted in image sharing platforms like Flickr and Instagram. In many cases the labels given by the uploader of the image to platform can be retrieved. By detecting the location tag of an image we

10 Daniel Fernandez, Andrey Ponikar and Mikhail Fain (Cookpad Ltd). 'Good Old Pretraining (with New Tricks)', 5th Place in Kaggle Landmark Recognition Competition 2019

can annotate the landmark id with a specific location tag. The second step is to use manually annotated images from landmarks that are of interest to Fandango's end users and cross reference the predicted landmark ids of a landmark detector model. This process is able to provide the system with a big number of labels. Taking advantage of this system we can provide the platform with a landmark descriptor.

As is the case with all the modules in Fandango, this process is provided as a RESTful service that will return upon call the landmark descriptor of an image, given either the url of this image or the image itself in base64 encoding.

5. NEAR DUPLICATE IMAGES FOR TEMPORAL UNFOLDING

5.1. INTRODUCTION

One of the requirements of the Fandango platform is to inform the end user about the reuse of images in news articles. While an image may be perfectly valid, in the sense that it has not been tampered with, the visual content has to coincide with the textual context in the article that it is being used. Additionally the image has to be from the specific event that is being discussed or be clearly noted as archive material. Also, while a tampered image might not be easily detected, even with state-of-the-art methodologies, finding a very similar image in the wild, with minor tamperings, can offer both to the end user and the system a clear indication that one of them is tampered and the pair of images should be investigated more. To provide the system with temporal clues as to the reuse of images, the Fandango platform includes a near duplicate image detector.

5.2. State-of-the-Art

A lot of research has been done on the problem of image similarity based on visual features during the last years. Following is a presentation of the state-of-the-art methodologies based on deep learning approaches.

5.2.1. Image Matching

Melekhov et al. [54] focuses on the discrimination of similar and non-similar image pairs by representing them with neural network based feature vectors. These features vectors are extracted from a convolutional neural network (CNN) trained with matching and non-matching labeled image pairs. The image similarity is measured by the Euclidean distance of feature vectors. The Siamese network loss is calculated using a contrastive loss function layer. The model architecture is illustrated in *Figure 34*.

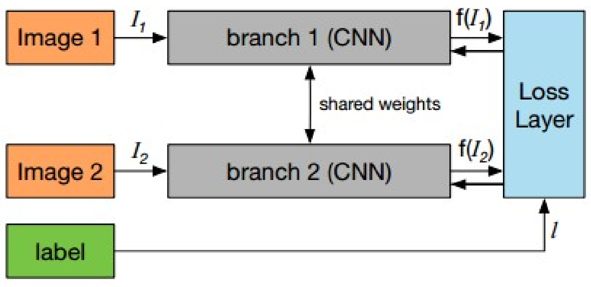


Figure 34: Siamese network architecture

In the beginning, a pair of images (I_1, I_2) goes through the network. Then, two features are extracted for each image ($f(I_1), f(I_2)$) and are fed to the loss layer. This layer tries to minimize the squared Euclidean distance for the features of similar images and maximize it for dissimilar pairs. For the network optimization, a cost function is implemented with the capability to distinguish the similar $(l=1)$from dissimilar pair $(l=0)$of images.

$$L=\frac{1}{2} l D^2 + \frac{1}{2}(1-l)\{max(0, m-D)\}^2, \text{ where}$$

l is a similarity binary indicator, m is the margin for dissimilar pairs and $D = ¿ f(I_1) - f(I_2) \vee ¿$ is the Euclidean Distance between the feature vectors.

This neural network is a pair-based (Siamese) network structure with sHybridCNN architecture, used for both object and scene image classification. The model structure consists of two identical branches that share weights and parameters. Each branch includes a set of convolutional layers, ReLU layers and fully-connected layers (FC). From the top three fully-connected layers $(fc6, fc7, fc8)$, the last one is removed and the $fc7$ is used for feature representation,aiming to learn the optimal feature representation.

5.2.2. Learning Near Duplicate Image Pairs

Zhang et al. [55] focuses on learning a general straightforward similarity function from raw image pairs. The proposed function substitutes the complicating handcrafted features extraction, by utilizing pairwise correlation information by the commonly processing.

This paper shows two possible architectures for detecting image similarity:

1. *The Siamese model*, which follows the traditional representation-matching strategy. This model has two branches sharing the same weights and architectures. Each branch consists of a set of convolutional layers, ReLU layers and Pooling layers. The output vectors, of each branch, are concatenated and go through the decision layer to give a 'yes' or 'no' answer. The branches are the representation generator, while the decision layer is the matching stage. This architecture is illustrated in *Figure 35a.*
2. *The Double-channel model*, which does not produce a single image representation generator. However, this model combines two images into one input resulting the output of bottom intermediate layer which could be regarded as extracted features. In the end, it is imported into the decision layer. This network structure is shown in *Figure 35b.*

Both these networks process raw images. However, there are some trade-offs as far as efficiency and accuracy are concerned. The double channel shows higher accuracy and better efficiency in training stage, while it needs more time to complete the test stage. The Siamese network takes into account a limited number of correlations (sharing the same weights and architectures) in contrast to the double-channel which provides better flexibility by processing the images jointly from the beginning.

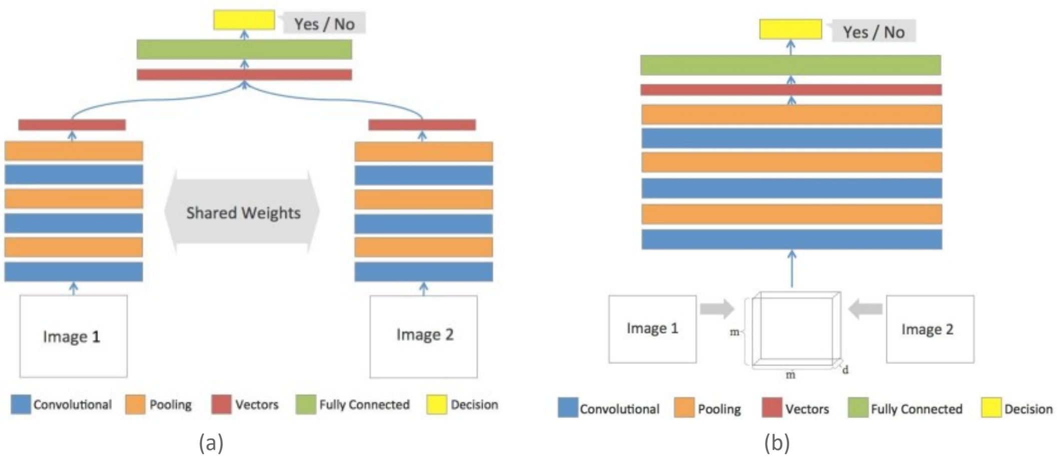


Figure 35: Illustration of two architectures.

They decided to train the second model, the double-channel network. The training process was strongly supervised. The I matrix is the image pair, which consists of the resize operation $r(\cdot)$ of two raw images I_1, I_2 (*equization 1.2*). The likelihood of the query image pair being similar is measured by a softmax regression on the final layer. The loss function is shown below:

$$I = \begin{bmatrix} r(i_1) \\ r(i_2) \end{bmatrix} \quad (1.2) \qquad Loss = -\frac{1}{n} \times \sum_{i=1}^{n} y_i \log f(x_i) + \lambda \sum_{k=1}^{L} sum \|W_k\|^2$$

This model does not extract single image features and that is why the method complexity is reduced significantly, requiring only data labeling from human.

5.2.3 Sketch-based image retrieval

Yonggang Qi et al. [56] propose a novel convolution neural network based on Siamese network for SBIR (Sketch-based image retrieval). The paper aiming both on minimizing the distance of feature vectors for the similar sketch-images and on maximizing it when they are dissimilar. This can be achieved by linking two convolutional networks to one loss function. In other words, the similarity metric $M_w(S,I)=$¿¿ should be small if sketch S and real image I belong to the same category and large otherwise. Two identical convolutional neural network compose the Siamese CNN branches which outputs are used to calculate the loss functions defined as:

$$L(W)=\sum_{i=1}^{N} l(W,(S,I,Y)¿¿i)¿$$, where

(S,I,Y) is the sketch (S), real image (I) pair and $Y=0$ when the pair is positive and $Y=1$ when the pair is negative.

$$l(W,(S,I,Y)^i)=(1-Y)L_p\left(M_w(S,I)^i\right)+YL_N\left(M_w(S,I)^i\right)$$, where

$(S,I,Y)^i$ is the ith pair and L_P, L_N are the losses for a positive and negative pair respectively. The model architecture is presented in *Figure 36*.

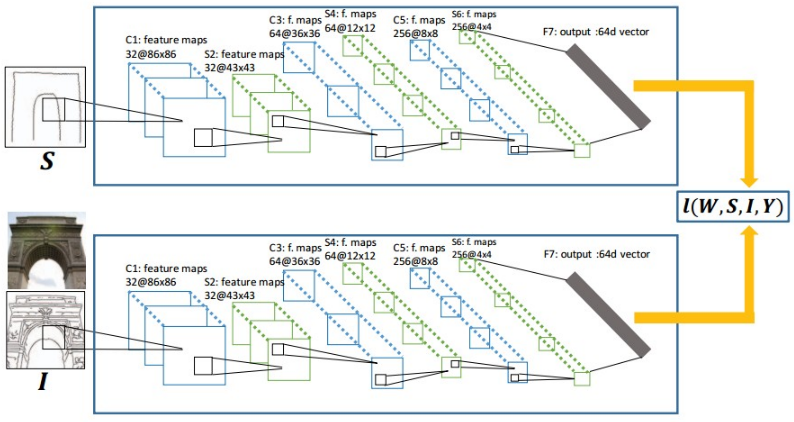


Figure 36: Siamese CNN overview.

Each branch consists of a sketch Siamese CNN with 7 layers, namely three convolutional layers alternate with three sub-sampling layers and in the end a fully connected layer. The output is a 64 dimensional feature vector. The retrieving procedures starts with a given sketch S. Then, a feature vector V_s is built by the aforementioned model representing the sketch and a feature vector V_I is constructed for each real image I by the same model. Finally, the real images are ranked according to the Euclidean distance $d(V_S,V_I)$.

5.2.4 Universal Image Manipulation Detection

Mazumdar et al. [57] propose a method that uses a deep siamese neural network for the classification of either similar or dissimilar processed images. The network tries to learn feature that can discriminate the different ways of an image manipulation. The block diagram of this architecture is presented in *Figure 37 (a)*. The twins neural networks trained to learn features that can discriminate whether a pair of images has been similarly edited. Each CNN network architecture is shown in *Figure 37 (b)*.

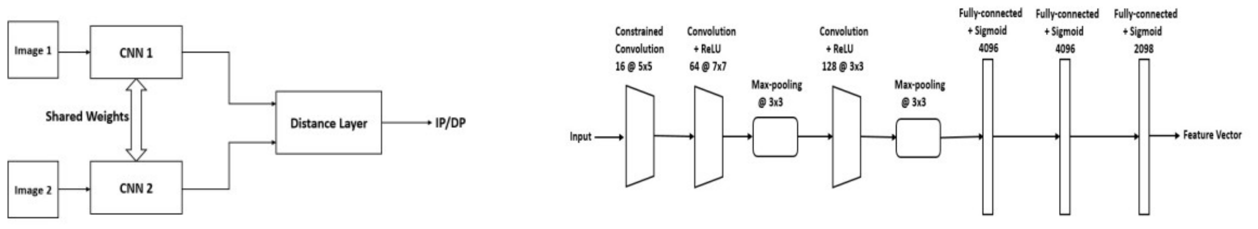


Image 1
CNN 1
Shared Weights
Image 2
CNN 2
Distance Layer
IP/DP
Constrained Convolution 16 @ 5x5
Convolution + ReLU 64 @ 7x7
Max-pooling @ 3x3
Convolution + ReLU 128 @ 3x3
Max-pooling @ 3x3
Fully-connected + Sigmoid 4096
Fully-connected + Sigmoid 4096
Fully-connected + Sigmoid 2098
Input
Feature Vector

(a) (b)

Figure 37: (a) Siamese network framework classifying Identical Processed (IP) and Differently Processed (DP). (b) CNN architecture used in each branch of the Siamese framework.

First there is a constrained convolutional layer. Then, there are two unconstrained convolutional layers each followed by a ReLU nonlinearity. Subsequently, there is a max-pooling layer and three fully-connected layers with a sigmoid nonlinearity. Finally, an image feature vector is extracted. As for the distance layer, given a pair of image patches (x_1, x_2) the Siamese framework extracts features (f_1, f_2). The distance layer computes the distance between them which is fed to a sigmoid output neuron. This neuron calculate the likelihood of the input image pair using the following metric:

$$p=\sigma\left(\sum_j a_j \left| f_1(j) - f_2(j) \right| \right)$$, where

σ is the sigmoid nonlinearity function and a_j a learnable significance indicator of the feature vectors in the classification of the patch-pair.

The network is trained by minimizing the average cross-entropy loss function C:

$$C=\frac{1}{M}\sum_{i=1}^{M} y\left(x_1^i, x_2^i\right)\log p\left(x_1^i, x_2^i\right) + \left(1-y\left(x_1^i, x_2^i\right)\right)\log\left(1-p\left(x_1^i, x_2^i\right)\right)$$, where

M is the number of images in each bach.

5.2.5 Multimodal Image-Repurposing Detection

Sabir et al. [58] introduce a new dataset, the Multimodal Entity Image Repurposing (MEIR), which has been created for better supporting research into the field of image repurposing detection. This dataset includes a variety of robust, labeled and coherent manipulations on real-world data sourced from Flickr. It is also presented a novel end-to-end deep multimodal learning model for detecting reproposing images, comparing the images' extracted information with the related information of a known database.

A new dataset with person, location and organization manipulations has been created. Examples of this dataset (MEIR) are presented in *Figure 38*. The red text indicates the manipulation in text and location data. There were three main stage which have to be done in order the MEIR dataset to be created. The first one was data curation in which it has to be ensured the data uniformity. The second stage was clustering packages by relatedness, helped to categorize packages to reference, training, development and test sets. The clustering stage consist of two substations. The first one was clustering by location and the second was image refining relating to image and text similarities. Finally, for image manipulation, StanfordNER's three class model was used for a variety of attacks in each package.

	Reference Dataset		Manipulated Dataset
	Related Package	Manipulation Source Package	Repurposed Query Package
Image			
Text	Ex Convento at Cuilapan De Guerrero completed in 1555	The Dorena Bridge has these neat windows , which kind makes you feel like you are in a house !	Ex Convento at Dorena Bridge completed in 1555
Location	Mexico, Cuilapam de Guerrero, Oaxaca, Cuilapam de Guerrero. GPS: 16.992657, -96.779133	United States, Lane, Oregon, Row River. GPS: 43.737553, -122.883646	United States, Lane, Oregon, Row River. GPS: 43.737553, -122.883646
Image			
Text	The 2001 Space pod modelled in Lego by Dilip	" It was a dream , and in dreams you have no choices: ..." - Neil Gaiman , American Gods (2001)	Space pod modelled in Lego by Neil Gaiman
Location	United Kingdom, West Sussex, England, Billingshurst. GPS: 51.024538, -0.450281	United States, Riverside, California, Moreno Valley. GPS: 33.900035, -117.254384	United Kingdom, West Sussex, England, Billingshurst. GPS: 51.024538, -0.450276
Image			
Text	The International Rally of Queensland Sunday competition images + podium	$10 , 000 , 000 00 is the estimated value for the first United States gold coin Numismatic Guaranty Corporation special display ...	The Numismatic Guaranty Corporation Sunday competition images + podium
Location	Australia, Gympie Regional, Queensland, Imbil. GPS: -26.460497, -152.679491	United States, Philadelphia, Pennsylvania, Philadelphia. GPS: 39.953333, -75.156667	Australia, Gympie Regional, Queensland, Imbil. GPS: -26.461131, 152.678117

Figure 38: Examples of location, person and organization manipulations from the MEIR dataset.

There are a few different modalities in a given package, so the process of repurposing detection begins with evaluating the significance of each edited modality and then continues with the manipulation detection approach.

