TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Karl Raud 193632IAIB

DATASET FILTERING FOR IMAGE CLASSIFICATION AND OBJECT DETECTION MODELS

Bachelor's Thesis

Supervisor: René Pihlak
MSc

Tallinn 2023

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Karl Raud 193632IAIB

ANDMESTIKU FILTREERIMINE PILDI KLASSIFITSEERIMISE JA OBJEKTITUVASTUSE MUDELITELE

Bakalaureusetöö

Juhendaja: René Pihlak
MSc

Tallinn 2023

Author's Declaration of Originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Karl Raud

22.05.2023

Abstract

Dataset filtering can reduce the training times of a model, lower the annotation costs of the dataset, and occasionally even improve the resulting model's performance in cases of poor dataset quality.

In this thesis, multiple clustering methods are employed to filter an image dataset on a semantic level. This is achieved by extracting features from a trained model to compare the specific neural network's perception of the data, as opposed to comparing the raw images themselves or letting other models create their own understanding of the dataset.

The proposed method categorizes the redundant images in the dataset into two types: images that are semantically too similar and outliers. Since the amount of filtered data is determined by the user, an application is also developed to facilitate a better understanding of the dataset's semantic content.

The proposed method is tested on multiple models and datasets, demonstrating that redundancies can be found in many datasets.

The thesis is written in English and is 36 pages long, including 6 chapters, 18 figures and 2 tables.

Annotatsioon

Andmestiku filtreerimine pildi klassifitseerimise ja objektituvastuse mudelitele

Andmestiku filtreerimine võib vähendada masinõppemudeli treenimise aega, andmestike annoteerimistasusid ning halva andmestiku kvaliteedi korral võib ka parendada mudeli tulemusi.

Selles töös kasutatakse mitmeid klasterdamis meetodeid, et filtreerida pildiandmestikku tähenduslikul tasemel. Seda tehakse treenitud närvivõrgu vahekihtidelt andmete võrdlemisega, et arvestada seda, mida just see närvivõrk andmetes tähenduslikult näeb.

Pakutud meetod poolitab mittevajalikud pildid andmestikus kahte osasse: liiga sarnased pildid tähenduslikus mõttes ja võõrväärtuslikud pildid, mis ei sobi ülejäänud andmestikku. Ka rakendus on loodud et kasutaja saaks andmestikku paremini hoomata ning hinnata, et kui palju pilte andmestikust välja filtreerida.

Pakutud meetod on testitud mitmete mudelite ja andmestikkudega, mille tulemused viitavad, et mittevajalikke pilte leidub paljudes andmestikkudes.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 36 leheküljel, 6 peatükki, 18 joonist, 2 tabelit.

List of Abbreviations and Terms

3D	Three Dimensional
CPU	Central Processing Unit
GPU	Graphical Processing Unit
HDBSCAN	Hierarchical Density-Based Spatial Clustering of Applications with Noise
mAP	Mean Average Precision
PCA	Principal Component Analysis
ReLU	Rectified Linear Unit
RGB	Red-Green-Blue
SGD	Stochastic Gradient Descent
UMAP	Uniform Manifold Approximation and Projection
YOLO	You Only Look Once algorithm

Table of Contents

1 Introduction **9**

2 Background **10**
2.1 Neural networks 10
2.2 Image datasets 12
2.2.1 Redundant images in the dataset 13
2.2.2 Outliers in the dataset 13

3 Related works **14**
3.1 The 10 % you don't need 14
3.2 FiftyOne 14

4 Proposed method **16**
4.1 Model and dataset 16
4.2 Filtering the feature maps 17
4.2.1 Outlier detection 19
4.2.2 Semantic similarity detection 19
4.2.3 Visualising the left out images 20
4.3 Application 20
4.4 Filtered dataset evaluation 22

5 Analysis **23**
5.1 Image classification 23
5.1.1 CIFAR-10 & CIFAR-100 23
5.1.2 Animal classification 26
5.2 Object detection 28
5.2.1 Cityscapes 28
5.2.2 Traffic signs 30

6 Summary **32**

References **33**

Appendix 1 – Non-Exclusive License for Reproduction and Publication of a Graduation Thesis **37**

Appendix 2 – Filterable interface **38**

Appendix 3 – Found redundancies in CIFAR-10 dataset **39**

Appendix 4 – Found redundancies in cityscapes dataset **41**

Appendix 5 – Found redundancies in traffic lights dataset **44**

List of Figures

1 An example of a neural network. 10
2 Predictions of a underfitted, robust and a overfitted model. 12
3 A 3D array and the resulting RGB image. 13
4 A pair of images from the traffic signs dataset. 15
5 A flowchart of the proposed method. 16
6 Visualization of a 2D point cloud being transformed to 1D by PCA. 18
7 Visualization of a 3D point cloud being transformed to 2D by UMAP. . . 18
8 2D point cloud and its minimum spanning tree. 19
9 CIFAR-10 dataset as a three dimensional point cloud. 21
10 Resnet32 model for CIFAR datasets. 23
11 CIFAR-10 filtered with $\%_{outlier} = 50\,\%$. 24
12 Comparison of different dataset filtering methods on the CIFAR-10 dataset. 25
13 Comparison of different dataset filtering methods on the CIFAR-10 dataset. 26
14 Comparison of different dataset filtering methods on animal classification task. 27
15 Comparison of model scores trained with different dataset filtering methods on the cityscapes dataset. 29
16 Semantically similar images in Cityscapes dataset. 29
17 Comparison of model scores trained with different dataset filtering methods on the traffic signs dataset. 30
18 Outliers in traffic signs dataset. 31

List of Tables

1 Average test accuracy over 5 trials on the CIFAR-10 dataset. 25
2 Average test accuracy over 5 trials on the CIFAR-100 dataset. 26

1 Introduction

Neural networks are widely used to solve various complex tasks from facial recognition to weather forecasting. The global market size was over $14.35 B in 2020 and is expected to reach $152.61 B by 2030. [1] Neural networks mimic the human brain and in order to teach it, neural networks need to be trained on data. [2]

Data science communities such as Kaggle [3] provide many datasets for training neural network models. As the amount of available datasets increases, machine learning engineers will value the quality of the dataset over quantity.

One of the quality metrics is the accuracy of the annotations which will impact the resulting model negatively [4]. The quality control of annotations is usually done by humans [5] and, therefore, is not easily automated.

Another metric is the quality of the dataset contents on a semantic level. Too many duplicate or similar images of the same scenario, as well as irrelevant images, can worsen the results of the trained model. [6]

The goal of this work is to detect and remove unneeded images from a dataset, comparing the contents on a semantic level. The resulting subset should represent the original dataset as accurately as possible, while improving or maintaining comparable model training results and reducing the number of irrelevant and similar images. Additionally, the algorithm must be generalized to work with any model and any dataset, even those without annotations. Therefore, the specifics of a single problem are not considered.

This paper consists of six sections. Introduction concludes the first. The second section describes the background information on the topic theoretically. The third section describes related works and other dataset analysis tools. The fourth section explains the proposed method. The fifth section presents the analysis of the method using multiple models and datasets. The last section provides a summary.

2 Background

This section describes the necessary details and technologies to help understand the proposed method.

2.1 Neural networks

Neural networks are a multi-layer networks of neurons that make a series of transformation on the data to generate their own understanding of it. It consists of three types of layers: input layer, hidden layers and output layer. [7] See Figure 1 for an illustration.

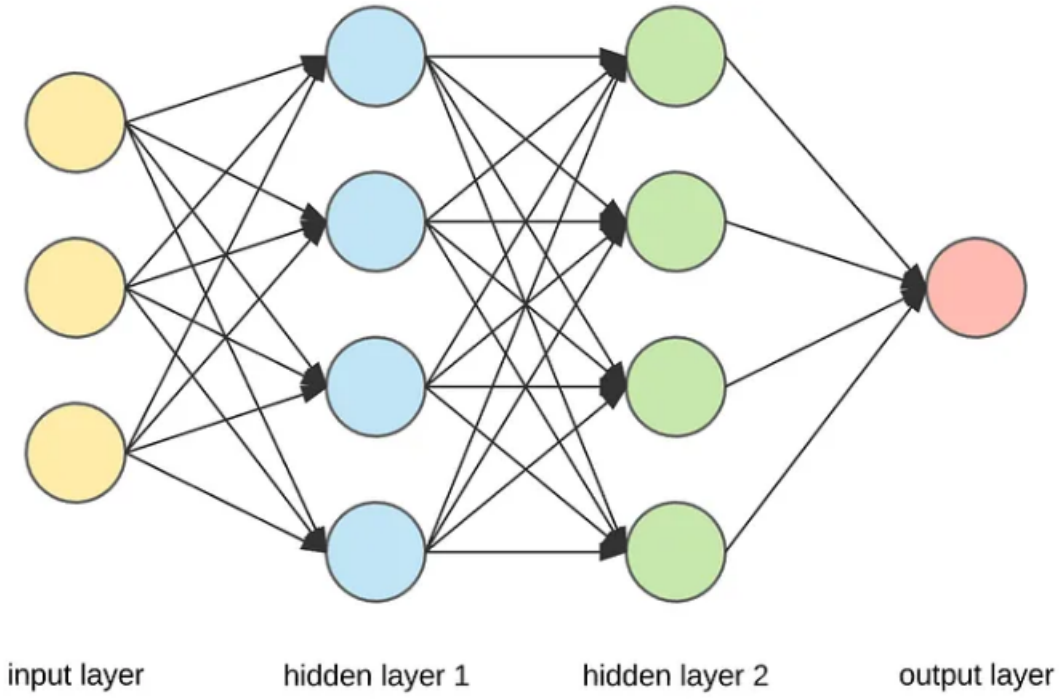

Figure 1. An example of a neural network. [7]

