BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Macek Jan** Personal ID number: **466126**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Computer Science**

Study program: **Software Engineering and Technology**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Dimensionality Reduction Methods for the Functional Map of the World Dataset

Bachelor's thesis title in Czech:

Metody redukce dimenze pro dataset funkční mapy světa

Guidelines:

The aim is to design and implement Deep Neural Network (DNN)[1, 2] based solution for the Functional Map of the World satellite imagery dataset (fMoW)[3] and compare it with the state-of-the-art results achieved as part of the original challenge organized by IARPA. Due to complex nature of the fMoW dataset understanding the data is essential and therefore dimensionality reduction method UMAP[4] should be implemented and used to explore and process the fMoW dataset. Comparison with related state-of-the-art in deep learning methods is integral part of the work. Instructions are as follows:
1. Explore the current state-of-the-art solutions of dimension reduction and deep learning algorithms, e.g. [1, 2].
2. Explore the fMoW[3] dataset using the UMAP[4].
3. Design and implement a new solution to the fMoW dataset in Python using TensorFlow [5] framework.
4. Evaluate the solution and compare it to the state-of-the-art methods.

Bibliography / sources:

[1] Goodfellow, Ian, et al. „Deep Learning", MIT Press, 2016
[2] He, Kaiming, et al. "Mask R-CNN" Proceedings of the IEEE international conference on computer vision. 2017.
[3] Christie, Gordon, et al. "Functional map of the world." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018. https://github.com/fMoW/dataset, https://www.iarpa.gov/challenges/fmow.html
[4] McInnes, Leland, John Healy, and James Melville. "UMAP: Uniform manifold approximation and projection for dimension reduction." arXiv preprint arXiv:1802.03426 (2018).
[5] Abadi, Martın, et al. "TensorFlow: Large-scale machine learning on heterogeneous systems, 2015." Software available from tensorflow.org.

Name and workplace of bachelor's thesis supervisor:

Ing. Michal Reinštein, Ph.D., Vision for Robotics and Autonomous Systems, FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **12.08.2019** Deadline for bachelor thesis submission: **14.08.2020**

Assignment valid until: **19.02.2021**

Ing. Michal Reinštein, Ph.D. Supervisor's signature	Head of department's signature	prof. Mgr. Petr Páta, Ph.D. Dean's signature

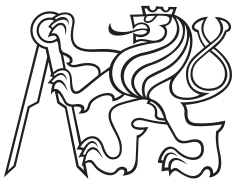
III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

. _____
Date of assignment receipt

_____
Student's signature

Bachelor Thesis



Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Computer Science

Dimensionality Reduction Methods for the Functional Map of the World Dataset

Jan Macek

Supervisor: Ing. Michal Reinštein, Ph.D
Field of study: Software Engineering and Technology
August 2020

Acknowledgements

I would like to thank the supervisor of this thesis, Ing. Michal Reinštein, Ph.D, for his support, understanding, and kindness. I am aware, that my decision and communications during writing this work weren't the best, but yet every time after our meeting I left with a sense of meaning in this work, which made me motivated and fulfilled.

Declaration

I hereby declare that I worked out the presented thesis independently and I quoted all the sources used in this thesis in accord with Methodical instructions about ethical principles for writing academic thesis.

Prague, August 20, 2020

Signature: .....

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 20. srpna 2020

Podpis: .....

Abstract

Dimensionality reduction algorithms are great for a deeper understanding of the dataset. In recent years we saw grow of feature selection algorithms more specifically subclass of neighbor graphs. In this work, we focused on the state of the art algorithm UMAP and applicated for the state of the art fMoW dataset. We then compare UMAP results, with older competing method t-SNE. We look at there strengths and weaknesses of both methods and possible difficulties in the application on the complex fMoW dataset. Based on these results, we implement and trained EfficientNet neural network on the fMoW dataset.

Keywords: Dimensionality Reduction, t-SNE, UMAP, fMoW, Convolutional Neural Network, Machine Learning, Python, Keras, TensorFlow, EfficientNet

Supervisor: Ing. Michal Reinštein, Ph.D
Karlovo náměstí 13,
121 35 Prague 2,
Czech Republic
room E225b

Abstrakt

Algoritmy redukce dimenze jsou skvělé pro hlubší pochopení datového souboru. V posledních letech jsme viděli růst algoritmů pro výběr prvků, konkrétně podtřídy sousedních grafů. V této práci jsme se zaměřili na nejmodernější algoritmus UMAP a aplikovali jsme na nejmodernější datový soubor fMoW. Poté porovnáme výsledky UMAP se starší konkurenční metodou t-SNE. Podíváme se na silné a slabé stránky obou metod a možné obtíže při aplikaci na komplexní datový soubor fMoW. Na základě těchto výsledků implementujeme a školíme neuronovou síť EfficientNet na datovém souboru fMoW.

Klíčová slova: Redukce Dimenze, t-SNE, UMAP, fMoW, Konvoluční Neuronová Sít, Strojové Učení, Python, Keras, TensorFlow, EfficientNet

Contents

Figures

Tables

Chapter 1

Introduction

This section focuses on basic introduction of this work.

This work is focused on experimenting with dimensionality reduction algorithms and neural networks. It is intended to find new approaches and ways to differentiate complex structures in the fMoW dataset and use these new approaches to design a neural network to this dataset.

1.1 Motivation

Main motivation for experimenting with new dimensionality reduction algorithm is deeper understanding of datasets. Which is further used in machine learning for training neural networks.

With increasing usages of neural networks in numerous scientific fields, we can observe increasing needs for BigData. In raw form, these datasets can be used in an unsupervised way, or a more efficient approach is adding labels to data, which can be later used in supervised learning. Supervised learning can have better accuracy, with fewer samples of data and computing time needed for training.

Dimensionality reduction algorithms used in this work can be divided into two main categories: Matrix Factorization and Neighbour Graphs. This work is mainly focused on Neighbour Graphs, more precisely on the difference between t-SNE[30] and UMAP[33] , which are two most of recent algorithms in this field.

Chapter 2
Related Work

This section focuses on related work in the areas of Dimensionality reduction methods, Image datasets, and Deep Neural Networks classification.

2.1 Dimensionality reduction methods

Most famous method from Matrix Factorization is principal component analysis (PCA)[47] from year 1987 which is widely used and is relatively computationally cheap. For instance as analysis method for association between bioactive compounds and functional properties in foods[18] or for analysis EEG data[4].

Newer method from Neighbour Graphs is T-distributed Stochastic Neighbour Embedding (t-SNE)[30] from year 2008. t-SNE is more computationally demanding but is also more accurate in grouping data into classes. Uses can be improved visualization of single-cell RNA-seq data[28] published in Nature, application to human genetic data[27] or identification of prognostic tumor subpopulations[2]. Multicore implementation of t-SNE exist on GitHub[45].

Newest method from Neighbour Graphs is (UMAP)[33] from year 2018. UMAP is more versatile than t-SNE and is computationally faster in comparison, UMAP is also more accurate in representing spaces between data groups. Uses can be visualizing single-cell data[6] published in Nature and to visualize physical and genetic interactions[16].

2.2 Image datasets

MNIST dataset[26] from year 1998 is most famous image dataset of hand-written digits and it is equivalent of "hello world" in image recognition. Dataset was later extended as EMNIST dataset[11] from year 2017 which includes hand-written digits and letters. In both datasets images are black and white.

Fashion-MNIST dataset[48] from 2017 is newer famous version composed of black and white images of clothing and shoes. Dataset is used in training of convolutional neural networks[40].

CIFAR dataset[24] from year 2009 is harder than MNIST-like datasets. Images are in RGB spectrum and each class can have subclass in more detailed version of dataset. Dataset is used in training of convolutional neural networks[43]. CIFAR can be seen as simpler version of ImageNet dataset[15].

fMoW dataset[10] from year 2017 is newest dataset composed of high-definition satelite images. Dataset was used in IARPA challenge[21], which took place in 2017. Good example as how hard is fMoW dataset, is HYDRA Ensemble of Convolutional Neural Networks[34] from third winner solution.

2.3 Deep neural networks classification

2.3.1 IARPA challenge

Here is closer look at each 3 best participant fMoW IARPA challenge[21] from year 2017.

Third winner solution is from usf-bulls[39] and it is HYDRA or Ensemble of Convolutional Neural Networks[34] which is composed of multiple instances of ResNet[19] and DenseNet[20] convolutional network architectures.

Second winner solution is from jianminsun[22] and it is using MXNet Framework[8] and ensemble of models as ResNet[19] and ResNeXt[49].

First winner solution is from pfr[37] and it is using ensemble of 12 deep convolutional network classifiers tuned from generic Dual Path Networks[9].

2.3.2 EfficientNet

EfficientNet[41] from year 2019 is a new approach to convolutional neural architecture, with a scaling method of depth, width and resolution dimensions in the convolutional neural network.

Repository implementing EfficientNet in Keras exist on GitHub[50].

Chapter 3

Theory

This section focuses on theoretical background in the areas of Dimensionality reduction and Artificial Intelligence.

3.1 Dimensonality reduction

Dimensionality reduction is process of reducing dimension in multi dimensional set of data. This process can be further divided into two categories: Feature selection and Feature projection. Main usages can be found in statistics, machine learning, clustering and anomaly detection.

3.1.1 Feature Selection

Feature selection is process of selecting specific set of features. This set is then used to distinguished data points between themselves in set of data. Main approaches for feature selection are filler, wrapper and embedded methods, which can be used for classification[13], regression and data analysis.

3.1.2 Feature Projection

Feature projection or extraction is process of generating new features from old set of features. Main two approaches for feature projection are Matrix Factorization and Neighbour Graphs.

3.2 Matrix Factorization

Matrix factorization is group of feature projection algorithms working with matrix operations in multi dimensional vector space. Most famous methods are PCA[47], Sparse PCA[14], Linear Auto-encoder, Latent Dirichent Allocation, Non-negative Matrix Factorization, Generalised Low Rank Models, Word2Vec[17], GLoVe[36] and Probalistic PCA[25].

The goal of Matrix Factorization is generally to find two matrices which product is approximately given matrix. Given matrix X is data and two searched matrices U and V are representation and archetypes.

$$X \approx UV$$

X is an N x D matrix
U is an N x d matrix
V is an n x D matrix

Adopted from [31]

In the equation below, the goal is to minimize loss between X and UV, with some constraints. This representation is fundamental to all Matrix Factorization techniques.

$$\sum_{i=1}^{N}\sum_{j=1}^{D} Loss(X_{ij}, (UV)_{ij})$$

Adopted from [31]

3.2.1 PCA

Principal Component Analysis[47] is method for reducing dimensionality of dataset, by reducing number of variables in the dataset and simultaneously preserving as much information as possible in linear projection. Process can be broken down into five steps:

1. Standardization of variables
2. Computation of covariance matrix
3. Computation of eigenvectors and eigenvalues of the covariance matrix
4. Construction of feature vector
5. Reorient the data by multiplying feature vector with transpose dataset

Equation

The queation goal is minimalization of square error loss.

$$\sum_{i=1}^{N}\sum_{j=1}^{D}(X_{ij} - (UV)_{ij})^2$$

Adopted from [31]

3.3 Neighbour Graphs

Neighbour Graphs is based on measuring distance between data points by defined metric and finding number of closest neighbours. Data points is then recreated in lower dimensional space with same neighbours relationships as in higher dimensional space before. Neighbour Graphs Most famous methods are Laplacian Eigenmaps, Spectral Embedding, Hessian Eigenmaps, Local Tangent Space Aligment, JSE, Isomap[5], t-SNE[30], Locally Linear Embedding, LargeVis[42], NerV[46] and UMAP[33].

3.3.1 t-SNE

T-distributed Stochastic Neighbor Embedding[30] is method for reducing dimensionality of dataset, by using local relationships between data points to create low-dimensional space with captured non-linear structure. Process can be broken down into two steps:

1. Creation of probability distribution which compute relationships between neighboring data points
2. Recreation of a low-dimensional space with probability distribution

Equation

G is set of all data points
$w(e)$ is weight of data point e in dimensional graph
h in $w_h(e)$ indicate high dimensional graph
ℓ in $w_\ell(e)$ indicate low dimensional graph

Adopted from [32]

The equation goal is minimalization of error between high and low dimensionality representation of graphs, which results as optimizing of clumping data points together.

$$\sum_{e \in G} w_h(e) \log \left(\frac{w_h(e)}{w_\ell(e)} \right)$$

Adopted from [32]

3.3.2 UMAP

Uniform Manifold Approximation and Projection for Dimension Reduction[33] is method for reducing dimensionality of dataset, by manifold approximation, fuzzy cover and union cross-entropy. Process can be broken down into six steps:

1. Finding nearest neighbours, creating simplices and approximating manifold
2. Defining a Reimannian metric on the manifold
3. Applicating fuzzy cover
4. Calculating fuzzy union
5. Optimizing embedding by fuzzy set cross-entropy and stochastic gradient descent
6. Constructing topological representation of the high dimensional data

Equation

G is set of all data points
$w(e)$ is weight of data point e in dimensional graph
h in $w_h(e)$ indicate high dimensional graph
ℓ in $w_\ell(e)$ indicate low dimensional graph

Adopted from [32]

The equation goal is minimalization of error between high and low dimensionality representation of graphs. Equation can be broken down to two parts. The first part is same as in t-SNE equation, which is responsible for optimizing of clumping data points together. The second part is responsible for optimizing of gaps between data points, which results in more meaning sparsing between data points in the graph.

$$\sum_{e \in G} w_h(e) \log\left(\frac{w_h(e)}{w_\ell(e)}\right) + (1 - w_h(e)) \log\left(\frac{1 - w_h(e)}{1 - w_\ell(e)}\right)$$

Adopted from [32]

Hyperparameters

UMAP has 4 main hyperparameters. It is number of neighbours around data point, minimal distance between data points, metrics of data and number of components which is dimensionality of output data.

