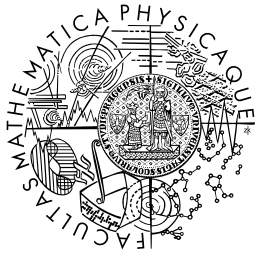


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

BACHELOR THESIS

Oliver Piter

Unsupervised Cloud Classification from Sky Imagery

Department of Software and Computer Science Education

Supervisor of the bachelor thesis: doc. RNDr. Elena Šikudová, Ph.D.
Study programme: Computer Science
Study branch: Artificial Intelligence

Prague 2024

I declare that I carried out this bachelor thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

I would like to thank supervisor, Elena Šikudová, for her guidance, Computer Graphics Group at Charles University for providing me with computational resources and training data, my family and friends for their support and patience.

Title: Unsupervised Cloud Classification from Sky Imagery

Author: Oliver Piter

Department: Department of Software and Computer Science Education

Supervisor: doc. RNDr. Elena Šikudová, Ph.D., Department of Software and Computer Science Education

Abstract: Cloud classification task is a task where we classify images of clouds into groups of similar images. Images are similar in terms of texture, shape, colour, size and other visual aspects. The aim of this work is to create an algorithm that can cluster images of clouds based on their sky coverage so that a user can request a specific type of cloud coverage from database of images and use the images for downstream tasks such as procedural sky generation. We use various models for feature extraction based on contrastive learning and image reconstruction and for clustering we use self-supervised learning and distance-based methods. These methods were used for clustering a subset of our data and evaluated based on visual consistency of clusters and cluster separation metrics.

Keywords: Unsupervised Classification Cloud image Contrastive learning

Contents

Introduction

Cloud classification is a very specific classification task when compared to other image classification tasks such as objects detection from images of objects from daily life. Instead of having high variance of data to work with, we only have very specific type of objects which need to be separated into groups based on their visual characteristic such as texture, colour, size, shape and so on. However, compared to everyday objects clouds are very homogeneous in this sense. They all appear in the sky, so usually their background has blue colour with the exception of situations when the cloud covers the whole sky. Also, other objects present in the sky, such as the Sun, represent noise which can make the process of classification even harder. When the Sun shine through clouds, it may change the colour or the visual perception of shape of the cloud as well as it may change the colour of the sky which might be considered by the model as a different object.

In recent years, new methods of classification were discovered for the task of general image classification. For *ImageNet-10* dataset [Deng et al., 2009] the state-of-the-art clustering method was introduced by Niu, Shan, and Wang [2022]. It utilizes contrastive learning as feature extraction technique by training the underlying network contrastive prediction task such as Chen et al. [2020] or He et al. [2020]. However, these models solve more general task than we do because they were evaluated on general image datasets such as *ImageNet-10* [Deng et al., 2009]. Other works have taken the concepts from these general classification tasks and applied them to cloud classification task. Lv et al. [2022] used contrastive learning with ResNet50 [He et al., 2016] backbone and MLP clustering head similarly to Niu et al. [2022]. We recognize that this work trained MLP in supervised manner with labelled dataset, however, our dataset contains only unlabelled data. Dematties et al. [2023] utilized Vision Transformer introduced by Dosovitskiy et al. [2021] and training was done in self-supervised manner from Caron et al. [2021].

In this work we aim to create an algorithm that can classify clouds according to cloud coverage of the sky. By having such algorithm we could label a database full of sky imagery and then query images containing specific types of clouds which can later be used for procedural sky generation in computer graphics. We use framework introduced by Niu et al. [2022] for its state-of-the-art performance on *ImageNet-10* [Deng et al., 2009]. Due to the framework having multiple components that can be modified, we exchange its feature extraction model MoCo [He et al., 2020] for autoencoder [Hinton and Salakhutdinov, 2006] with convolutional layers [Lecun et al., 1998] and its clustering head for KMeans [Macqueen, 1967] to get a comparison how each of the components influences the classification process.

We delve into the architecture and function of these models more in Chapter 1. Then, we describe the datasets used for training and their preprocessing in Chapter 2. Chapter 3 presents all the experiments conducted along with the parameters used for these models. After that, Chapter 4 shows the results of the classification with Silhouette coefficients Rousseeuw [1987] and visual analysis for clustering. Also, a separate analysis of image representation from feature extraction models using visualization techniques such as principal component analysis (PCA) [Maćkiewicz and Ratajczak, 1993] and t-SNE [van der Maaten and Hin-

ton, 2008] is included in this chapter. Chapter 5 describes code implementation and prerequisites required to run the code.

1. Background

The aim of this chapter is to introduce the problem of clustering to the reader and the approaches to tackle this problem. In later sections, concrete models and approaches used in this work are explained.

1.1 Clustering

Clustering is an unsupervised machine learning technique which separates data into some groups according to some condition. The number of clusters is either known in advance or it should be inferred.

In the case of sky imagery it is expected that during clustering similar clouds end up in the same cluster and different clouds end up in different cluster. Clouds are similar when their size, texture, colour and (visual) height are similar.

1.2 Clustering of images

When it comes to clustering images, usually, some pre-processing of the data is done before the clustering because working with raw images is not very convenient in most situations.