Because of large overlapping information during the package retrieving procedure this paper introduce a technique to find each modality importance. It uses similarity scoring for retrieving a package from reference dataset R. Top retrieved package $r*¿$ is identified by:

$$r* = \underset{r \in R}{argmax} \sum_{m} s\left(f_{qm}, f_{rm}\right)$$

where f_{qm}, f_{rm} are the feature vectors for modality m in query q and reference r packages respectively. VGG and word2vec are used for embeddings of image and text. Trying to estimate the importance of each modality for manipulation detection, a model consisting Gaussian random projection for dimensionality reduction was created. Each modality is reduced to a common feature dimension L and then these features were concatenated. Finally, a random forest classification combined with Gini impurity across all trees which is a feature importance measurement implemented.

This paper focus a deep multimodal, multi-task learning model for image repurposing detection, which consists of four modules. The first module is related to feature extraction and the second module is responsible for feature balancing. Then, there is a module for package evaluation and finally one for integrity assessment. The model overview shown in *Figure 39*. The model inputs are a query package and the top-1 related package, retrieved from a reference dataset, shown in *Figure 40*. All modalities are concatenated into a single feature vector.

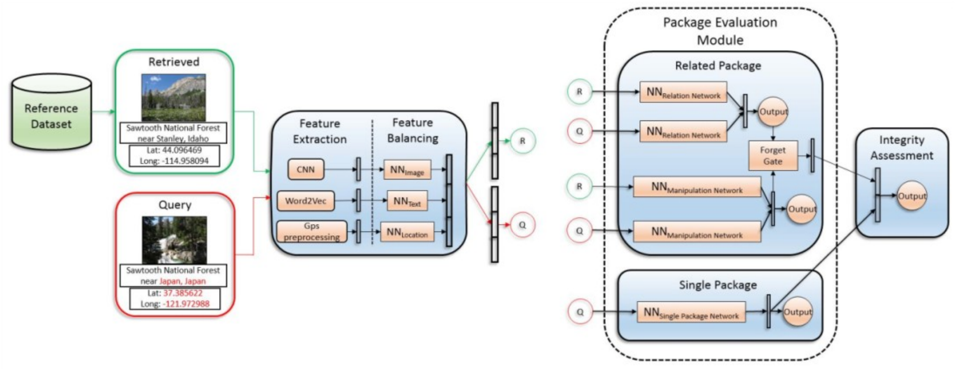


Figure 39: Query and Retrieved packages go through the feature extraction and feature balancing in order to create a concatenated feature vector. The evaluation module includes related and single package NNs layers.

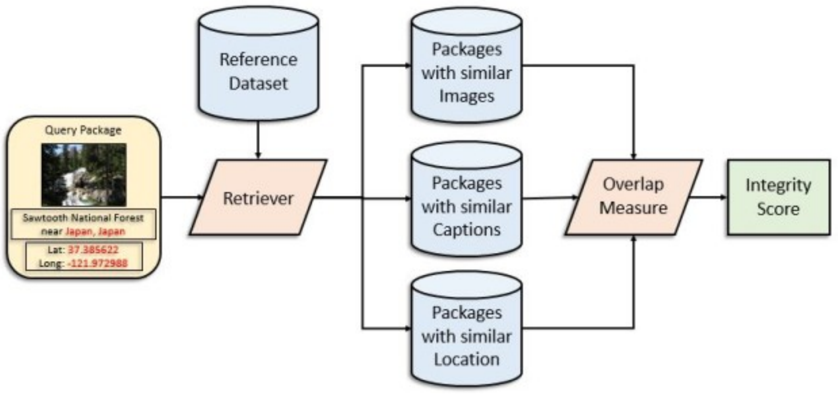

Reference Dataset
Query Package
Sawtooth National Forest near Japan, Japan
Lat: 37.385622
Long: -121.972988
Retriever
Packages with similar Images
Packages with similar Captions
Packages with similar Location
Overlap Measure
Integrity Score

Figure 40: The baseline semantic retrieval system (SRS). It focuses on retrieving similar concepts based on each package modality.

5.2.6 Beyond Pixels

Bharati et al. [59] tries to construct a graph which will give information about the provenance of an image based on conclusions about the scope and metadata applicability. It utilizes timestamps, geotags and camera IDs which give information about the image travel on the internet without large computational cost. In *Figure 41* there is an example of an Image Provenance Graph (IPG) showing some common manipulations applied on images and how they are inferred when it comes to provenance.

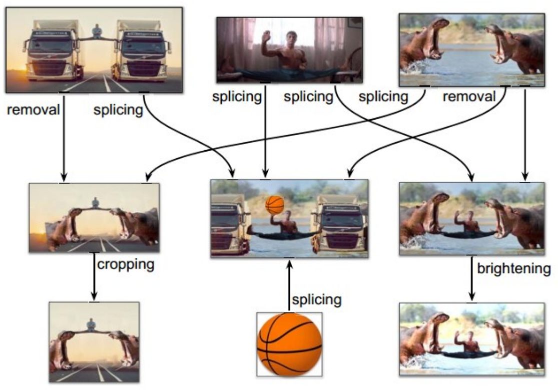


Figure 41: Example of an Image Provenance Graph (IPG)

The aim of image provenance analysis is the construction of a provenance graph with related images for each query image. This is a Directed Acyclic Graph (DAG) where each node represents an image in the set of similar images and the edges stand for the relationship of sharing duplicate content.

The proposed method starts with filtering images related to the query image Q. In this work, a subset of a million of images is retrieved based on the query. This solution use Optimized Product Quantization (OPQ) to store local Speeded-Up Robust Features (SURF) in an Inverted File index (IVF), with a large number (e.g., ~400k) of representative centroids. Then, the $k-most$ related images (R) are selected for pairwise image comparison for visual content, resulting in two types of $NxN(N=¿R\vee+1)$ adjacency matrix (D). Each index $D[i,j]$ indicates the similarity of image i,j. Besides the above matrix this work introduce also an metadata-based asymmetric adjacency matrix in order to determine the orientation of the pairwise image relations.

These matrices are used for formatting a direct provenance graph with the help of a specialized algorithm. The output graph can be represented as a binary adjacency matrix with each of its elements indicating the flow of content between a pair of images.

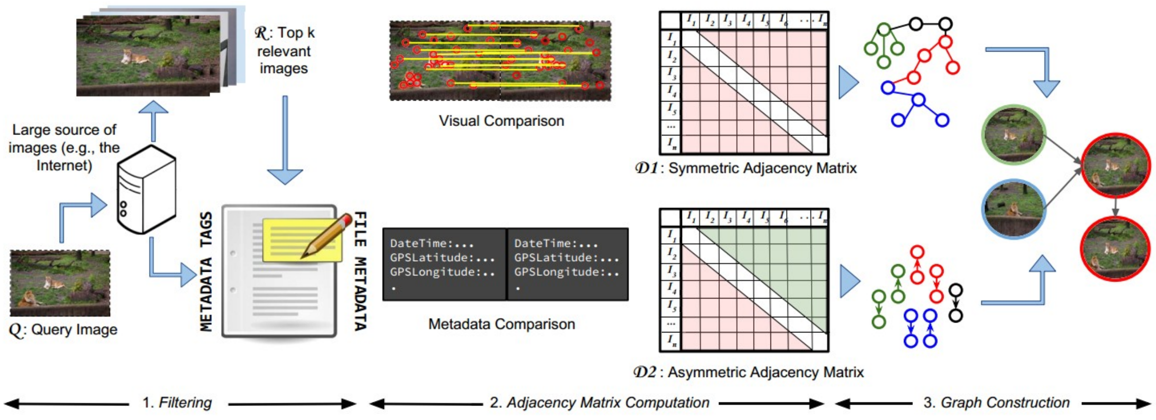

ℛ: Top k relevant images
Large source of images (e.g., the Internet)
Q: Query Image
METADATA TAGS
FILE METADATA
Visual Comparison
DateTime:... GPSLatitude:... GPSLongitude:...
Metadata Comparison
𝒟1: Symmetric Adjacency Matrix
𝒟2 : Asymmetric Adjacency Matrix
1. Filtering
2. Adjacency Matrix Computation
3. Graph Construction

Figure 42: Stages of image provenance analysis.

5.2.7 Fighting Fake News

Huh et al. [60] introduces a learning algorithm for detecting manipulations on images. His model was trained using automatically recorded EXIF metadata from real photographs as a supervisory signal so as the model being able to determine whether an image is self-consistent. Self-consistent definition indicates if each image different parts have been produced by a signal imaging pipeline.

The model focuses on predicting whether a pair of image patches are consistent with each other. Given two patches P_1, P_2 the model estimates the probabilities $x_1, x_2, \dots, x_n$ that they share the same values for each of n metadata attributes. Then, an overall consistency c_{ij} is calculated combining all n observations of metadata. More specifically an 83-dimensional vector (80 is all the attributes and 3 is an additional classification task per augmentation type) x of EXIF consistency is formed for pair of patches i and j.The overall consistency is estimated by:

$$c_{ij} = p_\theta(y \vee x)$$

where p_θ is a two layer MLP with 512 hidden units. The model is trained to predict if$i$ and $j$come from the same image ($y=0$ if they are different, otherwise $y=1$).

Moving from patch consistency to image self-consistency, these pairwise consistency probabilities have to be aggregated into a global self consistency score for the entire image. For each given patch a response map is produced corresponding to its consistency with every other patch in the image. Producing a single map which is going to merge all the patch responsive maps this paper implement mode-seeking using Mean Shift. The resulting map segments the image into consistent and inconsistent regions (*Figure 43*).

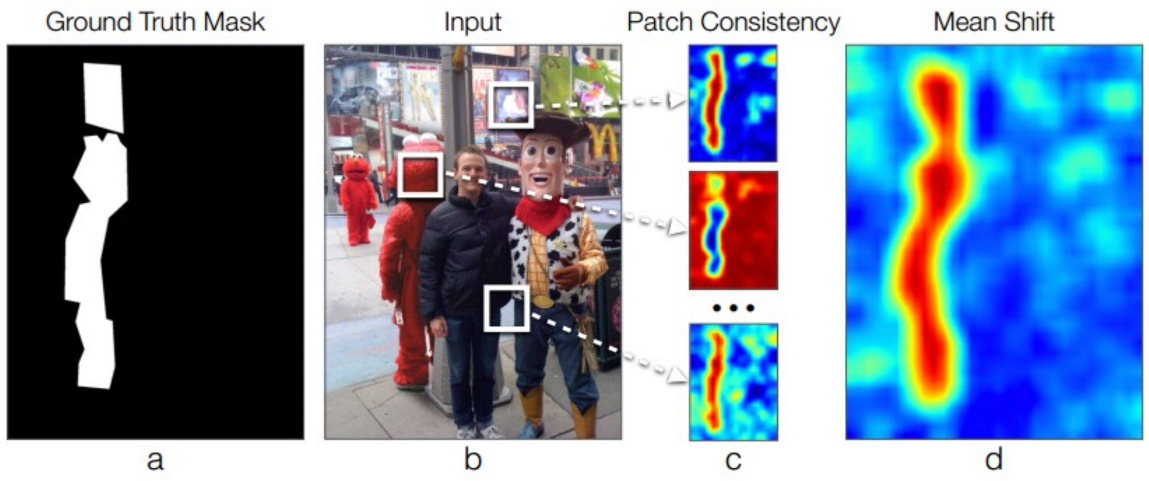


Figure 43: Test time. Starts with sampling patches from an input image and end with a prediction map indicating the whole image consistency.

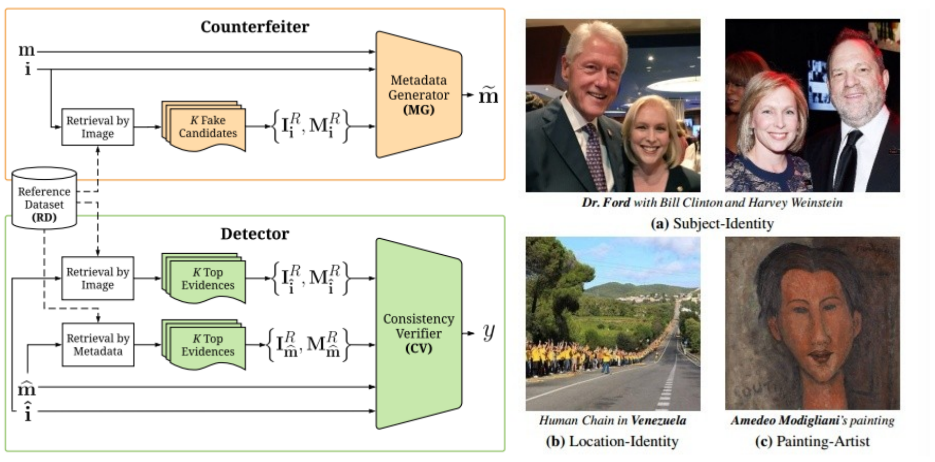
5.2.8 AIRD: Adversarial Image Repurposing Detection

Jaiswal et al. [61] presents a novel method which tries to simulate a real-world phenomenon between a bad actor who repurpose untampered images with reused or manipulated metadata to spread misinformation and a watchdog who verifies the semantic consistency between images and metadata. Both bad actor and watchdog have access to a reference dataset of verified content. This network can be trained with absence of training data containing manipulated data.

The proposed AIRD framework consists of two models illustrated in *Figure 44a*:

1. a counterfeiter (C): This model aims to create fake metadata for untampered real image which are in the reference dataset (*Figure 44b*),
2. a detector (D): This model tries to assess the semantic integrity of query packages by gathering evidence from the reference dataset.

Each modality of the multimedia packages is transformed into an information-rich representation using an encoder which is trained end-to-end with the metadata generator and consistency-verifier network. This encoding process helps the network to extract specific information and learn similarity between data instances.



(a) (b)

Figure 44: (a) Adversarial Image Repurposing Detection (AIRD), (b) Image repurposing in different domains from untampered images.

The counterfeiter has two mechanisms, namely fake candidates and metadata generator. Fake candidates mechanism queries the reference dataset retrieving $K-most$ similar images with dissimilar metadata aiming to manipulate these subject-identity metadata in order to reproduce K fake packages.

Both fake candidates and original package pass on to a metadata generator neural network (MG). It scores each of the K candidates by comparing them with the original image-metadata pair:

$s_k = CSSN((i, m), (i_k, m_k))$, where i_k and m_k are the *k-th* package in $\{I_i^R, M_i^R\}$.

In the end these scores are converted into a choice distribution through an attention-like softmax operation:

$$c_k = \frac{exp\left({}^{s_k}/_{\tau} \right)}{\sum_{j=1}^{K} exp\left({}^{s_j}/_{\tau} \right)} \; ; \tau \in (0,1]$$

The choice distribution c_k is multiplied element-wise with the metadata of the fake candidates m_k helping MG to produce fabricated metadata as the sum of these weighted candidate metadata.

$$\tilde{m} = \sum_{k=1}^{K} c_k \cdot m_k$$

There are also two mechanisms for the Detector model. The first one is gathering evidence, which is the process of collecting evidence from the reference dataset using them to validate query packages. This mechanism retrieves $k-most$similar packages from reference dataset using both image $(\hat{i})$ and metadata $(\hat{m})$ as queries. Thus, it collects two different kinds of evidence $\{I_{\hat{i}}^R, M_{\hat{i}}^R\}, \{I_{\hat{m}}^R, M_{\hat{m}}^R\}$which are image-based and metadata-based retrievals respectively.

In the subsequent stage, the consistency verifier neural network (CV) conducts semantic integrity verification between the retrieved evidence and the query package. The network performs first an aggregation of query and retrieved encodings and after that a cross-modality combination of information so as to evaluate the semantic integrity of query package. The final aggregated information used to make an integrity judgement using a final fully-connected layer is depicted below:

$$y=\sigma\left(W_y^T h_{cross}+b_y\right)$$

5.3. Methodology

Based on [62] we implement a similar Siamese Network for detecting near duplicate images with deep feature learning.

Feature Extraction

There are many different ways which can extract features for near duplicate image detection. For instance, Discrete Cosine Transform (DCT) extracts features which are coefficient-based, robust to noise but sensitive to rotate. Besides, there is Multi-Resoloution Histograms (MRH) and gist features which can produce quickly image features but these will be sensitive to geometric affine transformations. On the contrary, random transformation based High Order Invariant Moment (HOIM) forms embeddings which are robust to image rotation and scale variation but sensitive to local image editing. Furthermore, there are the Vector of Locally Aggregated Descriptors (VLAD), the Bag of Features (BoF) and the Scale Invariant Feature Transform (SIFT) which are cluster local features. These proposed features extraction techniques may achieve good performance on specific datasets mainly because they depend on human experience and skills but lack generalization capability.