Number of neighbours

Number of neighbours control balance between local and global structures in the data. Default value of neighbours is set to 15. Recommended range is generally between 2 and 200, but it can be set higher if needed.

Minimal distance

Minimal distance regulate how much neighbouring points can be packed together. Which can be useful for finer topological structures or preservation of global structure. Default value is set to 0.1 and recommended range of values is between 0.0 and 0.99.

Number of components

Number of components define dimensionality of output data. Which can be useful for 3D graphs or as intermediate step for another data analysis. Default state is set to 2 and recommended range of values is between 2 and 100.

Supervision

Supervised is non-default state in which algorithm use data and their labels for more precise dimension reduction. Supervision can be used for clearer separation of data, but it can be also biased by incorrect or too simplistic labels.

Data metrics

Metrics is used to compute distance in the ambient space of dataset. Default metric is set to euclidean, other prepared metrics are listed bellow:

Category	Metric
Minkowski style metrics	euclidean, manhattan, chebyshev, minkowski
Miscellaneous spatial metrics	canberra, braycurtis, haversine
Normalized spatial metrics	mahalanobis, wminkowski, seuclidean
Angular and correlation metrics	cosine, correlation
Metrics for binary data	hamming, jaccard, dice, russellrao, kulsinski, rogerstanimoto, sokalmichener, sokalsneath, yule

Table 3.1: UMAP data metrics

3.4 Artificial Intelligence

Artificial Intelligence is a field in Computer Science focused on finding algorithms to intelligence. Approaches can range from only mimicking inteligence to complex simulation of biological processes and evolution itself.

3.4.1 Machine Learning

Machine learning is a subfield of Artificial intelligence focused on self-learning approach. Algorithms can learn from specific datasets with predefined output in labels or iteratively interact with dynamic environment with reward feedback.

3.4.2 Deep Learning

Deep Learning is a subfield of Machine Learning specialized in Artificial Neural Networks.

3.5 Artificial Neural Networks

Artificial Neural Networks are computer recreation of biological Neural Networks. Simplest ANN have 3 types of layers (Input, Hidden and Output layer).

3.5.1 Deep Neural Networks

Deep Neural Networks are ANN with more hidden layers.

3.5.2 Convolutional Neural Networks

Convolutional Neural Networks are DNN with 2 repeating layers. Input data is usually images in 2D or 3D format.

Convolutional layers

Convolutional layers act as a filter which generate new dimension of data, that is called convolution features. Feature is calculated through iteration of filter. Filter is matrix which is multiplied with all data points on filter and sum of multiplication is a new point in feature. Each feature is calculated from different filter. Features can be understood as perspectives to highlight edges and other structures.

Pool layers

Pool layers is responsible for reducing size of input data. Data is reduced through use of mask and calculation of max or average value of all points on mask. Calculation of max value or Max Pooling is also noise suppressive and it usually performs better than Average Pooling.

Chapter 4

Datasets

This section focuses on sets of data used in dimensionality reduction algorithms (Chapter 3).

4.1 MNIST

The MNIST[26] dataset of handwritten digits consist of gray scale 28x28 images with pixel value range from 0 to 255. Dataset contains 60 000 training and 10 000 test examples and have label from 10 classes, which represent each different digit.

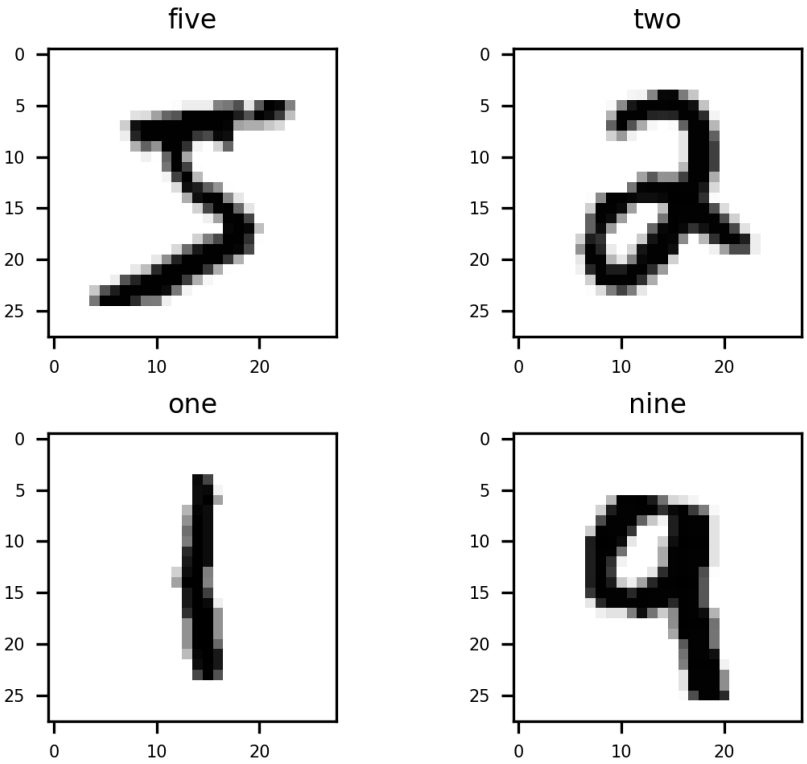


Figure 4.1: Examples of images in MNIST dataset

4.2 Fashion-MNIST

Fashion-MNIST[48] dataset consist of gray scale 28x28 images with pixel value range from 0 to 255. Dataset contains 60 000 training and 10 000 test examples and have label from 10 classes, which represent each different fashion pieces:

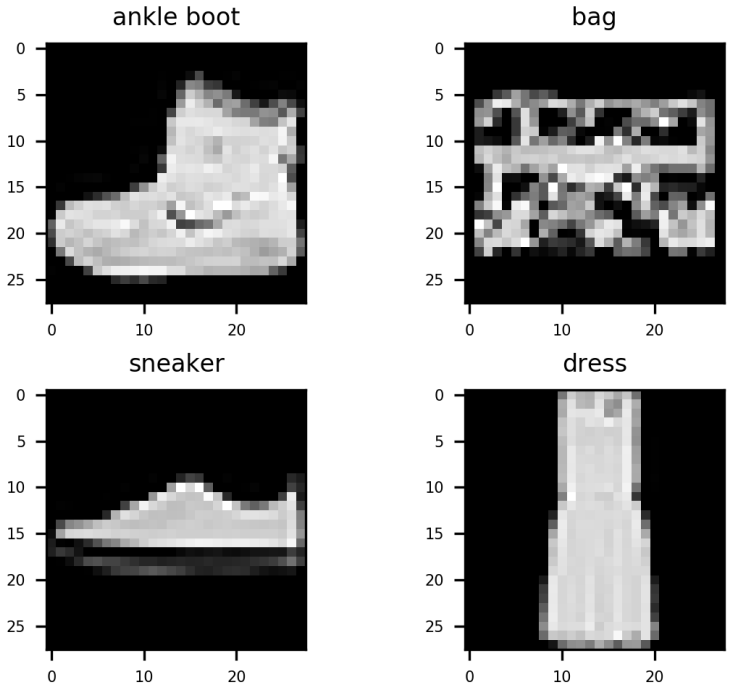


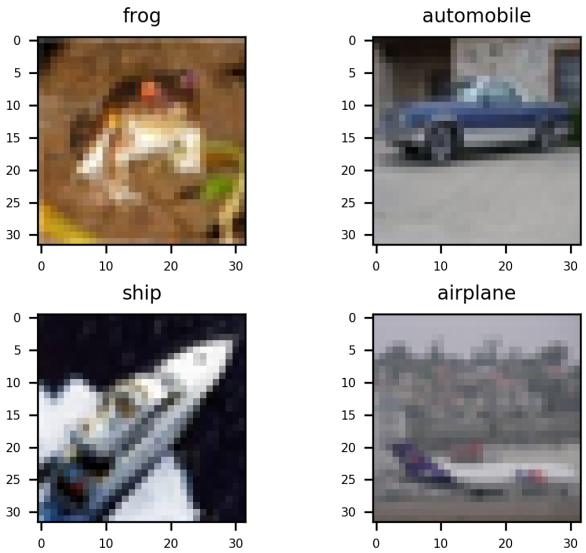
Figure 4.2: Examples of images in Fashion-MNIST dataset

N.	Classes
1.	t-shirt/top
2.	trouser/pants
3.	pullover shirt
4.	dress
5.	coat
6.	sandal
7.	shirt
8.	sneaker
9.	bag
10.	ankle boot

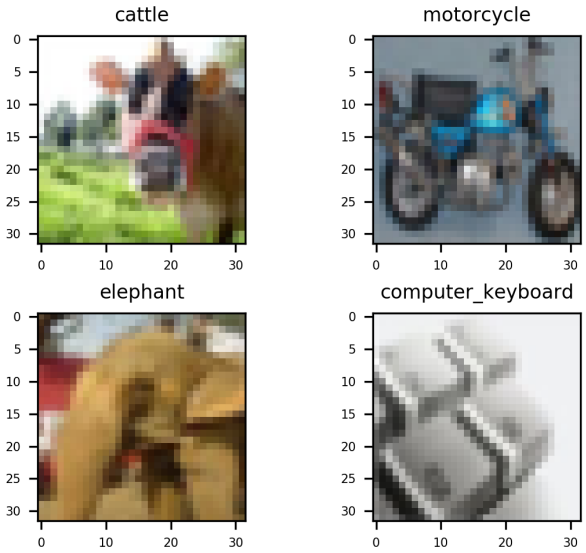
Table 4.1: Fashion-MNIST classes

4.3 CIFAR

CIFAR[24] dataset consist of RGB 32x32 images with pixel value range from 0 to 255 for each RGB layer. Dataset contains 50 000 training and 10 000 test examples in two versions.



(a) : CIFAR-10



(b) : CIFAR-100

Figure 4.3: Examples of images from two versions of CIFAR dataset

4.3.1 CIFAR-10

CIFAR-10 is simpler version of dataset, with 10 classes.

N.	Classes
1.	airplane
2.	automobile
3.	bird
4.	cat
5.	deer
6.	dog
7.	frog
8.	horse
9.	ship
10.	truck

Table 4.2: Cifar-10 classes

4.3.2 CIFAR-100

CIFAR-100 is more complex version of dataset have, with 100 classes grouped in 20 superclasses.

N.	Superclasses	Classes
1.	aquatic mammals	beaver, dolphin, otter, seal, whale
2.	fish	aquarium fish, flatfish, ray, shark, trout
3.	flowers	orchids, poppies, roses, sunflowers, tulips
4.	food containers	bottles, bowls, cans, cups, plates
5.	fruit and vegetables	apples, mushrooms, oranges, pears, sweet peppers
6.	household electrical devices	clock, computer keyboard, lamp, telephone, television
7.	household furniture	bed, chair, couch, table, wardrobe
8.	insects	bee, beetle, butterfly, caterpillar, cockroach
9.	large carnivores	bear, leopard, lion, tiger, wolf
10.	large man-made outdoor things	bridge, castle, house, road, skyscraper
11.	large natural outdoor scenes	cloud, forest, mountain, plain, sea
12.	large omnivores and herbivores	camel, cattle, chimpanzee, elephant, kangaroo
13.	medium-sized mammals	fox, porcupine, possum, raccoon, skunk
14.	non-insect invertebrates	crab, lobster, snail, spider, worm
15.	people	baby, boy, girl, man, woman
16.	reptiles	crocodile, dinosaur, lizard, snake, turtle
17.	small mammals	hamster, mouse, rabbit, shrew, squirrel
18.	trees	maple, oak, palm, pine, willow
19.	vehicles 1	bicycle, bus, motorcycle, pickup truck, train
20.	vehicles 2	lawn-mower, rocket, streetcar, tank, tractor

Table 4.3: Cifar-100 superclasses and classes

4.4 fMoW

Functional Map of the World[10] dataset consists of over 1 million satellite images in multiple sizes. Dataset is split into 4 sections: train, val, test and seq. Train and val sections are organized into folder structure by their classes. Each image is in pair with JSON file, which besides classes contains information about the location of the satellite image, positions of boxes containing objectives in images, code of the country of origin, image timestamp, the direction of scanning, cloud cover in image, and other extra data useful for more precise classification. Images are classified in 62 distinct classes and 1 class for false detection. Dataset is split into 4 sections: train, val, test and seq. Train and val sections are organized into folder structure by their classes. Test and seq sections are organized in numbered folders, and JSON files don't contain information about class of the image.

