REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

LARGE SCALE LANDMARK RECOGNITION USING GLOBAL DESCRIPTORS AND ENCODED FEATURES

Berk TAŞKIN

MASTER OF SCIENCE THESIS
Department of Computer Engineering
Program of Computer Engineering

Supervisor
Prof. Dr. Mine Elif KARSLIGİL

July, 2022

REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

LARGE SCALE LANDMARK RECOGNITION USING GLOBAL DESCRIPTORS AND ENCODED FEATURES

A thesis submitted by Berk TAŞKIN in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE** is approved by the committee on 19.07.2022 in Department of Computer Engineering, Program of Computer Engineering.

Prof. Dr. Mine Elif KARSLIGİL
Yildiz Technical University
Supervisor

Approved By the Examining Committee

Prof. Dr. Mine Elif KARSLIGİL, Supervisor
Yildiz Technical University _____

DOÇ.DR. GÖKHAN BİLGİN, Member
Yildiz Technical University _____

PROF.DR. ÇİĞDEM EROĞLU ERDEM, Member
Marmara University _____

I hereby declare that I have obtained the required legal permissions during data collection and exploitation procedures, that I have made the in-text citations and cited the references properly, that I haven't falsified and/or fabricated research data and results of the study and that I have abided by the principles of the scientific research and ethics during my Thesis Study under the title of Large Scale Landmark Recognition Using Global Descriptors and Encoded Features supervised by my supervisor, Prof. Dr. Mine Elif KARSLIGİL. In the case of a discovery of false statement, I am to acknowledge any legal consequence.

Berk TAŞKIN

Dedicated to my family

ACKNOWLEDGEMENTS

I would like to sincerely thank to my supervisor Prof. Dr. Mine Elif KARSLIGIL for her invaluable contributions and support throughout the research and writing of this thesis. I would like to thank my dearest friend Mustafa Furkan ESEOGLU for his ideas, inputs and guidance. I am forever grateful to have a family who have been there for me in this arduous process. To my beloved wife, Yagmur DEVIR TASKIN, who has been there whenever i need, i owe a great debt of graditute.

Berk TAŞKIN

TABLE OF CONTENTS

LIST OF SYMBOLS vii

LIST OF ABBREVIATIONS viii

LIST OF FIGURES x

LIST OF TABLES xi

ABSTRACT xii

ÖZET xiv

1 INTRODUCTION 1

1.1 Literature Review . . . 2

1.1.1 Publications on the Google Landmarks Dataset . . . 2

1.1.2 Research on dataset cleaning . . . 3

1.2 Objective of the Thesis . . . 4

1.3 Hypothesis . . . 4

2 METHODOLOGY 6

2.1 Feature Extraction . . . 7

2.1.1 Feature Extraction with image compression . . . 8

2.1.2 Local and Global Features . . . 11

2.1.3 Established Feature Extractors . . . 11

2.2 Dataset Cleaning . . . 13

2.2.1 Outlier Detection . . . 13

2.2.2 Clustering Based Approaches . . . 15

2.2.3 Classification Based Approach . . . 16

3 SYSTEM DESIGN 20

3.1 Dataset . . . 20

3.2 Data Preprocessing . . . 23

3.3 Feature Extraction . . . 23

3.3.1 Feature Extraction with image compression . . . 24

3.3.2 DELF and DELG 24
3.4 Dataset Cleaning 25
3.4.1 Outlier Detection 25
3.4.2 Clustering Based Approaches 26
3.4.3 Classification Based Approach 27

4 EXPERIMENTS **28**
4.1 Environment Setup 28
4.2 Image Compression 29
4.3 Feature Extractor Selection 32
4.3.1 DELF 32
4.3.2 DELG 34
4.4 Classification Architectures 35
4.4.1 Classification of 1000,2000,15000 classes 37
4.4.2 Thresholding 38
4.4.3 Dataset cleaning and reclassification 39

5 RESULTS AND DISCUSSION **42**

REFERENCES **45**

PUBLICATIONS FROM THE THESIS **48**

LIST OF SYMBOLS

μ	Mean value of a distribution
Σ	Standart Deviation of a distribution

LIST OF ABBREVIATIONS

CPU	Central Processing Unit
CIFAR	Canadian Institute for Advanced Research
DELF	Deep Local Features
DELG	Deep Local and Global Features
FAISS	Facebook AI Similarity Search
FLOPS	Floating-point Operations Per Second
GLD	Google Landmarks Dataset
GPU	Graphics Processing Unit
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
KL Loss	Kullback–Leibler divergence Loss
kNN	K-Nearest Neighbours
MNIST	Modified National Institute of Standards and Technology database
MS COCO	Microsoft Common Objects in Context
MSE	Mean Square Error
MS-SSIM	Multi Scale Sturcutal Similarity Index
Pascal VOC	Pascal Visual Object Classes
PCA	Principal Components Analysis
PyOD	Python Outlier Detection
RAM	Random Access Memory
RANSAC	Random Sample Concensus
SIFT	Scale-Invariant Feature Transform
SSIM	Structural Similarity Index
SURF	Speeded-Up Robust Features

SVM	Support Vector Machines
VAE	Variational AutoEncoder

LIST OF FIGURES

Figure 2.1 Images from 3 different landmark classes, Top: Real Landmark, Bottom: Outlier Images 7
Figure 2.2 Comparison between MSE and SSIM, showing same MSE value at different image qualities 10
Figure 2.3 DELF pipeline [7] 12
Figure 2.4 DELG pipeline [26] 13
Figure 2.5 Architectural layout of a Variational AutoEncoder 15
Figure 2.6 Current state of the art architectures in ImageNet dataset [36] . . . 17
Figure 3.1 Diagrams of architectural steps; Top: AutoEncoder, Middle : DELF&DELG, Bottom: Classification 21
Figure 3.2 Images not pertaining to any specific class for various reasons . . . 22
Figure 3.3 Noise image samples from the dataset, labeled because of proximity. Samples include other objects, drawings etc. 23
Figure 3.4 General architecture of the image compression AutoEncoder 24
Figure 4.1 Convolutional architecture of the decoder in AutoEncoder 30
Figure 4.2 Image recreation results from the image compression architecture, originals on left, recreations on right. Main observable differance is the absense or weakness of color red, making the recreation look duller compared to original 30
Figure 4.3 Sample images of museum exhibits that were marked as outliers by VAE because correlation between exhibits and specific museum landmark cant be made 32
Figure 4.4 Architecture of classification head, used with extracted feature vectors 33
Figure 4.5 Top-1 accuracy and model size of EfficientNet family of models [11] 36
Figure 4.6 Classification accuracies for extracted features and baseline models 37
Figure 4.7 Accuracy graph of the model prior to cleaning 38
Figure 4.8 Accuracy graph of the model after cleaning 40
Figure 4.9 System results from two different classes in rows: a, b, f, g: Correct class, not removed. c, h: Correctly cleaned. d, j: Noise sample, failed to clean. e, k: Wrongly removed 41
Figure 4.10 Final pipeline used in this research 41

LIST OF TABLES

Table 2.1 Dataset sizes throughout the years 19
Table 4.1 Available system resources of research environment 29
Table 4.2 Outlier detection scores of tested features 35
Table 4.3 Data subset and accuracies 38
Table 4.4 Thresholds effect on class number and image sample size 39

ABSTRACT

Large Scale Landmark Recognition Using Global Descriptors and Encoded Features

Berk TAŞKIN

Department of Computer Engineering
Master of Science Thesis

Supervisor: Prof. Dr. Mine Elif KARSLIGİL

The increasing importance of computer vision algorithms like object detection and segmentation in recent years made the research for these topics evermore popular. As a result, larger and larger datasets started surfacing. However, with the development of larger datasets, comes the effort of labeling. While in most studies, images are manually annotated, this labeling technique is outright infeasible for large collections of data therefore automatic annotations are used instead. In this research, an automated dataset cleaning technique for one such dataset has been outlined and performed using various outlier detection, clustering, and deep learning based classification algorithms.

Providing the basis for this study, Google Landmarks Dataset v2 is a dataset with ~200k classes and 5M images. This dataset is then first used in the extraction of three different feature vectors using an AutoEncoder, DELF and DELG. Extracted feature vectors are then used in outlier detection (kNN,Isolation Forest,VAE) and clustering (FAISS) schemes in order remove the noisy samples. Various degrees of success were achieved using different features.

Final part of this study used the aforementioned feature vectors and EfficientNet backbone in the construction of a classification architecture. Resulting landmark recognition validation accuracies of; 49% on AutoEncoder features, 55% on DELF features, 57% on DELG features, and 61% on EfficientNet-B5 were achieved prior to cleaning. In the light of these results, we decided to use EfficientNet for dataset cleaning purposes. Following a further training and fine-tuning step, the dataset was

cleaned using the prediction confidences of the trained model. Using the cleaned dataset, another training was then made from scratch achieving 71% validation accuracy. With this, it was observed that dataset cleaning is in fact possible to achieve using classification architectures.

Keywords: Dataset cleaning, deep learning, google landmarks dataset , efficientnet, outlier detection

ÖZET

Coğrafi Lokasyon ve Yer Sembolü Veri Kümelerinin Öznitelikler Yardımı ile Tanınması

Berk TAŞKIN

Bilgisayar Mühendisliği Anabilim Dalı
Yüksek Lisans Tezi

Danışman: Prof. Dr. Mine Elif KARSLIGİL

Obje tanıma ve anlamsal bölütleme gibi bilgisayarlı görü algoritmalarının öneminin artması ile bu alanda yapılan çalışmalar da son yıllarda popülerlik kazanmıştır. Bunun sonucunda kullanılan veri kümelerinin de boyutlarında artış gözlemlenmiştir. Fakat çalışılan veri kümelerinin boyutlarındaki artış işaretleme yükünü de beraberinde getirmektedir. Çalışmaların büyük çoğunluğunda kullanılan resimler insanlar tarafından işaretlenmiş olmakla beraber, çok büyük veri kümeleri için bu durum uygulanabilir olmaktan çıkmaktadır. Bu çalışmada veri kümesi temizliği için aykırı örnek tespiti, kümeleme ve derin öğrenme tabanlı sınıflandırma tekniklerini kullanan otomatik bir sistem tasarlanmış ve gerçeklenmiştir.

Çalışmanın temelini oluşturan Google Landmarks v2 veri kümesi, içerisinde ~200B sınıf ve 5M resim örneği bulunduran bir veri kümesidir. Bu veri kümesi öncelikle AutoEncoder, DELF ve DELG'den üç öznitelik vektörünün oluşturulmasında kullanılmıştır. Öznitelik vektörlerinin üretiminin ardından gürültü örneklerini ortadan kaldırmak için aykırı değer tespiti (kNN,Isolation Forest,VAE) ve kümeleme (FAISS) yöntemlerinden faydalanılmıştır.

Çalışmanın son bölümünde, bir sınıflandırma modelinin oluşturulmasında yukarıda bahsedilen öznitelik vektörleri ve EfficientNet mimarisi kullanılmıştır. Veri kümesi temizliğinden önce AutoEncoder özelliklerinde %49, DELF özelliklerinde %55, DELG özelliklerinde %57 ve EfficientNet-B5'te %61 doğruluk puanı elde edilmiştir. Bu sonuçlar ışığında veri kümesi temizleme amaçlı EfficientNet kullanımına karar verilmiştir. Temel performans değerlerının eldesi için yapılan detaylı eğitimde

%69 doğruluk puanı elde edilmiş ve bunu takiben, veri kümesi, eğitilen modelin tahmin güvenleri kullanılarak temizlenmiştir. Temizlenen veri kümesi kullanılarak sıfırdan yeni bir eğitime başlanmış ve %71 doğruluk puanı elde edilmiştir. Bununla birlikte, sınıflandırma mimarileri kullanılarak veri kümesi temizliğinin mümkün olduğu gözlemlenmiştir.

Anahtar Kelimeler: Veri kümesi temizleme, derin öğrenme, google landmarks veri kümesi, efficientnet, aykırı örnek tespiti

1
INTRODUCTION

Computer vision as a field of research has many academic and real world usages, among these use cases classification has always been one of the main benchmarks for evaluation of performance and accuracy. Although on its own classification doesn't necessarily directly correlate with any widely implemented computer vision solutions, it provides a fundamental understanding for image context. This image context can then be utilized for various tasks such as object detection [1], segmentation [2], scene understanding [3] etc. Likewise systems developed for image classification has been widely used as a backbone for many different computer vision algorithms [4] [5]. That is why ImageNet [6] dataset is still being used in ILSVRC (ImageNet Large Scale Visual Recognition Challenge) in order to test performances of developed architectures or as a way of creating a backbone model to then be fine-tuned for another use case.

Google Landmarks Dataset (GLD) [7] is another example for recognition oriented datasets and is also the main focus of this research. This massive dataset has the largest sample size of any landmark dataset by a comfortable margin, carrying ~200k classes and 5M images including test splits. Therefore the main reason for why this dataset is the proving ground of choice for this research is simple; as a problem of deep learning, recognition especially with this many classes is a problem of immense difficulty and thus provides countless opportunities for research to be made.

One such opportunity comes from, automatically labeled images in this dataset introducing considerable amount of noise to the distribution of classes. This means that without a strong supervision, learning samples of this scale is a difficult task. This brings us to the core of our research, dataset cleaning. In this study we proposed, developed and tested various architectures in order to create an automated solution to dataset cleaning. In the following sections these methods will be first introduced and then usages for them will be explained in detail.

1.1 Literature Review

With the increase of popularity for deep learning based approaches on all conceivable areas of computer vision research, countless architectures have been designed, developed and tested. Additionally since its inception, deep learning for classification problems have been on the forefront of advancements.