In recent years, many deep learning CNN-based hashing methods have been proposed to be trained so as to extract image features for large scale image classification which tend to substitute the aforementioned techniques. In general, these embeddings produced automatically by deep learning are more generalized and effective than the features designed by human operators.

Feature Indexing

Today's databases are very large and as a result there is a significant computational cost when someone tries to retrieve data from a big database. Detecting near duplicate images is a two-stage procedure. The first stage groups the similar images into the same class in order to reduce the number of candidate matches to a query. The second stage starts with an exhaustively search to the results of the previous stage in order to retrieve the $k-most$ near duplicate images. This model is known as coarse-to-fine.

The Model

Latest hash coding networks based on deep learning of features usually use Convolutional Neural Networks Hashing (CNNH) and the pairwise supervised hashing network. The CNNH uses a semantic similarity indicator matrix S_{ij} which elements is equal to one if the input image pair (images i, j) is similar, otherwise it will be zero. The matrix is decomposed into hash codes for samples which are used for CNN's training.

The proposed network consists of two identical CNNs which have same structures and parameters. Their initial parameters are taken from AlexNet pretrained model on ImageNet dataset. Pairs of images are inserted into the network. In each of the CNNs architectures the FC8 fully connected layer having 1000 nodes has been replaced by a FC8 fully connected layer having *d* nodes. A new layer was added between the fully connected layer FC8 and the loss function layer in the CNN. This latent layer H has V nodes which indicates the number of extracted features. This layer maps the features extracted for FC7 to hash codes. A hyperbolic tangent (tanh) function used as the activation function of H layer formulated as:

$$\text{tan: } h(x) = \frac{exp(x) - exp(-x)}{exp(x) + exp(-x)}$$

where x is an input real value. The range of $\tanh$ function is $[-1,1]$, which is appropriate for the hash coding task.

Loss Function

The loss function layer consists of two functions. The first one is the contrastive loss function, which measures the similarity of each input pair of images and the second one is the regularization function which adds a binary constraint to the output of the latent layer H.

Let $\delta \in \{0,1\}$ be the similarity indicator, where $\delta = 1$ means that the input pair of images is nearly duplicate, otherwise $\delta = 0$. Let the a and $b$be the V- dimensional vectors, called approximate hash codes, which are produced by the latent layer H of the two CNNs. The real hash codes can be obtained by transforming the components of the approximate hash code vectors into integers. If a_u and b_u are the $u-th$ value of a and $b$vectors the contrastive loss function is defined as:

$$E_c = \frac{1}{2V} \sum_{u=1}^{V} ¿¿$$, where

the margin is used to adjust the effect of not nearly duplicate image pairs on the entire loss function. Only when the pairwise features difference is less than margin, their loss is included in the loss function. When the pair of images is near duplicate the contrastive loss is equal to the pairwise distance between the images' approximate hash codes and is minimized by making the output approximate hash code as identical as possible. On the contrary, when the pair of images is not siamese then the contrastive loss is minimized by making the output approximate hash codes as dissimilar as possible.

Furthermore, a two-value binary constraint term added to the loss function in order to bring closer to a binary format the approximate hash codes. The Hamming distance between two hash code vectors h_i, h_j of two images i, j respectively can be represented using the scalar product of h_i, h_j.

$$dis_H(h_i, h_j) = \frac{1}{2}\left(V - \langle h_i, h_j \rangle\right),$$

where is transformed using the cosine distance

$$dis_H(h_i, h_j) = \frac{V}{2}\left(1 - \cos(h_i, h_j)\right), \ where \quad \cos(h_i, h_j) = \frac{\langle h_i, h_j \rangle}{\|h_i\| \|h_j\|}$$

Let $\hat{a}$ be the vector whose $u-th$element is $\hat{a}[u] = ¿ a_u \vee$. To increase the quality of produced hash codes the bellow regularization factor is added to the entire loss function:

$$E_h = -(\cos(\hat{a}, l) + \cos(\hat{b}, l)), \text{ where}$$

l is a V-dimension vector with all of its elements equal to one. This function calculates the cosine distance between the absolute values of approximate hash codes and the vector l.
Hence, the entire loss function for this deep constrain siamese hash coding network is defined as: $E_c = E_c + E_h$, which includes the near duplicate distance information and the hash coding constraint.

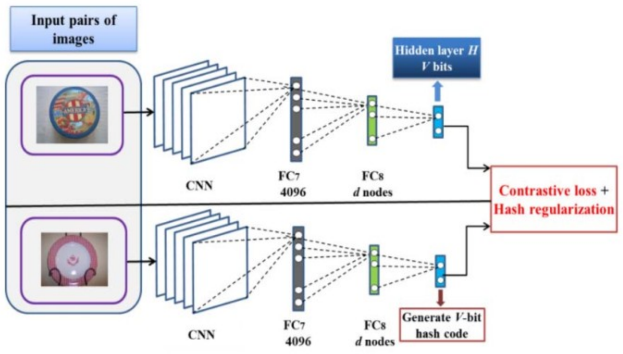


Figure 45: The architecture of deep constraint Siamese Network

Training phase

First of all, we consider similar or near duplicate pairs of images those pairs which consists of one original image and the corresponding manipulated one. The manipulated or siamese image may be the original one with different quality, size or shape. Also, manipulated images can be considered those which have subjected to some blur, cropping or photoshop editing. On the contrary, dissimilar images are those which is completely different.
We start training the model on our synthetic dataset which consists of images which had been edited. More specifically, it is composed of MSCOCO images on which some random elements have been added in order to create forged ones. We feed the network with pair of images indicating with delta parameter (δ) when this pair is similar or dissimilar. An example of similar pair of images is shown at *Figure 46*. The network has been trained using a 50 or 100 batch size depending on the existing resources.

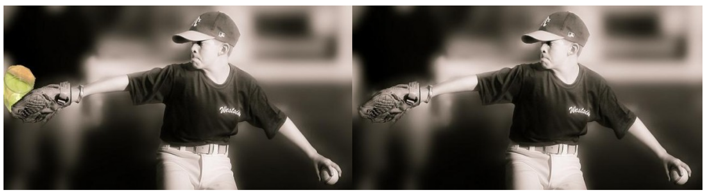


Figure 46: A potential image pair indicating similar images. The left one is forged while the right is the original one

Apart from this dataset we construct a new dataset using a variety of image augmentations to the original ones, such as flip (vertical or horizontal), gaussian noise, rotate, resize and crop. The original images which have been used came from MSCOCO dataset. We utilized 10,000 images with the above methods and produced a total of 50,000 manipulated images. Additional to them, we add another 10,000 edited images from the previous synthetic dataset (forged images). Hence, we have 60,000 manipulated images in total each of them is imported in the network with the corresponding original one, as a pair. Consequently, we have 60,000 pairs of images (one original image and the corresponding manipulated one) which are considered similar or near duplicates ($\delta=1$). Subsequently, we use only the MSCOCO original images to make 60,000 pairs with dissimilar images ($\delta=0$). So, we feed the network with equal number of similar and dissimilar pairs in order to learn how to extract better quality descriptors so as to distinguish satisfactorily both the similar and dissimilar images. We split this constructed dataset to 80% training set and 20% testing set. Also, from the training set we use the 80% for training and 20% for validation.

We make this augmented dataset so as to train our network capability to understand better the near duplicate images. That is why, in the first training we use only forged images from the first synthetic dataset. However, we testing this trained model with datasets such as COPYDAYS, which include either cropped images or images with different quality, and UK-BENCH, which has 2,550 different images each one with four different captions. So, we consider that if the network trained in a variety of manipulations, it may have better results when it comes to retrieve near duplicate images from a database.

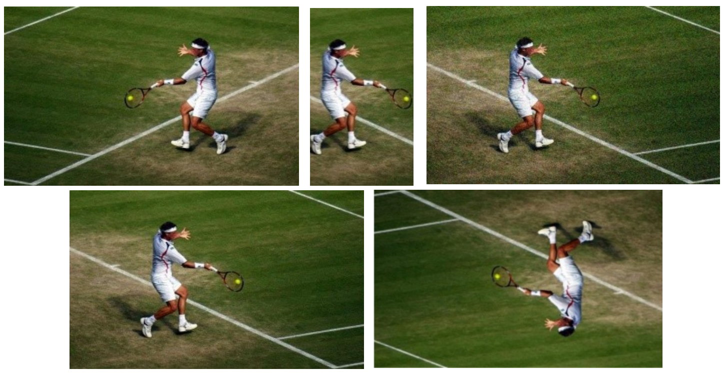


Figure 47: Augmentations implemented in images from MSCOCO dataset

Testing Phase

In the test phase, we started by building a database (DB) with image features extracted by our trained model, in order to be used for searching and retrieving similar images for every query image. Each image of our dataset comes first through the model (Deep Learning), then its features is transformed into a binary format (Binarization) and finally its binary feature vector is saved in a csv file. The extracted features are range between -1 and 1 and the corresponding binary vector is formated following the below conditions:

$$h_j = \begin{cases} 1 & out(H_j) > 0 \\ 0 & otherwise \end{cases}$$

After that, the query images are coming through the trained model which extracts binary features, following the same procedure like each image in the dataset, which has been used for the database creation. So, after the creation of the binary vector of a query image, we conduct an exhaustive search in our database and calculate its hamming distance from every image in the database.

The Hamming distance is the distance between two binary vectors. More specifically, it is equal with the number of different bits between the two vectors. This difference can be inferred as the number of dimensions in which the two images are different. Hence, first we calculate the hamming distance $d_h(q, I)$between each query q and all the images I in the database and then we sort in a descending order the database images based on their distance from the query. Finally, we retrieve the k- closer to the query images. The *Figure 48* shows the testing process.

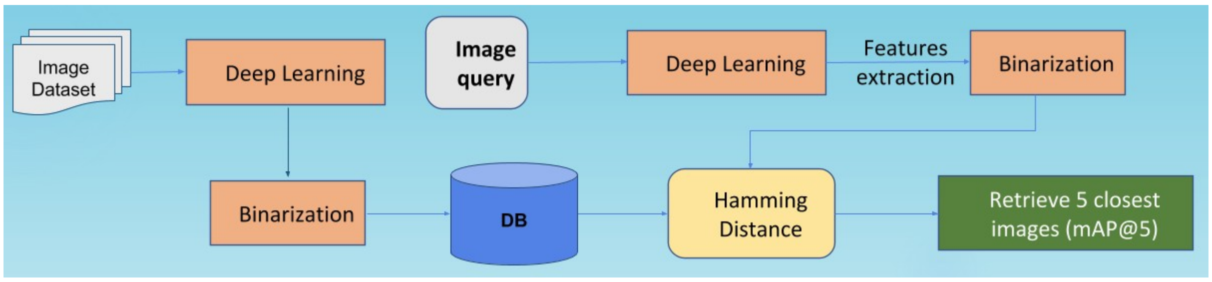


Figure 48: Testing process. First there is a creation of a database (DB) based on an image dataset. Then, a query is processed and after a database search system retrieves the five closest images

Implementation Details

Our implementation is based on the PyTorch framework. We use the *torchvision* library to inherit the CNN models. The *matplotlib.pyplot* library helps us to draw the loss function figure. We also use the *csv* library to write down the results of the network training and testing procedure. We conduct some experiments maintaining the *AlexNet CNN* architecture and after that we substitute it with *ResNet50*. Furthermore, we train different models being able to extract 32, 64, 128 bit features from the input images. All of the models are trained using the Accelerated Stochastic Gradient Descent (ASGD) optimizer with learning rate of 0.001 on Nvidia GeForce GTX 1070. We use also the Tesla K40c in order to train the models which extract 128bit vectors. The batch size was set to 100. Finally, in all the experiments the models are trained for 50 epochs.

Datasets

In order to specify the ability of our trained Siamese models in near duplicate image detection we conduct a variety of experiments. In the beginning, we searched both online and on the aforementioned papers for any available datasets created for this purpose. We found that there are such datasets like UKBENCH and INRIA COPYDAYS. The first one is a dataset containing 2,550 not similar images, each one captured in 4 different angles so as to create a dataset with a total of 10,200 images. Trying to adapt this dataset to our problem, we consider as near duplicate images, the pair of images which depicts the same image captioned in different angles. As for the second dataset, its folder contains 157 original images which have been suffered a variety of manipulations such as:

- Cropping: a 10% to 80% removing of original image area,
- Quality reduction: original images saved in a range of different JPEG quality factors (75, 50, 30, 20, 15, 10, 8, 5, 3)
- Strong attack**:** original images are subjected to print, scan, blur and paint attacks.

Furthermore, as we mentioned before we create one extra dataset both for training and testing. Trying to explain its creation better we start by downloading MSCOCO dataset which is a dataset with over 100K images and run some manipulations in those. We attacked in these images with different ways such as:

- Cropping: a random 50% of image is kept and the rest is deleted,
- Flip: every image suffered from both a horizontal and a vertical flip,
- Gaussian Noise: a random percentage of noise was added to each image,
- Items: these images are named forged and have an irrelevant object added in each one,
- rotate: a random angle rotate applied to each image,
- resize: a random percent applied to each image, changing each size, and
- rescale: a random percent applied to each image, changing each scale.

However, because of the fact that in today's world what matters is how the deep learning models perform in Big Data, we create a big dataset in order to evaluate our model better aiming to simulate how it will perform in more realistic conditions. When we talk about Big Data, it is considered that the corresponding dataset will contain over 1-2M (million) images.

In order to create one such dataset we take the advantage of *Open Images Dataset V5,* which is the latest version of *Open Images* (realized on May 2019). *Open Images* is a dataset of ~9M images annotated with image-level labels, object bounding boxes and segmentation masks and visual relationships. We download the provided *.tsv* files from a github repository[11] which were organized into test set, validation set and test set. The whole dataset is around 18TB so a python script was made which followed the below procedure:

1. Open the *.tsv* file, which contains the image urls,
2. Start download each image into a local file,
3. When it downloads 100 images, it drives them through our trained siamese models in order to create vectors which are the features of each image. The models which we used to extract these features were different variations of our trained Siamese Model (our variations include a range of trained Siamese models using either Alexnet or ResNet50 as CNN architecture and can produce 32-bit, 64-bit or 128-bit vectors for each image).
4. Then, we collect these vectors and we save them next to the corresponding image url into a new *.csv* file,
5. Finally, we delete the processed images and start downloading the next 100 images until we finish the *.tsv* file.

Hence, we managed to create a 1M image dataset in order to produce some reliable results about our models performance on finding near duplicate images in big datasets.

5.3.1. TESTING PROCESS

Overall, six models were trained and evaluated in different conditions. The three of them are using the Alexnet CNN architecture, while the other three are using the ResNet50 CNN architecture. The

11 https://github.com/cvdfoundation/open-images-dataset#download-full-dataset-with-google-storage-transfer

difference among the three trained siamese models using the same CNN architecture is that these models produce feature vectors of different dimensions (32-bit, 64-bit and 128-bit).

For the testing process, we pick some images from our constructed test-set to be used as queries. More specifically, we select 1,000 images from every manipulation category such as crop, flip, noise, etc. We keep also the original ones from those specific 1,000 attacked images. We use the originals to produce image descriptors and add them into our big dataset, the *.csv* file, which we create before. Finally, we conduct a test for each trained model (e.g. Alexnet-32bit, Resnet50-64bit, etc) for every group manipulation.

5.3.2. Metrics

It is a typical task in information retrieval for a user to provide a query to a database and retrieving information very similar to the query. To assess our models ability in retrieving similar information to a query, two evaluation metrics were used, namely mean average precision and mean average recall.

Mean Average Precision (mAP) is given by Equation (*6.1.2*) which indicates that, for a given query q, we calculate its corresponding *Average Precision* (AP), and then the mean of the all these APs scores would give us a single number, called the *mean Average Precision* (mAP), which quantifies how good our model is.

Calculation of AP

For a given query $q$we start calculating the distances $d(q,I)$ between the query and each image I in the database. Then, the images' distances are sorted either in ascending or descending order and the first k elements (*AP@k*) are returned.

$$AP@k = \frac{1}{GTP} \sum_{i=1}^{k} \frac{TPseen}{i} \quad (6.1.1)$$

The GTP refers to the ground truth positives for the query which in our occasion is set to 1 and the $TPseen$which refers to the number of true positives seen till k, which in our occasion also is either 1 or 0.