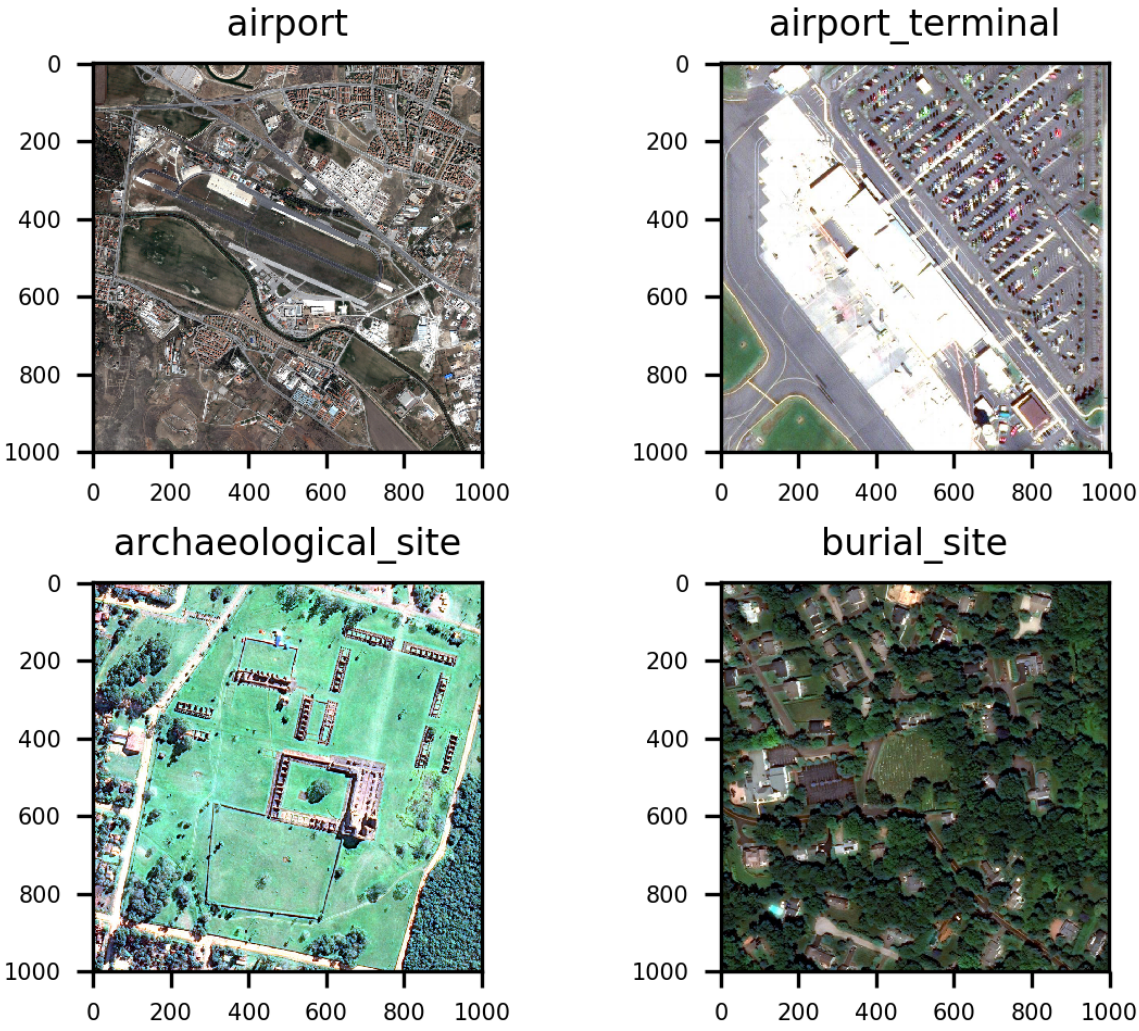


Figure 4.4: Examples of images in fMoW dataset with resolution 1000x1000

N.	Classes	N.	Classes
1.	airport	32.	office building
2.	airport hangar	33.	oil or gas facility
3.	airport terminal	34.	park
4.	amusement park	35.	parking lot or garage
5.	aquaculture	36.	place of worship
6.	archaeological site	37.	police station
7.	barn	38.	port
8.	border checkpoint	39.	prison
9.	burial site	40.	race track
10.	car dealership	41.	railway bridge
11.	construction site	42.	recreational facility
12.	crop field	43.	impoverished settlement
13.	dam	44.	road bridge
14.	debris or rubble	45.	runway
15.	educational institution	46.	shipyard
16.	electric substation	47.	shopping mall
17.	factory or powerplant	48.	single-unit residential
18.	fire station	49.	smokestack
19.	flooded road	50.	solar farm
20.	fountain	51.	space facility
21.	gas station	52.	stadium
22.	golf course	53.	storage tank
23.	ground transportation station	54.	surface mine
24.	helipad	55.	swimming pool
25.	hospital	56.	toll booth
26.	interchange	57.	tower
27.	lake or pond	58.	tunnel opening
28.	lighthouse	59.	waste disposal
29.	military facility	60.	water treatment facility
30.	multi-unit residential	61.	wind farm
31.	nuclear powerplant	62.	zoo

Table 4.4: fMoW classes

Chapter 5

Methods

This section focuses on specific settings for dimensionality reduction algorithms (Chapter 3).

5.1 Test computer HW

Test computer consist of:
- Intel 4 core CPU i5-4670, with 3.40GHz frequency
- 16 GB RAM
- Nvidia RTX 2070 GPU
- 2x 500 GB SSD and 1x 4 TB HDD
- OS Windows 10

5.2 Processing of datasets

MNIST and Fashion-MNIST are both imported through fetch_openml function from sklearn.datasets[12] python library.

CIFAR datasets are loaded from HDD, with a size of 340 MB for each of 2 variations. Loading from files is completed through load function from pickle[38] python library.

In both cases are data saved into the object of classes enveloping respective dataset. This is possible thanks to size small enough to be fully loaded into operational RAM memory.

5.2.1 fMoW

The processing of fMoW dataset is non-trivial due to of complex placement of files in the folder structure. Data was used only from files in /train and /val folders of fMoW dataset folder structure.

First step in processing is scan files in folder structure into a list, which is then saved into fmow_file_list.hdf5 file onto a specified location. Saving to the file is done with h5py python library[7].

The next step after obtaining a file list is to through a multi-threaded algorithm[44], is access, and transformation of each file. Each file have two versions .jpg and .json.

Because jpg image files are different in width and height pixel size, resized function from PIL.image python library[29] is used to uniform images size as INT8 datatype. The array of image pixels is then reshaped to a more suitable shape.

The string name of the class of the image is extracted from the JSON file with json python library[23], and formatted into the number corresponding position of a given class in an array of dataset classification classes as a label.

In the final step transformed images and labels are then saved into memmap array from NumPy python library[35], which is entirely saved on HDD and only access parts of the array are loaded into operational memory. This is done to limit the excessive use of memory when working with data, which in the case of fMoW dataset are in tens or hundreds of GB large. The largest transformed size of images was 1000x1000 pixels, which resulted in approximately 1,3 TB worth of data in memmap, generated this amount of data took 7 days in total.

5.3 Calculation of dimensional reduction algorithms

PCA, t-SNE experiments are processed only with default hyperparameters.

T-SNE algorithm is implemented in multicore variant, which makes a noticeable save of time.

PCA in combination with t-SNE experiments uses PCA with dimensionality parameter setup to number 50, t-SNE experiments are processed only with default hyperparameters.

UMAP experiments are processed with multiple hyperparameters settings, with a random state 42. The operational memory of the test computer is too small for the use of UMAP on fMoW dataset. The problem with memory is solved, by allocating one of 500 GB SSD as virtual operational memory. The virtual operation memory is able to extend the boundary of possible calculations to fMoW dataset, by sacrificing the shorter time of running graph calculations.

PCA in combination with UMAP experiments uses PCA with dimensionality parameter set up to number 50, UMAP experiments are processed with default hyperparameters and in supervision mode.

Chapter 6

Experiments evaluation

This section focuses on performed experiments with methods (Chapter 5) on datasets (Chapter 4).

6.1 PCA

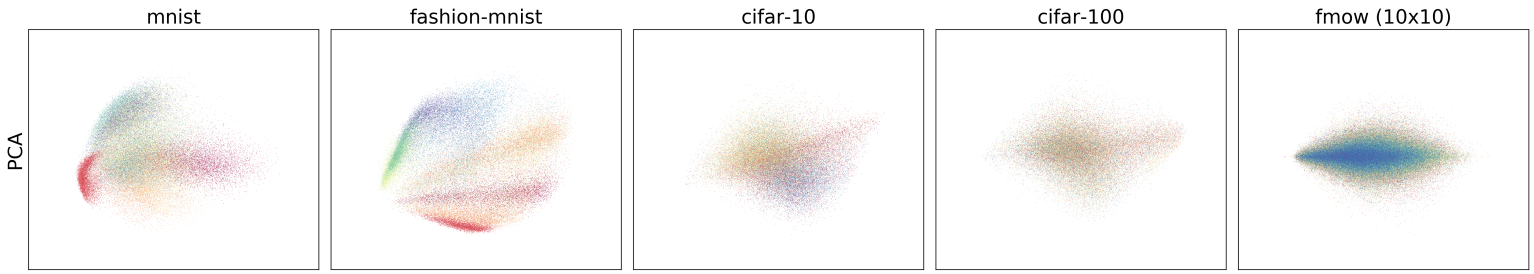


Figure 6.1: Comparison of PCA method

As can be seen in graph (Figure 6.1) the results with pca are quite fuzzy. On simpler datasets such as mnist and fashion-mnist classification in results can be fuzzy, but recognizable.

Thanks to this fuzziness it cannot be described as perfect. The results of more complex datasets begin to merge into one spherical point, where classes are overlapping each other and cannot be distinguished.

6.1.1 fMoW

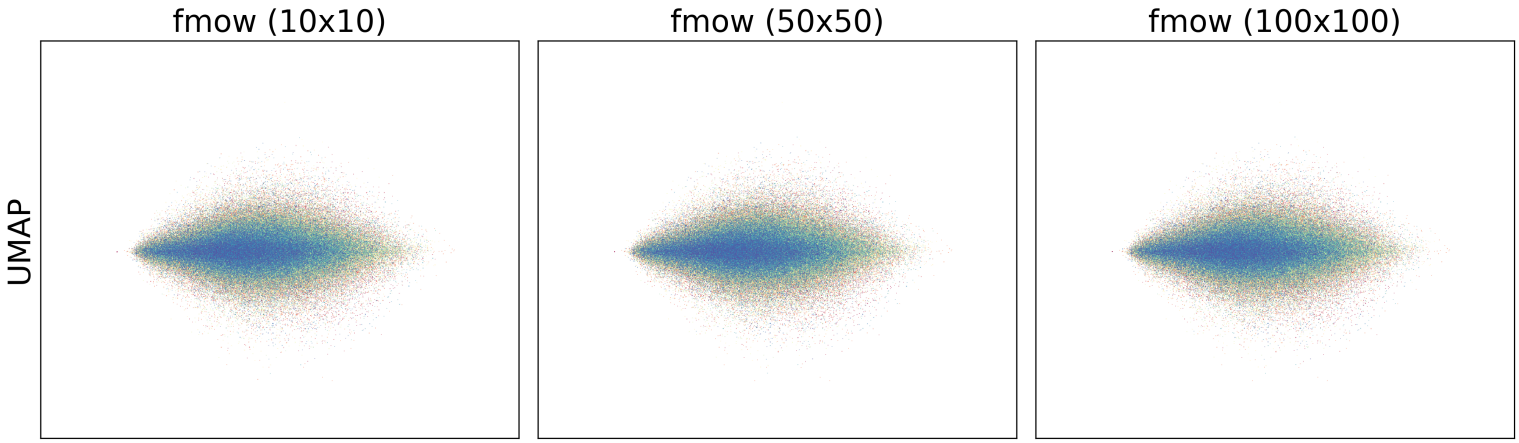


Figure 6.2: Comparison of PCA method on fMoW dataset

Application of PCA on fMoW dataset with scale down images shown no noticeable difference (Figure 6.2).

6.1.2 Performance

The PCA method is in terms of performance very fast and moderately memory intensive.

6.1.3 Conclusion

The PCA method alone seems only suitable for simpler datasets with easier classification.

The method cannot be recommended in case of complex datasets analysis or needs for clearer results.

6.2 t-SNE

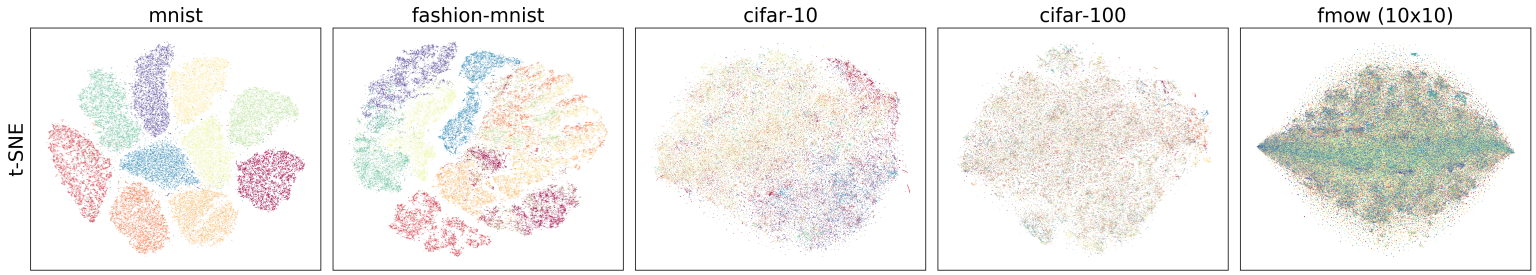


Figure 6.3: Comparison of t-SNE method

As can be seen in graph (Figure 6.3) t-SNE results are more distinctive. On simpler datasets such as mnist and fashion-mnist classification in results create distinctive groups, each group represents a specific class.

Spaces between groups are consistent, with no correlation to data. In more complex datasets results are rather scrambled and cannot be distinguished into groups. Spaces between groups are incoherent or missing altogether.

6.2.1 Performance

The t-SNE Method is in terms of performance and memory very intensive, which can be optimized with the multicore parallel version[45].

6.2.2 Conclusion

The t-SNE method is a reasonable test for distinctiveness of classes in dataset.

Result groups do not show a correlation between each other, therefore classes similarity cannot be analyzed.