Hidden layers are responsible for learning the mapping between input and the desired output. Each neuron has multiple inputs, weights, and an output, which is computed as shown in equation 2.1. The activation function $f(x) = y$ maps the resulting value into a desired range. Examples include Rectified Linear Unit (ReLU) $f(x) = max(0, x)$ and Linear $f(x) = x$ activation functions. [8] A **feature map** is the output of a specific hidden layer. [9]

$$output = activation\left(\sum (Weight_i \cdot Input_i) + Weight_{bias}\right) \tag{2.1}$$

Loss function computes the distance (loss) between the current and the expected output as shown in equation 2.2. [10] Cross-entropy loss is usually used in classification tasks, where the output of the network consists of probabilities. It is calculated as shown in equation 2.3 where M denotes the number of classes, $y_{o,c}$ is a binary indicator if class label c is the correct classification for observation o and $p_{o,c}$ is the predicted probability that o is of class c. [11]

$$loss = loss\_function(Output_{current}, Output_{expected}) \tag{2.2}$$

$$-\sum_{c=1}^{M} y_{o,c} \log p_{o,c} \tag{2.3}$$

Optimizer is an algorithm that minimizes the loss by gradually adjusting the weights of the model. [12] This is done using **back-propagation** algorithm which calculates gradients for each parameter of the model. An example of Stochastic Gradient Descent (SGD) optimization is shown in equation 2.4 where C is the loss function, w denotes the weights of the model and ϵ is the learning rate. The gradient $\frac{\partial C}{\partial w}$ describes the direction that the weights must be adjusted towards, to minimize the loss function. [13]

$$w = w - \epsilon \frac{\partial C}{\partial w} \tag{2.4}$$

Models are **trained**, by first randomizing all of the weights. The dataset is split into batches. In each **step**, one batch of data is forwarded through the model. After each step, the loss and the gradients are calculated and the weights are adjusted. An **epoch** is over, when the whole dataset has been forwarded through once.

As seen on Figure 2, an underfit model does not learn the relation between input and output enough, resulting in inaccurate guesses. An overfit model has learned the relation too well, so it does not generalize the task in order to do well on new data. The model is typically trained until the loss no longer decreases or the model has reached a state at which it could overfit. The latter can be detected by evaluating the model on another dataset (test dataset) after each epoch, which is not used for training.

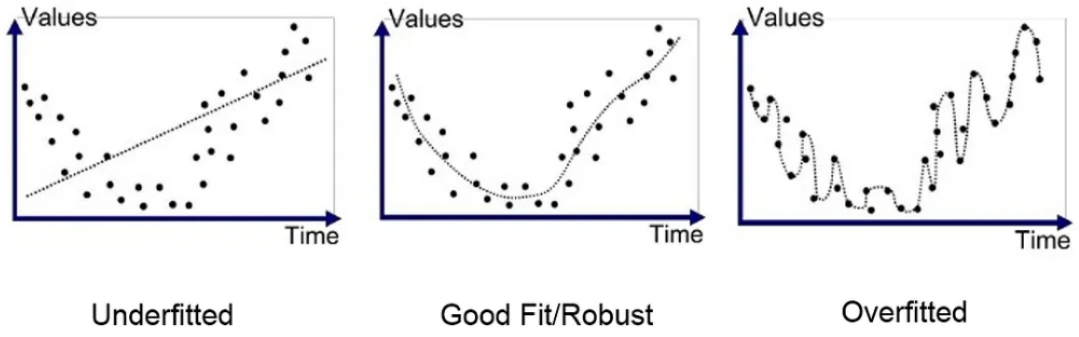

Figure 2. Predictions of a underfitted, robust and a overfitted model. [14]

Training performance

Training neural networks on the Central Processing Unit (CPU) is slower than on the Graphical Processing Unit (GPU) because they can handle more tasks at a time due to the much higher core counts and memory bandwidth. [15] **GPU cluster** is a shared group of computers that have a GPU on every node. [16]

To accelerate training, Taltech AI-LAB GPU cluster was used in this work. [17] It has six nodes that are controlled by Slurm workload manager. Software for the jobs is prepared on the head node and then the job is passed on to the workload manager which assigns the required hardware resources to run it. An example Slurm job with 40 % of a GPU, 8 CPU cores and 16 GB of memory is shown below.

```
sbatch --gres mps:40 --cpus-per-task=8 --mem=16G ./task.sh
```

2.2 Image datasets

Dataset size and quality are usually in direct correlation with the model's results. An image dataset usually consists of Red-Green-Blue (RGB) images and annotations.

RGB images are represented as 3D arrays, where one dimension describes color and the others represent locations of each pixel. See Figure 3 for an example.

Annotations are descriptions that reflect the visual contents of each image. For classification tasks, it can be an integer or a string describing the type of the image. Object detection annotations need to describe the location and type of the objects present. Location is often described as a bounding box, polygon or an anchor point. The majority of annotation is done by humans with a market over $1 B. [18]

$$\begin{bmatrix} (0,0,0) & (1,1,1) & (1,0,0) \\ (0,1,0) & (0,0,1) & (1,1,0) \\ (0,0,0.5) & (0.5,0.5,0) & (0,0,0) \end{bmatrix}$$

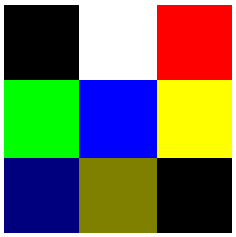

Figure 3. A 3D array and the resulting RGB image.

2.2.1 Redundant images in the dataset

It is proven, that 10 % of the CIFAR-10 [19] and ImageNet [20] datasets are not needed for image classification because of the semantic similarity of the images. [21] If the dataset has too many similar images, the model is in risk of getting biased. It will focus more on learning a specific scene or feature and it will reduce the generalization factor of the result, thus reducing test accuracy. These redundancies can occur more often on video datasets [22] and synthetic datasets, where multiple transformations of the same image or scene are used.

2.2.2 Outliers in the dataset

Outliers are images, that stand out from the rest of the dataset. These could include images that are incorrectly annotated, corrupted or noisy. FiftyOne [23] is a interactive tool to detect these anomalies. In this approach, images that are not semantically similar to rest of the dataset are considered as outliers.

3 Related works

This section describes other related approaches to the problem and discusses their potential shortcomings.

3.1 The 10 % you don't need

Semantic Redundancies in Image-Classification Datasets: The 10 % You Don't Need [21] filters the dataset by removing semantically similar images out of CIFAR-10, CIFAR-100 and ImageNet datasets. This was done using agglomerative clustering on the feature maps taken from the last layers of the Resnet32 network.

The results were positive on the CIFAR-10 and ImageNet datasets. It was proved that upon removing 5 % to 10 % of the dataset, the resulting model would have no drop in validation accuracy on average.

The mentioned approach filters the data belonging to each class independently, so it requires the dataset to be annotated. Also as some models can output feature maps with over 100 000 dimensions, the performance of agglomerative clustering could get worse.

3.2 FiftyOne

FiftyOne [23] is a tool which can detect similar images, annotation faults and datapoint uniqueness.

Similar image detection and uniqueness detection are done by comparing either the outputs of the model or the raw RGB images. Because the output of the model can contain less information about the semantic meaning of an image, this could lead to worse results than using middle layers of the network. [21]

Searching for similarities in RGB images themselves does detect duplicates but similarity detection can be inaccurate due to similar pixel values but different meanings of the image. An example of a potentially misleading pair is shown in Figure 4 in which the images look similar but the annotations of the signs are from different classes.

Figure 4. A pair of images from the traffic signs [24] dataset. The left image contains only “other” classified signs and the right image contains only "prohibitory" classified signs.

4 Proposed method

This section describes the details of the method used. The implementation is done using Python 3. PyTorch library is used to implement or import neural network models.

Sections 4.1 and 4.2 describe the proposed method in detail and a summarizing flowchart of the method is shown in Figure 5.