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \quad (6.1.2)$$

where N is the number of queries and $AP$is given from the equization 6.1.1.

Sometimes it does not matter the order of the returned correct images but the fact that in the returned image group there are the appropriate images. That is why we apply a second evaluation method, the mean average recall (6.1.4). For instance, a reporter cares more about finding an image among the k returned images from an over 1 million database rather than finding this specific image in the first place. Hence, we use the mean average recall to evaluate our model ability to include the correct image in the k returned images, regardless of the order.

$$recall@k = \begin{cases} 1, & when\ correct\ image\ in\ k\ returned \\ 0, & otherwise \end{cases} \quad (6.1.3)$$

$$mAR = \frac{1}{N}\sum_{i=1}^{N} recall_i \quad (6.1.4)$$

where N is the number of queries and $recall$ is given from the equization 6.1.3.

5.3.3. VISUALIZATION

Trying to visualize the results of our models, we pick some random image queries for which we depict the retrieval from the database in a picture, ensuring ourselves that our models perform fine. Thus, during the testing phase, we decide to retrieve the five closer images to a query and visualize them in a picture. Below, we present some *figures 49-53* of our successful tests visualizations. In the leftmost image we placed the query and the next five images are the retrieved ones, coming from the database. Each element of the Query results vector refers to the hamming distance between the query and each retrieved image.

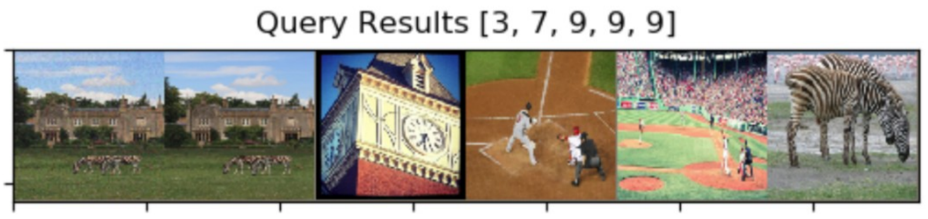


Figure 49: Test results on images which have been manipulated with gaussian noise.

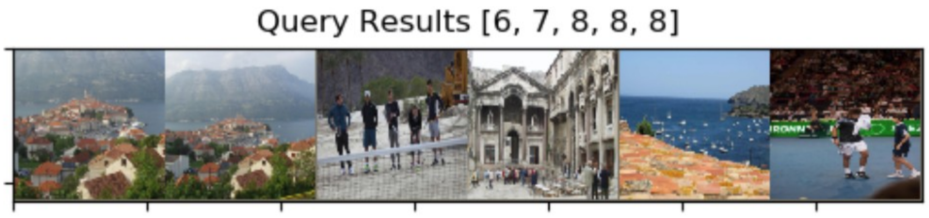


Figure 50: Test results on images which have been manipulated with cropping

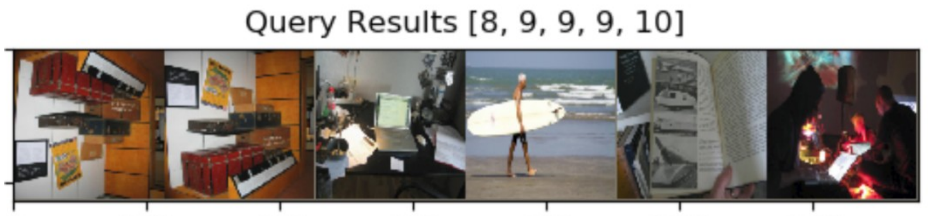


Figure 51: Test results on images which have been manipulated with horizontal flip

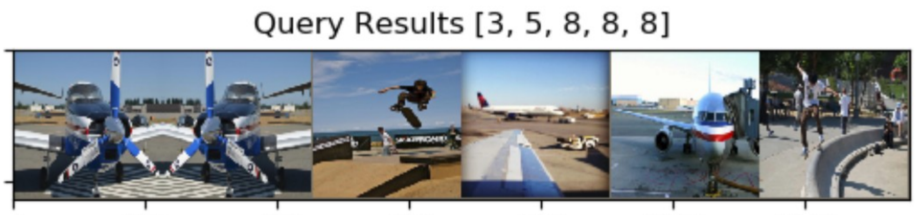


Figure 52: Test results on images which have been manipulated with vertical flip

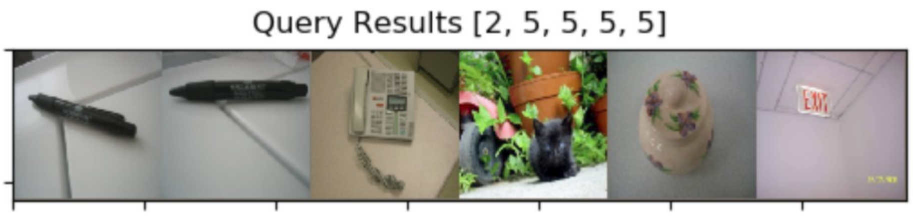


Figure 53: Test results on UKBENCH images

The *figures 54-56* present the performance of different models on the same query image. At *figure 54 Resnet50 32-bit* returns 5 images with close distances, none of which are similar to the query image. At *figure 55,* the results of *Resnet50 64-bit* are better as the model returns the correct image in third place. Finally, at the last experiment, *figure 56, Resnet50 128-bit* returns the correct image in the first place of the top five returned images. Hence, we can infer that as the vector size increases, the distance between dissimilar images are bigger and the results become better.

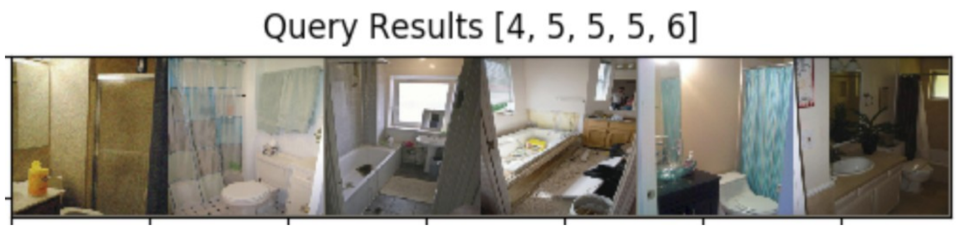


Figure 54: Test results on gaussian noise images from Resnet 32-bit

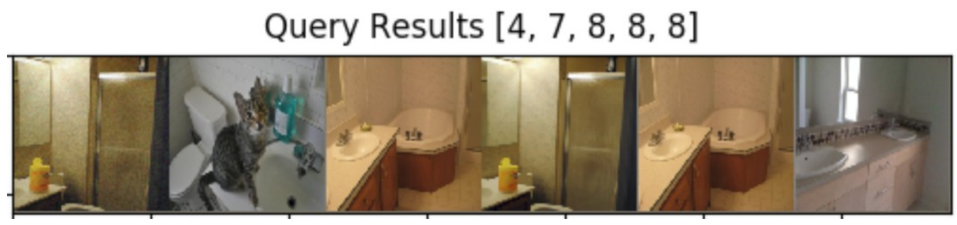


Figure 55: Test results on gaussian noise images from Resnet 64-bit

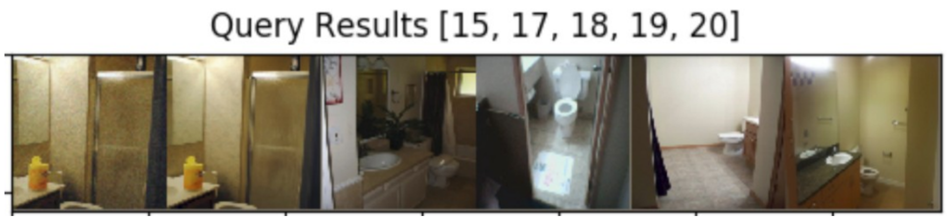


Figure 56: Test results on gaussian noise images from Resnet 128-bit

Experiments Results

In *tables 2 - 17* below we present our experiments results for our six trained models. We conduct tests on a variety of image manipulations such as flip, crop, forgery, etc. Also, in order to check our models' performance on different size database search, we conduct each experiment for 5 different database sizes (200K, 400K, 600K, 800K and 1M images). Hence, we are able to understand how the database size affects the precision of our models. Furthermore, from the below tables it seems clear that as we increase the extracted vector size, the results are improved.

At flip image test (*tables 2 and 3*) our Siamese model using Alexnet CNN architecture extracting 128-bit image feature vectors outperforms the others reaching 97,50% and 99,50% precision and recall respectively searching on 1M image Database.

mAP@15 - FLIP	Alexnet 32bit	Resnet50 32bit	Alexnet 64bit	Resnet50 64bit	Alexnet 128bit	Resnet50 128bit
200K	49,29%	43,29%	92,84%	73,70%	99,00%	91,43%
400K	46,41%	41,00%	91,08%	70,93%	98,32%	89,11%
600K	45,00%	40,23%	89,83%	68,41%	97,97%	87,54%
800K	43,80%	35,15%	88,58%	66,92%	97,95%	86,92%
1M	38,85%	29,72%	87,48%	59,88%	**97,50%**	86,44%

Table 2: Mean average precision (mAP@15) results for flip image test per model based on different database size

mAR@15 - FLIP	Alexnet 32bit	Resnet50 32bit	Alexnet 64bit	Resnet50 64bit	Alexnet 128bit	Resnet50 128bit
200K	68,00%	69,00%	100,00%	89,00%	100,00%	97,50%
400K	65,50%	67,20%	98,50%	89,00%	100,00%	97,50%
600K	64,50%	65,20%	96,50%	85,00%	100,00%	96,50%
800K	63,50%	61,50%	95,54%	83,00%	100,00%	94,50%
1M	59,00%	58,50%	95,06%	79,00%	**99,50%**	94,50%

Table 3: Mean average recall (mAR@15) results for flip image test per model based on different database size

In gaussian noise image test (*tables 4 and 5*) our Siamese model using Alexnet CNN architecture extracting 128-bit image feature vectors outperforms the others reaching 93% precision and 96,50% recall searching on 1M image Database for gaussian noise test. As for random angle rotate image test (*tables 6 and 7*) we achieve 35% precision and 45,50% recall with the same model.

mAP@15 - NOISE	Alexnet 32bit	Resnet50 32bit	Alexnet 64bit	Resnet50 64bit	Alexnet 128bit	Resnet50 128bit
200K	39,87%	24,81%	83,17%	46,31%	95,84%	77,82%
400K	39,64%	23,84%	80,42%	43,64%	95,02%	74,44%
600K	38,71%	23,47%	77,24%	42,63%	94,21%	71,67%
800K	38,28%	21,81%	74,08%	41,69%	93,90%	70,53%
1M	31,35%	19,13%	72,25%	38,56%	**93,00%**	65,37%

Table 4: Mean average precision (mAP@15) results for gaussian noise image test per model based on different database size

mAR@15 - NOISE	Alexnet 32bit	Resnet50 32bit	Alexnet 64bit	Resnet50 64bit	Alexnet 128bit	Resnet50 128bit

200K	58,67%	44,09%	94,12%	60,50%	99,00%	92,00%
400K	57,00%	42,80%	91,00%	58,00%	99,00%	90,00%
600K	55,50%	39,50%	90,00%	56,50%	98,00%	87,50%
800K	54,00%	38,00%	89,22%	56,00%	97,50%	84,50%
1M	51,00%	35,50%	87,25%	52,50%	**96,50%**	82,35%

Table 5: Mean average recall (mAR@15) results for gaussian noise image test per model based on different database size

mAP@15 - ROTATE	Alexnet 32bit	Resnet50 32bit	Alexnet 64bit	Resnet50 64bit	Alexnet 128bit	Resnet50 128bit
200K	11,44%	11,44%	31,67%	20,87%	44,03%	35,43%
400K	10,29%	10,19%	28,77%	20,22%	40,67%	30,91%
600K	9,97%	9,08%	26,93%	19,69%	38,58%	29,38%
800K	9,66%	8,60%	26,63%	18,82%	36,98%	27,43%
1M	6,93%	7,07%	26,25%	16,39%	**35,04%**	24,33%

Table 6: Mean average precision (mAP@15) results for random angle image rotate test per model based on different database size

mAR@15 - ROTATE	Alexnet 32bit	Resnet50 32bit	Alexnet 64bit	Resnet50 64bit	Alexnet 128bit	Resnet50 128bit
200K	16,67%	24,67%	43,14%	31,00%	54,00%	47,00%
400K	15,00%	22,20%	37,50%	28,50%	51,00%	45,00%
600K	14,50%	17,50%	34,00%	27,50%	49,50%	43,00%
800K	14,50%	16,50%	32,27%	25,50%	48,00%	41,50%
1M	12,50%	12,00%	31,33%	23,00%	**45,50%**	40,00%

Table 7: Mean average recall (mAR@15) results for random angle image rotate test per model based on different database size

In random 50% crop image test (*tables 8 and 9*) our Siamese model using Resnet50 CNN architecture extracting 128-bit image feature vectors outperforms the others. However, its results at top 15 returned images are not good, so we conduct the same experiment on Alexnet and Resnet 128-bit models returning the top 150 images (*tables 10 and 11*). Thus, the precision improved noticeably and the recall metric is good enough as the Resnet50 model returns at almost 73% the right image from 1M database.

mAP@15 - 50% CROP	Alexnet 32bit	Resnet50 32bit	Alexnet 64bit	Resnet50 64bit	Alexnet 128bit	Resnet50 128bit
200K	0,84%	0,69%	7,01%	11,96%	19,42%	25,79%
400K	0,67%	0,51%	5,36%	10,90%	16,29%	21,60%
600K	0,62%	0,49%	4,32%	10,25%	14,57%	19,86%
800K	0,53%	0,49%	2,68%	7,69%	11,68%	17,94%
1M	0,42%	0,40%	1,96%	5,83%	9,13%	**16,04%**

Table 8: Mean average precision (mAP@15) results for 50% random image crop test per model based on different database size

mAR@15 - 50% CROP	Alexnet	Resnet50	Alexnet	Resnet50	Alexnet	Resnet50

	32bit	32bit	64bit	64bit	128bit	128bit
200K	2,67%	1,47%	16,67%	24,00%	31,50%	45,50%
400K	2,50%	1,18%	14,50%	20,50%	24,00%	41,00%
600K	2,00%	1,18%	11,50%	19,50%	24,00%	36,50%
800K	2,00%	1,15%	8,50%	18,00%	23,50%	30,50%
1M	1,50%	0,85%	6,00%	14,50%	21,00%	**26,00%**

Table 9: Mean average recall (mAR@15) results for 50% random image crop test per model based on different database size

mAP@150 - CROP	Alexnet 128bit	Resnet50 128bit
200K	57,02%	56,79%
400K	48,51%	53,22%
600K	45,61%	50,07%
800K	42,51%	47,31%
1M	40,79%	**42,56%**

Table 10: Mean average precision (mAP@150) results for 50% random image crop test per model based on different database size. Here we return the top 150 images

mAR@150 - CROP	Alexnet 128bit	Resnet50 128bit
200K	78,37%	86,49%
400K	75,68%	78,38%
600K	74,43%	76,65%
800K	72,97%	73,68%
1M	70,04%	**72,98%**

Table 11: Mean average recall (mAR@150) results for 50% random image crop test per model based on different database size. Here we return the top 150 images

In 50% crop image test per model from COPYDAYS Dataset (*tables 12 and 13*) Alexnet 128-bit model outperforms the others reaching 47% and 67% precision and recall respectively at top 15 returned images from 1M database.

mAP@15 - COPYDAYS 50% CROP	Alexnet 32bit	Resnet50 32bit	Alexnet 64bit	Resnet50 64bit	Alexnet 128bit	Resnet50 128bit
200K	5,30%	20,49%	24,49%	26,00%	56,23%	39,54%
400K	2,66%	17,42%	20,42%	20,39%	53,22%	37,62%
600K	2,15%	10,81%	17,81%	18,66%	50,37%	35,21%
800K	1,96%	8,96%	16,83%	16,85%	48,63%	33,95%
1M	1,57%	6,17%	12,88%	14,95%	**47,04%**	31,65%