6.3 UMAP

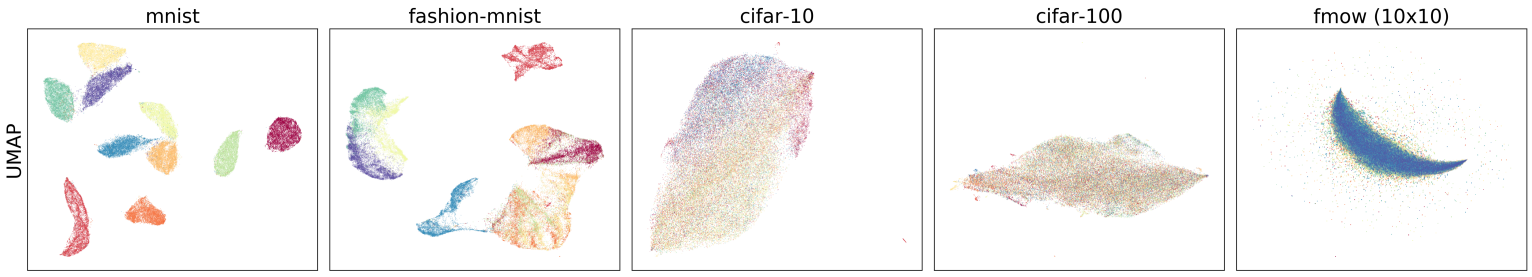


Figure 6.4: Comparison of UMAP method

As can be seen in graph (Figure 6.4) UMAP shows most promising results. On simpler datasets such as mnist and fashion-mnist, classes are in distinct groups with dynamic spacing. Spacing between groups visualize similarities between classes. In more complex datasets results are more mixed together in one group, with distinct subgroups inside.

6.3.1 Supervision

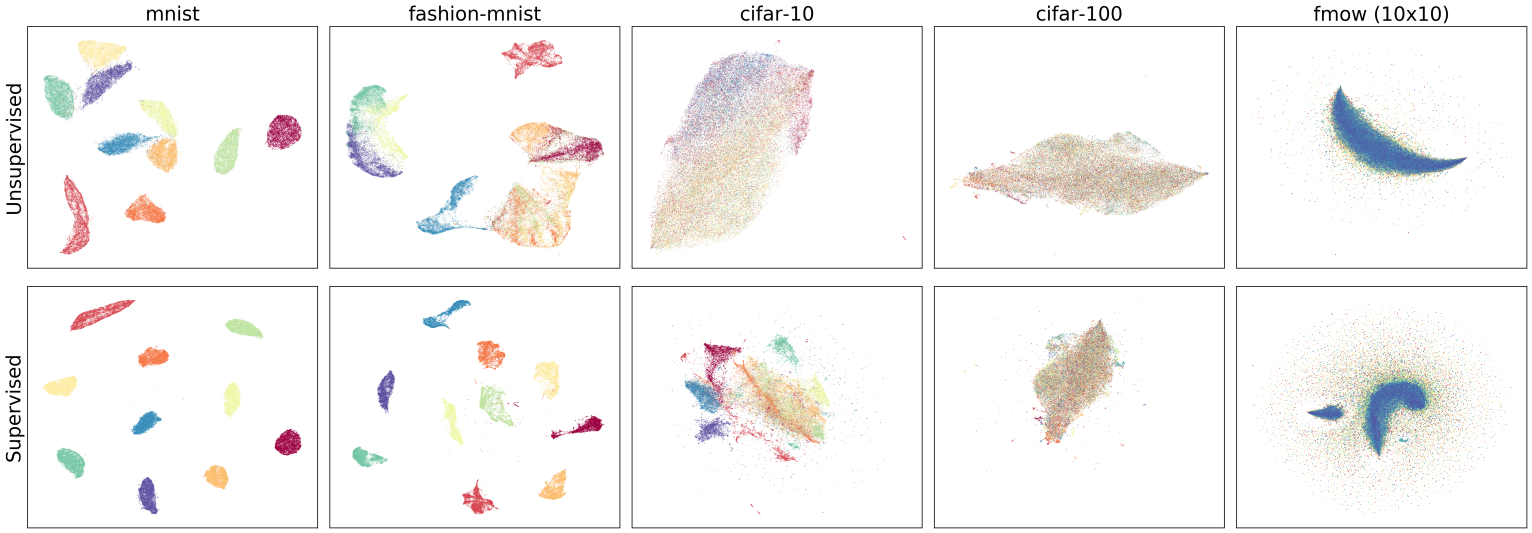


Figure 6.5: Comparison of UMAP method with supervision

A comparison in the graph (Figure 6.5) shows how results change with UMAP supervision. On simpler datasets such as mnist and fashion-mnist, classification with supervision end with clearly separated groups.

The cifar-10 dataset results under supervision are no longer in one group, but in separate groups, which overlap each other.

The cifar-100 dataset results on the other hand are still in one group and the fMoW dataset is broken into two groups with overlapping classes and a corona of points around them.

6.3.2 Neighbours and distance

The first set of experiments in the euclidean metric shows the relation between a number of neighbors on the y-axis and minimal distance between points on the x-axis. On simplest datasets like mnist (Figure *umap_mnist_plot_metric=euclidean.png* on CD B) and fashion-mnist (Figure *umap_fashion-mnist_plot_metric=euclidean.png* on CD B), change can be observed only in the minimal distance on the x-axis.

The second set of experiments shows the difference of the relation between a number of neighbors on the y-axis and minimal distance between points on the x-axis in euclidean and correlation metric. On the cifar-10 dataset in euclidean metric (Figure *umap_cifar-10_plot_metric=euclidean.png* on CD B) change can be observed in both axes.

The number of neighbors change how much are the graphs spread out and minimal distance dictates spacing between points in the graph. On the cifar-10 dataset in correlation metric (Figure *umap_cifar-10_plot_metric=correlation.png* on CD B) is group structure more separate, but the trend of change of axis parameters is the same.

The cifar-100 dataset results in euclidean (Figure *umap_cifar-100_plot_metric=euclidean.png* on CD B) and correlation (Figure *umap_cifar-100_plot_metric=correlation.png* on CD B) metrics are less clear, but the overall trend is the same as in Cifar-10.

6.3.3 Metrics

The metric experiment (Figure 6.6) shows how can be dataset (cifar-10 on this example) be differently interpreted with relation to different metrics.

With Minkovsky style metrics (euclidean, manhattan, chebyshev, minksovky), Miscellaneous spatial metrics (canberra, braycurtis) and Angular and correlation metrics (cosine, correlation) dataset graphs can be seen as classifiable.

In hamming metric data points are in horseshoe-like shape and classes are scrambled. Finally fice and yule metrics shows completely scrambled data in sphere shape.

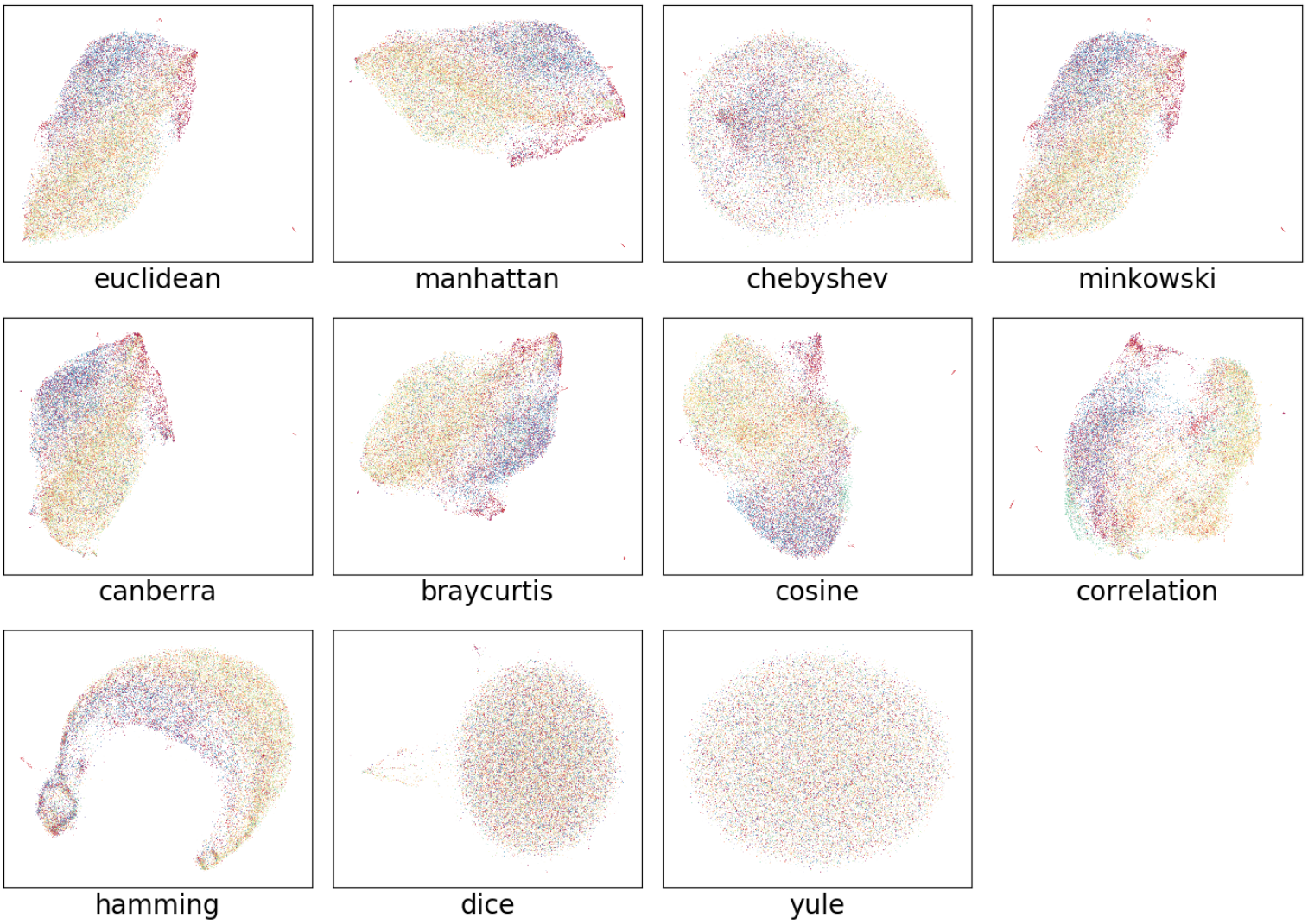


Figure 6.6: Comparison of UMAP method on CIFAR-10 with different metrics

6.3.4 fMoW

The experiment with the fMoW dataset (Figure 6.7) shows the increasing categorization of classes in dependency of image resolution in the dataset, which can collate with increasing pattern fidelity.

Cleaner categorization in higher dataset resolution can be seen as proof, that increasing resolution of the fMoW dataset will lead to better UMAP results. The fMoW resolution is limited to size of 75x75 pixels, due to the heavy memory requirements result (which are discussed in Performance section 6.3.5).