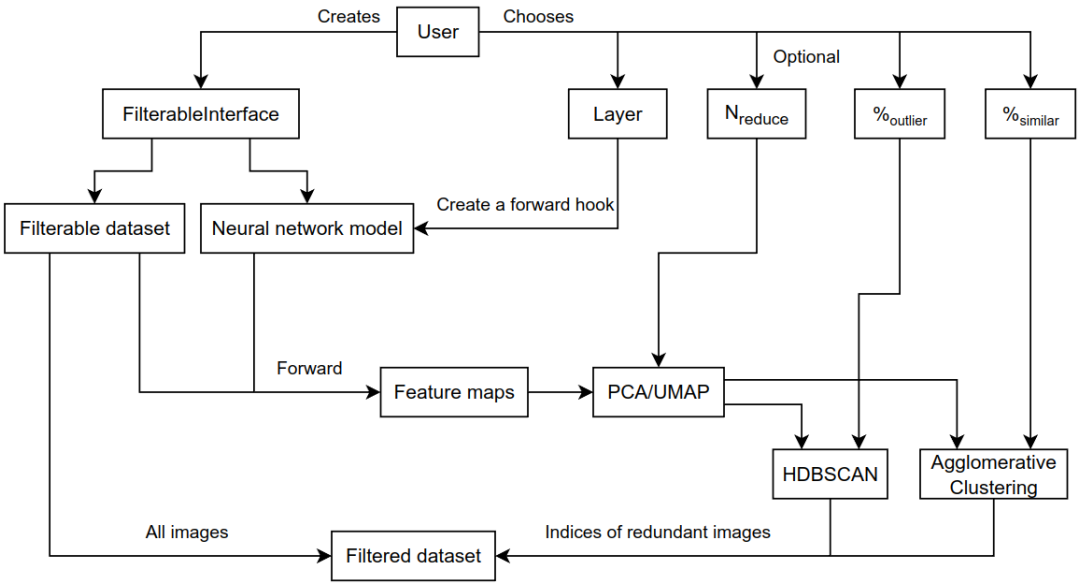

Figure 5. A flowchart of the proposed method.

4.1 Model and dataset

The method proposed assumes that there already exists a model and a dataset. For the model to create as accurate semantic representations of the dataset as possible, a part of the dataset must be initially annotated. The annotated dataset was divided into train and test datasets. The model is initially trained on the annotated train dataset until it converges. To avoid overfitting, the best performing model of the training process, chosen by test accuracy, will be considered as the baseline model.

Feature map extraction is done by using a forward hook in the baseline model in order to extract feature maps of all images in a chosen layer. These images are not required to be annotated.

The chosen layer to extract feature maps from should represent the whole input image on

a semantic level. Layers toward the start of the model tend to contain more information about the pixel values of the image. As the output of the network contains only semantic information about the image, a layer towards the end of the model should be chosen. Due to the last layer often being low in dimensionality, often containing too little information about the original image, it should also not be chosen. Fully-connected layers before the output layer have shown the best performance. [21]

4.2 Filtering the feature maps

To filter the extracted feature maps, the user must specify the percentage $\%_{outlier}$ of which outliers are removed and the percentage $\%_{similar}$ of which semantically similar images are removed from the dataset. These numbers should be estimated by the user based on the quality of the dataset.

Since the amount of memory and time required process and store the feature maps scales with their amount N_{maps} and their dimensionality $N_{features}$ like $N_{variables} = N_{maps} \cdot N_{features}$, feature map dimensionality $N_{features}$ is reduced with Principal Component Analysis (PCA) or Uniform Manifold Approximation and Projection (UMAP).

Depending of the amount of memory available, the user can specify a parameter N_{reduce}, the reduced feature count. The final dimensionality is calculated as $N_{dim} = min(N_{reduce}, N_{maps}, N_{features})$. It is assumed that if more dimensions are kept then the feature map will retain more accurate semantic information. So N_{reduce} is a trade-off parameter between computational cost and the accuracy of the filtering process.

Principal Component Analysis

PCA rotates the angle of the point cloud in order to find a good fit to represent the data in lower dimensions. Figure 6 illustrates a 2D point cloud being rotated. In the rotated point cloud $pc2$ axis contains little information about the rotated dataset so it is eliminated. We can represent the original 2D dataset with a single dimension $pc1$ that retains most of the information.

Uniform Manifold Approximation and Projection

UMAP [26] reduces the dimensionality of a point cloud by first creating a graph representation of it and then optimizes a lower-dimensional graph to be as structurally similar as possible to the original one. [27] Figure 7 shows how UMAP reduces the point cloud of a 3D elephant to a 2D point cloud while preserving the global relative structure of the points.

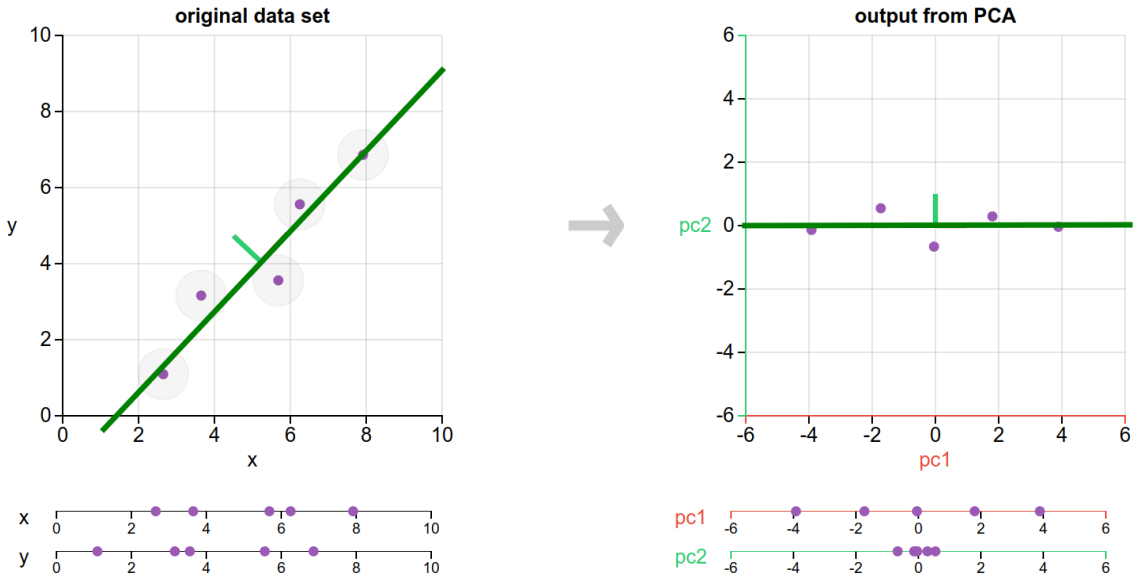

Figure 6. Visualization of a 2D point cloud being transformed to 1D by PCA. [25]

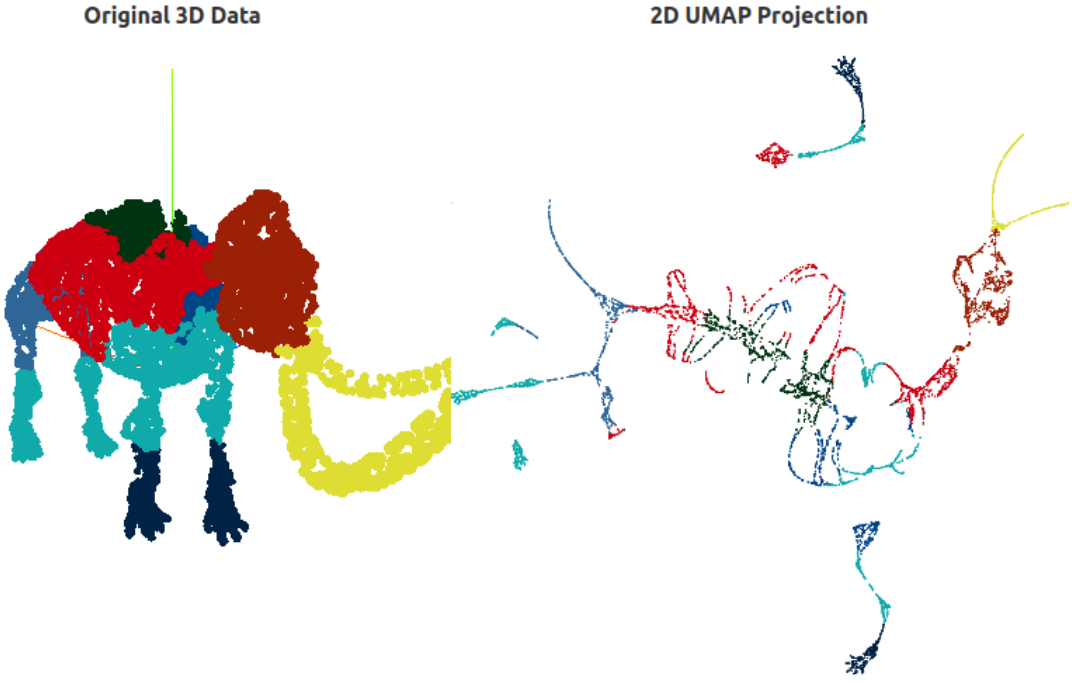

Figure 7. Visualization of a 3D point cloud being transformed to 2D by UMAP. [27]

4.2.1 Outlier detection

Outliers in the dataset should stand out by their dissimilarity (distance) to other feature maps. To detect these outliers, Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) was used. It is chosen due to its performance on high-dimensional data. Because HDBSCAN is much slower on the CPU [28, 29], RAPIDS core library `cuml` [30] was used to accelerate the computation on the GPU.