Increased computational power introduced by the usage of GPU‘s made increasing model depth and parameter count possible, however with these improvements, training of larger models necessitated ever growing datasets to be developed. While in the age of information, acquisition of raw image data is trivial, labels for these samples are difficult to come by, requiring for the most part manual labeling. Google Landmark Challenge [7] dataset, is one such example and the basis for this research. In this dataset, labels are acquired semi-automatically from image metadata and geolocation information. Similarly Google JFT [8] dataset is collected in crowd sourced manner.

1.1.1 Publications on the Google Landmarks Dataset

Google Landmark Challenge, unlike JFT is a publicly accesible dataset. Introduced in the 2018 Kaggle Landmark Recognition Competition version one of this dataset, aptly named GLDv1 includes 15000 classes and 1.2M image samples. Following year as a continuation of the challenge GLDv2 with 200k classes and 4.6M image samples were published. Since public availability of large datasets are limited, myriad of research followed this release. In Nguyen et. al. [9] manually labeled, clean version of this dataset is split into three parts with no class overlap afterwards one of these splits then used to train a transformer based clustering algorithm. Trained transformer is then used for clustering and testing of the other splits. Authors of this paper have achieved higher clustering performance by at least 3% compared to previously achieved best results. Overall results of this paper shows that transformer based clustering algorithms could be utilized in quantization of large scale datasets.

Another research made in the retrieval context of previously mentioned competition by Xu et. al [10] utilized three different model backbones in the form of EfficientNet [11], Swin-L [12] and CvT [13]. Feature output of these backbones are then combined and resulting vector is fed through a series of layers in order to create class predictions which are trained with ArcFace loss. Each if the backbone decisions are aggregated in an ensamble, which resulted in final predictions. In order to facilitate retrival of images, logit output for the classification is used. Score for the retrival is calculated using raw retrival similarity, classification logit and distraction score from non landmark images. This research achieved third place in GLDv2 retrival

challenge. Authors approach of using an ensamble is promising because intuitively different architectures tend to focus on different structural information in an image.

Kim et. al. [14] proposes an approach in order increase representativeness of the dataset. Authors of this paper correctly state that, locations for most landmarks are not correlated with the respective populations of the countries/states they are located in. For example some landmarks in China are grossly underrepresented in their number of image samples in spite of population of respective Chinese city, creating a bias for recognition. In order to compensate for this represantation issue in retrival, a probabilistic calculation dependent to anonimized information of the contributor distribution is made. This information utilizes age, gender and country of residence to create relevance scores for 50k landmarks. With these scores at hand, resampling of the GLDv2 dataset is then made. Authors show that newly created subset has overall better representation within the user demographic.

1.1.2 Research on dataset cleaning

Research by Ozaki et. al. [15] used known clean dataset of GLDv1 to train a classifier model. Then after the model is trained, image samples of GLDv2 are processed through this model, acquiring image representations. These image representations are then utilized to obtain k nearest neighbours from the train samples. Followed by a spatial verification step to acquire at maximum top 100 images with the same label. If samples passing the verification exceeds the pre determined threshold, original sample is added to the new "clean" dataset. Proposed cleaning method resulted in 53% reduction for training images with 49% reduction in class counts. Authors reported an overall increase of 9% in mAP@100 with the cleaning.

Xu et. al. [16] proposed a scalable data cleaning pipeline for web crawled parallel two language corpora providing another use case for data cleaning in NLP research. In this study authors first map sentence pairs into a proprietary feature space, then train a logistic regression based classifier that seperates known good sample pairs from sythetically generated bad ones. Trained model is then used for cleaning noisy dataset, with better scoring pairs being added to clean "subset". Resulting clean dataset is then observed to be effective in increasing accuracy with at minimum 2% improvement.

A novel approach to learning from noisy datasets have been made by Veit et. al [17]. In this study, authors trained a model to create mapping between noisy and clean datasets. In order to achieve this, large amount of noisy labels in conjunction with the small sample of clean labels are taken, then a multi-label image classifier with two heads have been trained. While Label cleaning part of this network learns to map

clean and noisy samples, classifier part simply creates multi label class information. As a result average precision increase of 2% over the previous best is reported.

In another applicaton regarding face datasets, Ng et. al. [18] introduces a dataset cleaning method to be used on large scale face recognition datasets. In order to build the dataset, authors compiled a list of celebrities and their genders followed by a google image crawl in order to acquire images for each celebrity. Using a face detector, their research then locates, resizes and centers all face images. This dataset is assumed to be noisy because image samples acquired from google image search might not have the labeled celebrity. After several steps for obtaining a descriptor for all images are finished, 2 labels are then added, name of the celebrity and gender. Then a quadratic problem to be minimized is constucted which tries to assign similar labels to similar apperances and labels mismatched outputs as outliers. Authors ran this optimization approach for each individual, cleaning the available samples. Compared to naive approach of classification in all samples, a 20% improvement in F1 score is reported for correctly predicted outliers in best case. This study highlights another use case for data cleaning in face recognition algorithms.

1.2 Objective of the Thesis

Encompassing wide field of current research, this study proposes an automated dataset cleaning approach for automatically labeled noisy datasets. To this end several different methods for detection and removal of noise samples, have been proposed and tested with a classification network. Detailed methodology for these methods along with their results are then provided in their respective sections.

1.3 Hypothesis

Although several hypotheses have been suggested, guiding idea behind this study was and still is, to achieve higher values for accuracy in datasets that are known to be difficult. In order to accurately create and test approaches following assumptions that are backed by observation are made.

- It is difficult for massive datasets to be manually labeled thefore most of them are labeled automatically.
- Automatical labeling innately introduces some noise to the collection of samples simply because variations between collected samples.

- Noise is an especially considerable problem for large datasets when class distributions are imbalanced because it introduces another factor to skew the results of classifier.

Considering these assumptions, a hypothesis was formed; in a dataset with known noise and imbalanced class distribution, cleaning of these noise samples should improve the accuracy without any modification to the model.

For feature extraction steps this hypothesis is then expanded to; if an AutoEncoder with sufficient learning capacity is trained, then the resulting encodings should carry the necessary information for differentiation of images and thus can be used for dataset cleaning.

In the last stage of the study our final hypothesis ended up being; for a classification architecture of sufficient capability, it should be possible to differentiate noise samples from clean ones with classification confidences of architectures.

Combined hypothesis can then be stated as; Using a sufficiently trained classifier, it should be possible to differentiate and remove noise samples from a dataset, such that remaining samples can be better learned by the same network with the same parameters.

2
METHODOLOGY

Fundamental objective of this research is automated dataset cleaning. While in the recent years acquisition of large amounts of data became easier, labeling of images are still done manually. This creates an immense cost in model training especially for architectures that require more detailed labels like object bounding boxes or segmentation boundaries. For massive datasets, manually labeling images is not feasible therefore automatically acquired labels are used instead. This process of label acquisition however, introduces some noise and labeling mistakes to the dataset. Googles JFT, for example is collected from captcha images that are labeled in a crowd sourced manner. While they use a voting mechanism [8], it is still possible for this data to have some inherent errors. Similarly Google Landmark Challenge dataset acquires it's labels from geolocation metadata and image tags. These informations on image metadata are heavily susceptible to human error since images near landmarks dont only include the landmark itself, rather they can include objects near landmarks, exhibits within landmarks, part of landmarks, or completely unrelated objects simply because of proximity. This behavior can be easily observed from the dataset as it can be seen in the Figure 2.1.

Additionally most of the large datasets include vast number of classes since the end goal of computer vision algorithms is to recognize every object possible in a given scene. This creates an issue, where however many images exists there will always be less samples for some objects, in comparison to the other ones. While there are some techniques that can learn from a limited number of samples [19],[20], dataset cleaning is still important for creating best represantation of the class in the image subset that can be trained.

Currently, there are number of methods that are being widely used for dataset cleaning, these can be listed as;

- Noisy sample removal: In noisy sample removal approaces, samples in a given dataset are evaluated through mathematical means either inside their classes or



Figure 2.1 Images from 3 different landmark classes, Top: Real Landmark, Bottom: Outlier Images

across the entire dataset. If a given sample is falls outside of its dataset/class representation boundaries that sample is removed.

- Mislabeled sample removal: In a large dataset, removal of mislabeled samples is rather difficult, this method is used when acquisition of a majority vote for classes is possible.
- Deletion of empty fields/classes: Mainly used for tabular data, deletion of samples with empty fields is a widely used approach when imputation of missing information is either impossible or not desired. In this approach sample with more empty columns than a treshold value are deleted.
- Removal of less populated classes: Specifically used on datasets with high number of classes, this dataset cleaning method simply removes entire classes with samples less then a set threshold value.

2.1 Feature Extraction

Feature extraction is a common concept that has been used throughout the history of computer vision. Early examples like Harris corner detector and SIFT relied on mathematically calculated local features, and correlation between said features for

following steps like panoramic image stitching [21]. With the improvements on computer vision however, especially since deep learning algorihms were popularized, deep learning based feature extraction have been a fixture in almost all of the computer vision research. This is because feature extraction does two things quite efficiently; it alleviates some of the issues of higher input dimensions from images and most importantly creates a backbone that does not need to be extensively trained for every use case, instead simpy fine-tuned. Rather than going through training for every conceivable data, it is far simpler to just finetune a model if necessary for a short amount of time then use the outputs in myriad of other tasks like predictions, segmentations, detections etc.

While feature extraction was prominent prior to deep learning, deep learning made achieving better representative image features such that simplest architectures still outperformed the traditional methods. Since this research focuses solely on computer vision/image processing, it must be stated that field of computer vision has been almost completely dominated by deep learning algorithms and architectures mostly thanks to previously stated feature extraction capability of deep networks.

2.1.1 Feature Extraction with image compression

Image compression on its own is not an intrinsically deep learning oriented concept, hovewer there are several approaches within the field of deep learning that can facilitate a system to compress images to vectors. Main idea behind these methods is the distillation of features that represent the entire image so that recreation can be made as losslessly as possible. Therefore usually these architectures include a feature extraction step. In order to recreate the image as needed, it is a necessity for this extractor to be able to capture as much information from the image as possible. In this study we hypothesized that features from one such extractor can be used to group/cluster images such that detection of the outliers can be achieved.

AutoEncoders [22] are one example that can be used to achieve both feature extraction and decompression stages. In an AutoEncoder, an input array is fed through a network to be "encoded", then a second part of the architecture called a decoder takes this encoding to create various outputs. These outputs may vary from the original array to classification results etc.

In its simplest form an AutoEncoder is an unsupervised algorithm since it does not require any additional information for its basic purpose. Hovewer as a deep learning algorithm, training will include an expected output and calculations for error on this expected output. Following section will detail loss functions that are utilized to

calculate loss and train the model.

2.1.1.1 Loss Functions

For the context of this study recreation of an image means that both inputs to the system and the expected outputs are the exact same image. Therefore loss functions for comparing original image with the network outputs will be used. Commonly used fuctions for these purposes can be listed as;

- Mean Square Error (MSE);

  Mean Square Error or MSE is an error metric in which square of deviation between two data points are calculated, summed and divided by total number of data points pairs. As it can be observed from the Equation 2.1. In this equation y represents ground truth and $\hat{y}$ represents model output. When used as a loss function for a deep learning algorithms approximation, ideal result is achieving "0" which means perfect match between target and model output.

$$\text{MSE} = \frac{1}{m}\sum_{i=1}^{m}(y^{(i)} - \hat{y}^{(i)})^2 \tag{2.1}$$

- Cross-Entropy;

  In it‘s essence Cross-Entropy calculates entropy between expected target output of y_c and the approximated output of p_c. Equation 2.2 shows how Cross-Entropy between these vectors of y_c and p_c is calculated. In machine learning a loss fuction that creates appropriate results in accordance with the wrongness of the assumption is needed in order to facilitate accurate training. Since Cross-Entropy depends on the probability of an expected outcome, Cross Entropy as loss function rewards the correct prediction with low loss, conversely for a lower probability i.e. wrong prediction function will output a high value for loss.

$$\text{Cross-Entropy} = -\sum_{c=1}^{N} y_c log(p_c) \tag{2.2}$$

  As a loss function for recreation, cross entropy again calculates data point distances without additional context input. Hovewer since output is calculated probabilistically results are generally better than MSE could achieve since smoothing of boundaries are punished more severely, being a distinct "wrong" prediction.

- Multi Scale Structural Similarity (MS-SSIM);

  While using distance measures like MSE or MAE for image recreation loss, these algorithms only look at a single value and its distance from a target. This causes the model to learn a blurred version of the image, since averaging of the neighbouring pixels results in low loss and therefore finishes the training procedure. Structural similarity hovewer is different, it calculates three distinct measures; change in luminance, change in contrast and change in structral information. This structural change takes relation with the neighbouring pixels into account meaning that a smoothing arcitecture results in low recreation scores. In the Figure 2.2, comparison between MSE and SSIM can be seen.