Table 12: Mean average precision (mAP@15) results for 50% crop image test per model, from COPYDAYS Dataset based on different database size

mAR@15 - COPYDAYS 50% CROP	Alexnet 32bit	Resnet50 32bit	Alexnet 64bit	Resnet50 64bit	Alexnet 128bit	Resnet50 128bit
200K	17,00%	31,00%	41,00%	53,00%	76,00%	60,50%
400K	11,00%	24,00%	34,00%	43,00%	72,00%	55,50%
600K	8,00%	20,00%	30,00%	39,00%	70,00%	52,50%
800K	7,00%	18,00%	27,00%	37,00%	68,50%	51,00%
1M	6,00%	16,00%	27,00%	36,00%	**67,00%**	50,00%

Table 13: Mean average recall (mAR@15) results for 50% crop image test per model, from COPYDAYS Dataset based on different database size

At forged image test (*tables 14 and 15*), Resnet50 reach 83,50% precision and 85% recall performing better than the other models on 1M database size.

mAP@15 - FORGED	Alexnet 32bit	Resnet50 32bit	Alexnet 64bit	Resnet50 64bit	Alexnet 128bit	Resnet50 128bit
200K	56,72%	57,22%	84,83%	73,35%	86,69%	87,22%
400K	55,84%	53,02%	82,66%	72,25%	83,50%	86,44%
600K	54,13%	50,07%	81,36%	71,40%	82,64%	85,00%
800K	48,86%	49,18%	80,44%	70,49%	81,51%	84,13%
1M	47,94%	47,75%	80,07%	69,14%	80,36%	**83,50%**

Table 14: Mean average precision (mAP@15) results for forged image test per model based on different database size

mAR@15 - FORGED	Alexnet 32bit	Resnet50 32bit	Alexnet 64bit	Resnet50 64bit	Alexnet 128bit	Resnet50 128bit
200K	69,00%	76,00%	86,00%	83,00%	88,00%	85,00%
400K	69,00%	73,00%	84,50%	83,00%	85,00%	85,00%
600K	68,00%	71,00%	83,00%	82,00%	84,52%	85,00%
800K	61,00%	71,00%	81,50%	81,00%	83,63%	85,00%
1M	58,50%	68,00%	80,00%	81,00%	83,00%	**85,00%**

Table 15: Mean average recall (mAR@15) results for forged image test per model based on different database size

At UKBENCH Dataset image test (*tables 16 and 17*), Alexnet CNN architecture presents better results than the other models on 1M database size reaching almost 70% precision and 83% recall.

mAP@15 - UKBENCH	Alexnet 32bit	Resnet50 32bit	Alexnet 64bit	Resnet50 64bit	Alexnet 128bit	Resnet50 128bit
200K	14,21%	15,82%	56,09%	32,67%	76,52%	62,56%
400K	11,01%	12,03%	50,42%	29,78%	74,56%	59,37%
600K	9,14%	10,57%	46,35%	27,05%	72,03%	56,52%
800K	8,29%	9,04%	45,89%	24,37%	70,68%	54,63%
1M	6,93%	7,92%	41,66%	22,97%	**69,56%**	52,66%

Table 16: Mean average precision (mAP@15) test results per model on UKBENCH Dataset based on different database size

mAR@15 - UKBENCH	Alexnet	Resnet50	Alexnet	Resnet50	Alexnet	Resnet50

	32bit	32bit	64bit	64bit	128bit	128bit
200K	28,00%	34,00%	77,00%	45,00%	85,00%	79,00%
400K	21,00%	27,00%	64,00%	40,00%	85,00%	77,00%
600K	18,00%	23,00%	59,00%	37,50%	83,00%	73,00%
800K	17,00%	21,00%	56,00%	36,00%	83,00%	73,00%
1M	16,00%	18,00%	53,00%	35,00%	**83,00%**	73,00%

Table 17: Mean average recall (mAR@15) test results per model on UKBENCH Dataset based on different database size

Overall, our results about images attacked by flip, forgery and gaussian noise are good enough. More specifically, our high scores in all experiments are achieved only when we extract 128-bit feature vector size as image representation. We achieve very good precision at *Flip* and *Noise* tests, while apart from *Rotate* and *Crop* tests we achieve a recall above 85% in all other experiments searching on 1M database. Also, considering that our *Crop* experiments are conducted on images with random 50% crop, in other words losing a random half of image information, we reach almost 73% recall at top 150 images on 1M dataset.

Trying to improve the model performance on UKBENCH dataset, on rotate image attack and especially on crop image attack we have already considered some possible improvements. First and foremost, we will study the semantics of cropping images, as random image crop may produce useless images with none semantic content which would be impossible to be used in any kind of article. Besides, as UKBENCH images and rotate image attack concern the spatial transformation of an original image we will implement a learnable layer like [120] which improves the performance of spatial transformer networks. Finally, we will train the Resnet50 model for more epochs as the above comparisons became for the same number of epochs, although the Resnet50 network is considerably larger than Alexnet.

SERVICES

Apart from retrieving similar images locally, this service is expanded online and is established onto a platform. Now, any person can paste an image url or upload an image file as a query to our user interface and search into our database for near duplicate images.

This database was held on ElasticSearch. We search in millions of articles from which we extracted every image they contain. In that way, we manage to collect a huge amount of images. More specifically, we created an independent index in ElasticSearch to store an array of image urls found into each article (each article may contain more than one image). So, after this procedure, we started process each url in this index in order to create a searchable database from which we will retrieve the k-most near duplicate images. We had to download all of these images locally and fed them to our siamese trained model in order to create image representations.

The model's output vectors with the corresponding image urls and image identifiers (ids) are stored into a separate ElasticSearch index named *image-descriptors*. According to our conducted experiments results we decided to extract image features from our 128 bit resnet50 trained model, as it had the best mean average precision (mAP) and recall (mAR) among all the other implemented models.

Also, we create an identifier for each image based on its url. This identifier is unique for each image and help us with image search into the database. More specifically, this unique *id* improves the image retrieval from the database and also it is used as a parameter to decide if a new input image already exists in the database. Thus, we will not have to process the same image again every time the service receives a url, we calculate the identifier and then search if this identifier exists into the database. If it does, the service does not download the image and proceed to the next url, otherwise it downloads the image into a local folder. After some time in which the folder gather a predefined number of images, equal to a batch size, the service fed these images into our siamese model and collects the vectors for each image.

Thus, for every image we create a document, which follows a JSON format described in *Table 1* and it is saved in the image descriptor index of ElasticSearch database:

field	type	example
identifier	SHA-256 hash object	eaa337ec8c17a379503c5036cc6504094868cdbe1ef700ea3e29f56600936b747db194109a
url	string	https://images.pexels.com/photos/414612/pexels-photo-414612.jpeg
descriptor	binary vector	[0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1]

Table 18: The table shows the fields and the types of an image descriptor JSON format which is stored in the Elasticsearch database, describing an image. Also it gives an example for every field in JSON file

This would be a useful tool for everyone who want to find near duplicate images. There is a user interface in which one can upload an image or paste a url and by pressing the “search” button our service will satisfy his/her request. First, the service will take the url or the image itself and it will calculate the query image identifier. The image identifier is calculated in a step-by-step procedure. We use the image url which first we encode into *UTF-8.* Then, using the sha256() we create a SHA-256 hash object from the *hashlib* library. In the subsequent step we ask for the *digest* of the concatenation of the data fed to it using the hexdigest() methods.

Having the image identifier, the service will search on the online database, which we described above, to find if this identifier exists. If it does, it will return the same image first, otherwise it will download the image and then it will feed the image into our siamese model. In the subsequent state, we will obtain an image descriptor vector which would be the query vector to our database.

At the beginning, in order to find the k closest images to the query, we calculate the hamming distance between the query image and every image from the whole database conducting an exhaustive search. The images which vectors differ less to the query will be returned and shown in the user interface. The difference is measured in the number of different bits between two binary vectors. More specifically, the service will create a dictionary with images' identifiers as keys and the images' distances from the query image as values. After that, it will sort that dictionary in a descending order and then it will return the k first image identifiers to the callback function which will be responsible to show to the user the most similar images.

LOAD-BALANCED LSH

Finding nearest neighbours is a very common task. You can think of applications like finding near duplicate or similar documents, images, audios or videos. Although using brute force to check for all possible combinations, will give you the exact nearest neighbour, it's not the optimal solution based on the computational cost. Approximate algorithms, which is an area of active research trying to accomplish this task reducing the computational cost. Despite, these algorithms do not guarantee to give always the right answer, their results indicate that they will provide a good approximation frequent enough. These algorithms are faster and scalable. Locality sensitive hashing (LSH) is one such algorithm. LSH has many applications such near-duplicate detection in which LSH is commonly used to deduplicate large quantities of documents, webpages, and other files.

Trying to improve the performance of our service's time search we implement a Locality Sensitive Hashing (LSH) algorithm. LSH corresponds to a family of functions (known as LSH families) to group data points into buckets so that data points *near each other are located in the same buckets with high probability*, while data points *far from each other are likely to be in different buckets (Figure 57)*. This makes it easier to identify observations with various degrees of similarity. Furthermore, it is an effective way of significantly reducing the dimensionality of data, while preserving the differentiability.

So, in general LSH is a hash function that aims to maximize collisions for similar objects following the definition below:

1. $Probability\left(h\left(a\right)=¿h\left(b\right)\right)$ is high if a and b are near,
2. $Probability\left(h\left(a\right)=¿h\left(b\right)\right)$ is low if a and b are far,
3. Time complexity to identify close items is sub-linear.

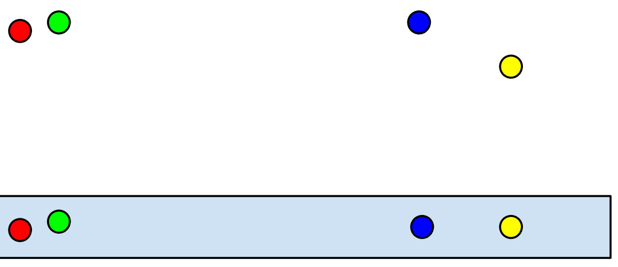


Figure 57: Locality sensitive hashing. The close data points (green and red) are placed into the same bucket while those which are not close (e.g. green and yellow) are placed into far away buckets

We followed the idea described in [88] and we implemented a load-balanced LSH algorithm. The key idea is that this implementation will hash image feature vectors into buckets with a balanced load. By doing this, the query vectors will not drop into large buckets improving the efficiency of index structure. So, we enforce an upper bound limit to each bucket. Considering B be the maximum number of buckets per hash table, n the number of samples, d the number of bits of hash codes and L the number of hash tables, the upper bound of load-balanced hash bucket is defined as:

$$\Delta_{LB} = next\ possitive\ integer\left[\frac{\left(dn + n^{1+1/c^2}\right)}{LB}\right]$$

We drive through four stages to implement the load-balanced LSH algorithm:

1. *Initialization*: To construct the basic LSH mapping function we use either a family of Hamming LSH functions (1) or a family of Euclidean distance LSH functions (2).

$$F = \{\eta : \eta(x_i) \rightarrow \{0,1\}\}_{i=1}^{d} \quad (1.1)$$

$$\eta_{w,b}(x) = \frac{w^T x + b}{r} \quad (1.2)$$

2. *Basic Hashing*: At the beginning using one of the aforementioned constructed functions $\{g_l()\}_{l=1}^{L}$corresponding to L hash tables, each sample is mapped into a bucket $g_l(x)$in each hash table l without considering the upper bound Δ_{LB}.

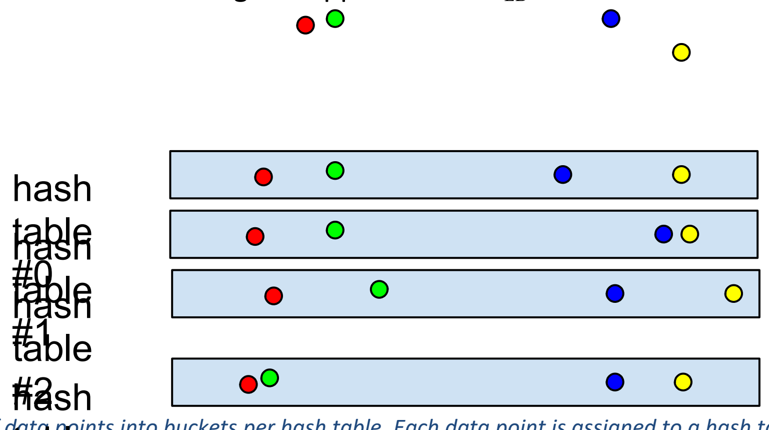


Figure 58: Distribution of data points into buckets per hash table. Each data point is assigned to a hash table bucket depending on the corresponding function gl for this hash table. Because of the fact that each hash table has different function the data points might not be in the same bucket per hash table

3. *Local Redistribution*: This step is responsible for balance the load of the buckets under the upper bound limit. First, using the initial hash tables we compute every bucket's virtual center VC which is the mean of feature vectors of initial tables per bucket:

$$VC_t = \frac{1}{n_t} \sum_{x \in bucket(t)} x$$

where $n_t$is the initial number of samples per bucket. After the calculation of virtual centers for every bucket per hash table, we iterate over each bucket and check if its number of samples exceed the upper bucket bound. If it does, we measure the distances between the samples and the corresponding bucket virtual center and we form a dictionary which after we sort in an ascending order and pop out the k samples which are over the predefined number of samples. These samples are transferred to the next bucket of the same hash table. The redistribution process is continued going through each bucket for every hash table and it is terminated when all buckets number of samples are below the upper bound limit.

4. *Neighbor-Probe Search*: In order to be compatible with the Redistribution procedure, during the search phase, the load-balanced LSH probes more than one buckets in order to retrieve the appropriate image. Thus, a query image q is first mapped into a bucket by applying to it the above created function $\{g_l()\}_{l=1}^{L}$ per hash table. Then, we probe the specific bucket $g_l(q)$ and then the next φ buckets to find the k most near duplicate images per hash table. The φ parameter is determined as follows:
$$\varphi = \frac{\Delta_{LB}}{\Delta_{LB} - M_l}$$
where the M_l is the average number of samples of every bucket per hash table. As a result, finally we conduct an exhaustive search among the retrieved images per hash table and we return the $k-most$ nearby to query. This is a way which help to retrieve near duplicate images more efficiently.

After the implementation of the load-balanced LSH algorithm we start processing every image descriptor in order to construct the initial LSH map. Hence, we obtain a LSH map with L hash tables and Bbuckets per hash table. Then, we proceed to the third step in order the buckets to be conformed with the upper bucket limit. After that, we create an elasticsearch index *"LSH-map"* which will contain the hash tables and the buckets per hash table identifiers. Also, each bucket will include the images' identifiers which have been located in that bucket. Thus, after we complete the fourth step we obtain the bucket's identifiers which we should retrieve per hash table. We make a query to the elasticsearch database to retrieve the exact buckets and then we use each bucket of image identifiers to take the image descriptors from the *"image-descriptors"* index and calculate the distance between them and the query. Finally, we return the image identifiers with the lowest distance from the query.

6. NEAR DUPLICATE VIDEOS FOR TEMPORAL UNFOLDING

In a similar fashion as with the images, the platform has to detect if a video, or part of a video, has been reposted in the past.

6.1. State-of-the-Art

6.1.1 NetVLAD

Arandjelovic et al [63] proposed NetVLAD, a leanable VLAD layer able to be plugged into any CNN architecture. A CNN architecture, without the last convolutional layer, is designed in order to learn image representation and output a dense descriptor. The network outputs a $HxWxD$map which is a D-dimensional vector for each HxW spatial location. So, a new pooling layer, NetVLAD, was designed to pool the extracted descriptors into a single image representation.

VLAD is a popular descriptor pooling method for both image classification and instance retrieval. Generally, given N D- dimensional local image descriptors (x_i) as input and K cluster centers (c_k) as VLAD parameters, the network output would be a V KxD dimensional matrix which then, after normalization, is transformed into a vector. The (j,k) element of V is computed as:

$$V(i,j) = \sum_{i=1}^{N} a_k(x_i)\left(x_j(j) - c_k(j)\right)$$

where $x_i(j)$ and $c_k(j)$ is the $j-th$ dimension of the $i-th$ descriptor and $k-th$ cluster center respectively. The $a_k(x_i)$ indicates how close is the c_k cluster to the descriptor x_i (set to 1 if it is the closest, otherwise 0). In order to make the operation differentiable, the a is substituted with a soft assignment of descriptors to multiple clusters. The final form of VLAD descriptor is:

$$V(j,k) = \sum_{i=1}^{N} \frac{e^{w_k^T x_i + b_k}}{\sum_{k'} e^{w_{k'}^T x_i + b_{k'}}}\left(x_i(j) - c_k(j)\right) \qquad (6.1)$$

where w_k, b_k, c_k are sets of trainable parameters for each cluster k. The NetVLAD layer aggregates the first order statistics of residuals $(x_i - c_k)$ in different parts of the descriptor space, weighted by soft-assignment $a_k(x_i)$ of descriptor $x_i$to cluster k. The final result is a trainable layer end-to-end producing image representation on the target task.