Considering the feature maps in N_{dim}-dimensional space as a complete graph weighted with euclidean distances, HDBSCAN creates a minimum spanning tree. [31] A 2D illustration is show on Figure 8. After the vertexes are sorted decreasingly by distances to their neighbors, outliers can be detected by taking the first $\lceil \%_{outlier} \cdot N_{maps} \rceil$ feature maps. The rest of the points were detected as a single cluster.

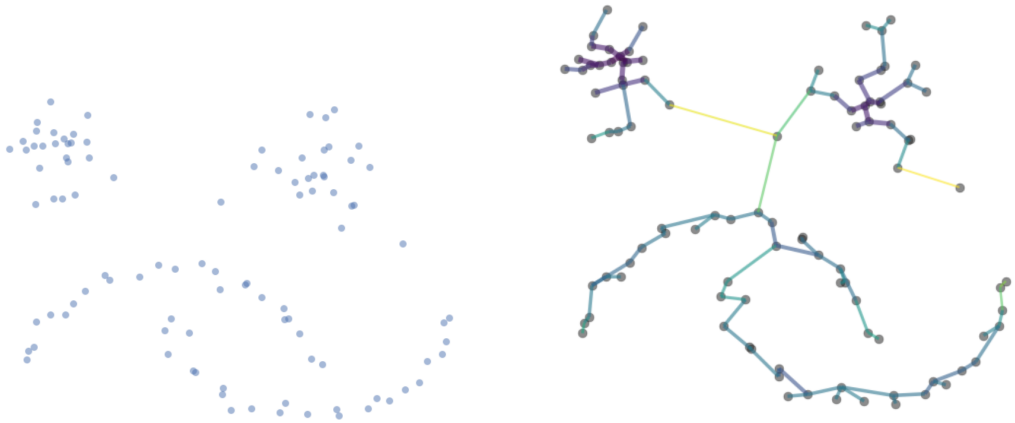

Figure 8. 2D point cloud and its minimum spanning tree. [31]

4.2.2 Semantic similarity detection

Semantic similarity appears when multiple feature maps are close to each other. This can be detected by using **agglomerative clustering**. [21, 30]

Agglomerative clustering starts by assigning each feature map to its own cluster. Two closest clusters by cosine distance (equation 4.1) are merged until $\lfloor (100\% - \%_{outlier} - \%_{similar}) \cdot N_{map} \rfloor$ clusters are left. In each cluster, the feature map closest to the cluster center is kept. The rest are discarded as too semantically similar to it.

$$Distance(M_1, M_2) = 1 - \frac{\vec{M_1}\vec{M_2}}{||M_1|| \cdot ||M_2||} \tag{4.1}$$

4.2.3 Visualising the left out images

Due to the high dimensionality of the images and feature maps, visualising the whole dataset on a semantic level is hard. [32] To counteract this, PCA or UMAP was used to reduce the data to three dimensions. In three dimensions, the dataset can be plotted. In all of dataset visualization plots, **green points** represent feature maps that are not filtered out. **Red points** represent outliers. **Blue points** represent too semantically similar datapoints. The lines from blue points point to the feature map that is in the center of its cluster. An example can be seen on Figure 9.

Due to the finding of semantically similars and outliers is done with feature maps of N_{dim} dimensions, the representation and locations of the left-out feature maps is not completely accurate. As we lower N_{dim}, less information about the feature maps is retained so it is assumed that filtering the dataset in higher dimensions is more accurate. For illustration, filtering at $N_{dim} = 3$ the representation becomes accurate as shown in the lower image of Figure 9.

4.3 Application

An application with a graphical user interface was made using Python tkinter [33] library. The application requires the user to first create an implementation of an interface as shown in Appendix 2. The PyTorch model returned should be trained so once the dataset is forwarded through the model, the feature maps extracted contain as accurate semantic information as possible. The filterable dataset returned should be an iterable of batches of data. Each batch must also have metadata attached which describes each item in the batch. This could be a file name, an index or anything else unique.

After the user has chosen the layer for the feature maps to be extracted from, has specified $\%_{outlier}$ and $\%_{similar}$ and optionally the parameter N_{reduce}, the dataset can be filtered. The application plots the dataset as a three dimensional point cloud with colored outliers and similars. If optional get_image function in the Python interface is also implemented, the application shows the left-out images and the reason (other image) why it is filtered out. Finally, the user can export the metadata of each item in the filtered dataset to a text file.

The application is open source and publicly available in a GitHub repository. [34] The repository includes a "readme" file that describes the setup process and a detailed description of the usage. Two example models with datasets are provided to either be filtered using the application or with Python code.

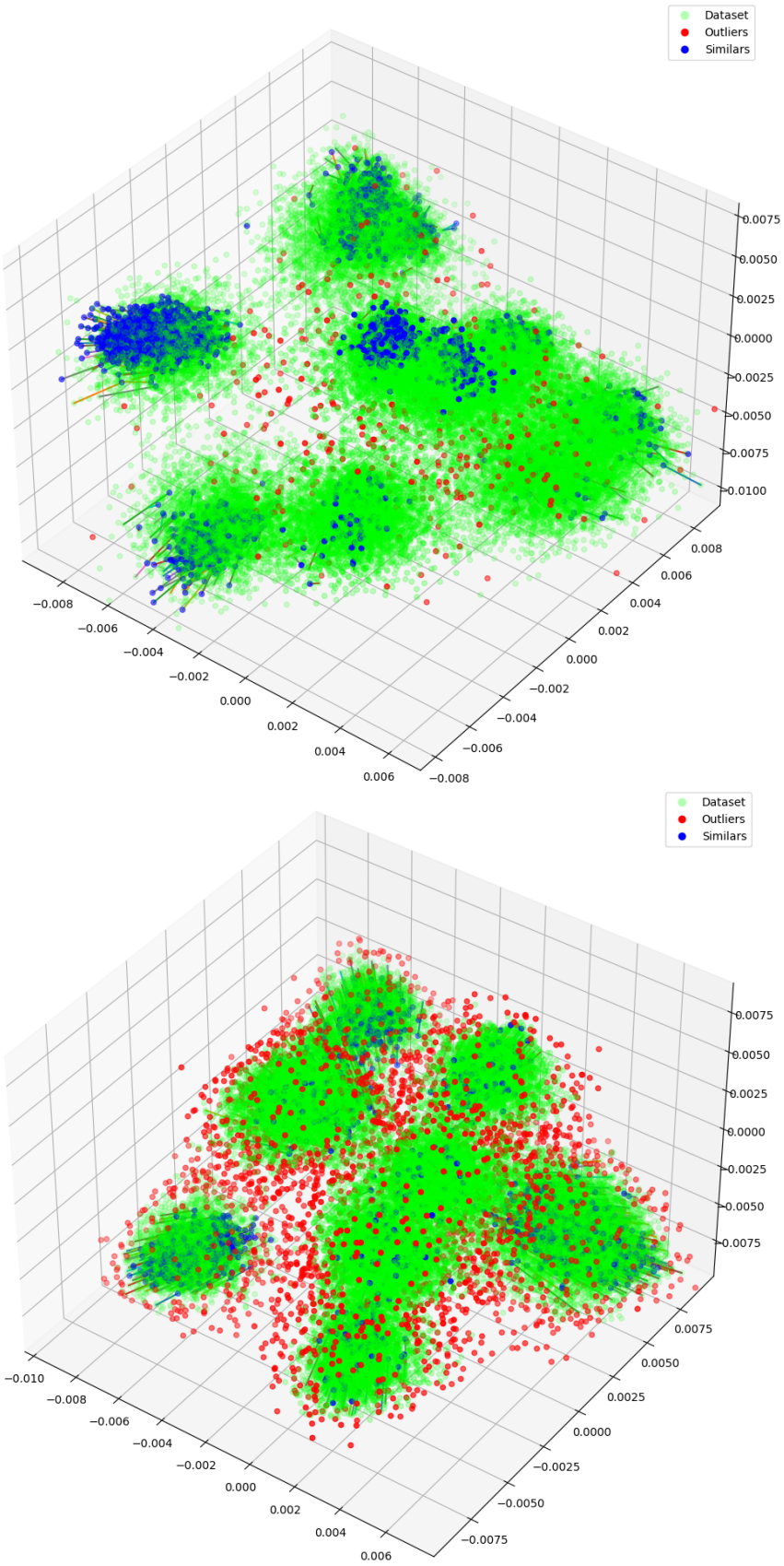

Figure 9. CIFAR-10 [19] dataset as a three dimensional point cloud filtered with $\%_{outlier} = 5\,\%$ and $\%_{similar} = 5\,\%$. Upper image $N_{dim} = N_{features}$. Lower image $N_{dim} = 3$. Downscaled using PCA.

4.4 Filtered dataset evaluation

To compare the quality of a found filtered dataset by the proposed method, the dataset is also filtered by other methods: FiftyOne [23] and random subsampling.

Random subsampling involves dropping random images from a dataset until the target dataset size is attained.

To filter a dataset using **FiftyOne**, image uniqueness computation was used to find similar images from a dataset. For each image, a uniqueness value was calculated, and the least unique images were dropped until the target dataset size is attained.