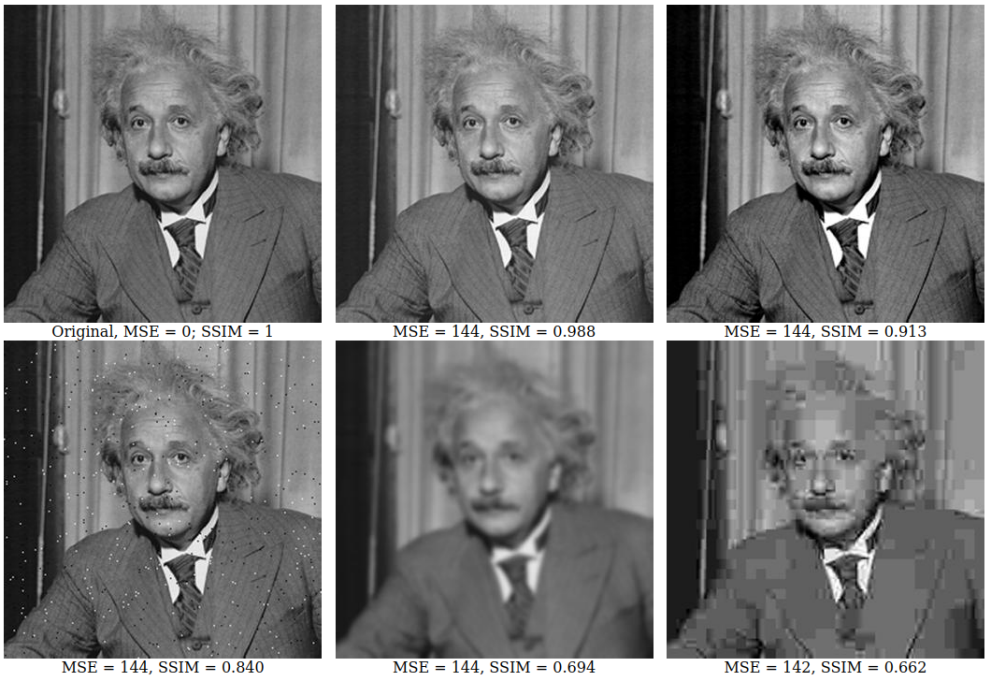


Figure 2.2 Comparison between MSE and SSIM, showing same MSE value at different image qualities

MS-SSIM[23], is a differentiable version of SSIM to be implemented as a loss function. Main differance being its implementation of subsampling data on multiple scales resulting in a more robust albeit slower algorithm. Formulas for both SSIM and MS-SSIM can be seen in the Equations 2.3 and 2.4. In these equations, l, c, s, depicts; luminance, contrasts and structural comparisons in given order. Since the result of perfect match for SSIM is 1, DMS-SSIM is used as loss metric.

$$\mathrm{SSIM}(x,y) = [l(x,y)]^{\alpha} \cdot [c(x,y)]^{\beta} \cdot [s(x,y)]^{\gamma} \tag{2.3}$$

$$\text{MS-SSIM}(x, y) = [l_M(x, y)]^{\alpha_M} \cdot \prod_{j=1}^{M} [c_j(x, y)]^{\beta_j} [s_j(x, y)]^{\gamma_j} \tag{2.4}$$

2.1.2 Local and Global Features

Feature extraction for this research, utilized two different mechanisms. Local features, have existed since before deep learning achieved its current popularity. Early techniques such as SIFT [24] used mathematical approaches to locate and extract features from areas of import in an image. These local features can then be used to match images together with algorithms like RANSAC [25]. Strength of local features compared global features is exactly this. While global features are known to create better image representations, they lack the necessary spatial information to perform image matching. Instead global descriptors are shown [26] to be better for retrieval tasks. In the following section, two architectures that were used for this study will be detailed.

2.1.3 Established Feature Extractors

Deep Local Feature Extractor (DELF) is an algorithm that has been trained on Google Landmark Challenge dataset by researchers at Google for landmark recognition and retrieval tasks. In its core, model creates local features for prominent objects in a given image. In order to achieve this, DELF employs two deep models in conjunction. First model which is a ResNet-50 architecture, extracts the initial descriptors. This model is pretrained on ImageNet [6] and then finetuned on Google Landmarks dataset. After the extraction of initial features, second model starts. Main objective for this network is to create keypoints using attention. It is theorized by the authors of the paper that given an image, if attention model decides to activate some parts of the original feature map specifically then those regions should be relevent local features of the image. Original paper at this point uses Principal Component Analysis (PCA) for dimentionality reduction to achieve vectors of size 40 which authors present as a good trade-off between compactness and discriminativeness.

As it can be seen from the Figure 2.3 for model pipeline of DELF, it utilizes a classification model, loss for this model is calculated using cross-entropy. Authors have observed that connecting modules directly and training from scratch made it difficult for the model to converge, therefore after the fine-tuning of the initial classifier, second attention module has been trained to create necessary local features. Which was again trained with cross-entropy loss hovewer with added rules.

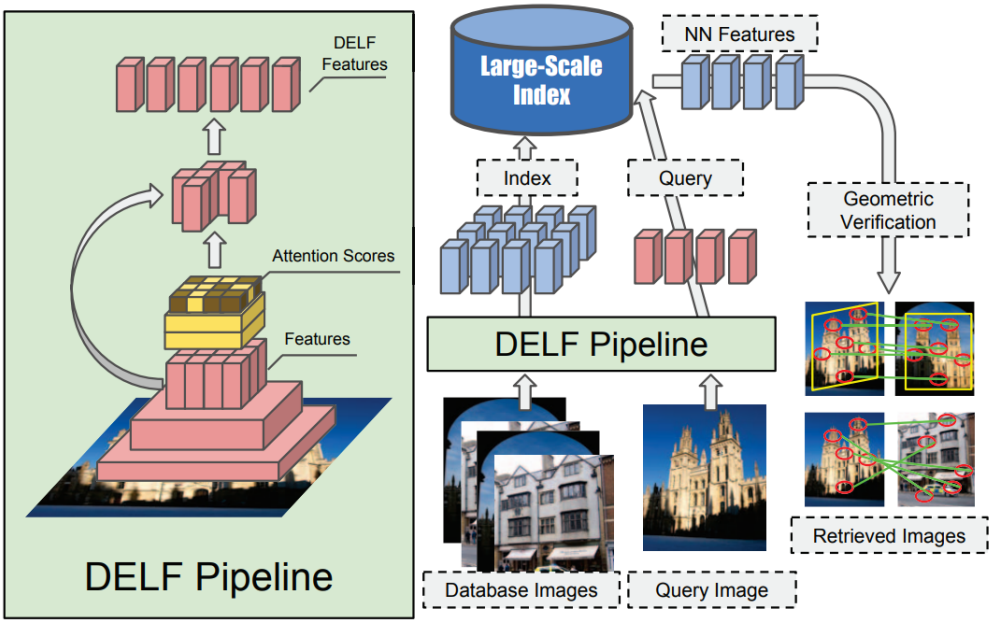


Figure 2.3 DELF pipeline [7]

As an iteration of DELF, Deep Local and Global feature extractor (DELG) aims to create features from images, hovewer it creates both local and global descriptors. Training methodology for this architecture involves three different losses. In order to provide a better overall view for the architecture, model diagram can be seen in the Figure 2.4. First head of the network is used in acquiring global descriptors from the image and is the only head that trains the backbone of the architecture. ArcFace loss which transforms logits to a hyperspace with the radius of s, and adds a value of m as an angular margin penalty; can be seen in Equation 2.5. This loss function is used in classification step [27]. Both second and third heads of the model are used to create local features. For this output, an AutoEncoder has been trained for reconstuction of the image using MSE as its reconstruction loss, simultaneously an attention layer is created prior to main dimentionality reduction of the model backbone. Then combining outputs from this attention map with AutoEncoder recreation, another cross-entropy loss is calculated. This combination provides the local features to be used in models main goal of image retrival.

$$\text{ArcFace Loss} = -\frac{1}{N}\sum_{i=1}^{N} log\frac{e^{s*(cos(\theta_{y_i}+m))}}{e^{s*(cos(\theta_{y_i}+m))}+\sum_{j=1,j\neq y_i}^{n} e^{s*cos\theta_j}} \tag{2.5}$$

Both of the aforementioned architectures are constructed for image retrival tasks, therefore have more steps after what is mentioned above, hovewer this study is only

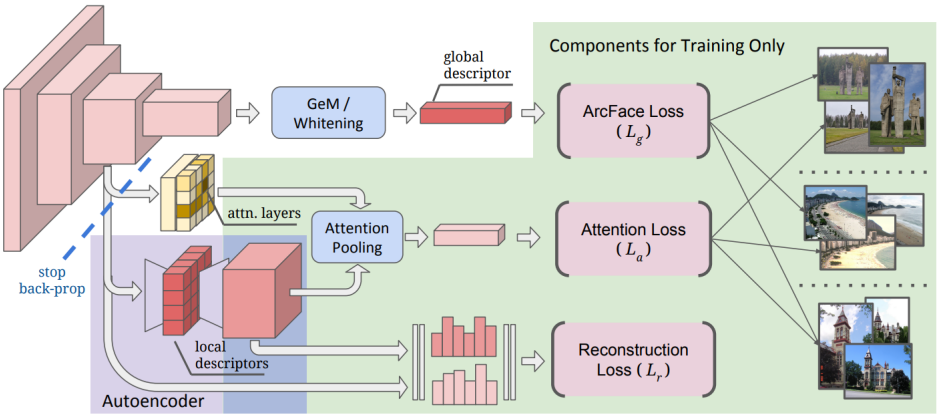


Figure 2.4 DELG pipeline [26]

interested in feature outputs that are generated. Thus explanations have only been made until acquisition of said feature vectors.

2.2 Dataset Cleaning

With the advent and increasing popularity of deep learning, necessity for larger and larger datasets has grown exponentially. Most of the current benchmark dataset has older counterparts that were not as massive, transition from Pascal Visual Object Classes [28] to Microsoft Common Objects in Context (MS COCO) [29] being a good example. Hovewer even with this clear increase in size, some tasks still require even larger sample sizes for efficient training. This is especially true for tasks that need high number of classes to be recognized. Current dataset for this research has roughly ~200000 classes and 4.6M samples and it is highly unbalanced. Manual labeling for datasets this size are not feasible, therefore labels are usually either automated or crowd sourced. This approach inevitably introduces some noise to the dataset.

Google Landmark Challenge dataset is known to be an especially noisy one for the priorly stated reasons. Dataset cleaning therefore, is required to improve classification/recognition performance of trained models.

2.2.1 Outlier Detection

Concept of outlier detection is not a new one. In a grouping of samples, an outlier can be defined as a sample that is outside of the norm for the general group. While some outliers in the context of machine learning can be useful in detecting overfitting of models, too many outliers might hinder the learning process. As such, a way of detecting and removing outliers might result in better distribution of samples, thus

improving the results of a machine learning model. Conversely removing too many outliers could also adversely affect a model by imparing generalization capabilities of an architecture. In this section outlier detection methods used in this study will be explained.

- kNN;

  kNN or k-Nearest Neighbours is a traditional, supervised classification/regression algorithm. For "k" nearest samples in a space, kNN takes the dominant class as a prediction for the query sample. Without a query sample hovewer, kNN can be used for outlier detection [30]. Since the labels for classes exist, samples without k same class neighbours within some set distance can be accepted as outliers.

- Isolation Forest;

  Isolation forest [31] like any tree ensamble method, relies on decision trees and used explicitly for anomaly detection. Since by definition outliers are anomalies in sample distribution, isolation forest can be utilized in detection of anomalies. Intuition behind isolation forest is that normal i.e. non outlier samples in a dataset ought to be further away from the root because they should have some similarities with other samples. Outliers on the other hand should have data points that cause them to fall closer to the root of the tree. By using random distribution of samples in a forest, samples that consistently have shorter paths from the root could then be classified as outliers with this method.

- Variational AutoEncoder;

  Variational AutoEncoders [32], much like their previous iteration, AutoEncoders, involve deep learning unlike previously explained machine learning based outlier detectors. In AutoEncoder based outlier detections, reconstruction of input is the main form of discrimination, if a sample can be reconstructed with minimal reconstruction error than that sample is closer to the original dataset that the AutoEncoder has been trained on. Thus if reconstruction error for a sample is unusually high that sample is labeled as being an outlier. Using such architecture for outlier detection hovewer requires an AutoEncoder that is well trained on a similar dataset at the very least. A graph for a Variational AutoEncoder can be seen in the Figure 2.5.

  Variational AutoEncoders on the other hand, instead of learning a direct vectoral representation of the input data, learns a function that can represent the input image. This function consists of two main components that can be combined to generate the latent vector for decoder to reconstruct expected output. Loss

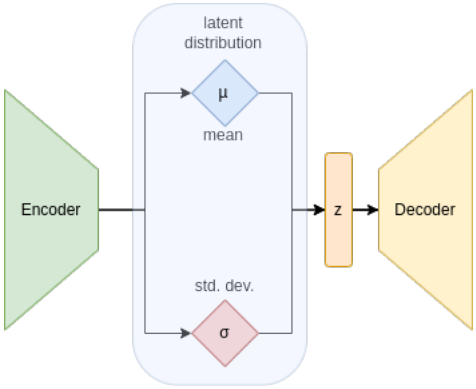


Figure 2.5 Architectural layout of a Variational AutoEncoder

function for a VAE is given in the Equation 2.6, two components of this loss function are reconstruction loss ($\mathbb{E}$) and KL loss ($\mathbb{KL}$), this KL loss is a score that forces encoders distribution of feature vector towards prior probabilistic normal of the trained samples. Thus moving the combined latent vector to "normal". This enforced normal reduces the strict similarity requirement of training samples when used with a pretrained model.

$$l_i(\theta,\phi) = -\mathbb{E}_{z\sim q_\theta(z|x_i)}[log p_\phi(x_i|z)] + \mathbb{KL}(q_\theta(z|x_i)||p(z)) \tag{2.6}$$

All of the aforementioned algorithms used previously extracted features from compression, DELF and DELG in order to provide a comparison between their respective results.

2.2.2 Clustering Based Approaches

Clustering is a machine learning method that focuses on creating groupings and centeroids for supplied input samples then using various distance measures to classify query images according to their clusters. For most use cases, clustering methods are among the unsupervised algorithms. This makes them a popular option to be used for seperating unlabeled data.