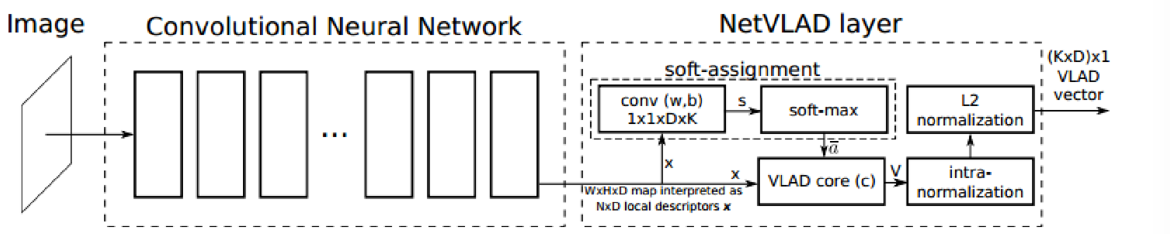


Figure 59: CNN architecture followed by a NetVLAD layer which is implemented using convolutional layers, softmax L2-normalization and VLAD core to perform aggregation in equation (6.1)

6.1.2 Near-Duplicate Video Retrieval by Aggregating Intermediate CNN Layers

Kordopatis-Zilos et al. [64] tries to solve the Near-Duplicate Video Retrieval (NDVR) problem by aggregating the intermediate (layer-based) CNN features in a global video descriptor.
In this work, visual features are extracted using three deep network architecture, namely AlexNet, VGGNet and GoogleNet. Using a pre-trained CNN network C with L convolutional layers ($L^1, L^2, \dots, L^L$), a total of L feature maps are generated, denoted as: $M^l \in R\left(n_d^l x n_d^l x c^l\right), with\left(l=1, \dots, L\right)$, where $n_d^l x n_d^l$ is the dimension of every channel for convolution layer and L^l and c^l the total number of channels. There are two main methods for aggregating features from layers into a single descriptor:

1. *Vector Aggregation*: A vector $u^c$is formated from the concatenation of individual layer features. Then, a bag-of-words scheme is applied on $u^c$to create a codebook of k visual words denoted as $C_k=\{t_1, t_2, \dots, t_k\}$.Following this, every video keyframe is assigned to the nearest visual word and subsequently each frame f descriptor u_f^c is aggregating to the nearest visual word $t_f = NN\left(u_f^c\right)$ hence its $H_f$contains only a single visual word.
2. *Layer Aggregation*: For each intermediate CNN layer a Kwords codebook is generated. These L layer-specific codebooks extract separate bag-of-words representations. So, every frame f is represented by a frame-level histogram H_f, which is the outcome of the concatenation of each layer-specific histogram.

The corresponding outcome of both of the above methods would be a frame level histogram H_f representing a frame. Finally, by applying a plain summing on every H_f, a video level histogram H_u is derived (*Figure 60*).

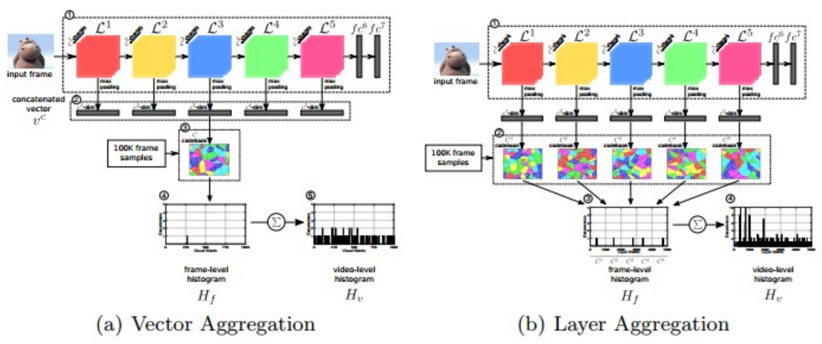


Figure 60: The two aggregation schemes and the final video representation

Given a video d with $¿F\vee¿$ keyframes, $\left(F=\{f_1, f_2, \dots, f_F\}\right)$, the video-level histogram H_u is derived by $H_u = \sum_{f_i \in F} H_{f_i}$.For finding video similarity the following equation is used: $w_{td} = n_{td} \cdot \log \vee D_d \vee ¿ n_t$, which calculates $tf-idf$ weights for every visual word in every video collection D_d. As for the feature extraction and aggregation steps for a query video q are the same. A final histogram H_u^q is extracted form q an inverted file indexing structure and placed into an inverted file indexing for fast retrieval. All videos are sorted in a descending order based on their cosine similarity with the q, using the corresponding $tf-idf$ representations.

6.1.3. ActionVLAD

Girdhar et al. [65] it is proposed a new way of constructing video embeddings for action classification that aggregates the local convolutional features from whole video spatio-temporal extent. The integrated model consists of a two stream state-of-the-art networks focusing on learnable spatio-temporal feature aggregation. In the beginning, the network samples frames from videos and produce features per frame for both their appearance (RGB) and their motion (flow) using "action words" vocabulary. Then, it aggregates them into a single video-level fixed-length vector. This representation goes through a classifier which decide the final classification score (*Figure 61*).

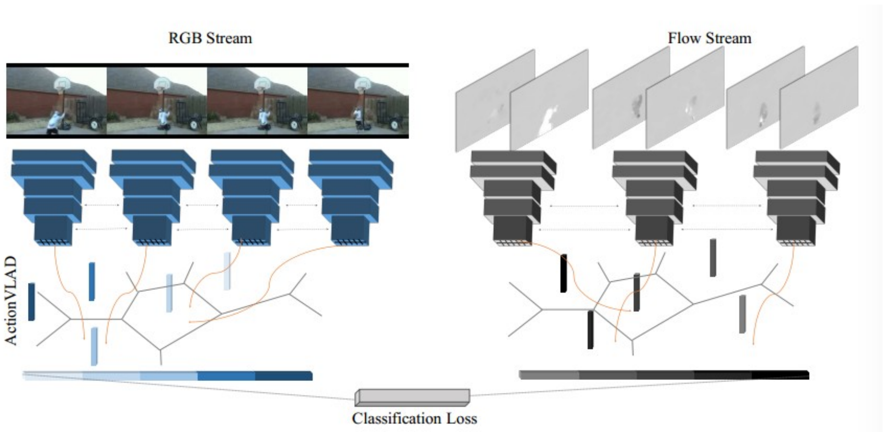


Figure 61: Network architecture. A VGG-16 CNN architecture is used to extract features from appearance and motion frames of video which then are pooled across space and time using the trainable end-to-end ActionVLAD layer with the classification loss

Let $x_{i,t} \in R^D$ be a D- dimensional local descriptor extracted from video spatial location $i \in \{1 ... N\}$ and frame $t \in \{1 ... T\}$. The goal is to aggregate these descriptors both spatially and temporally without losing any information from the video. This procedure starts with dividing the descriptor space R^D into K cells using K "action words". These are represented by anchor points $\{c_k\}$ to which each $x_{i,t}$ is assigned, represented by a residual vector $x_{i,t} - c_k$, indicating the difference between descriptor and anchor point. The Vmatrix represents the $V[\cdot, k]$ aggregated descriptor in the $k-th$ cell:

$$V[j,k] = \sum_{t=1}^{T} \sum_{i=1}^{N} \frac{e^{-a\|x_{it}-c_k\|^2}}{\sum_{k'} e^{-a\|x_{it}-c_k\|^2}} \left(x_{it}[j] - c_k[j] \right)$$

where $x_{i,t}[j]$ and $c_k[j]$are the $j-th$ component of the descriptor vector and anchor respectively and a is a tunable hyper-parameter. The first factor represents the aggregation over time and the second one the aggregation over space. All the columns of V matrix are intra-normalized, stacked and L2-normalized into a single descriptor $u \in R^{KD}$. This spatio-temporal extension is denoted as *ActionVLAD.*Generally, the spatio-temporal aggregation layer can be placed anywhere in the network to pool the corresponding feature maps. First a frame-level classifier is trained using all videos' frames and then at the testing phase, the network averages the predictions of T uniformly sampled frames. This frame-level pre-trained model is used as a feature generator providing input from different frames to the trainable ActionVLAD layer. There were few different layers activations which were possible to be pooled, such as the output of fully-connected layers ($1 x 1 4096$- dimensional output) or the convolutional layers outputs (conv4_3, conv5_3). But after conducting experiments, it was shown that the best performance obtained by pooling features at highest convolutional layer (conv5_3 for VGG-16). There are three main strategies for combining the appearance and action streams (*Figure 59*). The first one is a single ActionVLAD layer on the top of the concatenated flow and RGB features ((a) Concat Fusion), resulting in better correlations between visual and flow features for codebook construction. Second, a single ActionVLAD layer over all flow and RGB features ((b) Early Fusion), which helps model learn a single descriptor x_{ij} (visual and flow features) exploiting redundancy in features. Finally, the (c) Late Fusion which is a method using two separate ActionVLAD layers for weighted average of the appearance and motion features, learning specific representations for each input modality.

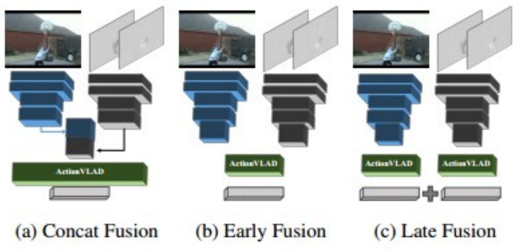


Figure 62: The three strategies for combining the appearance and motion streams

6.1.4. Aggregating Frame-level Features

Chen et al [66] explored a standard RNN and several variants as a way to learn a global descriptor for frame level features. Also, a trainable VLAD layer was implemented to aggregate features in temporal dimension. Besides, feature transformation is employed to train models on features from different time scales. The model produces a single video-level representation by aggregating the frame features and then this representation goes through a Mixture of Experts (MoE) classifier (*Figure 60*). In this paper the frame level representations are extracted either by using *variants of RNNs* (e.g. LSTMs, GRUs) or by using *VLAD Aggregation.* Generally, an RNN takes a sequence $(x_1, x_2, \ldots, x_T)$ as input and operates on this step by step from $t=1$ to $t=T$, producing an output h_t at cell state c_t. The RNN cell function is: $h_t, c_t = f(x_t, c_{t-1})$. Having a sequence of $(c_1, c_2, \ldots, c_T)$, the $c_T$is considered that represents the whole sequence of data.
On the contrary, the VLAD layer (described above) is used to pool video frame features on the temporal axis. Given the sequence of frame features $(x_1, x_2, \ldots, x_T)$, which is TxD- dimensional (T depends on video length), they pool frame features into a fixed length KxD- dimensional descriptor. K is a parameter which is adjusted following a trade-of between computation cost and

performance. They first randomly sample S out of T frame features denoted by $R=(r_1, r_2, \ldots, r_T)$, which is a SxD-dimensional matrix. The cluster mean is $(u_1, u_2, \ldots, u_K)$ which is a KxD- dimensional trainable parameter. Then, a matrix $A=(a_1, a_2, \ldots, a_S)$ is computed by $1\,D$- convolving R into a SxK - dimensional. Then, a soft-max is applied to A so that $\sum_{k=1}^{K} a_{ik} = 1$. The aggregated descriptor is computed by:

$$v_k = \sum_{i=1}^{S} a_{ik}\left(r_i - u_k\right)$$

These descriptors are concatenated to construct the new video representations. This method compared with variants of RNNs has lower computational cost. The proposed model use label filtering in the training procedure to better predict labels with small occurrence probability, while discard labels with high occurrence probability because they are focused on other models with full set of labels. This filtering is useful when a dataset has imbalanced class distribution. The final prediction is an outcome of multiple models linear weighted combination. First, there is an intra-model fusion stage in which the predictions are calculated by fusing the checkpoints predictions saved after each epoch for each model. Second, they fuse the predictions, generated from different models in the previous stage, to get the final prediction score. This stage is denoted as inter-model fusion. To decide the fusion weights they try the upcoming simple techniques: empirical fusion weights, brute-force search of fusion weights and learning for fusion weights.

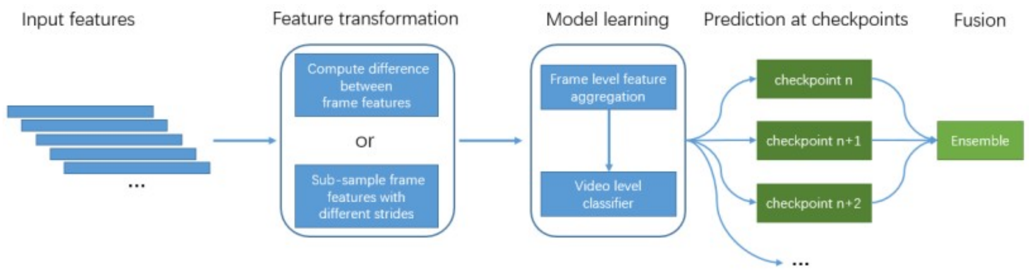

Input features
...
Feature transformation
Compute difference between frame features
or
Sub-sample frame features with different strides
Model learning
Frame level feature aggregation
Video level classifier
Prediction at checkpoints
checkpoint n
checkpoint n+1
checkpoint n+2
...
Fusion
Ensemble

Figure 63: The frame-level prediction model pipeline procedure. The checkpoints are different predictions saved in different iteration during training phase. In the end, these checkpoints are fused producing a final prediction

6.1.5 Learnable pooling with Context Gating

Miech et al. [67] investigates alternative methods for temporal aggregation proposing a two branch architecture aggregating audio and visual features by exploring the clustering-based aggregation. Then, a learnable non-linear unit, named Context Gating, is introduced modeling interdependencies among network activations. This work contributes in video and audio classification by (a) introducing a state-of-the-art architecture aggregating video and audio frames features, (b) Context Gating and (c) indicates the benefits of clustering-based aggregation in contrast with LSTM and GRU.There are three main subsequent modules which this work follows for video classification (*Figure 61*). First, they extract the input features from video and audio signals. In the subsequent stage, a single compact video representation, enhanced by Context Gating layer is constructed by aggregating the input features with the pooling module, treating visual and audio features separately. Finally, the MoE (Mixture-of-Experts) classification module takes the video embedding and calculate a score for a pre-defined set of labels, which is followed by another Context Gating layer.

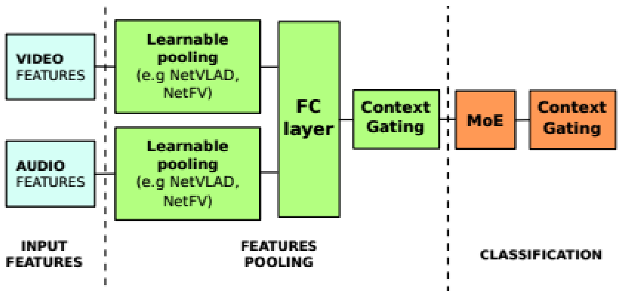


Figure 64: Overview of network architecture for video classification with MoE and Context Gating

The transformation of video embedding from X to Y by: $Y=\sigma(WX+b)\circ X$ has two-fold meaning. First, the Context Gating module help model to predict human-generated tags for a video by re-weighting both the features and the output labels of this architecture. This module create dependencies between visual activations by assigning different weights according to the importance of each network activation. Furthermore, the Context Gating can create dependencies among output class scores, helping modeling biases in label annotations. This paper focuses on non-recurrent aggregation techniques due to the fact that recurrent models have demanding computational cost and their sequence modeling is not necessary for feature aggregation problem. The NetVLAD architecture has been proposed for place recognition and then is extended to action recognition, as it is described above. Its main idea is to calculate a descriptor x_i hard assignment to the cluster k as a soft assignment:

$$a_k(x_i) = \frac{e^{w_k^T x_i + b_k}}{\sum_{j=1}^{K} e^{w_j^T x_i + b_j}}$$

The NetVLAD descriptor x_i to cluster k can be written as:

$$VLAD(j,k) = \sum_{i=1}^{N} a_k(x_i)\left(x_i(j) - c_k(j)\right)$$

computing the weighted sum of residuals $x_i - c_k$of descriptors x_v from learnable anchor point c_k in cluster k. By implementing this cluster soft-assignment idea on BOW (Bag-of-visual-words) and Fisher Vectors, they managed to obtain a differentiable representation. For BOW:

$$BOW(k) = \sum_{i=1}^{N} a_k(x_i)$$

The advantage of BOW over NetVLAD is, that given a predefined number of clusters, the first method aggregates a group of feature descriptors into a more concrete representation. On the other hand, the disadvantage is, the BOW aggregation needs a considerable larger number of clusters to produce a good representation.