Neural network models are trained on a training dataset, separately filtered by each filtering method.

For image classification tasks, the test accuracy was measured. Higher test accuracy indicates a better trained model, thus a higher quality dataset. For object detecion tasks, a score metric (see Section 5.2) on a test dataset was measured. Higher model score indicates a higher quality dataset.

5 Analysis

This section evaluates and compares the method proposed on multiple models and datasets. Analysis of the results are also provided with visual explanations.

5.1 Image classification

5.1.1 CIFAR-10 & CIFAR-100

For CIFAR-10 and CIFAR-100 [19] datasets, a Resnet32 model [35, 36] was used. SGD was used with momentum coefficient of 0.9, weight decay of 0.0001, and the learning rate was cosine annealed from 0.1 to 0 over 100 000 steps with a linear warmup for the first 2 500 steps. The training was done for 100 000 steps with a batch size of 128, and the best model based on test accuracy was chosen to avoid overfitting. For filtering, the layer to extract features from was the output of the average pooling layer. An image that shows the location of average pooling layer in Resnet32 model is seen on Figure 10. The feature maps extracted contained 64 features, so dimensionality reduction was not necessary. The training process was identical to [21] but the results are not directly comparable because in the experiments done in this work, the average test accuracies were consistently higher.

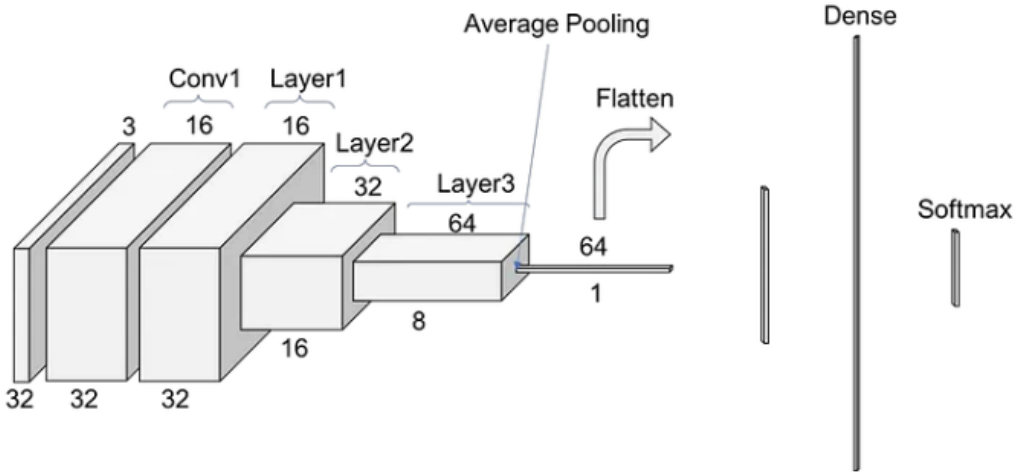

Figure 10. Resnet32 model for CIFAR datasets. [37]

CIFAR-10

CIFAR-10 dataset consists of 60 000 RGB images with dimensions of 32 pixels × 32 pixels and ten classes. 10 000 images are used for testing and 50 000 for training. During training, each image was normalized. To increase the variability of the dataset, during each epoch,

the training images were sometimes horizontally flipped or cropped from a random side by four pixels.

Table 1 compares the average test accuracy measured over five trials in multiple dataset sizes and subset finding methods. Using randomly chosen subsets of the dataset (random subsampling) shows that the drop in test accuracy is linear. Using the FiftyOne uniqueness method, mostly greater results are found over random subsampling.

With the proposed method, filtering only outliers from the dataset gives greater results when $\%_{outlier}$ is not over 10 %. This indicates that there are not many outliers in the dataset. It should also be noticed that by removing over 10 % of outliers, the model performs very poorly. This could be due to HDBSCAN detecting a whole part of the dataset as an outlier cluster so part of the dataset representation is completely lost as seen in Figure 11.

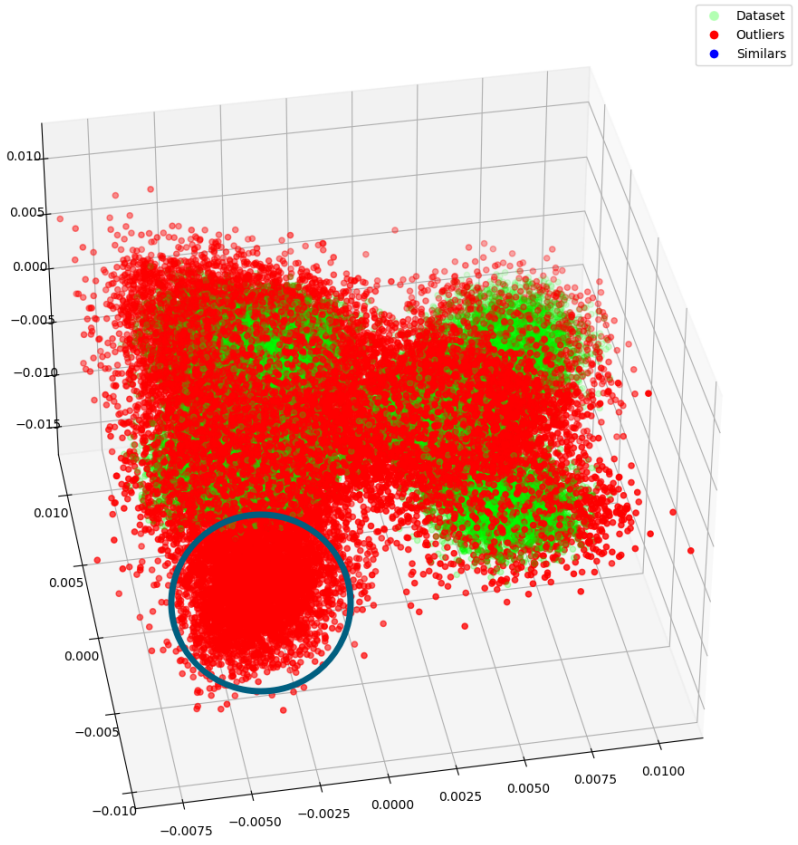

Figure 11. CIFAR-10 filtered with $\%_{outlier} = 50\,\%$. A part of the dataset circled with blue is detected as an outlier cluster.

Filtering semantically similar images from the dataset generally results in the best resulting model accuracies. When filtering both types of redundancies equally at $\%_{outlier} = 10\,\%$ and $\%_{similar} = 10\,\%$, the resulting model accuracy is still higher compared to using the whole dataset. Some redundant images can be seen in Appendix 3.

Table 1. Average test accuracy over 5 trials on the CIFAR-10 dataset.

Dataset percentage	**Random subsampling**	**FiftyOne**	**Semantically similar**	**Outliers**	**Half outliers, half semantically similar**
100 %	**93.12**	-	-	-	-
95 %	93.09	92.93	**93.25**	93.05	93.08
90 %	92.76	**93.20**	**93.20**	93.17	93.13
80 %	92.67	92.92	93.00	89.30	**93.19**
50 %	90.75	91.06	**91.97**	59.43	86.75

Figure 12 illustrates the resulting model accuracy comparison when filtering the dataset with Random subsampling, FiftyOne uniqueness method and the proposed method.

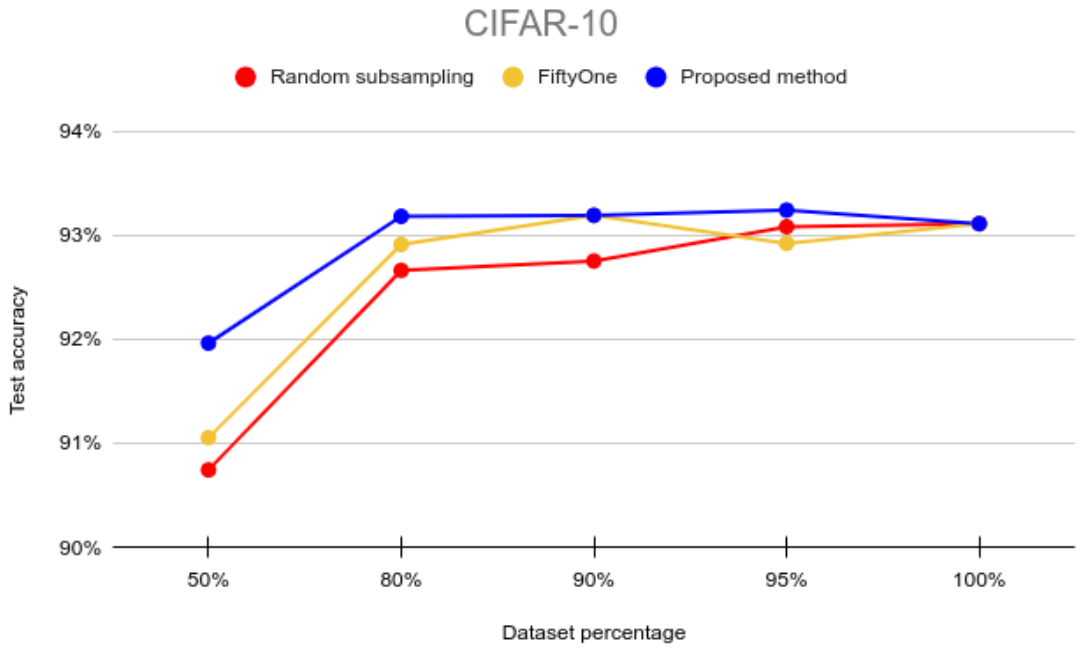

Figure 12. Comparison of different dataset filtering methods on the CIFAR-10 dataset.