Facebook AI Similarity Search (FAISS) [33] is an index clustering approach which indexes the given data according to their similarities using a distance function (L2). For each queried sample, FAISS returns "k" closest samples. Cluster for the queried sample can then be decided by taking either cluster for the top-1 sample or majority from the top-N samples. Unlike regular clustering algorithms, FAISS is implemented with BLAS libraries with multithreading thus it works efficiently on GPU‘s.

Dataset cleaning with FAISS utilizes aforementioned querying approach. With a higher

"k" value it is expected that majority of cluster centers for the returned samples to match with the class of the queried sample. If no clear majority exists, then the sample can be removed. Hovewer, this approach requires training and testing on the same dataset simply because FAISS's requirement of prior indexing. For that reason efficacy of initial indexing should be tested with a validation set.

2.2.3 Classification Based Approach

Prior to domination of deep learning in computer vision, classification of images were largely academic in their form and generally involved simpler datasets with less classes like CIFAR and MNIST. In this context methods like Random Forests, SVMs and Perceptrons used flattaned image data to make their decisions on image object. Currently however, classification algorithms are only there to create a feature vector from the given image so that other algorithms can use it. Even though feature extraction had already been used with algorithms like SIFT and SURF these approches lacked the learnability and flexibility of a trained model.

Starting with Google LeNet [34], since most of the classification models were being used as backbone for other approaches, a more efficient way of computation was needed. As the years passed deep learning has continued to increase it's status as an integral part of computer vision research. While reported accuracy of models increased, so did the required computational power with each iteration. Among these iterations, ResNet should be mentioned as a stepping stone for most approaches. Before development of ResNet, depth of deep learning models were mathematically limited because of the intrinsic properties of model training. Increasing the depth of models (ie. number of layers) caused gradients on backpropogation to vanish and approach zero, making them unable to update any of the previous weights. ResNet with its residual connections made it possible to move gradients to previous layers. Residual connection structure can be further explained as follows; during training of deeper models, gradients of the last layers get smaller with each backpropogation step towards previous layers. In this architecture, previous layers are connected to the later ones with a multiplication process. During backpropogation these connections allows the transfer of gradients from the later layers to flow in two directions at once, one for linear structure, as it has previously existed, and one for a multiplication connection that was made directly. This is why these architectures can be virtually limitless in their depth since almost all of the loss can be carried throughout the network if necesssary. With this development deeper and deeper models could be developed. While deeper models improved accuracy, this came with an immense cost in computational power requirements, additionally reliance on GPU capability made it difficult to train ever

growing models efficiently on publicly accesible devices. MobileNet [35] architectures were developed to adress this issue. Utilizing different structures like depthwise seperable convolutions these architectures reduced the number of parameters in a given layer while retaining most of the learning capaticy gains.

In the recent studies EfficientNet family of classifiers had achieved state of the art performance for classification benchmarks like Imagenet and CIFAR datasets. Recent performance results for current architectures can be seen in the figure 2.6.

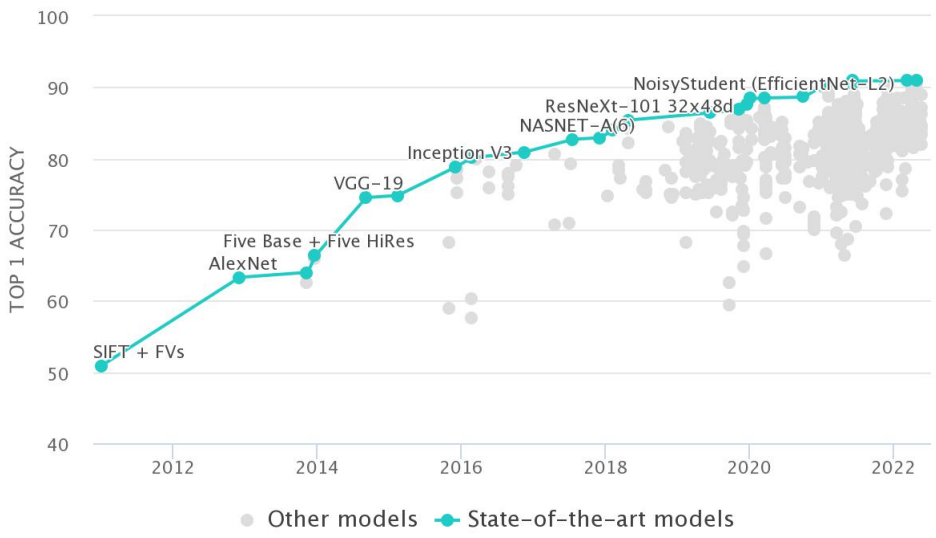


Figure 2.6 Current state of the art architectures in ImageNet dataset [36]

EfficientNet family of models has a two dimentional compound scaling in accordance with the size of the input image. EfficientNet shows that scaling the model width and model depth simultaneously improves learning capacity of the model. This works because intuitively, higher resolution input images not only benefit from deeper models but also improved receptive field that comes from an increase in model width. Initial construction of EfficientNet family of networks have been made with neural architecture search, while the idea of compound scaling can be proven effective, a good baseline model was also required to make the architecture efficient. To that end, a reinforcement learning based approach have been introduced. In this, a baseline model named EfficientNet-B0 has been optimized for both accuracy and floating point operations (FLOPS).This optimization Equation for the aforementioned approach can be seen in Equations 2.7,2.8, 2.9 and 2.10. In these equations; $\hat{H}$, $\hat{W}$ and $\hat{C}$ denotes predefined base model tensor sizes while $\hat{F}_i$ dictates the respective layer in the model. Scaling of the architecture is then made with adjusting the values of d, w and r for model depth, width and resolution respectively.

$$\max_{d,w,r} Accuracy(N(d,w,r)) \tag{2.7}$$

$$s.t. N(d,w,r) = \bigodot_{i=1\ldots s} \hat{F}_i^{d\cdot\hat{L}_i}(X_{\langle r\cdot\hat{H}_i, r\cdot\hat{W}_i, r\cdot\hat{C}_i\rangle}) \tag{2.8}$$

$$Memory(N \leq target_memory) \tag{2.9}$$

$$FLOPS(N \leq target_flops) \tag{2.10}$$

Correlation between model depth and contextial complexity of these features are well documented. First few layers of a convolutional network generally learn low level features like edges while deeper the model goes higher concepts like faces and objects can be recognized. Similarly, models with more width have been shown to extract better fine-grained information from an input image [35]. Compound scaling of these factors for EfficientNet has been made with an Equations that can be seen in 2.11,2.12,2.13,2.14. Scaling coefficient of ϕ, is used to alter all the model parameters together, such that model size changes accordingly. However, while model depth effects the model size linearly, width and resolution changes are quadratic. This is the reasoning behind Equation 2.14.

$$depth : d = \alpha^{\phi} \tag{2.11}$$

$$width : w = \beta^{\phi} \tag{2.12}$$

$$resolution : r = \gamma^{\phi} \tag{2.13}$$

$$s.t. |\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \text{ and } \alpha \geq 1, \beta \geq 1, \gamma \geq 1 \tag{2.14}$$

While model accuracy and size increased need for larger datasets became apperent. Since larger models require more data to effectively train, size of datasets has also skyrocketed with each passing year as in can be seen in the Table 2.1. Google JFT

Table 2.1 Dataset sizes throughout the years

	Dataset Name	Year	Number of Training Samples
Classification	MNIST	1998	60k
	CIFAR	2009	60k
	CelebA	2005	202k
	ImageNet	2012	14.2M
	JFT	2017	300M
Obj. D.	Pascal VOC	2012	11.5k
	ADE20K	2015	20k
	MS COCO	2017	123k

is one such example, although one with a restricted access. Efficient models like MobileNet evolved to deeper and scalable architectures like EfficientNet and utilized this improvement in training data availability to achieve State Of The Art status.

3
SYSTEM DESIGN

Current state of deep learning research is largely centered on large datasets. However acqusition of labels for these datasets are, for the most part, either impractical or difficult. For that reason automatical labeling methods are used instead. But with the ease of automatic labeling comes the issue of noise, thus making some effort on dataset cleaning, a necessity. This research aims to provide an automatic dataset cleaning method as a solution to the problem of noise.

Understanding behind this study has been structured and then implemented as a series of solutions to dataset cleaning problem. As a result of this, most of the architecural improvements have been made incrementally after observing the shortcomings of previously hypothesized method. Thus final system has a clear line of improvements that are going to be laid out in this section of the thesis.

In order to provide a better view of overall approach, steps taken has been listed below and will then be further elaborated on, in their own sections;

- Compression based feature extractor;
- DELF / DELG features;
- Outlier detection;
- Classification based cleaning;

Following diagrams in the Figures 3.1 shows the inputs and outputs for their relative processes.

3.1 Dataset

Google Landmark Challenge for recognition and retrieval dataset is a massive ~200k class, 4.5M image sample dataset. It is highly imbalanced, with top 1 percent of the

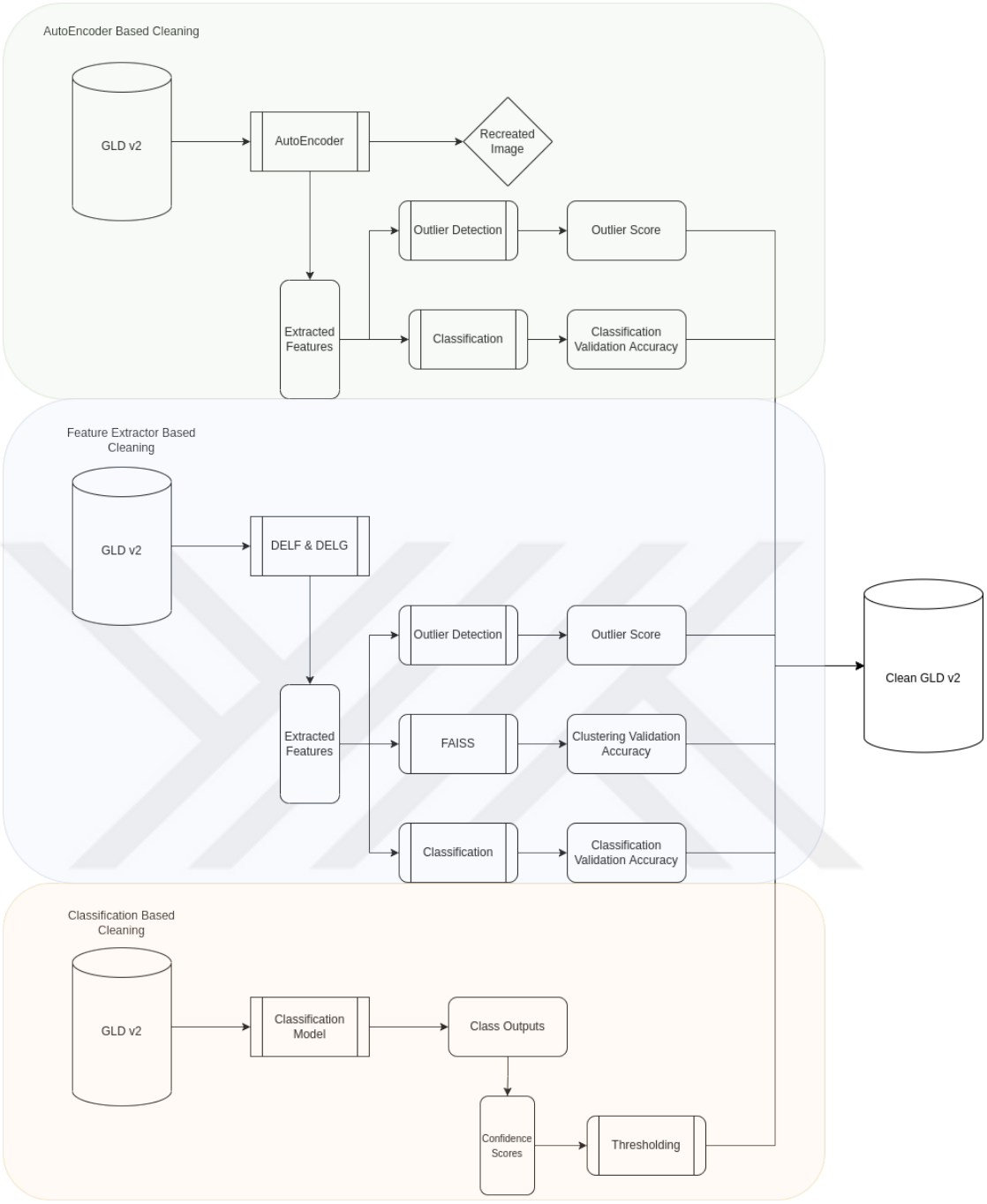


Figure 3.1 Diagrams of architectural steps; Top: AutoEncoder, Middle : DELF&DELG, Bottom: Classification

classes having nearly quarter of all images. This creates two main issues, firstly it gets exceedingly difficult to learn classes with less then 10 samples, secondly creating a fully connected layer with 200k classes is extremely space inefficient and causes the last layer of the model to have 400M parameters just for predictions (assuming a standard feature layer size of 2000). In order to alleviate some of these issues we conducted our experiments in 3 seperate dataset instances; Top 1000,2000 and 15000 classes. While using most populated 1000 classes is beneficial in solving these

problems it also simplifies the dataset to a degree where any solution for the overall dataset needs to be tested on a higher class count subset. Thus best approach for this research has been additionaly tested on the other aforementioned (2000,15000) subsets.