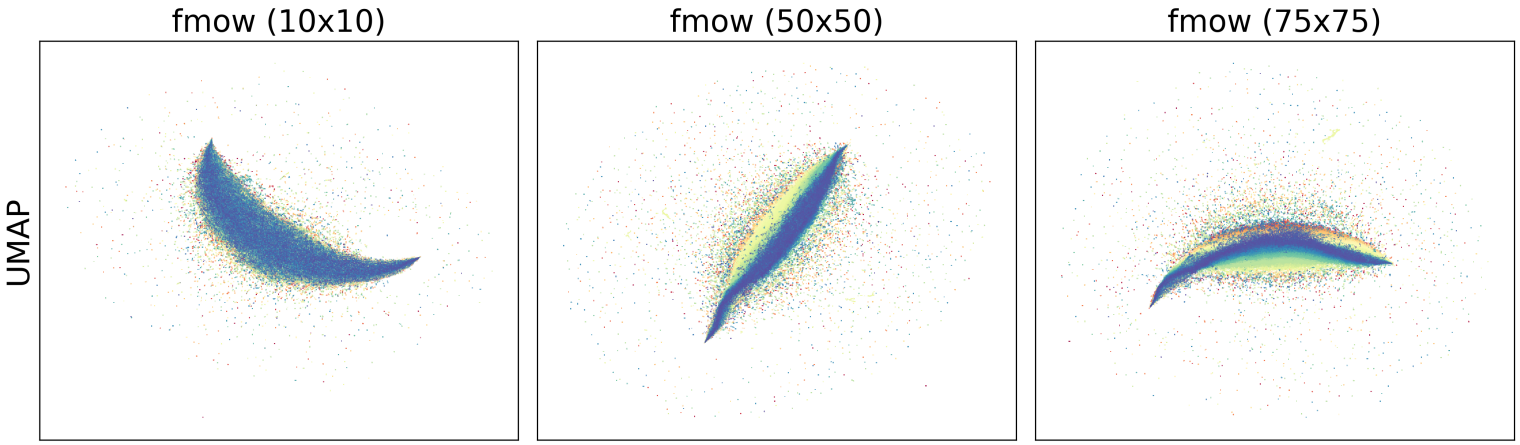


Figure 6.7: Comparison of UMAP method on fMoW dataset

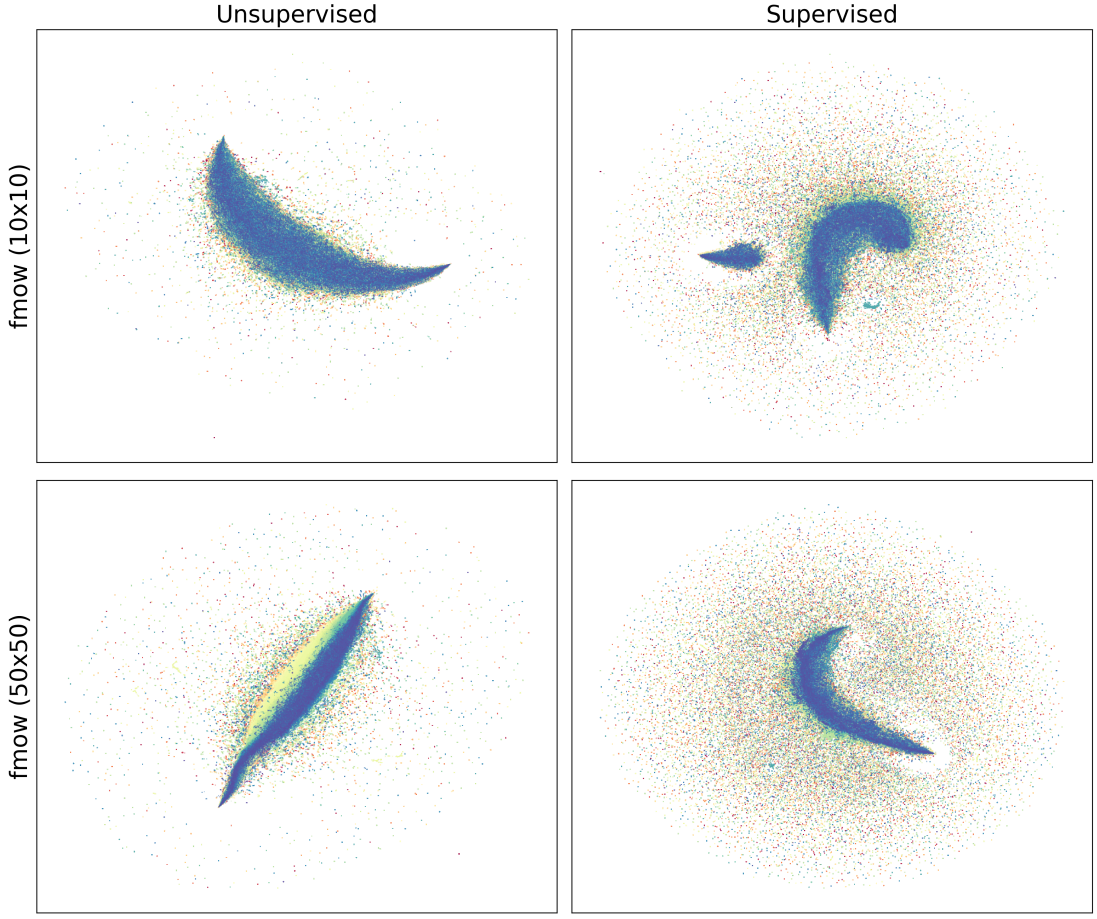


Figure 6.8: Comparison of UMAP method on fMoW dataset with supervision

Supervision

The experiment of the supervision (Figure 6.8) shows the failure of a supervised calculation approach of fMoW dataset, with small image resolution.

The graphs comparison of the unsupervised and supervised approach of the fMoW dataset images in 50x50 resolution shows the increased categorization without supervision, probable cause is due to too small image resolution, which does not include all detailed patterns in data.

6.3.5 Performance

The UMAP method is in terms of performance and memory very intensive. Poor performance is mainly caused by the inability to use multiple cores.

The high memory requirements can be solved by use the memory virtualization of operational memory on SSD or HDD (preferred option is SSD due to an order of magnitude lower response times).

Due to the exponentially large increase in memory requirements, related to the more comprehensive size of the datasets as fMoW, the full potential of the UMAP method could not be tested.

6.3.6 Conclusion

The UMAP is a very promising method for finding distinctiveness of known classes or discovering hidden unlisted classes in the dataset.

The results of more complex datasets clearly show the need for the search for untrivial UMAP parameter combinations.

6.4 PCA in combination with t-SNE

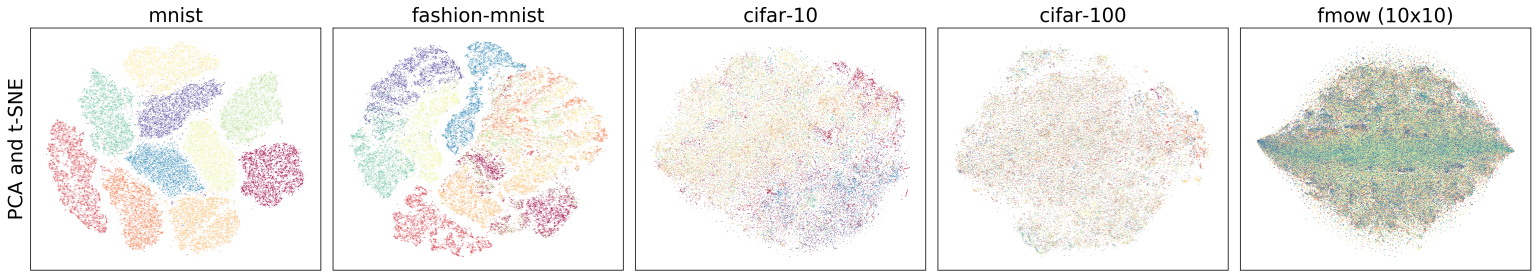


Figure 6.9: Comparison of PCA in combination with t-SNE method

As can be seen in graph (Figure 6.9) the PCA in combination with the t-SNE method produced essentially the same results as use of the t-SNE method alone.

This finding is very useful from a performance standpoint (which is discussed in Performance section 6.4.1).

6.4.1 Performance

The PCA in combination with the t-SNE method is in terms of performance and memory is essentially the t-SNE method with easier HW requirement thanks to a reduced number of dimensions on input.

6.4.2 Conclusion

The PCA in combination with the t-SNE method seems like a good solution for very big datasets thanks to the lower needed system requirement, but it still has some form of downside in the form of a more complicated parameter fine-tuning for ideal results.

6.5 PCA in combination with UMAP

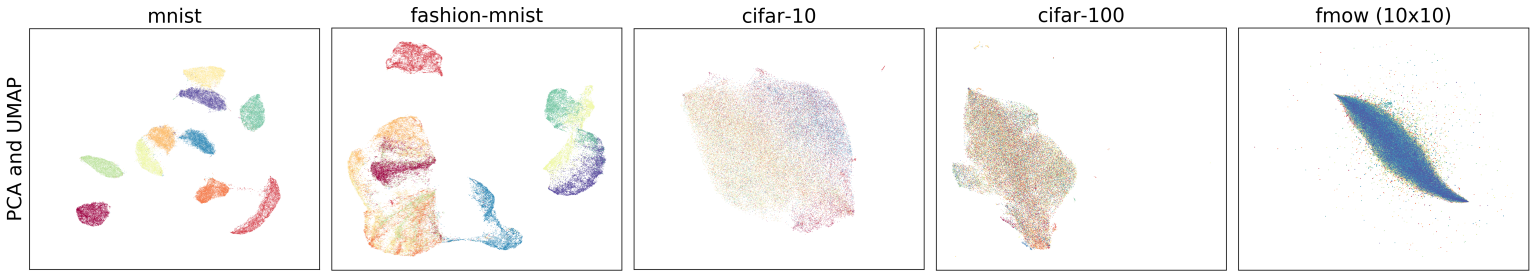


Figure 6.10: Comparison of PCA in combination with UMAP method

As can be seen in graph (Figure 6.10) the PCA in combination with the UMAP method produced very similar results as the UMAP method alone, with minimal differences.

6.5.1 Supervision

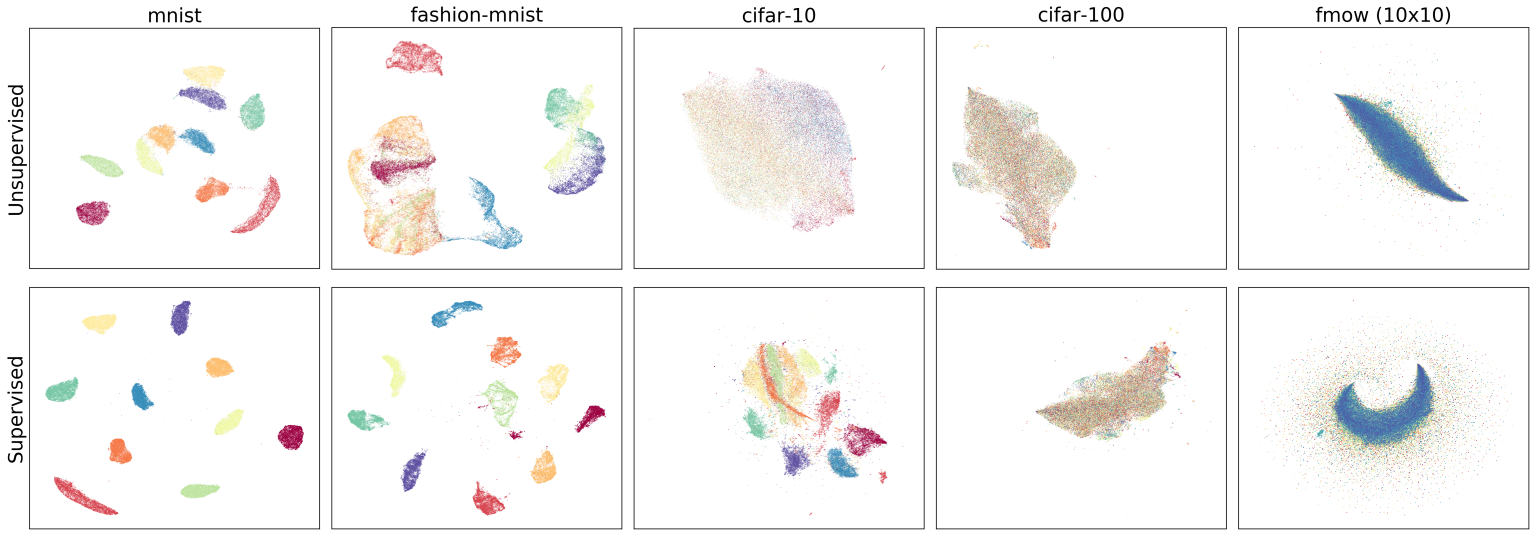


Figure 6.11: Comparison of PCA in combination with UMAP method with supervision

A comparison in the graph (Figure) shows very little difference between the supervised and unsupervised versions of PCA in combination with the UMAP method.

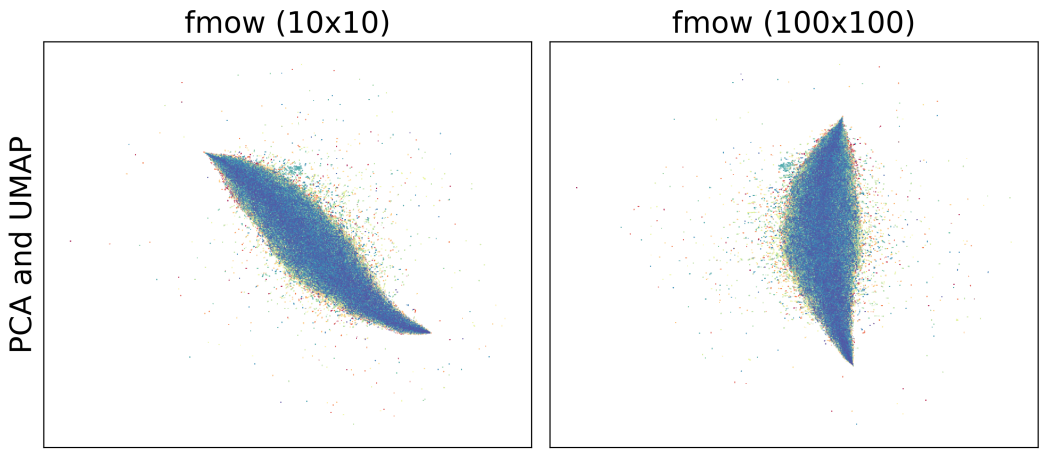


Figure 6.12: Comparison of PCA in combination with UMAP method on fMoW dataset

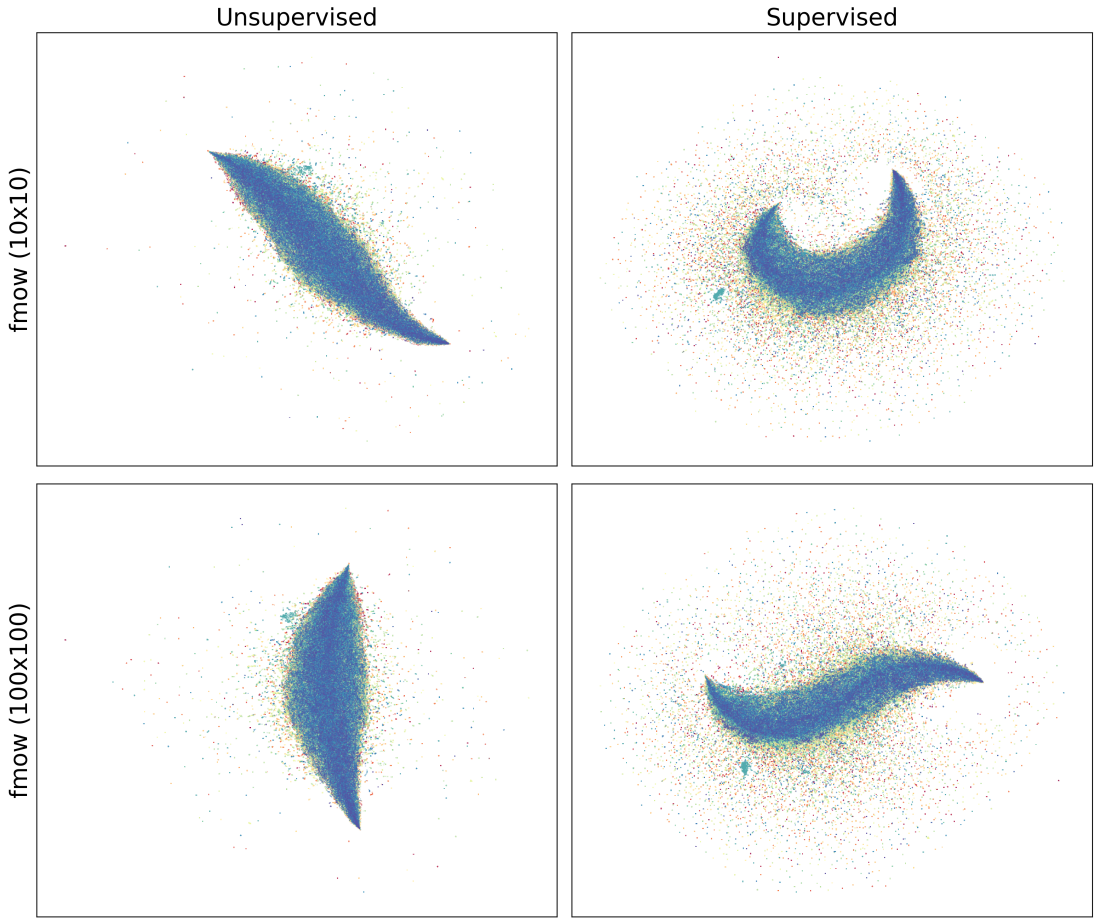


Figure 6.13: Comparison of PCA in combination with UMAP method on fMoW dataset with supervision

6.5.2 fMoW

The experiment with the fMoW dataset on the PCA in combination with the UMAP method (Figure 6.12) shows surprising failure compared to result with the UMAP method, which can indicate the loss of integral patterns in data through the use of PCA. More fine-tuning of PCA output size and parameters are needed for a more comprehensive answer.