CIFAR-100

CIFAR-100 dataset has the same shape and size as the CIFAR-10 dataset but it has images of 100 classes. The image preprocessing used was also identical.

Table 2 represents the average test accuracy measured over five trials with multiple dataset sizes and subset finding methods on the CIFAR-100 dataset. As seen in the table, redundancies in the dataset were mostly found on similar images. Removing outliers impacted the results negatively.

It should be noted that random subsampling outperformed both filtering methods using only 50 % of the dataset. This could be due to removing too many images of a single class which is less likely to occur with random subsampling.

Table 2. Average test accuracy over 5 trials on the CIFAR-100 dataset.

Dataset percentage	**Random subsampling**	**FiftyOne**	**Semantically similar**	**Outliers**	**Half outliers, half semantically similar**
100 %	**68.86**	-	-	-	-
95 %	68.74	68.74	**69.21**	68.24	68.68
90 %	68.28	68.08	**68.79**	67.69	68.59
80 %	67.16	66.88	**67.56**	65.95	67.51
50 %	**62.13**	61.11	59.18	58.41	61.58

Figure 13 compares the resulting model accuracy when filtering the dataset with random subsampling, FiftyOne uniqueness method and the proposed method.

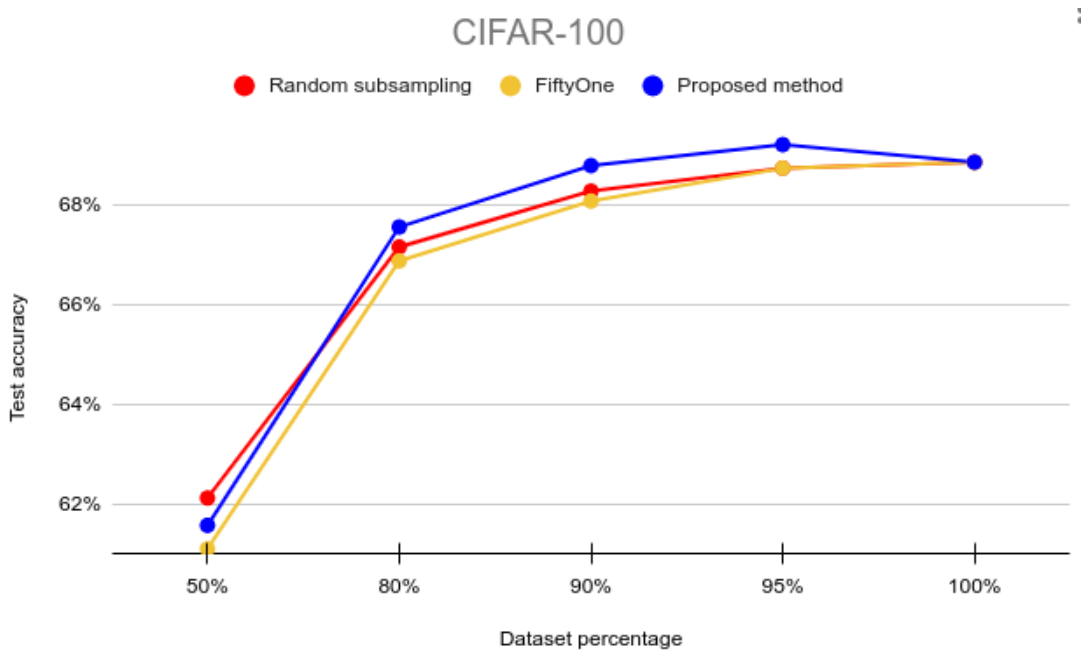

Figure 13. Comparison of different dataset filtering methods on the CIFAR-10 dataset.

5.1.2 Animal classification

Due to the worse results on the CIFAR-100 dataset when finding a smaller ($\leq$ 50 %) subset of a dataset, an animal classification dataset was tested. Similarly to [38], only 3 classes (Cheetah, Hyena and Tiger) were used from the Cheetah, Hyena, Jaguar and Tiger [39]

dataset. The resulting dataset consists of 2 700 training and 300 testing RGB images with dimensions 400 pixels × 400 pixels.

For the neural network, a pretrained Resnet18 model [36, 40] was used. Similarly to Resnet32, the last avgpool layer was used for extracting feature maps. For training the model, SGD optimizer was used with a learning rate of 0.001. The model was trained until no test accuracy improvement was found during the last five epochs. For each following result, the model was trained 15 times and the median test accuracy was measured.

The model achieved a maximum accuracy of 99.33 % with the full dataset. Figure 14 summarizes the results of finding smaller subsets of the dataset. It is seen that the proposed and FiftyOne filtering methods are significantly worse compared to random subsampling.

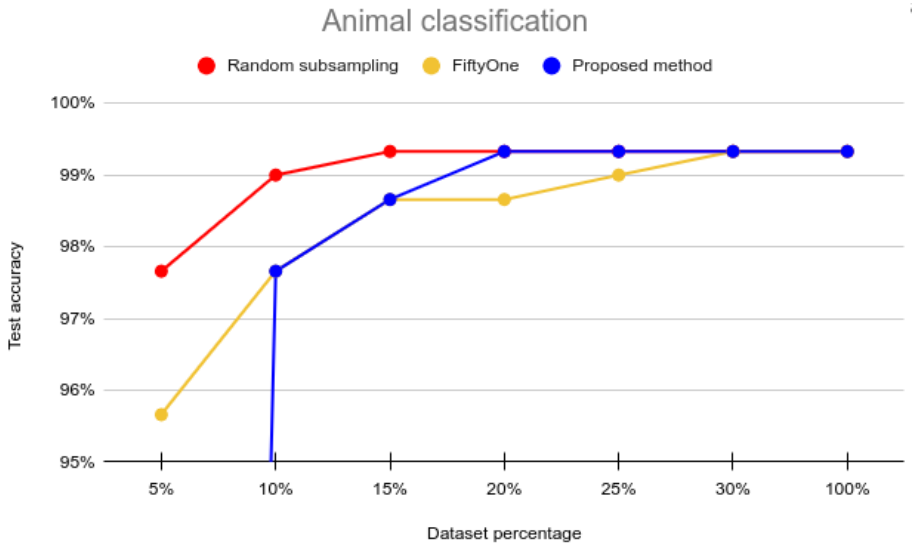

Figure 14. Comparison of different dataset filtering methods on animal classification task.

This behaviour is similar to the 50 % result with the CIFAR-100 dataset (Table 2). Not retaining the ratios of the image classes causes these results. For example, the animal classification dataset has 900 images of each class and if one of the classes is sparsely represented in the filtered dataset, the resulting trained model can not distinguish this class from the others well enough.

A solution to this would require the dataset to be annotated in order for each class to be represented more equally in the filtered dataset. Another solution is to first annotate the dataset as accurately as possible with the model itself. Then it is possible to filter using these approximated annotations. However, using ground truth values for input data filtering is outside of the scope of this thesis.

5.2 Object detection

For object detection tasks, YOLOv5 [41] model was used. During feature map extraction for the proposed method, the output of the detect layer in the head of YOLOv5 model is chosen. Similarly to section 5.1, the datasets are filtered with various parameters: all similars, all outliers and half of each.

For training YOLOv5 models, SGD optimizer was used with a learning rate of 0.01. All images were scaled to 640 × 640 resolution. After each training, the best performing model was picked by the Mean Average Precision (mAP) metric. A combination of mAP50 and mAP50-95 scores weighted 0.1 and 0.9 were used respectively and the total score of the model is calculated as shown in equation 5.1.

$$score = 0.1 \cdot score_{mAP50} + 0.9 \cdot score_{mAP50-95} \tag{5.1}$$

5.2.1 Cityscapes

A part of the Cityscapes dataset [42] is used with 2 500 images for the train and 475 images for the test dataset. The YOLOv5 model was trained for 120 epochs. For each result, the average score was measured over 3 trials. As the extracted feature map dimensionality $N_{features} = 327\,600$ was large, the feature maps were reduced to $N_{dim} = 2\,500$ using PCA or UMAP.

The results are shown in Figure 15. It is apparent that PCA mostly outperformed UMAP. It should be noted that by filtering the dataset with the proposed method, the model's results are consistently higher compared to random subsampling, especially when $\geq 80\,\%$ of the dataset is retained.

Figure 16 shows resized similar images detected in cityscapes dataset. The left image was the center-most of its cluster during agglomerative clustering so it is kept in the dataset. The others (on the right) are discarded due to being too similar to it. Additional redundant images can be seen in Appendix 4.