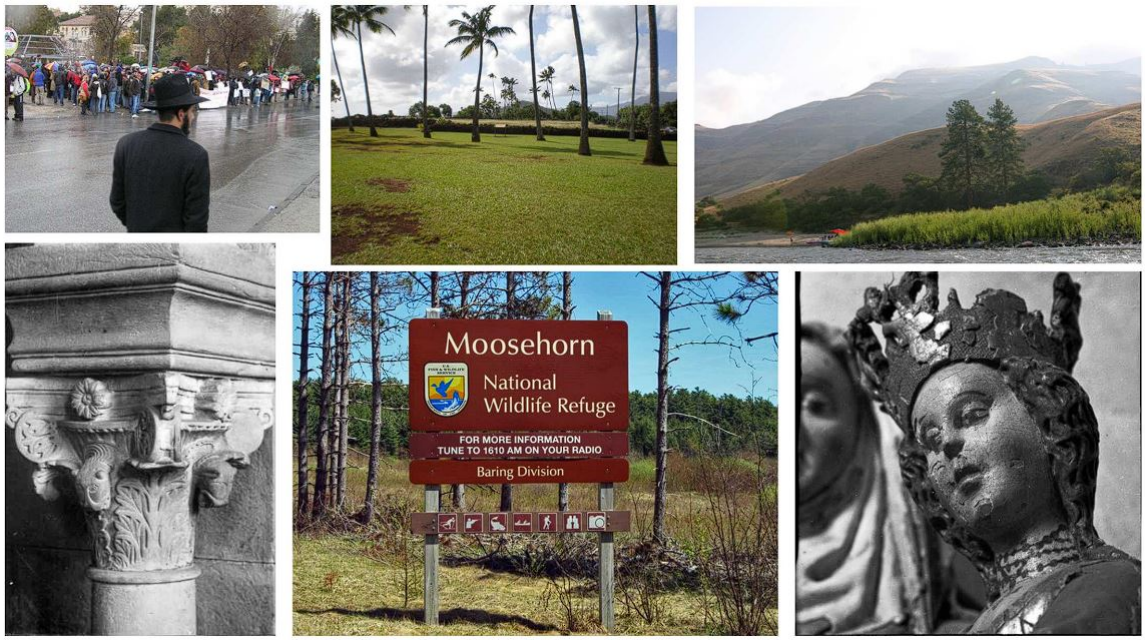


Figure 3.2 Images not pertaining to any specific class for various reasons

Apart from the issues stemming from class counts, another problem was the existence of non representative images. As it can be seen in the Figure 3.2, some of the images include people in front of landmarks, similar structures or exhibits in museums and images of gardens and flowers. These are non class descriptive images, since non of the focus objects included are landmark specific. As an automatically labeled dataset, Google Landmark Challenge data is also vulnerable to mislabeling in images. For example, images taken near a landmark has been observed to also be labeled as that landmark on several occasions as it can be observed in Figure 3.3. Therefore need for dataset cleaning has emerged.

During this study, Google Landmark Recognition dataset has been trained and tested with various architectures. Over the years, classification arcitectures reached a point of saturation for commonly used benchmarks like CIFAR or ImageNet. Therefore even with its apperent difficulties usage of this dataset has been decided on. Perhaps with hopes to provide a solution to some of the problems that were explained.



Figure 3.3 Noise image samples from the dataset, labeled because of proximity. Samples include other objects, drawings etc.

3.2 Data Preprocessing

In order to facilitate various training methodologies througout the research, 3 subsets from the original data hes been seperated. These subset are, as previously mentioned; 1000,2000,15000 most populated classes. No further balancing steps for this dataset has been made, since point of the research is cleaning an imbalanced dataset to a degree that it improves overall performance.

In addition to subset creation, for all training and test methodologies, same train/test split has been used in order to reduce the run to run variations between different methodologies throughout the study. Equally important, whenever possible, same hyperparemeters has been used to keep model outputs as replicatable as possible with consistent outputs.

3.3 Feature Extraction

As a core concept of deep learning, feature extraction has found countless use cases in computer vision research. The reason for this is simple, whatever the use case is, whether it be object detection, semantic segmentation, or classification, context-level image understanding is a fundamental requirement in computer vision. To this end,

the usage of feature extraction in the concept of dataset cleaning, since it aims to distill image information only makes sense.

In this study, several feature extractors are utilized for first, distillation of image information and then classification, clustering and grouping of image samples. In this section specific use cases for feature extraction throughout this research will be detailed.

3.3.1 Feature Extraction with image compression

Image compression in the context of feature extraction revolves around the concept of autoencoders. Autoencoders are architectures that, through a series of convolutional layers reduce the image dimensions to "encode" image data and then use the encoded image vector to achieve various tasks. These tasks could be achieving pixel based outputs for segmentation or in the scope of this study, recreation of the image as best as possible. Architecture for this model can be seen in the Figure 3.4.

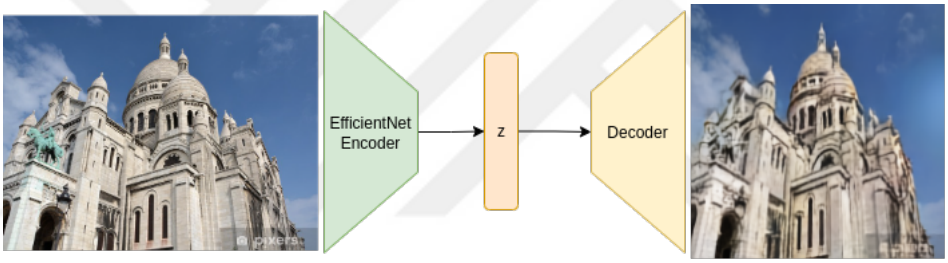


Figure 3.4 General architecture of the image compression AutoEncoder

Training of the autoencoder has been made on the previously mentioned 1000 subsample dataset in order to keep feature extraction consistent with the initial tests. After training encoder and decoder parts are decoupled and encoder outputs are then saved as a list of features for following steps in dataset cleaning part.

3.3.2 DELF and DELG

Following the results of the previous experiment, necessity of using a more specialized feature extractor was apparent. Because of this, first DELF and then DELG, has been integrated to our pipeline. Initial experiment with DELF, similar to compression features was, extract the features then use these features for the following cleaning steps. However, since DELF creates local features as opposed to global image features, another step needed to be taken. In order to facilitate this, much like the original DELF paper, PCA was used to create a fixed size (2048) vector. Resulting features are then saved to a dataframe along with labels for each of the images.

DELG, unlike DELF is a feature extractor that also creates global descriptors from images. Since it has been trained on the same dataset our research uses, relative subset is passed through the network then the resulting feature vectors are saved to a seperate dataframe with their class labels.

Training for these architectures are computationally expensive. During a trial run, each epoch for a 1000 class subset of the data took 7 days to complete on DELG. This makes finetuning for better losses in this specific use case incredibly difficult. Thats why model outputs are used directly for both of these architectures. Saved feature outputs from the pre-trained models are then used in the following outlier detection and clustering steps.

3.4 Dataset Cleaning

Process of dataset cleaning has some profound effects on deep learning as a whole. Most of the architectures that are currently being used, rely on structured, well labeled inputs from which they understand/extract necessary information. While most of the commonly used datasets have been prepared in a way to provide this input data, structuring and labeling large datasets manually is an expensive endeavor. Recent research therefore tried to move away from this necessity in hopes of achieving comparable results with minimal manual effort. Methods of said approaches vary greatly; from few shot/zero shot learning to automatically labeled samples. In this study, mostly automatically labeled images from Google Landmark Challenge dataset have been used. These labels are acquired from geolocation information and location tags in the image metadata. Hovewer context analysis on the images makes it clear that some cleaning is required. Since geolocation metadata is used for labeling, image gets labeled as that specific landmark whether landmark itself is present in the image or not. Similarly, taken out of context some landmark features are quite similar across multiple classes making images from these areas not representative for a single class.

In this study, several different methods have been utilized for dataset cleaning in order to achieve higher accuracies. Sections below outlines the approaches taken during this research.

3.4.1 Outlier Detection

Figuring out which of the sample images could be removed without adversely affecting the dataset (reduction in class count, introduction of further imbalance to the dataset) can be achieved with different methods. Outlier detection is what comes to mind first simply because, outliers are, by definition, samples that does not belong to the regular

class distribution. Hence removal of said samples should in theory help the model learn remaining samples better.

At this point of the research, there are 3 extracted feature vectors for each image, which can be utilized for dataset cleaning. To this end, outlier detection part of this research employed three commonly used outlier detectors; kNN, Isolation Forest and VAE seperately for all of the previously mentioned features.

Outlier detection approaches are all implemented in the same way. Subset of the data has been passed through the outlier detection algorithm with its labels. Then a score for outlier detector has been calculated simply by testing with a prepared subset of 100 feature vectors. Of these 100 feature vectors 10 of them were outliers and resulting score is the accuracy of the system on this subset. Out of the three outlier detector architectures, VAE consistently outperformed the other two methods. Therefore next step of the architecture was to use VAE in a per-class basis to detect and then remove the outliers in each class. Due to highly unbalanced nature of the dataset, this cleaning process became problematic. While some classes with the more samples performed better, sample distributions for less populated classes was not able to produce a distinct center. Because of this cleaning procedure for these classes produced lower overall outlier scores.

3.4.2 Clustering Based Approaches

Clustering for dataset cleaning, is an approach that had been previously used in Nguyen et. al[9]. Since the main objective of clustering is to create groups of samples, outlier detection in this context simply requires finding samples that are outside of their respective group labels. As a similarity search algorithm FAISS can group samples by their distances. For that reason, it has been used on this research as a clustering approach.

During this step, previously extracted features were fed through the FAISS clustering module to produce cluster centers. After this step a seperate validation split was then tested to see whether they could be clustered correctly or not. With the previous best performant features from DELG, it was clear that clustering efforts with this many classes and restricted feature vector size was not feasible. Best case scenario for this approach had ended up being just slightly better then random guessing.

3.4.3 Classification Based Approach

Primary objective during this research is to clean a massive dataset to a degree that there is an observable accuracy improvement. While priorly mentioned methods go through this process utilizing other means, here with the classification based approach, single model is used for classification, cleaning and benchmarking. Three different architectural iterations from EfficientNet family of models are trained directly with the image samples and compared to accuracies of ResNet classification results from DELF, DELG and compression features in order to decide on which of the models provide a better performance/accuracy balance. After this decision making process, best performing model, is then trained on the dataset split for 1000 classes. Using the trained model, confidence scores for each of the samples are acquired. When models predicted class confidence for an image sample drops below one of the 6 previously decided threshold (20%,17,5%,15%,12,5%,10%,7,5%) they are recorded in a seperate dataframe. For each treshold a new "clean" dataset is then prepared. Another decision making at this step has been made in order to decide which threshold split should to be used. Among the 1000 classes, removal of samples with this method, also removes some of the classes in their entirety, since classes with low number of samples are not efficiently learned thus, are more succeptible to this elimination process. Previously mentioned decision therefore has been made to remove as much of the dataset as possible with as less of an impact on the classes as possible. To this end threshold of 12,5% has been decided on. On further improvements to the model it was possible to remove data without causing the reduction on the number of classes, so that procedure is followed. Which will be explained in detail in the experiments section.

After this cleaning, model has been trained from the scratch, with the same hyperparemeters and train/test splits. Accuracy improvements can be seen, again, in the following section. Both baseline and clean data training has been performed for 10 epochs, first 5 epochs was performed with no class weights and for the remaining 5 class weights were enforced on loss calculation. This is because during initial testing with class weights due to nearly 50% of the image data belonging to first $\sim$50 classes, losses from bottom 95% of classes made progression of the model nearly impossible. Starting with no class weights solved this issue by eliminating pressure on the top classes and letting the model learn these samples for 5 epochs. After 5 epochs, class weights were activated, since the network had already learnt top 5% of classes, remaining samples could be enforced without negatively effecting the models learning process.

4
EXPERIMENTS

In this study different methods for dataset cleaning are utilized but with the exception of one approach, all of the presented methodologies relied on feature extractions and grouping of said features. First of these approaches was to create a global feature extractor that has a managable feature size, since it is needed to save/cluster/calculate differences on 4.5m images. To that end decision to train an image compression model and use the compresses encoder output as extracted features have been made. Following self developed feature extractor, various tests with established feature extractors were started. In the last step of the research, classification based method is trained, tested and used to clean the dataset.

In order to create a baseline benchmark for all developed models. A ResNet-50 model is trained in 1000 class subset of the data, since this model will be used as a benchmark for comparison no fancy training time methodologies were employed. From it‘s result validation classification accuracy to pass has been established as 53%.

4.1 Environment Setup

During the entire development process, usage of a large dataset made the sizeble storage space for images and features a necessity. Additionally, models and architectures employed required considerable amounts of computational power. On top what was previously mentioned, some of the index based methods working of of RAM also made high memory availablity a necessity. In Table 4.1, environment resources that are utilized throughout the research can be observed. Although resources used are better then the bare minimum needed, it should be noted works published on this dataset usually employ much higher computational power.

Architectures of DELF and DELG have been implemented on TensorFlow 2 Framework [37], AutoEncoders, EfficientNet and custom architectures were all implemented using PyTorch 1.8.1 [38].