Supervision

The experiment of the supervision with the fMoW dataset on the PCA in combination with the UMAP method (Figure 6.13) shows similar results as with the UMAP method.

6.5.3 Performance

The PCA in combination with the UMAP method is in terms of performance and memory is significantly faster, because of the reduced number of dimensions on the input of the UMAP method.

6.5.4 Conclusion

The PCA in combination with the UMAP method can be promising, but it depends if failure with the fMoW dataset can be solved. More experiments in that regard are needed, with wider scope of parameters.

Chapter 7

Analysis

This section focuses on the analysis of the appropriate neural network for the fMoW dataset, based on conducted experiments (Chapter 6).

7.1 Finding right CNN architecture

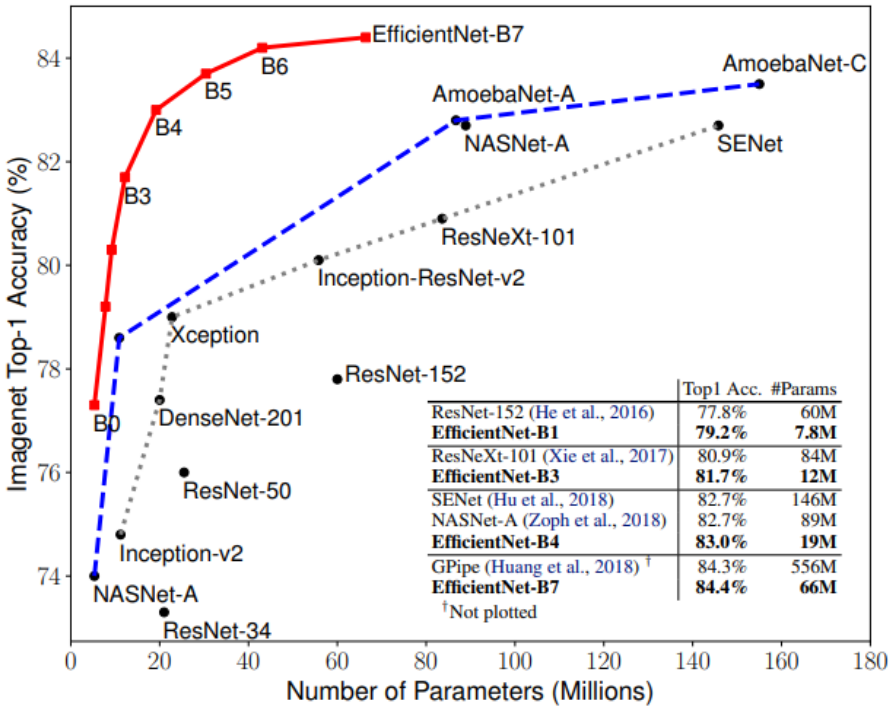


Figure 7.1: EfficientNet comparsion with different NN techniques on ImageNet[15] dataset, adopted from [41]

The experiments with the UMAP method conducted on the fMoW dataset, indicate the emerging classification in relation to increasing dataset resolution. This leads to the conclusion, that maximization of possible dataset resolution results in favor of using the deeper pattern in data.

Due to the limited computing power of test PC 5.1, a possible suitable architecture must have as few number of parameters as possible. A short search on the internet for CNN architectures shows EfficientNet state of art neural network from the year 2019 [41] as a very promising choice.

After a deeper look at the comparison with other competing CNN architectures on Imagenet[15] dataset, which coincidentally loosely matched with architectures (Figure 7.1) used in winning solutions of IARPA challenge. This makes use of EfficientNet even more exciting and interesting for this work.

7.2 Basic description of EfficientNet

EfficientNet is essentially a model with the CNN compound scaling method, which uniformly scale width, depth all CNN layers, and resolution of each layer.

Basic building blocks (Figure 7.2) and modules (Figure 7.3) for layers in EfficientNet contained convolutional layers, pooling, batch normalization, rescaling, and dropout.

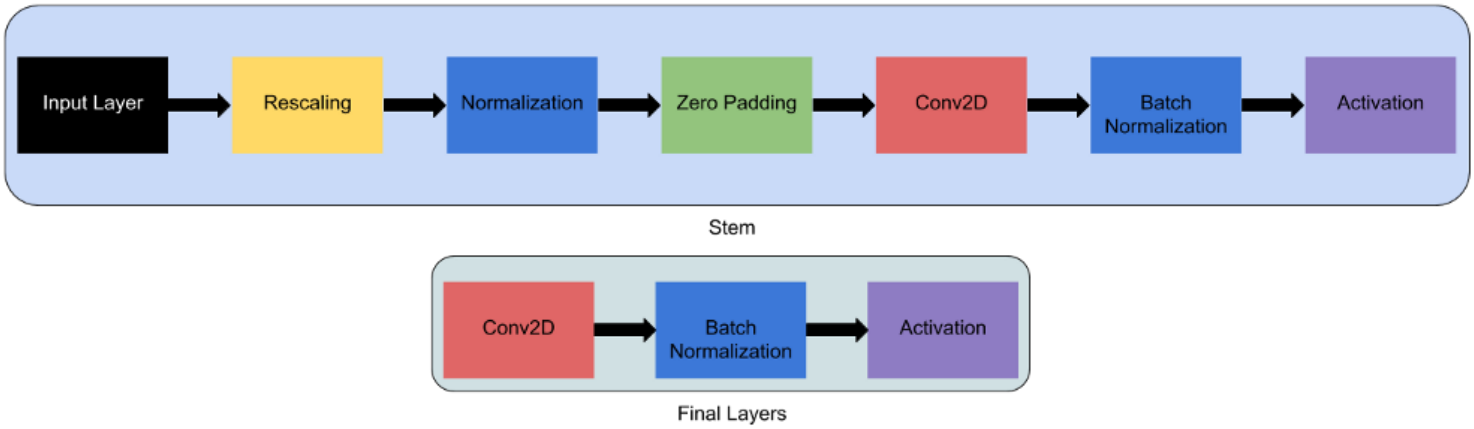


Figure 7.2: Stem and Final layers of EfficientNet, adopted from [3]

With these building blocks is then built whole EfficientNet network model, which has overall 8 versions, differentiated by different dimensional data input (Table 7.1).

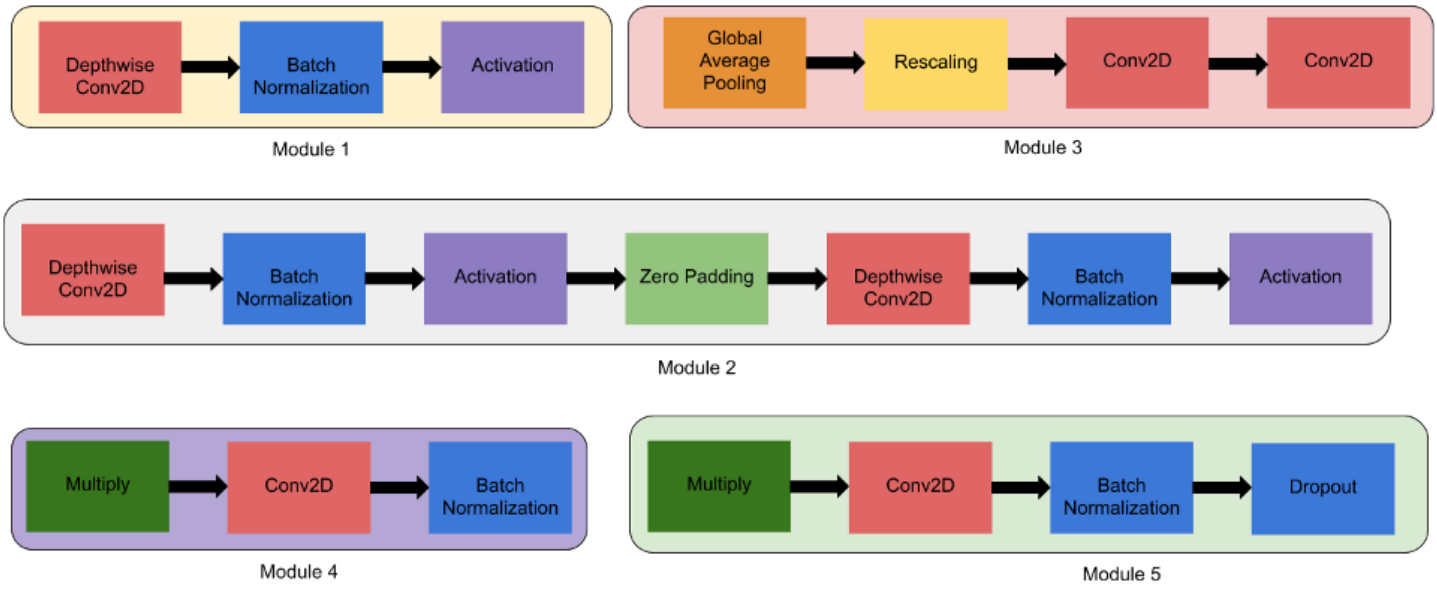


Figure 7.3: Module types of EfficientNet, adopted from [3]

Base model	Resolution
EfficientNetB0	224
EfficientNetB1	240
EfficientNetB2	260
EfficientNetB3	300
EfficientNetB4	380
EfficientNetB5	456
EfficientNetB6	528
EfficientNetB7	600

Table 7.1: EfficientNet base models

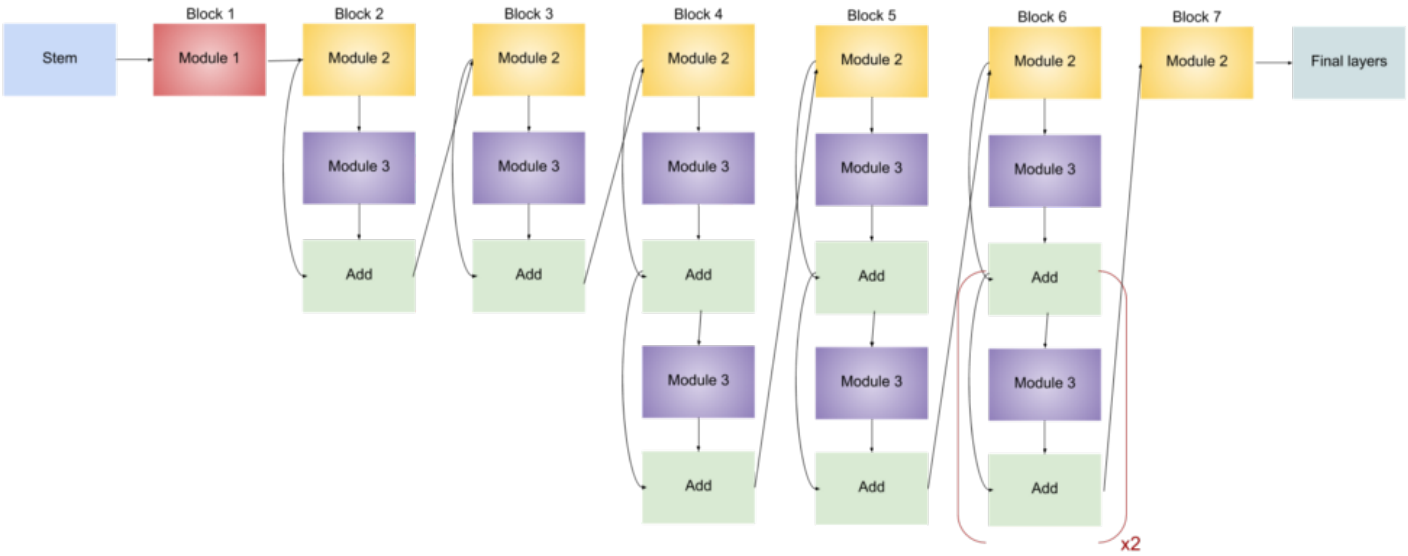


Figure 7.4: Layers layout of type B0 of EfficientNet, adopted from [3]

The most suitable choice from EfficientNet base models seems to be EfficientNet version B0 with the lowest input resolution 224x224.