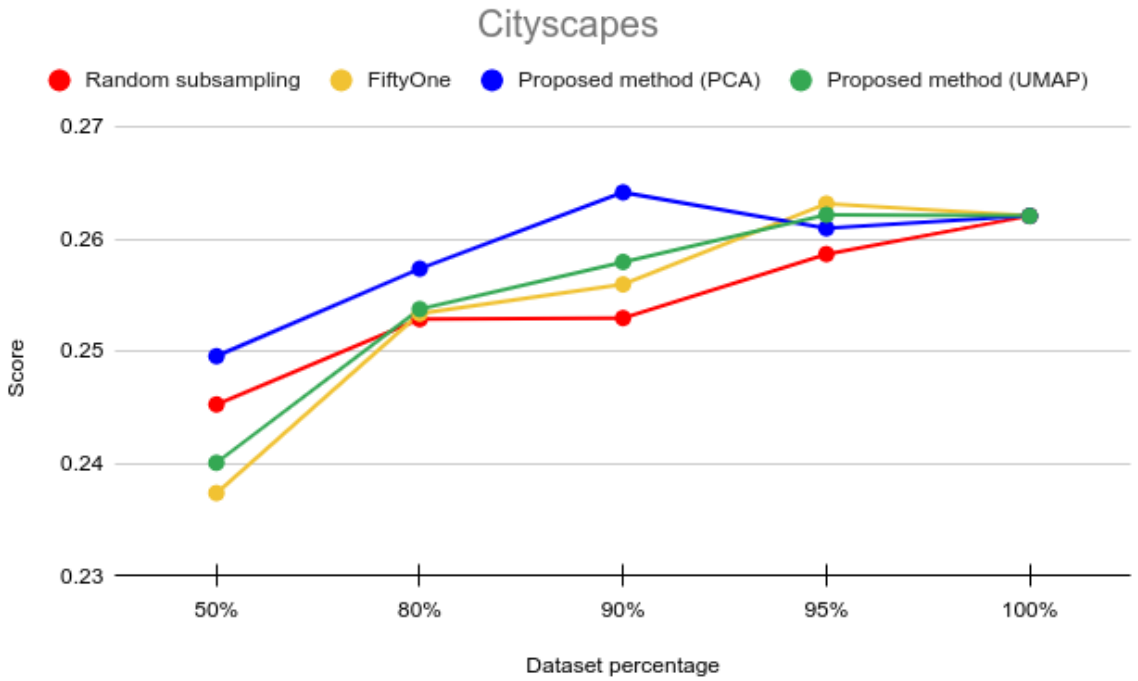

Figure 15. Comparison of model scores trained with different dataset filtering methods on the cityscapes dataset.

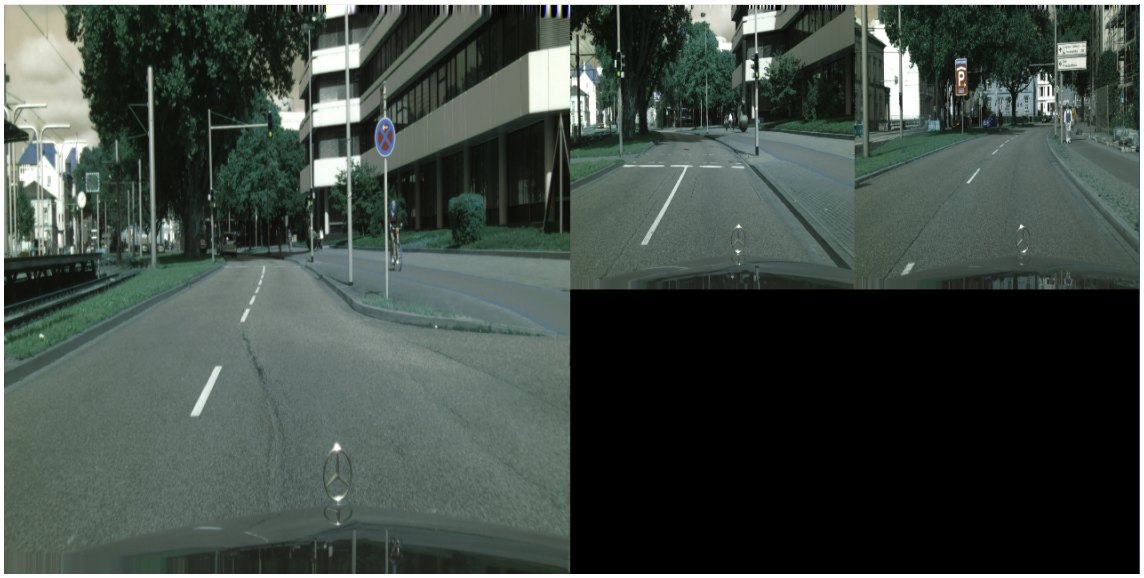

Figure 16. Semantically similar images in Cityscapes dataset.

5.2.2 Traffic signs

The traffic signs dataset [24] contains 630 training images and 111 test images with 4 different classifications for traffic signs. The extracted feature map dimensionality $N_{features} = 226\,800$ was large, they were reduced to $N_{dim} = 630$ using PCA or UMAP. For each result, the YOLOv5 model was trained five times for 300 epochs and the average result was calculated.

Results for the traffic signs dataset are shown in Figure 17. By removing 2.5 % of similars and 2.5 % of outliers, the model indicates significant improvement in measured score using UMAP dimensionality reduction.

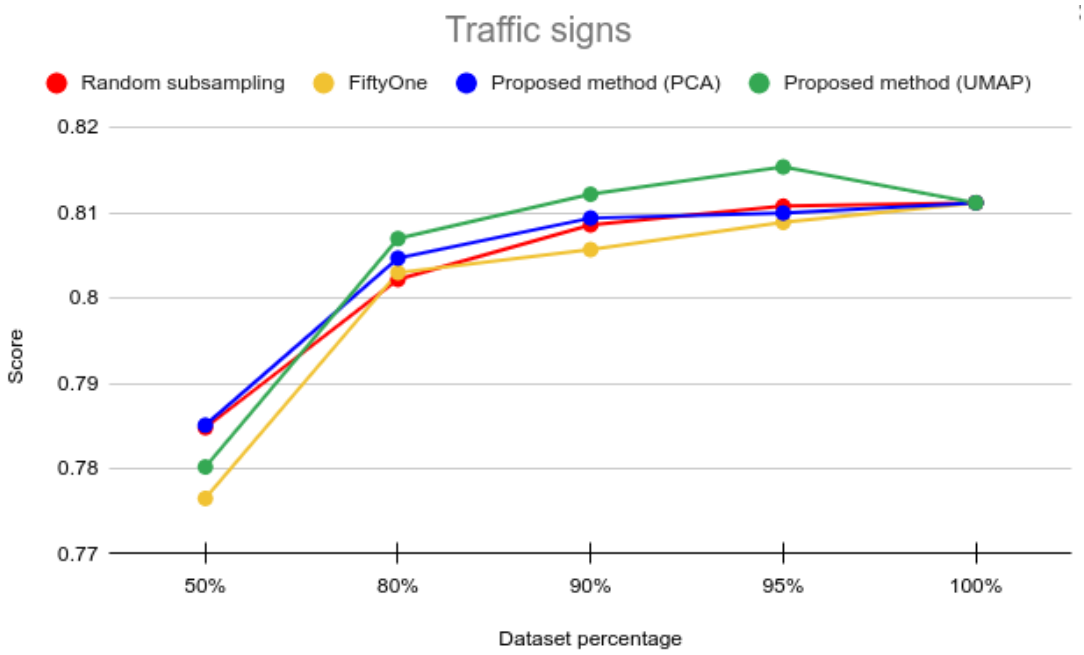

Figure 17. Comparison of model scores trained with different dataset filtering methods on the traffic signs dataset.

Some of the detected outliers are shown in Figure 18. The left-most image is blurry, so it may stand out. The center image has too bright lighting compared to the rest of the dataset. Right image has the traffic sign barely visible due to the darkness. Additional redundant images can be seen in Appendix 5.

Figure 18. Outliers in traffic signs dataset.

6 Summary

Filtering an image dataset for training a neural network is useful when the quality of the dataset contents is not thoroughly checked. The purpose of this work was to remove redundancies in datasets that minimally reduce the performance of a neural network model trained on it. To find these redundancies, an algorithm with an application was proposed.

The proposed algorithm creates a semantic representation of a dataset from the perspective of the trainable neural network. The representation can be used to analyse the dataset visually and computationally. Multiple clustering and high-dimensional data algorithms with GPU acceleration were used to effectively detect outliers and semantically too similar images in the dataset that potentially reduce the quality of the dataset.

Multiple datasets and models were tested and it was observed that upon removing certain amounts of the detected redundancies, the model's accuracy does not decrease. In contrast, when a random subset of a dataset is found, the drop in model accuracy was in correlation with the decreasing subset sizes. The proposed method was also compared with FiftyOne [23] image uniqueness filtering method and it was found that in most cases, the proposed method significantly outperformed it.