Exploiting the idea of Fisher Vector encoding, they try to transform the NetVLAD architecture so as to proceed in learning of second order feature statistics within the clusters. The NetFV representation can be written as:

$$FV1(j,k) = \sum_{i=1}^{N} a_k(x_i)\left(\frac{x_i(j) - c_k(j)}{\sigma_k(j)}\right)$$

$$FV2(j,k) = \sum_{i=1}^{N} a_k(x_i)\left(\left(\frac{x_i(j) - c_k(j)}{\sigma_k(j)}\right)^2 - 1\right)$$

with $FV1$ and $FV2$capturing first and second order statistics respectively. Finally they propose another method of aggregation, the NetRVLAD (for Residual-less VLAD), averaging the actual descriptors instead of residuals. This method requires less parameters and computing operations and can be described as:

$$RVLAD(j,k) = \sum_{i=1}^{N} a_k(x_i)(x_i(j))$$

6.1.6. Non-local NetVLAD Encoding

Tang et al. [68] there is a fusion of six different video descriptor sub-models into a single computational graph, which are categorized into three families for the video classification task. The first family is the model with non-local operations following the NetVLAD encoding, the second is Soft-BoF and third is GRU.As it has been described before, the VLAD descriptor uses cluster centers c_k to represent features, while the NetVLAD uses soft assignment descriptor to produce the local feature descriptors. To enhance the information of NetVLAD descriptors they correlate the relations between different local clusters by applying the non-local block proposed by Wang and adopting the embedded Gaussian function to compute the non-local relations: $f(v_i, v_j) = e^{\theta(v_i)^T \varphi(v_i)}$. So, the non-local NetVLAD descriptor $\widehat{v_k}$ of cluster k is: $\widehat{v_k} = W y_i + v_i$, where v_k is the NetVLAD descriptor and $y_i = \frac{1}{Z(v)} \sum_{\forall j} f(v_i, v_j) g(v_j)$, where $g(v)$ is a linear transformation.

In this system there are three different non-local NetVLAD methods, complementary with each other. The Late-fused Non-local NetVLAD (LFNL-NetVLAD) and the Late-fused Non-local NetRVLAD (LFNL-NetRVLAD) operates with the same way. First, the pre-extracted visual and audio features are encoded independently and after that they go through the non-local NetVLAD or NetRVLAD pooling method respectively, to perform aggregation. Then, both vectors proceed to context gating module and the MoE equipped with video level context gating. On the contrary, the Early-fused Non-local NetVLAD (EFNL-NetVLAD) aggregates the video and audio features before they pass through a non-local NetVLAD layer. The frame level context gating and video level MoE are also used in this model. For the Bag-of-Feature encoding they utilize soft-assignment of descriptors to feature clusters to obtain the distinguishable representation. For Gated Recurrent Unit, they put two GRU layers back to back and the results saw that it is complementary with the non-local NetVLAD and Soft-BoF producing a considerable increase at the results with the model combination. The fusion of models is a common technique for improving the final results by providing better feature expressions and relief overfitting.

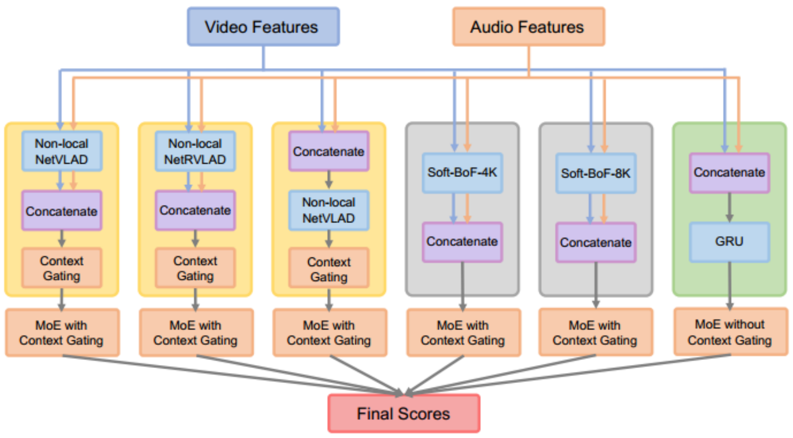


Figure 65: The framework of proposed system for video classification

6.1.7. CIRCULANT MATRICES FOR VIDEO CLASSIFICATION

Araujo et al. [69] contributes to define a novel architecture for video classification based on circulant matrices. Besides, it is proposed a new pooling technique improving the Deep Bag-of-Frames embedding. Also, by fine-tuning their architecture, they achieve the best trade-off between size and accuracy. Finally, it is presented a fusion of models into a single one, in order to produce a new model trained end-to-end which would be better than each model individual.A nxn circulant matrix C is a matrix where each row is a cyclic right shift of the previous one as illustrated below. One benefit of this type of matrix is that it can be represented in memory using one vector of n real values. Furthermore, it is computationally efficient especially on GPUs.

$$C = circ(c) = \begin{bmatrix} c_0 & c_{n-1} & c_{n-2} & \cdots & c_1 \\ c_1 & c_0 & c_{n-1} & & c_2 \\ c_2 & c_1 & c_0 & & c_3 \\ \vdots & & & \ddots & \vdots \\ c_{n-1} & c_{n-2} & c_{n-3} & & c_0 \end{bmatrix}$$

The proposed architecture for video classification is presented in *Figure 66*. The network produces embeddings for visual and audio random input samples frames independently, using Deep Bag-of-Frames. Then, two separated Fully-Connected layers reduce the dimensionality of video and audio embeddings vectors and merge them into a single one. The final results are classified with Mixture-of-Experts and Context Gating layer to calculate the final probabilities.

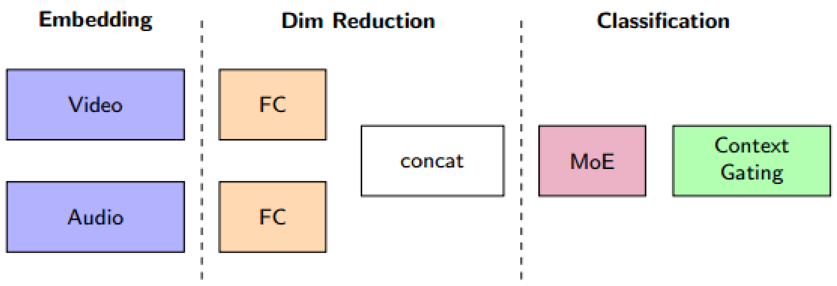


Figure 66: The architecture used for video classification on [95]

In order for the base model of BoF to perform better it is proposed a new pooling method which takes several samples of frames, applying upsampling followed by the maximum pooling to these samples and then averaging over all. The output of the robust-DBoF is: $\frac{1}{n}\sum max\{uxW : u\in S_i\}$. In order to take advantage of several model embeddings, the *Figure 67* below shows a transformation of the first architecture. First video and audio frames levels are processed by several embeddings models which then go through FC layer so as to reduce their dimensions. Then, that vectors are averaged and concatenated and fed into the final classification block.

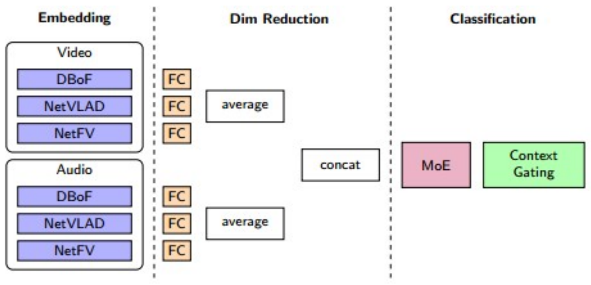


Figure 67: The evolution of architecture in Figure 62 to an ensemble architecture with several embeddings

6.1.8. Sequential Video VLAD

The proposal of recurrent convolutional networks (RCNs) is a native framework for learning the spatio-temporal video features. Youjiang Xu et al. [70], developed the SeqVLAD, a novel sequential vector of locally aggregated descriptor (VLAD) layer. This layer tries to combine the VLAD trainable encoding procedure and the RCNs architecture in order to produce sequential convolutional feature maps extracted from successive video frames. These maps are fed into the RCNs to learn spatio-temporal assignment parameters aggregating both detail spatial and fine montion information. Furthermore, this paper proposed an improved version of GRUs, the Shared GRU-RCN, employed to learn spatio-temporal assignments.

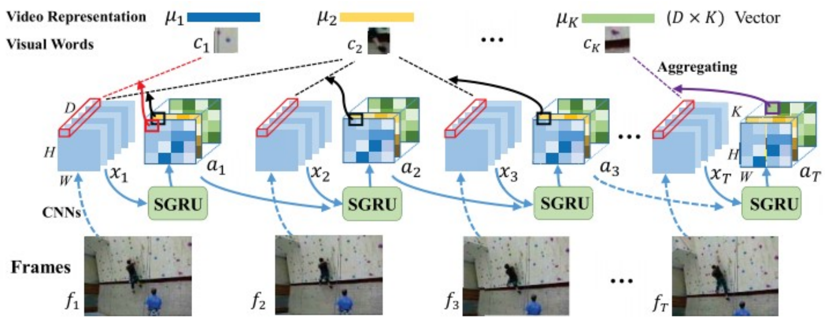


Figure 68: The flowchart of the proposed Sequential Video VLAD

The RCNs focus more on global appearance changes and neglect the flow information among successive video frames. The GRU-RCN model replace the fully-connected unit with convolution operation in a recurrent unit. To capture temporal cues that take place in different spatial resolution, the Stacked-RCN was proposed to take advantage of hidden representations from different depth layers (*Figure 66*). However, this proposed network has negative effect on model performance and also comes closer to overfitting. So, it is proposed in this paper a Shared GRU-RCN which reduces considerably the number of parameters and besides avoids the overfitting.

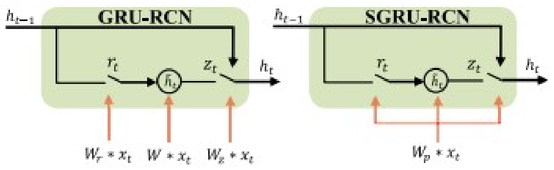


Figure 69: Illustration of GRU-RCN (left) and SGRU-RCN (right)

Based on the above proposed module which manage to maintain the spatial topology by convolution operation and capture temporal information by RNN architecture. So, they use this module to train the aggregation both locally and temporally. The Sequential VLAD encoding is:

$$\mu_k = \sum_{t=1}^{T} \sum_{j=1}^{W} \sum_{i=1}^{H} a_t^k(i,j)\left(x_y(i,j) - c_k\right)$$

where $x_t(i,j)$is the local descriptor at location (i,j).

Trying to compare the SeqVLAD with the NetVLAD and ActionVLAD it can be inferred that the SeqVLAD learns the aggregation parameters not only from the locally spatial content of each frame but also from the temporal information of successive frames. Also, by combining the trainable VLAD and RCN, it might have a positive potential to create a discriminative video representation with fine and rich spatio-temporal encoded information.

6.2. Methodology

6.2.1. Frame-Level Descriptors

Video is actually a stack of images, which are denoted as frames. Between the sequential frames of each video there are both spatial and temporal dependencies, which we try to represent into feature vectors aiming to obtain a single video descriptor which would describe a video preserving as much information as possible.

To begin with, for every inserted video we extract three frames per second considering that we will not lose any valuable information for video similarity and subsequently we do not overload our system. Due to the fact that we face the near video detection problem, we determine that we should use our near duplicate image detection model so as the similar frames of different videos to have the same as possible feature vectors. Thus, we drive each video frame through our aforementioned trained Siamese model, extracting for each frame a single N- dimensional vector, and as a result we obtain a MxNmatrix, where M is the number of extracted frames per video, which represents a whole video. In the subsequent stage, we try to aggregate these video frames features into single compact vector for video representation which would include as much as possible information about the video.

Hence, we decided to implement the NeXtVLAD [71] aggregation network which is a state-of-the-art model in order to acquire a single D- dimensional vector for every video MxNmatrix which goes through this model.

6.2.2. The NeXtVLAD Aggregation Network

Every video frame go first through our Siamese model and is represented by a descriptor .Then, the NeXtVLAD model follows (*Figure 67*). The input frame level descriptor x_i is inserted into a linear fully-connected layer (FC) which expands it as $\dot{x}_i$ with a λNdimension. Following that, a reshape operation is applied to $\dot{x}_i$ transforming it from $(M, \lambda N)$ to $\tilde{x}$ with shape $(M, G, \lambda N/G)$, where G is the size of groups and M is the number of frames per video. In other words, the $\dot{x}_i$ is splitted into Glower-dimensional feature vectors $\{\tilde{x}_i^g \mid g \in \{1, \ldots, G\}\}$. Each one of these vectors is represented subsequently as a mixture of residuals from cluster anchor point c_k, in the same lower-dimension space:

$$u_{ijk}^g = a_g(\dot{x}_i)\, a_{gk}(\dot{x}_i)\left(\tilde{x}_{ij}^g - c_{kj}\right)$$
$$g \in \{1, \ldots, G\}, i \in \{1, \ldots, M\}, j \in \{1, \ldots, \lambda N/G\}, k \in \{1, \ldots, K\}$$

where the proximity measurement for cluster k of the decomposed vector $\tilde{x}_i^g$consists of two parts: the a_{gk}(*eq. 6.2.1*) which measures the soft assignment of $\tilde{x}_i^g$to the cluster$k$and the a_g (*eq. 6.2.2*) which can be considered as an attention function over groups.

$$a_{gk}(\dot{x}_i) = \frac{e^{w_{gk}^T \dot{x}_i + b_{gk}}}{\sum_{s=1}^{K} e^{w_{gs}^T \dot{x}_i + b_{gs}}} \qquad (6.2.1)$$

$$a_g(\dot{x}_i) = \sigma\left(w_g^T \dot{x}_i + b_g\right) \qquad (6.2.2)$$

where $\{w_k\}$, $\{b_k\}$and $\{c_k\}$are sets of trainable parameters for each cluster k.

Finally, the video descriptor vector is derived from the below equation which aggregates the encoded vectors over time and groups:

$$y_{jk} = \sum_{i,g} u_{ijk}^g$$

After that it is applied an intra-normalization operation and a dimension reduction fully-connected layer (FC). The final representation ends up to a video-level classifier. In that way the dimension of video level descriptor would be reduced by Gtimes compared to NetVLAD, and as a result, this fact reduces also the number of network parameters which total number is:

$$\lambda N(N+G+K(G+(H+1)/G))$$

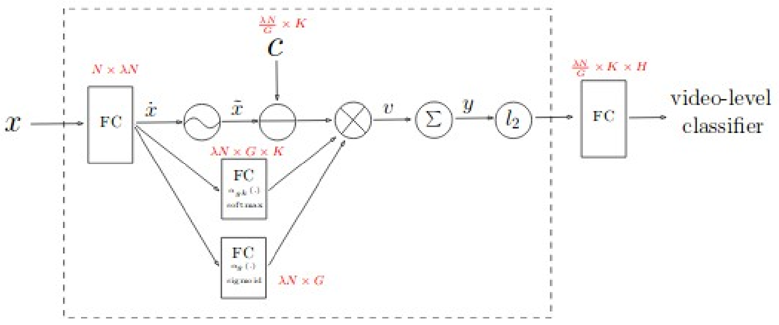


Figure 70: The NeXtVLAD architecture for video classification. The red descriptions show the number of parameters and the wave operation is the reshape transformation

6.2.3. NeXtVLAD Model combined with Context Gating

Video and audio features are encoded and aggregated separately going through a two stream network. These two aggregated vectors are then concatenated into a final representation which is enhanced by a SE Context Gating module with purpose to model the dependencies among labels. At the end, there is a logistic classifier with sigmoid activation for video-level multi-label classification (*Figure 71*).