Table 4.1 Available system resources of research environment

System Specifications	
CPU	16 Core
GPU	2 x Nvidia 2080 TI's
RAM	64 GB
Storage	1 TB of SSD, 6 TB HDD (for data)

4.2 Image Compression

Starting approach for this research was to compress images to such a degree that resulting feature vector represents the entire image as uniquely and as concisely as possible. Afterwards main idea was to cluster all features of images belonging to the same class and remove the outlier samples as non representative images. To ensure this, an AutoEncoder with Multi Scale Structural Similarity Index (MS-SSIM) as a loss function has been trained. Unlike L2 and L1 norms, SSIM heavily penalizes smoothed images, since it calculates similarity not as a distance between given images but as the intrastructural change between neighbouring pixels. Implemented architecture used an EfficientNet-B7 backbone for its encoder structure. Feature extraction output of this model was then directly connected to a custom designed decoder architecture. Since skip connections between encoder and decoder would mean that some of the relevant information could be transferred between encoder and decoder without passing from the feature extraction phase, it was decided to not use any skip connections in between. While main goal of the model is to train the encoder, it is still important for decoder to produce accurate outputs. Reason for this is that output of the decoder is where losses for the entire system are calculated. So a decoder that does not approximate would mean that errors can not propogate through the network correctly. Contradictarily it is also not desired for the decoder to be so successful in case it reduces the need for well extracted features. To this end, three decoder architectures with different complexities were prepared. Architectural details for one of these models can be seen in Figure 4.1. First and simplest of these decoders used a simple image upscaling scheme that consists of 5 max unpooling units each followed by convolution, normalization and activation layers. End result for this architecture achieved 0.55 SSIM score which is then decided to improve upon. Second architecture used upscaling convolutions instead of max unpooling and in addition, layer count was increased to 7 in order to create a smoother transition to higher resolution output image. This approach improved average SSIM score drastically to 0.73. Third and last approach was to utilize self attention mechanic in the decoder. Similar to previous method, upscaling convolution layers are used but on each convolution step another head extracted an attention vector which was then multiplied with the output, followed by normalization and activation layers. This approach resulted in the best

result yet with an SSIM score of 0.79.

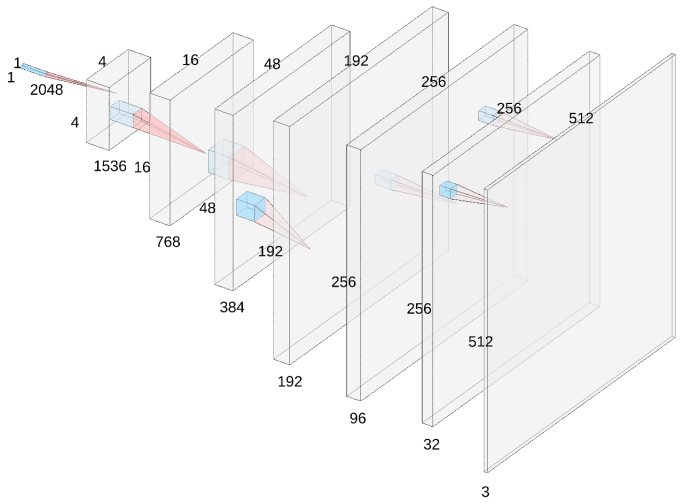


Figure 4.1 Convolutional architecture of the decoder in AutoEncoder

Ignoring the requirement for the resized input images, developed recreation model compresses the input image with dimentions of 512x512x3 to a floating point vector of size 2048, resulting in mathematical compression ratio of ~99% (assuming short integer size of 2 bytes and double size of 8 bytes). Even with this compression; recreation of the images are surprisingly succesful, as it can be seen in the Figure 4.2.

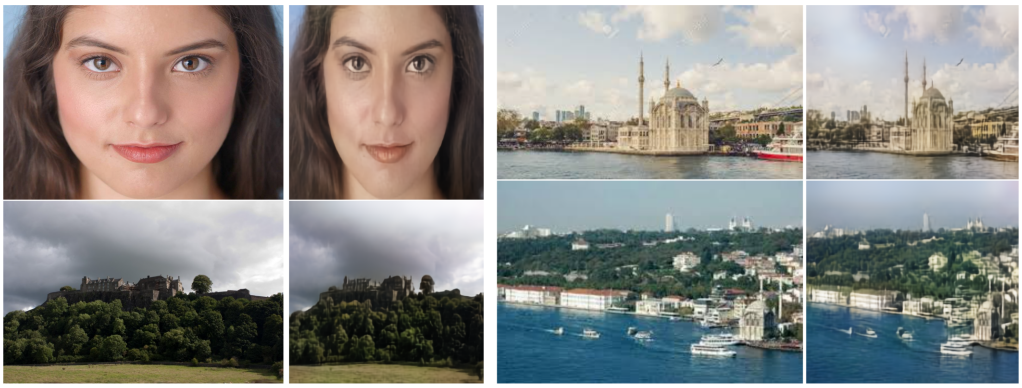


Figure 4.2 Image recreation results from the image compression architecture, originals on left, recreations on right. Main observable differance is the absense or weakness of color red, making the recreation look duller compared to original

However, closer inspection on the interclass distances utilizing the latent vectors extracted by AutoEncoder model described above, it was apperent that self attention decoder was too strong for the encoded features. Resulting features were, nearly random, floating point numbers without an apparent structure that the decoder has

learnt to make sense of. When same distance calculation is tried on the results from the previous decoder architectures, second approach with upscaling convolutions provided the best seperation between classes. Using this information, prior to the usage for feature extraction purposes, Encoder-Decoder architecture has been further fine-tuned with longer training until convergence.

After training, feature vector for each of the image samples are extracted. In order to be able to test different outlier detection methods, PyOD [39] is utilized. Out of the various approches in PyOD; kNN, isolation forest and VAE for a distribution of traditional, ensemble and deep learning based methods have been tested. For outlier detection trial; two distinct test subsets consisting of 100 and 2000 curated samples were selected. Each subset had 10% of their samples manually selected from outliers with the remaining 90% being clear landmarks. While test set of 2000 samples achieved 15% higher accuracy compared to 100 sample subset, best performing model remained VAE in both cases by a comfortable margin followed by isolation forest. Due to computational constraints and repetitive nature of the experiments, usage of 100 sample subset for decision making purposes has been decided on.

With the outlier outputs from VAE for the entire dataset at hand, a visual inspection of the removed samples have been made. This inspection revealed that samples for some of the classes are too "unique" thus ended up being wrongly removed. For instance, since these are landmark images, images from museums which can be seen in Figure 4.3 were removed simply because of the existence of images from exhibits that can't be correleted with the overarching class without context. Understanding that both of these images could be labeled as outliers if taken out of context, it was apparent that it was necessary to design an architecture that can learn the context and use class informations for its learning.

Solution to this issue was to use the extracted features to train a classifier, then use the classifier confidence for dataset cleaning. It is hypothesized that samples with low confidence scores had low confidence scores because similar images existed in more than one class thus were not representative for any class. In order for this approach to be succesful however it was crucial to achieve highest classification performance possible. First architectural limitation occured at this point. Since the initial reason for this dataset cleaning was to increase classification accuracy itself, proposed testing methodology needed to include direct comparisons with non-cleaned data baselines. This meant that if at any point classification performance dropped below baseline performance this would mean that either model or training or said model was insufficient.



Figure 4.3 Sample images of museum exhibits that were marked as outliers by VAE because correlation between exhibits and specific museum landmark cant be made

With previously mentioned requirements in mind, a classifier has been constructed which took feature vector of size 2048 and through series of 5 convolutional layers followed by 2 fully connected layers similar to what was previously used for ImageNet classification in the last layers of ResNet architecture [40]. Architecture of this model can be seen in the Figure 4.4. This model ended up achieving 49% accuracy with 1000 classes after 10 epochs. Compared to accuracy of 53% that was achieved directly classifying images with ResNet-50 after 10 epochs, it was obvious that previously extracted features were underperforming significantly. In order to make sure suggested logic was sound in theory, decision to use features extracted by DELF and DELG has been made.

4.3 Feature Extractor Selection

Following the tests performed on AutoEncoder features, importance of better features became apperant. To this end decision to use original model that had been introduces with the dataset have been made. In conjunction with this network, iterative improvement on the original architecture have also been employed.

4.3.1 DELF

After receiving unsuccesful but promising results from compression approach next step was to prove hypothesized concepts merit with an established feature extractor that has been trained in the same dataset that this research aimed to clean. Deep

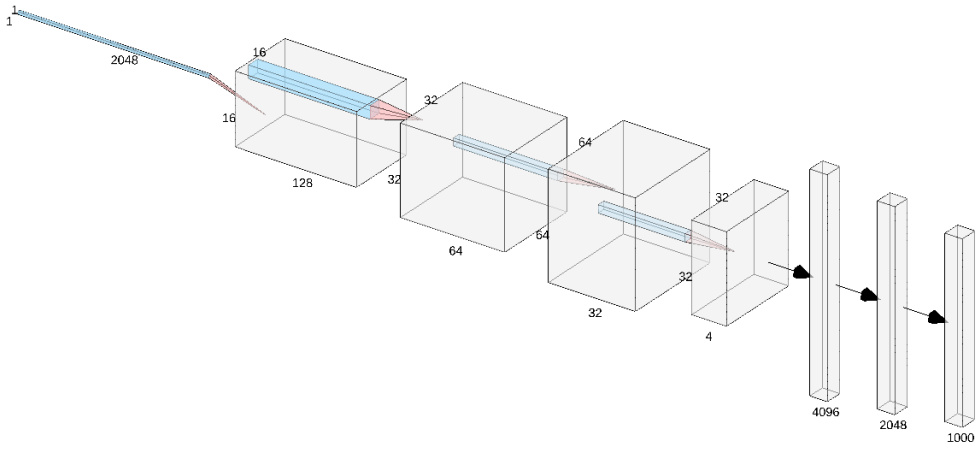


Figure 4.4 Architecture of classification head, used with extracted feature vectors

Local Feature extractor (DELF), is specifically trained on Google Landmark Challenge Dataset for recognition and image retrival. In order to retrieve similar images from an existing dataset, DELF extracts the features of a given image then find images with similar feature vectors. Since these feature vectors need to be respresentative of the total image they can also be used for recognition/classification tasks. With current resources extracting DELF features without training takes around 7 days. These features are then saved and used for the following approaches.

Experiments for this architecture was performed in two ways, firstly extracted features were processed through three different architectures for a labeled class based outlier analysis, these methods are the same ones that were used in compression approach; kNN, Isolation Forest and VAE. This was performed in order to decide whether extracted feature vectors were seperable inside their respective classes. Then a classifier has been trained in order to create an accuracy baseline so that comparison with the other approaches are possible and to clean the dataset in case this method ended up being highest performing one. While outlier detection methods showed better seperation of samples within samples of the same class compared to compression, it still was not able to distinguish between non representative samples clearly. This was observed through outlier scores shown in Table 4.2.

As an addition at this point, idea of utilizing FAISS architecture for outlier detection as another trial has come up. FAISS, being relatively recent, FAISS was expected to perform better than tested outlier detecion methods of kNN, Isolation Forest and VAE albeit still, it was ineffective for creating a clean dataset with least possible disturbances in the overall dataset. All of the implemented outlier detection methods, if tried on the entire dataset, would remove nearly quarter of the images as outliers.

While maybe accurate, this would worsen the already apperant class imbalance problem.

Using FAISS, clustering on validation samples are also made, in order compare DELF and DELG. Top-5 most similar images were processed for each of the validation features, then majority of the returned samples classes were taken as the "predicted" class. Accuracy for this approach for DELF features was 1%. Although statistically significant, considering random result would have been 0,1%, results were subpar, moving the current research towards another direction.

Next step for the extracted DELF features was to train a basic classifier, similar to the one trained for the compression approach. Reasoning behind this was to compare baseline accuracies for these features in order to find the best baseline approach. This would enable cleaning the dataset using prediction confidences as opposed to outlier detection systems. Main goal of training with DELF features was again to surpass the baseline model accuracy of 53% on Resnet-50. This would mean that DELF features were more representative of the overall dataset compared to the internally extracted ones from the ResNet-50 model. After training for 10 epochs, these features achived 55% accuracy on 1000 most prominent classes with 500000 image samples. Which is higher than that of ResNet-50 meaning better features did indeed yield better results as hypothesized. After this, iterational improvement and successor of DELF was the next step for this research.

4.3.2 DELG

Main issue of DELF is creating local features then from these local features acquiring a fixed length image feature vector of given size. Deep Local and Global Feture extractor (DELG) adresses this issue and creates both local and global features within the network using attention and pooling mechanisms. DELG uses a weakly supervised and supervised two stage architecture where local image descriptors are extracted and used for reconstruction of the image together with an attention map. In the second stage, model also extracts global features with an AutoEncoder like network from previously extracted attentive local descriptors. This approach ensures that model learns where in a given image main object of interest is, then creates global features from that area of the image. During training three different losses for the model is used, ArcFace loss is used for the classification performance of the global features, attention loss is used for attention mapping for local descriptors and reconstruction loss is used in order to effectively train local descriptor selection process. While extracting initial features with DELG took roughly the same time with DELF, size of the resulting features were radically more managable considering computational

Table 4.2 Outlier detection scores of tested features

Outlier detectors	AutoEncoder	DELF	DELG
kNN	0.10	0.15	0.15
Isolation Forest	0.23	0.35	0.37
VAE	0.32	0.41	0.42

constraints of this research. Tests showed that although achieving similar scores, outlier detection with DELG features took less then half the time of DELF features.

DELG being the latest iteration of feature extractors (as the time of writing) trained in Google Landmark Challenge dataset by Google; it is assumed that if any architecture could create a feature vector that was distinctively seperable between classes, it was DELG. Since it doesn't rely on other feature space dimentionality reduction schemes, extracting and saving DELG features was faster than DELF. However interclass boundaries were still not distinct enough for outlier detection approaches to seperate. This can be seen from the outlier detection score results in the Table 4.2.

Just as it was done with DELF, FAISS was again employed to create a cluster from the samples. Then a validation accuracy is calculated using the classes for top-5 most similar samples from the train dataset. This approach achieved 1,5% accuracy in validation. While it outperformed DELF, reliable cleaning was not feasible using this method.