The proposed solution could be improved in multiple ways. Firstly, the user must specify multiple hyperparameters about how much of the dataset is filtered. Finding these parameters is a time consuming task because for accurate results, it requires the model to be trained with each generated subset and the resulting models must be compared. Secondly, the algorithm can be improved by including generated annotations in the filtering process to avoid removing too much data of a single type. This is detailed and demonstrated in Section 5.1.2 where the proposed method failed to find a sufficient small subset of a dataset.

As the proposed method is as generalized as possible: it works with any deep neural network and with any (including unannotated) dataset, the results should also be verified in other machine learning applications such as natural language processing or numerical analysis.

References

[1] Y Beesetty et al. *Neural Network Market Statistics: 2030*. [Accessed: 22-05-2023]. URL: https://www.alliedmarketresearch.com/neural-network-market.

[2] Larry Hardesty. *Explained: Neural networks*. [Accessed: 22-05-2023]. URL: https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414.

[3] *Kaggle: Your Machine Learning and Data Science Community*. [Accessed: 19-04-2023]. URL: https://www.kaggle.com.

[4] Andreas Lindholm, Luca Caltagirone, and Olof Wahlström. *Why more training data cannot make up for poor annotations*. [Accessed: 19-04-2023]. URL: https://www.kognic.com/articles/why-more-training-data-cannot-make-up-for-poor-annotations/.

[5] Jordan Carlson. *Will AI Replace the Humans In the Loop?* [Accessed: 19-04-2023]. URL: https://blog.cloudfactory.com/will-ai-replace-humans-in-loop.

[6] SelectStar. *Creating the Best Quality Image Dataset*. [Accessed: 19-04-2023]. URL: https://selectstar-ai.medium.com/creating-the-best-quality-image-dataset-720f612944ed.

[7] Prince Canuma. *Understanding Neural Networks*. [Accessed: 19-04-2023]. URL: https://prince-canuma.medium.com/understanding-neural-networks-22b29755abd9.

[8] Pragati Baheti. *Activation Functions in Neural Networks [12 Types & Use Cases]*. [Accessed: 19-04-2023]. URL: https://www.v7labs.com/blog/neural-networks-activation-functions.

[9] Chris Kevin. *Feature Maps*. [Accessed: 19-04-2023]. URL: https://medium.com/@chriskevin_80184/feature-maps-ee8e11a71f9e.

[10] Christophe Pere. *What are Loss Functions?* [Accessed: 19-04-2023]. URL: https://towardsdatascience.com/what-is-loss-function-1e2605aeb904.

[11] *Loss Functions*. [Accessed: 19-04-2023]. URL: https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html.

[12] Musstafa. *Optimizers in Deep Learning*. [Accessed: 19-04-2023]. URL: https://medium.com/mlearning-ai/optimizers-in-deep-learning-7bf81fed78a0.

[13] Simeon Kostadinov. *Understanding Backpropagation Algorithm*. [Accessed: 19-04-2023]. URL: https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd.

[14] Anup Bhande. *What is underfitting and overfitting in machine learning and how to deal with it*. [Accessed: 19-04-2023]. URL: https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76.

[15] Shachi Shah. *Do we really need GPU for Deep Learning? - CPU vs GPU*. [Accessed: 19-04-2023]. URL: https://medium.com/@shachishah.ce/do-we-really-need-gpu-for-deep-learning-47042c02efe2.

[16] *How to Build Your GPU Cluster*. [Accessed: 19-04-2023]. URL: https://www.run.ai/guides/multi-gpu/gpu-clusters.

[17] Oluwandabira Alawode et al. *Taltech AI-LAB documentation*. [Accessed: 19-04-2023]. URL: https://gitlab.cs.ttu.ee/ai-lab/doc.

[18] Cognilytica. "Data Engineering, Preparation, and Labeling for AI". In: (2019).

[19] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images". In: (2009).

[20] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[21] Vighnesh Birodkar, Hossein Mobahi, and Samy Bengio. "Semantic Redundancies in Image-Classification Datasets: The 10% You Don't Need". In: *arXiv preprint arXiv:1901.11409* (2019).

[22] Anas. *How to filter redundant data*. [Accessed: 19-04-2023]. URL: https://www.lightly.ai/post/how-redundant-is-your-dataset.

[23] *FiftyOne 0.18.0 documentation — voxel51.com*. https://voxel51.com/docs/fiftyone/. [Accessed 19-04-2023].

[24] Valentyn Sichkar. *Traffic Signs Dataset in YOLO format*. [Accessed: 19-04-2023]. URL: https://www.kaggle.com/datasets/valentynsichkar/traffic-signs-dataset-in-yolo-format.

[25] Victor Powell and Lewis Lehe. *Principal Component Analysis Explained Visually*. [Accessed: 19-04-2023]. URL: https://setosa.io/ev/principal-component-analysis/.

[26] Leland McInnes, John Healy, and James Melville. “Umap: Uniform manifold approximation and projection for dimension reduction”. In: *arXiv preprint arXiv:1802.03426* (2018).

[27] Andy Coenen and Adam Pearce. *Understanding UMAP*. [Accessed: 21-05-2023]. URL: https://pair-code.github.io/understanding-umap/.

[28] Nick Becker and Corey Nolet. *Faster HDBSCAN Soft Clustering with RAPIDS cuML*. [Accessed: 19-04-2023]. URL: https://developer.nvidia.com/blog/faster-hdbscan-soft-clustering-with-rapids-cuml/.

[29] Sebastian Raschka, Joshua Patterson, and Corey Nolet. *Machine Learning in Python: Main developments and technology trends in data science, machine learning, and artificial intelligence*. 2020. arXiv: 2002.04803 [cs.LG].

[30] Nvidia. *Cuml 23.02.00 API reference*. [Accessed: 19-04-2023]. URL: https://docs.rapids.ai/api/cuml/stable/api.html.

[31] Leland McInnes, John Healy, and Steve Astels. *How HDBSCAN Works*. [Accessed: 19-04-2023]. URL: https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html.

[32] Mario Köppen. “The curse of dimensionality”. In: *5th online world conference on soft computing in industrial applications (WSC5)*. Vol. 1. 2000, pp. 4–8.

[33] *tkinter — Python interface to Tcl/Tk*. [Accessed: 21-05-2023]. URL: https://docs.python.org/3/library/tkinter.html.

[34] Karl Raud. *Dataset filterer*. [Accessed: 21-05-2023]. URL: https://github.com/Kiili/DataFilterer.

[35] Yerlan Idelbayev. *Proper ResNet Implementation for CIFAR10/CIFAR100 in PyTorch*. https://github.com/akamaster/pytorch_resnet_cifar10. [Accessed: 19-04-2023].

[36] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[37] Pablo Ruiz. *ResNets for CIFAR-10*. [Accessed: 19-04-2023]. URL: https://towardsdatascience.com/resnets-for-cifar-10-e63e900524e0.

[38] Shubhankar Nandakumar. *Animal Classification using PyTorch and Convolutional Neural Networks*. [Accessed: 03-03-2023]. URL: https://towardsdatascience.com/animal-classification-using-pytorch-and-convolutional-neural-networks-78f2c97ca160.

[39] TODO. *Cheetah, Hyena, Jaguar and Tiger*. [Accessed: 21-05-2023]. URL: https://www.kaggle.com/datasets/iluvchicken/cheetah-jaguar-and-tiger.

[40] *Deep residual networks pre-trained on ImageNet*. [Accessed: 21-05-2023]. URL: https://pytorch.org/hub/pytorch_vision_resnet/.

[41] Glenn Jocher et al. *ultralytics/yolov5*. [Accessed: 19-04-2023]. URL: https://github.com/ultralytics/yolov5.

[42] Marius 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.

Appendix 1 – Non-Exclusive License for Reproduction and Publication of a Graduation Thesis[1]

I Karl Raud

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis "Dataset Filtering for Image Classification and Object Detection Models", supervised by René Pihlak
   1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
   1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.
3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

22.05.2023

[1]The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

Appendix 2 - Filterable interface

```
class FilterableInterface:

    def get_model(self) -> torch.nn.Module:
        """
        :return: pytorch model
        """
        raise NotImplementedError

    def get_filterable_dataset(self) -> Iterable:
        """
        batch: 1 batch of input to model.
        Batch contents must be with constant shape

        args: any metadata for each element in :batch:

        :return: Iterable. Each item like (batch, args)
        """
        raise NotImplementedError

    def get_image(self, arg) -> np.ndarray:
        """
        optional for viewing images in the application

        If :arg: is used as image file path, it can be implemented:
        PIL:    np.array(PIL.Image.open(arg))
        OpenCV: cv2.imread(arg)[:,:,::-1]  # BGR to RGB

        :return: channels-last RGB image from the
        filterable dataset specified by its :arg:
        """
        raise NotImplementedError
```

Appendix 3 – Found redundancies in CIFAR-10 dataset

Redundant similar cars

Redundant similar airplanes

Redundant similar ships

Redundant outliers

Appendix 4 – Found redundancies in cityscapes dataset

Redundant similarities

Redundant similarities

Redundant similarities

Redundant similarities

Redundant similarities

Redundant outliers

Appendix 5 – Found redundancies in traffic lights dataset

Redundant similarities

Redundant similarities

Redundant similarities

Redundant similarities

Redundant outliers