Because of the lowest number of parameters, input resolution size, and decent accuracy (Figure 7.1).

Chapter 8

Implementation

This section focuses on implementation of chosen neural network architecture in the analysis (Chapter 7).

8.1 Dimension reduction

The implementation of dimension reduction algorithm UMAP in a function.

```
def generate_umap_graph(dataset_name, data, labels, r_state=42
                                , n_ngb=15, m_dst=0.1, metric="
                                euclidean",
is_supervised=False, structure_name="umap"):

if is_supervised:
  file_string = path_graphs + "graphs/" + structure_name + "/"
                                 + dataset_name + "/" + "
                                supervised/" +\
  structure_name + "_" + dataset_name + "_n_neighbors=" + str(
                                n_ngb) + "_min_dist=" + str(
                                m_dst)\
  + "_metric=" + str(metric) + graph_format
else:
  file_string = path_graphs + "graphs/" + structure_name + "/"
                                 + dataset_name + "/" + "
                                unsupervised/" +\
  structure_name + "_" + dataset_name + "_n_neighbors=" + str(
                                n_ngb) + "_min_dist=" + str(
                                m_dst)\
  + "_metric=" + str(metric) + graph_format

if os.path.isfile(file_string):
  print("File already exist: "+file_string)
  return

reducer = umap.UMAP(random_state=r_state, n_neighbors=n_ngb,
                                min_dist=m_dst, metric=metric)
```

```
if is_supervised:
  embedding = reducer.fit_transform(data, y=labels)
else:
  embedding = reducer.fit_transform(data)

sns.set(context="paper", style="white")

fig, ax = plt.subplots(figsize=(12, 10))
  plt.scatter(
  embedding[:, 0], embedding[:, 1], c=labels, cmap="Spectral",
                                s=0.1
)
plt.setp(ax, xticks=[], yticks=[])
plt.axis("off")

fig.savefig(
  file_string,
  bbox_inches="tight")

plt.close(fig)

print("UMAP plot saved as " + file_string)

return
```

8.2 EfficientNet

The implementation of EfficientNet is done through EfficientNet Keras python library[50], which worked better than direct import from the TensorFlow. Import error is probably caused by incompatible version, at the time of writing this work.

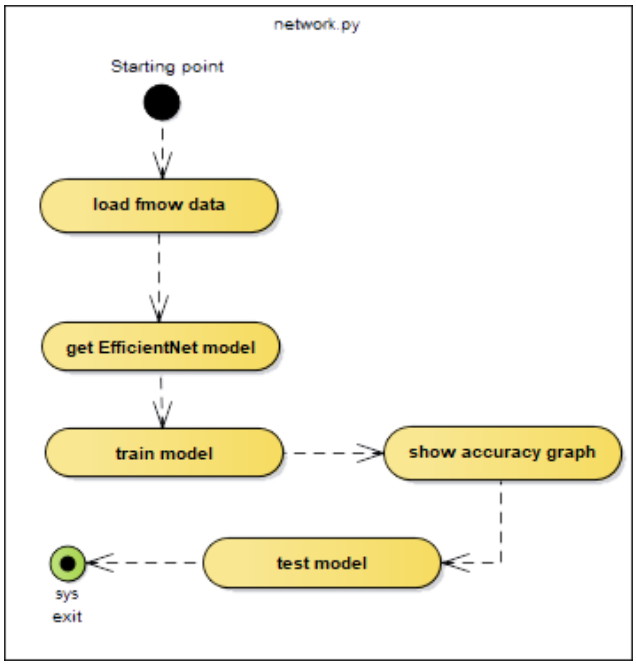
```
def get_model_effnetB0(input_shape, num_classes):
  inputs = keras.Input(shape=input_shape)
  outputs = efn.EfficientNetB0(include_top=True, weights=None,
                               classes=num_classes)(inputs)

  model = keras.Model(inputs, outputs)
  model.compile(
    optimizer="adam",
    loss="categorical_crossentropy",
    metrics=["accuracy"]
  )
  return model
```

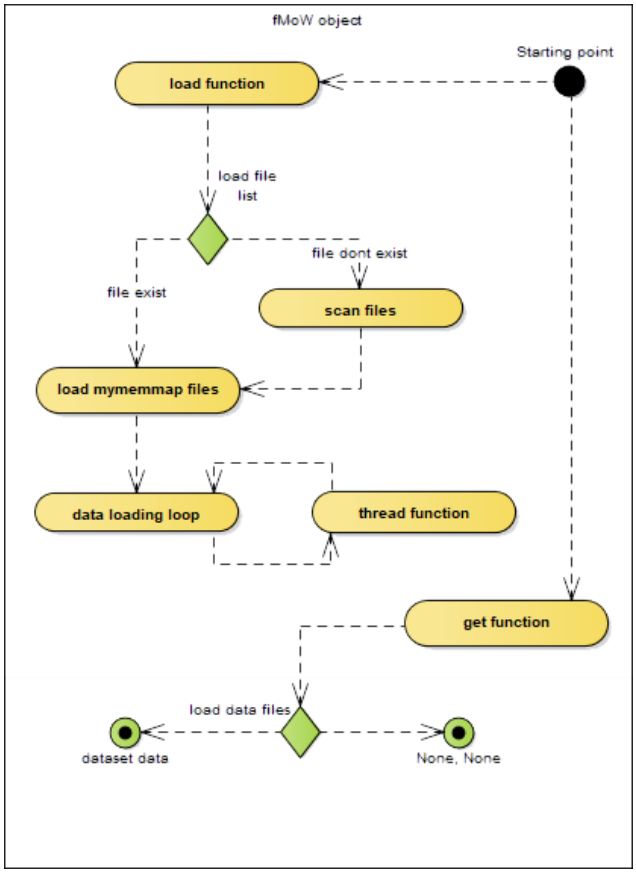
The model is trained without pre-trained weights, with default settings containing adam as an optimizer, categorical_crossentropy as loss, and accuracy as a metric.

Before the fMoW dataset can be used for neural networks, it is necessary to normalize data to specific data shapes. The array of image RGB pixels in N format need to reshaped into (X, Y, 3) shape (final number 3 indicates RGB layers in color images). Each image label then needs to be translated from one number representing the position of a specific class in classes array to new label array with a length of classes array and number 1 in the previous position of classes array.

```
def __normalized_thread(self, data_normalized, data,
                               labels_normalized, labels, i):
  data_normalized[i] = np.reshape(data[i] / 255, (self.
                               image_size, self.image_size, 3),
                                order="F")
  labels_normalized[i][labels[i]] = 1
```



(a) : network script



(b) : fMoW envelope object

Figure 8.1: Workflow diagrams

Chapter 9

Results

This section focused on description of results from implemented (Chapter 8) neural network.

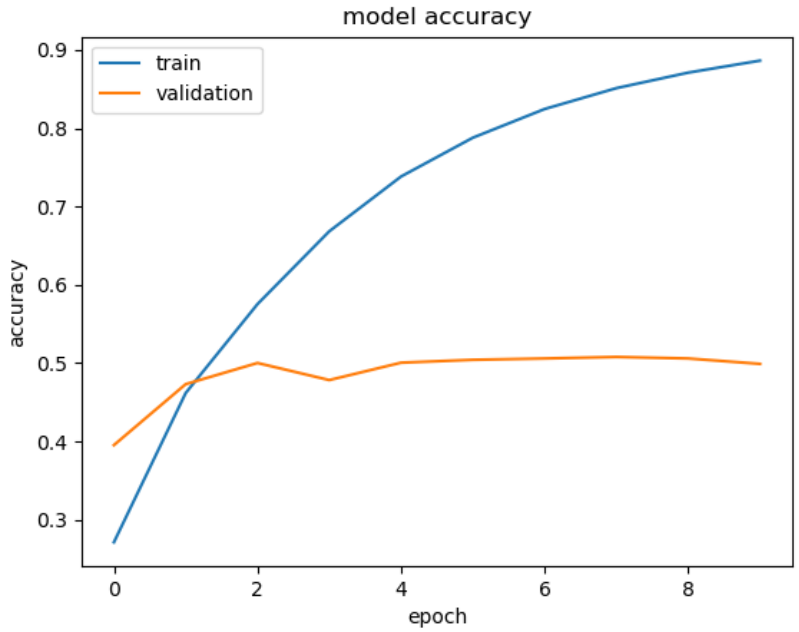


Figure 9.1: Training of fMoW dataset on EfficientNet B0 neural network

As can be seen in model accuracy graph (Figure 9.1) training the fMoW train dataset on the EfficientNet B0 base model after 10 epochs is stuck on approximately 50% accuracy metric. Training was set to only 10 epochs, because a very long computing time per epoch. With data memory maps saved, on HDD time per epoch is 3 hours and with SSD it is 1 hour per epoch. Which makes training of 10 epochs 20 hours computing task.

```
Test loss: 2.674571652876387
Test accuracy: 0.4991421699523926
```

The confusion matrix (Figure 9.2) is 2D grid with each point representing amount of right prediction. Each axis has 62 steps for each class. Only 2 classes were easily predictable, classification report (Table 9.1) contains the more fine analysis of the same data. This can probably indicates implicates deeper issues in the chosen model, which can really generalize the fMoW dataset. Due to limited HW capabilities, a bigger and deeper version of EfficientNet was not trained and tested.

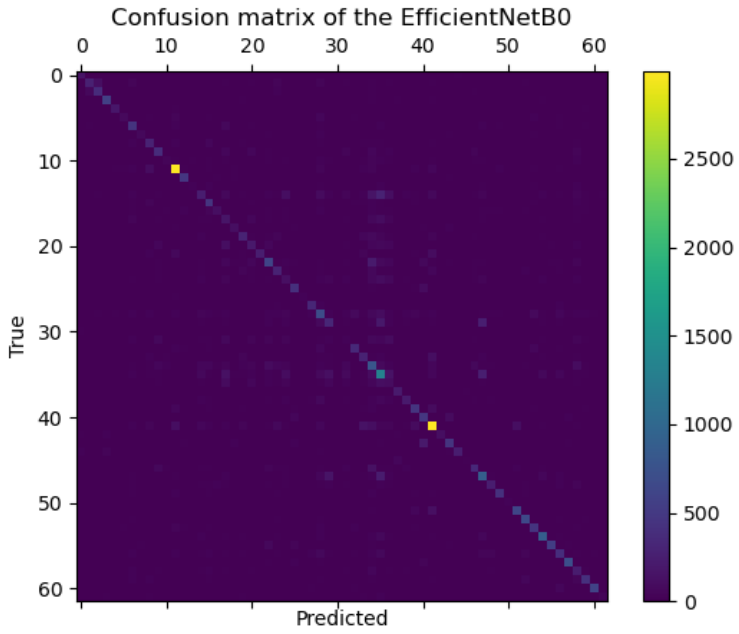


Figure 9.2: Confusion matrix

The more detailed analysis of the confusion matrix is in the classification report (Table 9.1). Precision metric means how many percent was correct, the lowest precise class is office_building and the highest precise class is space_facility. The recall metric means how many positive cases were caught, the lowest recall class is construction_site and the highest recall classes are crop_field and port. The F1-score metric means how many positive cases were correct, the class with the lowest F1-score is construction_site and the classes with highest F1-score are crop_field and wind_farm. The support metric means how many samples of that specific class is in the test dataset.

$$TN = True\ Negative \quad TP = TruePositive$$

FN = False Negative FP = False Positive
Precision = TP/(TP + FP) Recall = TP/(TP+FN)
F1_score = 2*(Recall * Precision) / (Recall + Precision)

Classes	Precision	Recall	f1-score	Support
airport	0.64	0.70	0.67	226
airport_hangar	0.62	0.40	0.49	703
airport_terminal	0.60	0.62	0.61	597
amusement_park	0.77	0.68	0.72	827
aquaculture	0.65	0.74	0.69	243
archaeological_site	0.34	0.20	0.25	372
barn	0.41	0.54	0.47	840
border_checkpoint	0.34	0.25	0.29	247
burial_site	0.35	0.46	0.40	626
car_dealership	0.48	0.49	0.48	781
construction_site	0.14	0.07	0.09	502
crop_field	0.81	0.83	0.82	3607
dam	0.70	0.78	0.74	603
debris_or_rubble	0.17	0.07	0.10	335
educational_institution	0.29	0.14	0.19	1668
electric_substation	0.40	0.53	0.45	860
factory_or_powerplant	0.32	0.19	0.24	547
fire_station	0.14	0.23	0.18	736
flooded_road	0.35	0.35	0.35	326
fountain	0.39	0.35	0.37	852
gas_station	0.27	0.27	0.27	787
golf_course	0.84	0.42	0.56	567
ground_transportation_station	0.46	0.40	0.43	1535
helipad	0.32	0.39	0.35	720
hospital	0.16	0.16	0.16	778
interchange	0.63	0.72	0.67	570
lake_or_pond	0.54	0.31	0.40	105
lighthouse	0.64	0.65	0.64	499
military_facility	0.37	0.36	0.36	1961
multi-unit_residential	0.32	0.27	0.30	1182
nuclear_powerplant	0.25	0.22	0.23	32
office_building	0.04	0.03	0.03	819
oil_or_gas_facility	0.53	0.54	0.54	642
park	0.35	0.40	0.37	723
parking_lot_or_garage	0.29	0.40	0.34	1950
place_of_worship	0.40	0.46	0.42	2913
police_station	0.10	0.14	0.11	770
port	0.69	0.83	0.75	266
prison	0.44	0.34	0.38	637
race_track	0.51	0.53	0.52	845
railway_bridge	0.48	0.67	0.56	702
recreational_facility	0.75	0.71	0.73	4231
impoverished_settlement	0.91	0.55	0.69	193
road_bridge	0.65	0.55	0.60	804
runway	0.86	0.64	0.73	374
shipyard	0.37	0.25	0.30	76
shopping_mall	0.47	0.38	0.42	901
single-unit_residential	0.48	0.53	0.50	1689
smokestack	0.43	0.36	0.39	661
solar_farm	0.68	0.66	0.67	632
space_facility	0.93	0.61	0.74	44
stadium	0.76	0.72	0.74	865
storage_tank	0.65	0.78	0.71	839
surface_mine	0.54	0.61	0.57	746
swimming_pool	0.70	0.68	0.69	1287
toll_booth	0.78	0.66	0.72	831
tower	0.40	0.37	0.39	1056
tunnel_opening	0.75	0.76	0.76	940
waste_disposal	0.23	0.37	0.28	616
water_treatment_facility	0.52	0.55	0.54	746
wind_farm	0.87	0.78	0.82	765
zoo	0.25	0.17	0.20	244
Accuracy			0.50	53041
Macro avg	0.49	0.46	0.47	53041
Weighted avg	0.51	0.50	0.50	53041

Table 9.1: Classification Report

Chapter 10
Conclusions

This section focuses on discussion about results (Chapter 9) from implemented (Chapter 8) neural network and possible future use or work over this topic.