As it was previously published in [26] DELG generally outperforms DELF when it comes to general image understanding related tasks. Utilizing the same hypothesis that if classification methodology for dataset cleaning is accurate and if DELG creates better global features, then resulting classification accuracies should also be higher using DELG with the same network. Similar to the approach with DELF, DELG features were trained using this hypothesis with a 10 epoch training on the 1000 most prominent classes. Results, compared to DELF improved in accuracy with a 57%, however considering necessity of "best possible classifier", it is decided to create a new classification approach to test this cleaning method.

4.4 Classification Architectures

After some deliberation, it was understood that grouping of a dataset of this scale was not feasible through some extracted features. However it was apparent that classification networks that were tested so far had always achieved better results from their outlier detector counterparts. With this in mind, next step of this research was relatively clear. It was possible to clean the dataset purely from the results of classifiers.

It was probable, in theory, to train a model, then use the confidence scores from the said model to remove poorly classified samples within the dataset. Hypothesizing that poorly classified samples have either no distinct features for a class, or has similary enough objects with other classes that their existance might cause the overall system to behave sub-optimally.

Moving towards a classification based approach meant that secondary baseline purely for classification was needed. Since neither DELF nor DELG was being used for their intended purposes, we could not rely on classification results from their respective papers. Instead another architecture to serve as a point of comparison between every previously used feature was decided to be implemented.

EfficientNet is a scalable model that has various versions each with different depth, width and parameter size values that can be seen in Figure 4.5. Intuition behind this scaling is simple; larger images have more meaningful features that can be extracted, therefore require more parameters. Compound scaling approach for EfficientNet uses coefficients that are used to increase or decrease both depth and width of the baseline model. When this approach is used on previous works like ResNet and MobileNet these models also displayed an increse in accuracy albeit in lower margins. This highlights the fact that when scaling, baseline model is also important. In this research decision to use EfficientNet as both a secondary baseline and probable method for cleaning was then made.

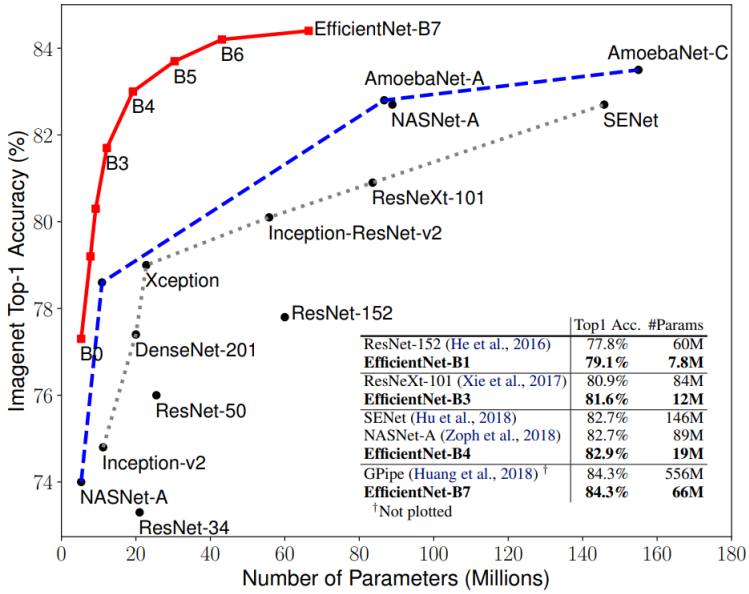


Figure 4.5 Top-1 accuracy and model size of EfficientNet family of models [11]

In this study, three of the EfficientNet models were trained in order to see which of the models performs best for GLDv2 dataset, within computational constraints. Resulting

observations were in-line with the papers authors that, increase in model size also increased accuracy. However, limitations for available computational power, rendered training the larger architectures in such a massive dataset infeasible. Therefore, of the three models that are tested, EfficientNet-B5 was the model of choice. In accordance with this decision, EfficientNet research accuracy report on ImageNet dataset shown in the Figure 4.5, which also shows a reduction in learning momentum after B5 model.

4.4.1 Classification of 1000,2000,15000 classes

Following the decision on the baseline model we trained the same 1000 class subset on the EfficientNet-B5 model for 10 epochs, initial training did not include any training methodology manipulations or class weight enforcements since main reason of the it was to create a baseline model. With the accuracy of 61% this method achieved highest classification accuracy so far.

In Figure 4.6, classification accuracies for all feature vectors used so far, can be observed. After decision to use EfficientNet-B5 has been made, Network has been

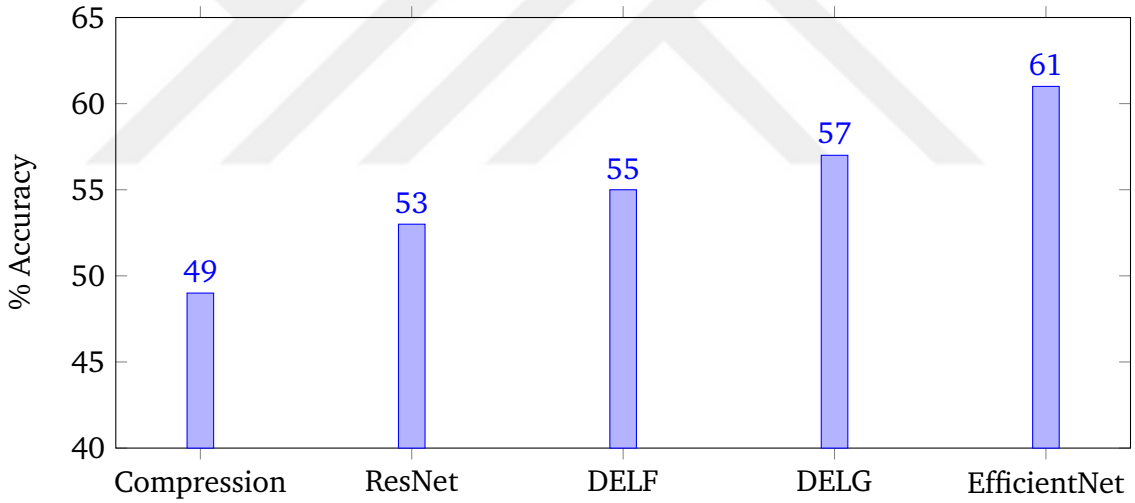


Figure 4.6 Classification accuracies for extracted features and baseline models

initialized with ImageNet weights and training has been started. Using categorical cross-entropy as loss function, stochastic gradient descent (SGD) with the momentum of 0.9 as optimizer and initial learning rate of 0.003 the training has been started.

Table 4.3 shows the accuracies for each subset of data, at the end of the training. From this it was apparent that increasing the number of classes was not causing a drastic reduction in accuracy. Therefore, since it provides a better distribution for the overall dataset, decision to use largest subset of 15000 classes has been made.

Following the decision of which subset to use, same EfficientNet-B5 model with basic

Table 4.3 Data subset and accuracies

Subset class count	Accuracy
1000	0.61
2000	0.60
15000	0.58

hyperparameters but this time with added learning rate scheduling factor of 0.95 every 50000 steps. This secondary training has been carried out for 5 epochs without class weights. Reason for this came from initial observations during previous trials, showing adverse affects for early introduction of class weights on converging speed for the model. After first 5 epochs training procedure was continued with class weights in order to enforce the model to better learn underrepresented classes. Since the architecture had 5 epochs to learn more represented classes, this introduction had no adverse effects. Figure 4.7 shows the validation accuracy graph for this training.

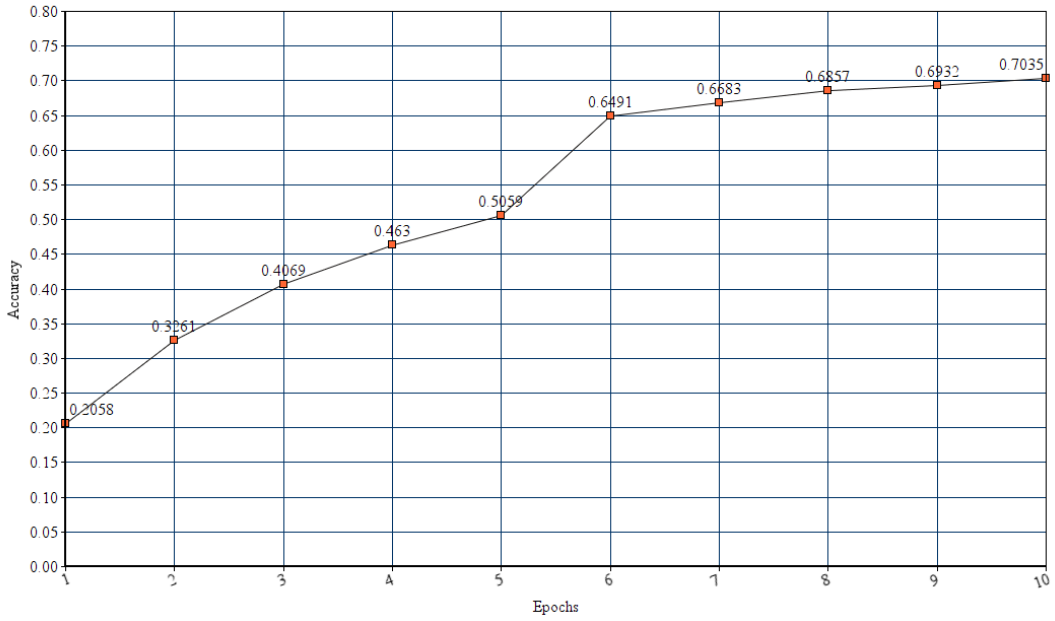


Figure 4.7 Accuracy graph of the model prior to cleaning

4.4.2 Thresholding

Before dataset cleaning starts, images that are going to be removed should be decided on. In order make this decision 6 seperate thresholds have been made with confidence levels of 20%,17,5%,15%,12,5%,10%,7,5%. Previous best model of EfficientNet-B5 on 15000 classes, was then ran on evaluation mode. Using this model 12 different dataframes were extracted, each threshold values had two dataframes of image names, one for "lower than threshold" confidence scores and one for higher. Table 4.4 shows

Table 4.4 Thresholds effect on class number and image sample size

Threshold	# of Removed images	# of Remaining classes
7,5	92	15000
10	2400	15000
12,5	36000	15000
15	147000	14900
17,5	381000	14400
20	682000	13000

number of classes remaining, number of samples remaining and percentile for the "lower than threshold" subset.

As it can be seen from the previously mentioned Table 4.4, lower thresholds of 10% and 7,5% did not remove substantial amount of the samples from the dataset. On the other hand higher threshold values of 20% and 17,5% caused reduction in the number of available classes. Remaning values of 15% and 12,5% were then investigated further in order to reach a decision. Decision boundary of 15% while not causing a drastic reduction in class count have been observed to reduce samples in some of the less populated classes significantly. For example in addition to completely removed ones, some classes have lost 90% of their images. This result meant that boundary of 15% was disruptive for the dataset as a whole and therefore decision to use threshold value of 12,5% has been made.

4.4.3 Dataset cleaning and reclassification

With model, subset and threshold of samples to be removed decided on, cleaning of the dataset began. All the samples in the below 12,5% confidence threshold dataframe were removed from the dataset. Of these samples, roughly 15% were actually correctly classified. Importance of this is as follows. Due to both method of performance measurement and means of cleaning being classification accuracy, this approach is highly susceptible towards the learning bias of the model. For example with a reduction of worst performing 10% of the dataset, accuracy increase of roughly 10% on the remainder of the dataset is trivial since it does not indicate any increase in models learning capability. Hovewer if the 20% of removed data is actually classified correctly, this result would mean that at least that many new samples are now learned by the model. This is because total number of correctly classified samples may not change if the only samples removed were mislabeled ones. Therefore definitive proof of increase in model performance could be stated as an increase in correctly classfied samples.

In order to test the cleaned dataset, same EfficientNet-B5 has been trained with exact same hyperparameters and methodologies, including the 5-5 epoch split. Distribution of train/test samples were also kept identical. Validation accuracies for this model can be found in the Figure 4.8 below. With these results, it can be seen that cleaning of the dataset led to an increase of 1,5% in accuracy, additionally even after the removal of 36628 samples, correctly labeled images went from 1277050 to 1279883. Considering the roughly 5000 correctly classified but removed samples, model learned to classify 7000 new images after cleaning is performed. Visual representation of success in this research can be defined in two seperate ways; removal of outliers and keeping true samples. Likewise failure can also be defined as failing to remove outliers or removal of true samples. In the Figures 4.9 some of the successful and unsuccessful data cleaning results can be seen.