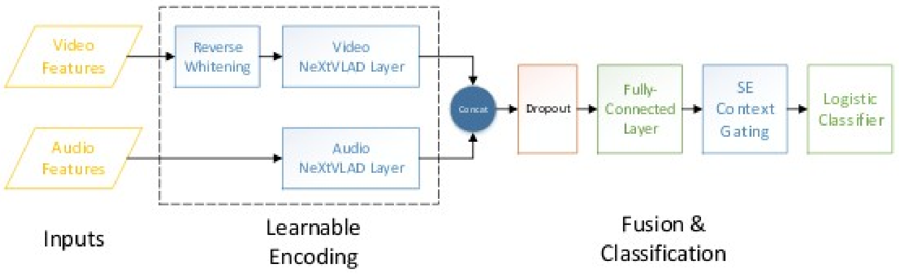


Figure 71: Overview of the NeXtVLAD model for video classification for Youtube-8M dataset

The proposed SE (Squeeze-and-Excitation) Context Gating (*Figure 72*) consists of two fully-connected layers with less parameters than the original Context Gating layer [67]. The total number of parameters is given by $(2F^2)/r$, where r is the reduction ratio and F is the feature size of the descriptor. Reversing the whitening process applied after performing PCA dimensionality reduction of frame-level features is beneficial for the performance of NeXtVLAD model. The reason for this is that feature space is transformed by eliminating different contributions between feature dimensions with regard to distance measurements, which may help the encoder to find better anchor points and soft assignments for each input feature. The whitening process is reversed by: $\hat{x}_j = x_j * \sqrt{(e_j)}$, where x_j and $\hat{x}_j$ are the input and the reverse vector respectively.

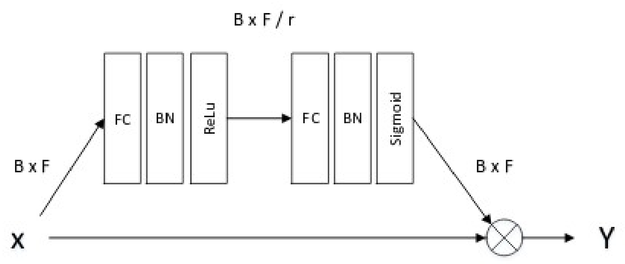


Figure 72: The architecture of the proposed Context Gating. The FC and BN stands for Fully-Connected layers and Batch Normalization respectively. B represents the batch size while F the feature size

7. CONCLUSIONS

The Fandango platform aims to provide the tools necessary for someone to discern if a news item is valid or not. In order for a news item to be valid, all of the modalities of the article must be authentic and unaltered but also the different modalities must be consistent in semantic, spatial and temporal level.

In section 2, the tools that were developed and employed to provide the system with clues on all three levels from textual information are presented. Using topic modeling each article is described, on the semantic level, in a fixed number of topics per language. Additional semantic information can be extracted by detecting the people and organizations that are referred to in the text. The locations that are detected by the named entity recognition process offer an estimation of the spatial context of the articles.

The detection of objects in images and their relative location is very important in understanding the semantics of an image. In section 3, the state-of-the-art in instance segmentation is analyzed and our methodology for extracting semantic information from images entered into the platform, is presented. Based on the needs of the end users, the currently implemented system can be retrained in order to detect specialized object categories, instead or additionally to everyday objects.

The problem of geo-localizing an image based on the visual content is still an open research issue. Extensive work is currently being done in detecting if an image depicts a specific location from a set of locations (i.e. landmarks) as well as for the retrieval of images, from a pool of images that depict the same location as a query image. By using the methodology presented in section 4, a location for every image in the platform can be provided, based on the visual content of the image.

A complete pipeline of the ingestion, indexing and retrieval process for detecting similar images is presented in section 5. The vector representing the image is extracted from a siamese model's last layer. The representation vector is subsequently binarized so that binary operations can be performed for calculating the similarity distance. In order to tackle the huge amount data that need to be searched, the LSH hashing algorithm is used. The training of the siamese model is on an augmented dataset where multiple forms of manipulation attacks have been performed on the original images. The retrieved images can be put on a timeline, offering a fast insight on the temporal context of an image as well as an image provenance graph even for parts of an image.

A video is actually a stack of images with a high dependency between them both on the spatial and the temporal axis. Therefore the same model as in image similarity can be used to provide a per frame representation of a video. The frame representations are aggregated so that a summary video representation is extracted.

8. References

[1] R. Panda, A. Pensia, N. Mehta, M. Zhou, and P. Rai, “Deep Topic Models for Multi-label Learning,” *Proc. Mach. Learn. Res.*, vol. 89, pp. 2849–2857, 2019.

[2] H.-F. Yu, P. Jain, P. Kar, and I. S. Dhillon, “Wasabi2,” *Proc. 31th Int. Conf. Mach. Learn. (ICML ’14)*, vol. 32, pp. 593–601, 2014.

[3] B. Jang and A. Hero, “Minimum Volume Topic Modeling,” vol. 89, 2019.

[4] M. Yurochkin and X. Nguyen, “Geometric Dirichlet Means algorithm for topic inference,” no. 2012, 2016.

[5] P. Jähnichen, F. Wenzel, M. Kloft, and S. Mandt, “Scalable Generalized Dynamic Topic Models,” vol. 84, 2018.

[6] M. C. Hughes *et al.*, “Semi-Supervised Prediction-Constrained Topic Models,” vol. 84, no. 2003, 2018.

[7] K. Huang, X. Fu, and N. D. Sidiropoulos, “Learning Hidden Markov Models from Pairwise Co-occurrences with Application to Topic Modeling,” 2018.

[8] S. Arora *et al.*, “A Practical Algorithm for Topic Modeling with Provable Guarantees,” vol. 28, 2012.

[9] H. Zhao, L. Du, W. Buntine, and M. Zhou, “Inter and Intra Topic Structure Learning with Word Embeddings,” *Proc. 35th Int. Conf. Mach. Learn.*, vol. 80, pp. 5892–5901, 2018.

[10] M. Zhou, Y. Cong, and B. Chen, “Augmentable gamma belief networks,” *J. Mach. Learn. Res.*, vol. 17, pp. 1–44, 2016.

[11] B. Esmaeili, H. Huang, B. C. Wallace, and J.-W. van de Meent, “Structured Neural Topic Models for Reviews,” vol. 89, 2018.

[12] F. Rodrigues, M. Lourenco, B. Ribeiro, and F. C. Pereira, “Learning Supervised Topic Models for Classification and Regression from Crowds,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2409–2422, 2017.

[13] D. M. Blei, “www.cs.princeton.edu/~blei/papers/Blei2012.pdf,” *Cs.Princeton.Edu*, pp. 77–84.

[14] S. Zhang, L. He, S. Vucetic, and E. Dragut, “Regular Expression Guided Entity Mention Mining from Noisy Web Data,” pp. 1991–2000, 2019.

[15] J. Xie, Z. Yang, G. Neubig, N. A. Smith, and J. Carbonell, “Neural Cross-Lingual Named Entity Recognition with Minimal Resources,” pp. 369–379, 2019.

[16] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, “Neural Architectures for Named Entity Recognition,” *Proc. 2016 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol.*, pp. 260–270, 2016.

[17] J. Shang, L. Liu, X. Gu, X. Ren, T. Ren, and J. Han, “Learning Named Entity Tagger using Domain-Specific Dictionary,” pp. 2054–2064, 2019.

[18] A. Graves, J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM

networks.", *Proc*. IEEE International Joint Conference on Neural Networks, 2005.

[19] P. F. Felzenszwalb, R. B. Girshick, D. Mcallester, and D. Ramanan, “Object detection with discriminatively trained part-based models._IEEE transactions on pattern analysis and machine intelligence_2010_Felzenszwalb et al.pdf,” pp. 1–20, 2009.

[20] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 580–587, 2014.

[21] M. für I. W. Forschung und Technologie des Landes Nordrhein-Westfalen, “Ziel- und Leistungsvereinbarungen 2007 -- 2010. Hochschulen in Nordrhein-Westfalen,” pp. 91–99, 2015.

[22] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 779–788, 2016.

[23] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei, “Fully convolutional instance-aware semantic segmentation,” *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 4438–4446, 2017.

[24] J. Dai, K. He, Y. Li, S. Ren, and J. Sun, “Instance-sensitive fully convolutional networks,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9910 LNCS, pp. 534–549, 2016.

[25] J. Dai, Y. Li, K. He, and J. Sun, “R-FCN: Object Detection via Region-based Fully Convolutional Networks,” 2016.

[26] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 770–778, 2016.

[27] G. P. Mallat Stephane, *A Wavelet Tour of Signal Processing The Sparse Way ´ phane Mallat*. 2009.

[28] K. He, G. Gkioxari, P. Dollar, and R. Girshick, “Mask R-CNN,” *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2017-Octob, pp. 2980–2988, 2017.

[29] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, “Path Aggregation Network for Instance Segmentation,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 8759–8768, 2018.

[30] C. Michaelis, I. Ustyuzhaninov, M. Bethge, and A. S. Ecker, “One-Shot Instance Segmentation,” 2018.

[31] J. Snell, K. Swersky, and R. S. Zemel, “Prototypical Networks for Few-shot Learning,” no. Nips, 2017.

[32] W. Liu *et al.*, “SSD: Single shot multibox detector,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9905 LNCS, pp. 21–37, 2016.

[33] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” 2018.

[34] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, “YOLACT: Real-time Instance Segmentation,” 2019.

[35] T. Y. Lin *et al.*, “Microsoft COCO: Common objects in context,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8693 LNCS, no.

PART 5, pp. 740–755, 2014.

[36] B. Zhou, A. Lapedriza Garcia, J. Xiao, A. Torralba, and A. Oliva, "Learning Deep Features for Scene Recognition using Places Database," *MIT web domain*, 2014.

[37] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 Million Image Database for Scene Recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 6, pp. 1452–1464, 2018.

[38] H. Noh, A. Araujo, J. Sim, T. Weyand, and B. Han, "Large-Scale Image Retrieval with Attentive Deep Local Features," *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2017-Octob, pp. 3476–3485, 2017.

[39] M. Teichmann, A. Araujo, M. Zhu, and J. Sim, "Detect-to-Retrieve: Efficient Regional Aggregation for Image Search," 2018.

[40] F. Perronnin, Y. Liu, and J. M. Renders, "A family of contextual measures of similarity between distributions with application to image retrieval," *2009 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work. CVPR Work. 2009*, vol. 2009 IEEE, no. July 2009, pp. 2358–2365, 2009.

[41] K. Ozaki and S. Yokoo, "Large-scale Landmark Retrieval/Recognition under a Noisy and Diverse Dataset," 2019.

[42] M. a Fischler and R. C. Bolles, "Paradigm for Model," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[43] S. Sun, J. Pang, J. Shi, S. Yi, and W. Ouyang, "FishNet: A Versatile Backbone for Image, Region, and Pixel Level Prediction," pp. 1–11, 2019.

[44] J. Hu, L. Shen, and G. Sun, "Squeeze-and-Excitation Networks," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 7132–7141, 2018.

[45] K. Chen, C. Cui, Y. Du, X. Meng, and H. Ren, "2nd Place and 2nd Place Solution to Kaggle Landmark Recognition andRetrieval Competition 2019," pp. 1–6, 2019.

[46] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-Up Robust Features (SURF)," *Comput. Vis. Image Underst.*, vol. 110, no. 3, pp. 346–359, 2008.

[47] G. Medioni and Y. Yasumoto, "Corner Detection and Curve Representation Using {B}-Splines," *Comput. Vision, Graph. Image Process.*, vol. 39, no. 3119, pp. 267–278, 1987.

[48] K. Mikolajczyk *et al.*, "A comparison of affine region detectors," *Int. J. Comput. Vis.*, vol. 65, no. 1–2, pp. 43–72, 2005.

[49] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 5987–5995, 2017.

[50] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," 2016.

[51] Y. Gu, "Team JL Solution to Google Landmark Recognition 2019," 2019.

[52] C. Chang, G. Yu, and L. Ai, "Explore-Exploit Graph Traversal for Image Retrieval," *Cvpr 2019*, pp. 9423–9431, 2019.

[53] F. Radenovic, A. Iscen, G. Tolias, Y. Avrithis, and O. Chum, “Revisiting Oxford and Paris: Large-Scale Image Retrieval Benchmarking,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 5706–5715, 2018.

[54] I. Melekhov, J. Kannala, and E. Rahtu, “Siamese network features for image matching,” *Proc. - Int. Conf. Pattern Recognit.*, pp. 378–383, 2017.

[55] Y. Zhang, Y. Zhang, J. Sun, H. Li, and Y. Zhu, “Learning near duplicate image pairs using convolutional neural networks,” *Int. J. Performability Eng.*, vol. 14, no. 1, pp. 168–177, 2018.

[56] Y. Qi, Y. Z. Song, H. Zhang, and J. Liu, “Sketch-based image retrieval via Siamese convolutional neural network,” *Proc. - Int. Conf. Image Process. ICIP*, vol. 2016-Augus, pp. 2460–2464, 2016.

[57] A. Mazumdar, J. Singh, Y. S. Tomar, and P. K. Bora, “Universal Image Manipulation Detection using Deep Siamese Convolutional Neural Network,” pp. 1–6, 2018.

[58] E. Sabir, W. AbdAlmageed, Y. Wu, and P. Natarajan, “Deep Multimodal Image-Repurposing Detection,” 2018.

[59] A. Bharati *et al.*, “Beyond pixels: Image provenance analysis leveraging metadata,” *Proc. - 2019 IEEE Winter Conf. Appl. Comput. Vision, WACV 2019*, pp. 1692–1702, 2019.

[60] M. Huh, A. Liu, A. Owens, and A. A. Efros, “Fighting Fake News: Image Splice Detection via Learned Self-Consistency,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11215 LNCS, pp. 106–124, 2018.

[61] A. Jaiswal, Y. Wu, W. AbdAlmageed, I. Masi, and P. Natarajan, “AIRD: Adversarial Learning Framework for Image Repurposing Detection,” 2019.

[62] W. Hu, Y. Fan, J. Xing, L. Sun, Z. Cai, and S. Maybank, “Deep Constrained Siamese Hash Coding Network and Load-Balanced Locality-Sensitive Hashing for Near Duplicate Image Detection,” *IEEE Trans. Image Process.*, vol. 27, no. 9, pp. 4452–4464, 2018.

[63] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, “NetVLAD: CNN Architecture for Weakly Supervised Place Recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 6, pp. 1437–1451, 2018.

[64] G. Kordopatis-Zilos, S. Papadopoulos, I. Patras, and Y. Kompatsiaris, “Near-duplicate video retrieval by aggregating intermediate CNN layers,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 10132 LNCS, no. August, pp. 251–263, 2017.

[65] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell, “ActionVLAD: Learning spatio-temporal aggregation for action classification,” *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, no. typically 25, pp. 3165–3174, 2017.

[66] S. Chen, X. Wang, Y. Tang, X. Chen, Z. Wu, and Y.-G. Jiang, “Aggregating Frame-level Features for Large-Scale Video Classification,” pp. 1–6, 2017.

[67] A. Miech, I. Laptev, and J. Sivic, “Learnable pooling with Context Gating for video classification,” pp. 1–8, 2017.

[68] Y. Tang, X. Zhang, J. Wang, S. Chen, L. Ma, and Y. G. Jiang, “Non-local NetVLAD encoding for video classification,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect.*

Notes Bioinformatics), vol. 11132 LNCS, pp. 219–228, 2019.

[69] A. Araujo, B. Negrevergne, Y. Chevaleyre, and J. Atif, "Training compact deep learning models for video classification using circulant matrices," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11132 LNCS, pp. 271–286, 2019.

[70] Y. Xu, Y. Han, R. Hong, and Q. Tian, "Sequential Video VLAD: Training the Aggregation Locally and Temporally," *IEEE Trans. Image Process.*, vol. 27, no. 10, pp. 4933–4944, 2018.

[71] R. Lin, J. Xiao, and J. Fan, "NeXtVLAD: An efficient neural network to aggregate frame-level features for large-scale video classification," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11132 LNCS, pp. 206–218, 2019.