10.1 Conclusion

The following 4 paragraphs represent analysis of 4 points from the assignment of this work.

The current state of the art solutions of dimension reduction is explored from a theory perspective in Theory chapter 3 and from practical use in Experiments evaluation chapter 6. The current state of the art solutions of deep learning algorithms is applied partly in one solution within Analysis chapter 7.

The fMoW dataset is explored using the UMAP in Experiments evaluation chapter 6. During exploration was found, that with increasing the fMoW dataset resolution of images, increased UMAP performance in differentiating data by class. The resolution of the dataset is limited by increasing memory requirements, for the biggest resolutions were SSD used as virtual operational memory. Because of these limitations, experiments with UMAP alone were limited to 75x75 image resolution. The PCA in combination with UMAP reduced memory requirements so much, that 100x100 image resolution was used, but results were worse than without PCA so further fine-tuning of parameters is needed. The UMAP supervised mode with fMoW dataset was also explored, but it had no real desired results.

The new solution to the fMoW dataset in Python using TensorFlow [1] framework is implemented in Implementation chapter 8, but due to solution using CNN compound scaling method and predefined layout the solution is not designed.

The new solution to the fMoW dataset has approximately 50% success rate with accuracy as performance metric which is inadequate, and cannot be compared to other state-of-the-art solutions of the fMoW dataset, due to

the different and unknown scoring rules on topcoder.com site referenced from IARPA challenge[21]. The confusion matrix (Figure 9.2) and classification report (Table 9.1) is in the Results9 chapter, with more in depth class analysis on the test dataset of fMoW dataset.

10.2 Future work

For the future work continuing in these steps, would be advisable to have bigger parameters scope for Dimensionality reduction algorithms, and with the help of better HW equipment train bigger EfficientNet models on more epochs. The use of cloud computing or other similarly powerful method is well recommended. The UMAP method with big dataset like the fMoW datasets has also increasing HW requirements. During the time of experimentation with UMAP, multicore parallel version was not available, if in the future parallelized version will be available, run time can be better.

Bibliography

[1] Martín Abadi et al. "Tensorflow: A system for large-scale machine learning". In: *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*. 2016, pp. 265–283. URL: `https://www.tensorflow.org/` (visited on 2020).

[2] Walid M Abdelmoula et al. "Data-driven identification of prognostic tumor subpopulations using spatially mapped t-SNE of mass spectrometry imaging data". In: *Proceedings of the National Academy of Sciences* 113.43 (2016), pp. 12244–12249.

[3] Vardan Agarwal. "Complete Architectural Details of all EfficientNet Models". In: *towards data science* (May 24, 2020). URL: `https://towardsdatascience.com/complete-architectural-details-of-all-efficientnet-models-5fd5b736142` (visited on 2020).

[4] Fiorenzo Artoni, Arnaud Delorme, and Scott Makeig. "Applying dimension reduction to EEG data by Principal Component Analysis reduces the quality of its subsequent Independent Component decomposition". In: *NeuroImage* 175 (2018), pp. 176–187.

[5] Mukund Balasubramanian and Eric L Schwartz. "The isomap algorithm and topological stability". In: *Science* 295.5552 (2002), pp. 7–7.

[6] Etienne Becht et al. "Dimensionality reduction for visualizing single-cell data using UMAP". In: *Nature biotechnology* 37.1 (2019), p. 38.

[7] Thomas A Caswell. *The h5py package is a Pythonic interface to the HDF5 binary data format.* 2017. URL: `https://www.h5py.org/` (visited on 2020).

[8] Tianqi Chen et al. "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems". In: *arXiv preprint arXiv:1512.01274* (2015).

[9] Yunpeng Chen et al. "Dual path networks". In: *Advances in Neural Information Processing Systems*. 2017, pp. 4467–4475.

[10] Gordon Christie et al. "Functional Map of the World". In: *arXiv preprint arXiv:1711.07846* (2017). URL: `https://github.com/fMoW` (visited on 2020).

[11] Gregory Cohen et al. “EMNIST: an extension of MNIST to handwritten letters”. In: *arXiv preprint arXiv:1702.05373* (2017).

[12] David Cournapeau. *scikit-learn Machine Learning in Python*. 2007. URL: `https://scikit-learn.org/` (visited on 2020).

[13] Manoranjan Dash and Huan Liu. “Feature selection for classification”. In: *Intelligent data analysis* 1.1-4 (1997), pp. 131–156.

[14] Alexandre d’Aspremont et al. “A direct formulation for sparse PCA using semidefinite programming”. In: *Advances in neural information processing systems*. 2005, pp. 41–48.

[15] 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.

[16] Michael W Dorrity et al. “Dimensionality reduction by UMAP to visualize physical and genetic interactions”. In: *bioRxiv* (2019), p. 681726.

[17] Yoav Goldberg and Omer Levy. “word2vec Explained: deriving Mikolov et al.’s negative-sampling word-embedding method”. In: *arXiv preprint arXiv:1402.3722* (2014).

[18] Daniel Granato et al. “Use of principal component analysis (PCA) and hierarchical cluster analysis (HCA) for multivariate association between bioactive compounds and functional properties in foods: A critical perspective”. In: *Trends in Food Science & Technology* 72 (2018), pp. 83–90.

[19] 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.

[20] Gao Huang et al. “Densely connected convolutional networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4700–4708.

[21] IARPA. *Functional Map of the World Challenge*. 2017-2018. URL: `https://www.iarpa.gov/challenges/fmow.html` (visited on 2020).

[22] jianminsun. *fMoW second place solution*. 2017. URL: `https://github.com/fMoW/second_place_solution` (visited on 2020).

[23] *json — JSON encoder and decoder*. URL: `https://docs.python.org/3/library/json.html` (visited on 2020).

[24] Alex Krizhevsky. *Learning multiple layers of features from tiny images*. 2009. URL: `https://www.cs.toronto.edu/~kriz/cifar.html` (visited on 2020).

[25] Neil Lawrence. “Probabilistic non-linear principal component analysis with Gaussian process latent variable models”. In: *Journal of machine learning research* 6.Nov (2005), pp. 1783–1816.

[26] Yann LeCun, Corinna Cortes, and Christopher JC Burges. "The MNIST database of handwritten digits, 1998". In: *URL: http://yann.lecun.com/exdb/mnist* 10 (1998), p. 34. URL: http://yann.lecun.com/exdb/mnist (visited on 2020).

[27] Wentian Li et al. "Application of t-SNE to human genetic data". In: *Journal of bioinformatics and computational biology* 15.04 (2017), p. 1750017.

[28] George C Linderman et al. "Fast interpolation-based t-SNE for improved visualization of single-cell RNA-seq data". In: *Nature methods* 16.3 (2019), p. 243.

[29] Fredrik Lundh. *Python Imaging Library (PIL)*. 1995. URL: https://www.pythonware.com/products/pil/ (visited on 2020).

[30] Laurens van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE". In: *Journal of machine learning research* 9.Nov (2008), pp. 2579–2605.

[31] Leland McInnes. *A Guide to Dimension Reduction*. URL: https://speakerdeck.com/lmcinnes/a-guide-to-dimension-reduction (visited on 2020).

[32] Leland McInnes. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. URL: https://speakerdeck.com/lmcinnes/umap-uniform-manifold-approximation-and-projection-for-dimension-reduction (visited on 2020).

[33] Leland McInnes, John Healy, and James Melville. "Umap: Uniform manifold approximation and projection for dimension reduction". In: *arXiv preprint arXiv:1802.03426* (2018). URL: https://github.com/lmcinnes/umap (visited on 2020).

[34] Rodrigo Minetto, Maurício Pamplona Segundo, and Sudeep Sarkar. "Hydra: an ensemble of convolutional neural networks for geospatial land classification". In: *IEEE Transactions on Geoscience and Remote Sensing* (2019).

[35] Travis Oliphant. *NumPy the fundamental package for scientific computing with Python*. 1995. URL: https://numpy.org/ (visited on 2020).

[36] Jeffrey Pennington, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[37] pfr. *fMoW first place solution*. 2017. URL: https://github.com/fMoW/first_place_solution (visited on 2020).

[38] *pickle — Python object serialization*. URL: https://docs.python.org/3/library/pickle.html (visited on 2020).

[39] Maurício Pamplona Segundo. *fMoW third place solution*. 2017. URL: https://github.com/fMoW/third_place_solution (visited on 2020).

[40] Yian Seo and Kyung-shik Shin. “Hierarchical convolutional neural networks for fashion image classification”. In: *Expert Systems with Applications* 116 (2019), pp. 328–339.

[41] Mingxing Tan and Quoc V. Le. “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”. In: *CoRR* abs/1905.11946 (2019). arXiv: 1905.11946. URL: https://arxiv.org/abs/1905.11946.

[42] Jian Tang et al. “Visualizing large-scale and high-dimensional data”. In: *Proceedings of the 25th international conference on world wide web*. International World Wide Web Conferences Steering Committee. 2016, pp. 287–297.

[43] Vignesh Thakkar, Suman Tewary, and Chandan Chakraborty. “Batch Normalization in Convolutional Neural Networks—A comparative study with CIFAR-10 data”. In: *2018 Fifth International Conference on Emerging Applications of Information Technology (EAIT)*. IEEE. 2018, pp. 1–5.

[44] *threading — Thread-based parallelism*. URL: https://docs.python.org/3/library/threading.html (visited on 2020).

[45] Dmitry Ulyanov. *Multicore-TSNE*. 2016. URL: https://github.com/DmitryUlyanov/Multicore-TSNE (visited on 2020).

[46] Jarkko Venna et al. “Information retrieval perspective to nonlinear dimensionality reduction for data visualization”. In: *Journal of Machine Learning Research* 11.Feb (2010), pp. 451–490.

[47] Svante Wold, Kim Esbensen, and Paul Geladi. “Principal component analysis”. In: *Chemometrics and intelligent laboratory systems* 2.1-3 (1987), pp. 37–52.

[48] Han Xiao, Kashif Rasul, and Roland Vollgraf. “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms”. In: *arXiv preprint arXiv:1708.07747* (2017).

[49] Saining Xie et al. “Aggregated residual transformations for deep neural networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1492–1500.

[50] Pavel Yakubovskiy. *EfficientNet Keras (and TensorFlow Keras)*. 2019. URL: https://github.com/qubvel/efficientnet (visited on 2020).

Appendix A

Dictionary

A.1 HW Terminology

CPU	Central Processing Unit
GPU	Graphics Processing Unit
RAM	Random Access Memory
SSD	Solid State Drive
HDD	Hard Disk Drive
CD	Compact Disk
RGB	Red Green Blue

A.2 Dimensionality reduction terminology

PCA	Principal Component Analysis
t-SNE	t-distributed stochastic neighbour embedding
UMAP	Uniform Manifold Approximation and Projection for Dimension Reduction
MNIST	Modified National Institute of Standards and Technology
CIFAR	Canadian Institute For Advanced Research
fMoW	Functional Map of the World

A.3 AI and SW terminology

AI	Artificial Intelligence
NN	Neural Network
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
JSON	JavaScript Object Notation

Appendix B

Tree of CD directories

```
CD
├── dim-reduce-generator
│   ├── data
│   │   ├── dataset_preview.py
│   │   └── datasets.py
│   ├── graph_algortihms
│   │   ├── compare_graphs.py
│   │   ├── generate_graph_pca.py
│   │   ├── generate_graph_pca_tsne.py
│   │   ├── generate_graph_pca_umap.py
│   │   ├── generate_graph_tsne.py
│   │   └── generate_graph_umap.py
│   ├── conda_env.yml
│   └── params.py
├── fmow-solution
│   ├── data
│   │   └── fmow.py
│   ├── network
│   │   ├── network.py
│   │   └── weights
│   ├── conda_env.yml
│   └── params.py
├── graphs-cifar10
│   ├── umap_cifar-10_plot_metric=correlation.png
│   └── umap_cifar-10_plot_metric=euclidean.png
├── graphs-cifar100
│   ├── umap_cifar-100_plot_metric=correlation.png
│   └── umap_cifar-100_plot_metric=euclidean.png
├── graphs-fashion-mnist
│   └── umap_fashion-mnist_plot_metric=euclidean.png
├── graphs-mnist
│   └── umap_mnist_plot_metric=euclidean.png
└── graphs-comparisons
    └── ...
```