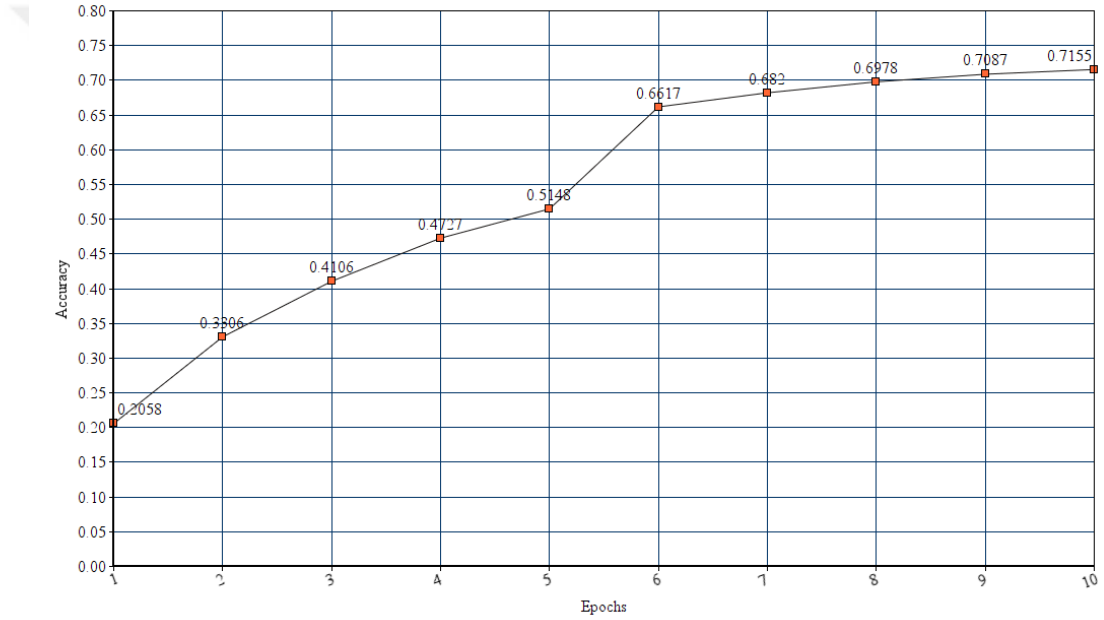


Figure 4.8 Accuracy graph of the model after cleaning

This shows, after cleaning, learning capability of the model has been improved, thus confirming our initial hypothesis. It is possible to automatically clean a dataset from noise. In the Figure 4.10 below, resulting pipeline for the developed architecture can be reviewed.



Figure 4.9 System results from two different classes in rows: a, b, f, g: Correct class, not removed. c, h: Correctly cleaned. d, j: Noise sample, failed to clean. e, k: Wrongly removed

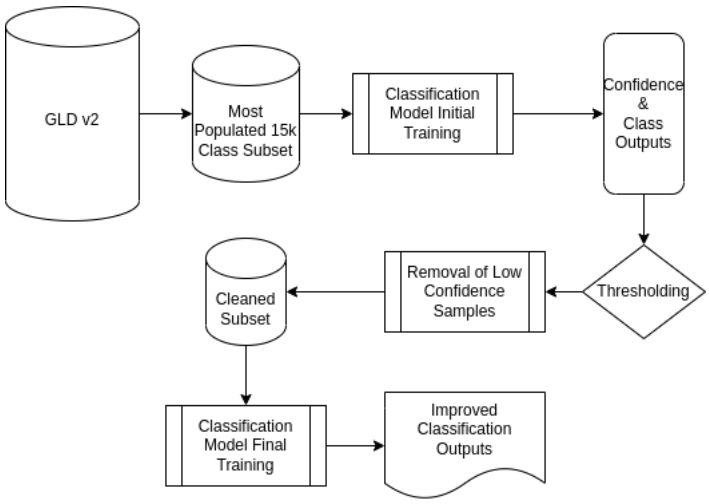


Figure 4.10 Final pipeline used in this research

5
RESULTS AND DISCUSSION

In this research, an automatical approach for cleaning an image dataset has been hypothesized, developed, evolved and explained. Proposed approach is then used on a large, noisy dataset to asses it‘s effecitiveness quantitatively. In this section result from the experiments will be evaluated verbally.

Initial understanding towards the dataset was formed from previous research on the Kaggle competition of 2019. While observations displayed that dataset cleaning was required, means on achieving said cleaning on such a large dataset needed more research and development. Since at this point extraction of DELF features were not feasible given computational constraints, a trial autoencoder was trained on a small subset of data for feature extraction. In the light of previous data cleaning attempts in other datasets , this features are then utilized for cleaning. This first attempt proved succesful in identifying the noise, in hindsight because of the small subset.

Proper research on the cleaning approach then began, with a fully developed AutoEncoder on a much larger subset of 1000 classes. While suprisingly succesful in recreating images with an SSIM of 0.79, this method ultimately was not fruitful in attempts of identifiying the noise due to larger number of classes making the class seperation difficult for AutoEncoder features. Achieving outlier scores of 10% in kNN, 23% in isolation forest and 32% in VAE. Utilizing another approach with FAISS, a clustering effort was then made in the extracted features achieving a clustering validation accuracy of 0,5%. Hovewer when classification architecture was ran through the extracted features, a classification accuracy of 49% was reached meaning that features could be made sense of using higher level representations.

In order to achieve higher level image features, DELF as a feature extractor is utilized, Using these features, previous methodoly repeated, this time achieveing outlier score of 15% in kNN, 35% in isolation forest and 41% in VAE. Similarly clustering validation accuracy of FAISS also improved to 1%. Classification approach is then tested achieving 53% validation accuracy. These results showed that accross the board better

features resulted in better outputs albeit they were not good enough to facilitate creation of a clean dataset.

As the next iteration for Google's family of feature extractors for GLD dataset, DELG was the next step in testing. Using global features of DELG, same trials have been performed, with the resulting outlier scores of 15% in kNN, 37% in isolation forest, 42% in VAE. FAISS validation accuracy also improved to 1,3%. Largest improvements was to classification validation accuracy hovewer, with 57%. At this point in the research, it was apparent that, without a training scheme on DELG or DELF features, achieving better results was wishful thinking.

Since all prior experiments displayed an observable tendancy towards classification, next step of the development was the training of a classfier for dataset cleaning purposes. In this approach three models from EfficientNet family of architectures are trained with a 1000 class subset of the data. Resulting accuracies were 61% for B5, 63% for B6 and 64% for B7. While using larger models increased accuracy, computational power requirement doubled going from B5 to B6, making B5 the most efficient model in this use case. Since the highest validation accuracy was achieved with EfficientNet family, classification for data cleaning was done on this architecture.

After decision to use EfficientNet-B5 has been made, previously established subset are then trained on this model for 10 epochs with no training time tricks. While 1000 class subset achieved highest accuracy, differance was minimal with 61% on 1000 classes, 60% on 20000 classes and 58% on 15000 classes. For this reason decision to use largest subset available has been made.

A 15000 class subset of the data is trained with previously decided EfficientNet B5 model; after training for 10 epochs, this model achieved 70% accuracy on the validation dataset, marking this highest accuracy achieved so far. Cleaning procedure is then made using confidence scores of the model, with a removal threshold being 12,5%. This approach removed 37000 samples from the total data of 1.8M. Remainder of the dataset is then trained with the same model from scratch, achiving accuracy of 71,5%, an increase of 1,5%.

Above result made it clear that very small number of noisy samples could in fact, disturb the training procedure of an architecture, and cleaning these helps improve results achieved. These observations however produce another question; if more data is removed would the results continue improving? To this end removal with a higher threshold value of 20% is made, causing the system to lose ~2000 of the classes along with 600k image samples. Although accuracy improvements continued with 81%, removal of 30% of available data constituted as greatly disturbing dataset distribution.

Aforementioned trials highlighted the importance of choosing a good threshold. In continuation of this research it is hypothesized that continuesly removing samples with lower tresholds could result in near complete cleaning of the dataset. In order to test this hypothesis, remaining dataset after this continual removal will be compared against existing manually cleaned version of the same dataset in order to acquire a quantifiable performance result.

While working with FAISS, it was observed that being an algorithm for similarity search between indexed samples and queried input, it intrinsically assumes clusters from the training samples are generated correctly. This makes methodology that this study used susceptible to clustering errors from noisy data. In order to alleviate this issue and test the hypothesis corretly, another idea was needed. To this end, idea of clustering classes internally was put forth. In this approach, creation of two clusters for each class was conceptualized. One for normal samples and one for anomalies. After training, cluster with the more samples can then be used as the "real" class while the samples on the other cluster can be removed as "noise". This approach, while promising in the initial test runs with more populous classes, fails to solve interclass noise within the dataset, thus requiring another step for further cleaning. However improvements on this idea is thought to be another future step for this research.

Additionally for further study; training EfficientNet architecture with attention and ArcFace loss could improve intraclass boundaries. While this approach shows promise in similar areas, training convergence time increases by tenfold, which made training this model in a timely manner relatively impossible given computation power constraints.

REFERENCES

[1] S. Ren, K. He, R. Girshick, J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[2] J. Long, E. Shelhamer, T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[3] M. Cordts *et al.*, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223.

[4] K. Simonyan, A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[5] K. He, X. Zhang, S. Ren, J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. arXiv: 1512 . 03385. [Online]. Available: http://arxiv.org/abs/1512.03385.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[7] H. Noh, A. Araujo, J. Sim, T. Weyand, B. Han, "Large-scale image retrieval with attentive deep local features," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 3456–3465.

[8] C. Sun, A. Shrivastava, S. Singh, A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 843–852.

[9] X.-B. Nguyen, D. T. Bui, C. N. Duong, T. D. Bui, K. Luu, "Clusformer: A transformer based clustering approach to unsupervised large-scale face and visual landmark recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2021, pp. 10 847–10 856.

[10] C. Xu *et al.*, "3rd place solution to google landmark recognition competition 2021," *CoRR*, vol. abs/2110.02794, 2021. arXiv: 2110 . 02794. [Online]. Available: https://arxiv.org/abs/2110.02794.

[11] M. Tan, Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*, PMLR, 2019, pp. 6105–6114.

[12] Z. Liu *et al.*, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 012–10 022.

[13] H. Wu *et al.*, "Cvt: Introducing convolutions to vision transformers," *CoRR*, vol. abs/2103.15808, 2021. arXiv: 2103.15808. [Online]. Available: https://arxiv.org/abs/2103.15808.

[14] Z. Kim *et al.*, "Towards A fairer landmark recognition dataset," *CoRR*, vol. abs/2108.08874, 2021. arXiv: 2108.08874. [Online]. Available: https://arxiv.org/abs/2108.08874.

[15] K. Ozaki, S. Yokoo, "Large-scale landmark retrieval/recognition under a noisy and diverse dataset," *CoRR*, vol. abs/1906.04087, 2019. arXiv: 1906.04087. [Online]. Available: http://arxiv.org/abs/1906.04087.

[16] H. Xu, P. Koehn, "Zipporah: A fast and scalable data cleaning system for noisy web-crawled parallel corpora," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 2945–2950. DOI: 10.18653/v1/D17-1319. [Online]. Available: https://aclanthology.org/D17-1319.

[17] A. Veit, N. Alldrin, G. Chechik, I. Krasin, A. Gupta, S. Belongie, "Learning from noisy large-scale datasets with minimal supervision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 839–847.

[18] H.-W. Ng, S. Winkler, "A data-driven approach to cleaning large face datasets," in *2014 IEEE international conference on image processing (ICIP)*, IEEE, 2014, pp. 343–347.

[19] W. Wang, V. W. Zheng, H. Yu, C. Miao, "A survey of zero-shot learning: Settings, methods, and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–37, 2019.

[20] T. B. Brown *et al.*, "Language models are few-shot learners," *CoRR*, vol. abs/2005.14165, 2020. arXiv: 2005.14165. [Online]. Available: https://arxiv.org/abs/2005.14165.

[21] Z. Min, Z. Jiguo, X. Xusheng, "Panorama stitching based on sift algorithm and levenberg-marquardt optimization," *Physics Procedia*, vol. 33, pp. 811–818, 2012.

[22] D. H. Ballard, "Modular learning in neural networks.," in *Aaai*, vol. 647, 1987, pp. 279–284.

[23] Z. Wang, E. P. Simoncelli, A. C. Bovik, "Multiscale structural similarity for image quality assessment," in *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, Ieee, vol. 2, 2003, pp. 1398–1402.

[24] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, Ieee, vol. 2, 1999, pp. 1150–1157.

[25] M. A. Fischler, R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[26] B. Cao, A. Araujo, J. Sim, "Unifying deep local and global features for image search," in *European Conference on Computer Vision*, Springer, 2020, pp. 726–743.

[27] J. Deng, J. Guo, N. Xue, S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4690–4699.

[28] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[29] T.-Y. Lin *et al.*, "Microsoft coco: Common objects in context," in *European conference on computer vision*, Springer, 2014, pp. 740–755.

[30] P. Yang, B. Huang, "Knn based outlier detection algorithm in large dataset," in *2008 International Workshop on Education Technology and Training & 2008 International Workshop on Geoscience and Remote Sensing*, IEEE, vol. 1, 2008, pp. 611–613.

[31] F. T. Liu, K. M. Ting, Z.-H. Zhou, "Isolation forest," in *2008 eighth ieee international conference on data mining*, IEEE, 2008, pp. 413–422.

[32] D. P. Kingma, M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[33] J. Johnson, M. Douze, H. Jégou, "Billion-scale similarity search with GPUs," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2019.

[34] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[35] A. G. Howard *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[36] "State of the art architectures on imagenet." Accessed: 2022-06-19. (2022), [Online]. Available: `https://paperswithcode.com/sota/image-classification-on-imagenet`.

[37] Martín Abadi *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: `https://www.tensorflow.org/`.

[38] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *CoRR*, vol. abs/1912.01703, 2019. arXiv: `1912.01703`. [Online]. Available: `http://arxiv.org/abs/1912.01703`.

[39] Y. Zhao, Z. Nasrullah, Z. Li, "Pyod: A python toolbox for scalable outlier detection," *Journal of Machine Learning Research*, vol. 20, no. 96, pp. 1–7, 2019. [Online]. Available: `http://jmlr.org/papers/v20/19-011.html`.

[40] K. He, X. Zhang, S. Ren, J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

PUBLICATIONS FROM THE THESIS

Conference Papers

1. B. Taskin, M. E. Karslıgil, "Cleaning Landmark and Geolocation Dataset Using Deep Learning Methods," 2021 29th Signal Processing and Communications Applications Conference (SIU), 2021, pp. 1-4, doi:10.1109/siu53274.2021.9478031.