The approach that is used in this work is divided into several steps:

1. Feature extraction
2. Clustering of extracted features

1.3 Feature extraction

Images that we are going to use contain a lot of unimportant information that does not help with the intended machine learning task. In Figure 1.1 it can be observed that in the top left corner there is information about the entity that is responsible for taking of the pictures. In the top right corner there is some additional information about the weather on the day the picture was taken. Additionally, the bottom part of the picture contains objects that are not related to clouds at all (trees, mountains and so on). Therefore, we need to use a method that would extract only important information about the clouds contained in the images and not consider the parts of the picture that do not contribute any information about clouds visible in the sky.

1.3.1 Convolutional autoencoder

The purpose of autoencoder is to learn a "compression" system that describes the input data well enough that the original data can be reconstructed from the compression up to some error ϵ. Formally, let us have encoder $\mathcal{E}$, decoder $\mathcal{D}$ and input data $x \in \mathbb{R}^{h \times w \times c}$. Let us also have some error function $\mathcal{L}$ which computes the error between the original data and reconstructed data. Then, the error rate of autoencoder reconstruction should be $\mathcal{L}(\mathcal{D}(\mathcal{E}(x))) < \epsilon$.

Figure 1.1: An example of image from sky dataset.

The loss function used in this work for autencoder is standard MSE (mean-square error) which is defined as

$$\mathcal{L}(x) = \frac{1}{h \times w \times c} \sum_{i=0}^{h} \sum_{j=0}^{w} \sum_{k=0}^{c} (x_{ijk} - \overline{x}_{ijk})^2$$

where $\overline{x} = \mathcal{D}(\mathcal{E}(x))$.

1.3.2 MoCo

MoCo (Momentum Contrast for Unsupervised Visual Representation Learning) from He et al. [2020] is a contrastive learning technique to learn features in unsupervised manner. During contrastive learning an underlying model is trained in a dictionary lookup fashion where each input image is associated with an encoded key. Query encoder and key encoder are two separate models with the same architecture. In this work the model is trained in such a way that if two pictures are just different views (for example different crops) their associated key should be the same. The underlying model used in this work is ResNet50.

In principle the training of MoCo is described in Figure 1.2. An instance $x \in \mathbb{R}^{m \times n}$ from training dataset is taken as query and alongside with a queue of $x_0^{key}, x_1^{key}, x_2^{key}, \cdots \in \mathbb{R}^{m \times n}$ key instances from the dataset. Then, encoded query and and key images from momentum encoder are taken and contrastive loss is computed. The momentum-based key encoder is moving averaged query encoder.

Contrastive loss

Let us have representation space $\mathcal{R}$ where each instance of input data has its representation. The representation can be a result computed by some model applied to input data. Contrastive loss measure the similarity a samples in this sample space.

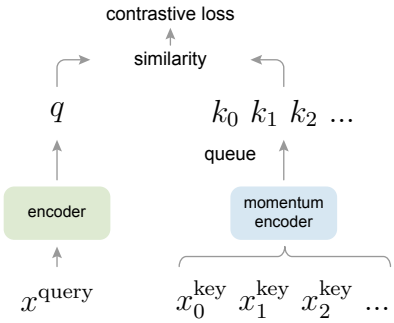


Figure 1.2: MoCo training diagram from He et al. [2020].

The loss used in this work measures how well a model performs in dictionary lookup task. The loss is defined in He et al. [2020] as

$$\mathcal{L}(q, k_n) = -\log\left(\frac{\exp(q \cdot k_+/\tau)/}{\sum_{i=0}^{K}\exp(q \cdot k_i/\tau)}\right) \tag{1.1}$$

where q is query example and $\{k_0, k_1, k_2, \cdots\}$ is a set of examples that are keys in the dictionary and k_+ is the instance of the positive example for given query q and τ is temperature hyperparameter Hinton et al. [2015].

Momentum update

As shown in Figure 1.2 encoder for query and encoder for keys are two separate models of the same type. Parameters for the query encoder θ_q are updated using back-propagation. For the key model to evolve more smoothly, its parameters are not updated using back-propagation but are updated as weighted averaged of query and key encoders

$$\theta_k \leftarrow m\theta_k + (1-m)\theta_q$$

where m is momentum parameter coefficient.

Image augmentations

Augmentations used in MoCo training are described by He et al. [2020] as: random resized crop, random grayscale, colour jitter and random horizontal flip.

1.4 Clustering

Clustering is a process of dividing input data into some groups such that similar data points end up in the same group and different data points end up in a different group.

1.4.1 SPICE clustering

SPICE (Semantic Pseudo-Labeling for Image Clustering) introduced in Niu et al. [2022] takes previously trained feature extraction model and clusters the data

based on the representations given by feature extraction model in the feature space.

The model is trained using EM framework (expectation-maximization). The EM framework is employed in the following manner:

1. Feature extraction model computes embeddings for original images e_i.
2. Clustering head assigns probabilities p_i over $\mathbf{K}$ classes to the embeddings of weakly transformed images.
3. Compute probabilities p_i' over $\mathbf{K}$ classes and optimize the clustering head using Equation 1.5.

Let the batch of input data be $\mathbf{X} \subset \mathbb{R}^{N \times w \times h \times c}$ where N is batch size, $w \times h \times c$ are the dimensions and number of channels of input images. Let us denote clustering head $\mathcal{C}$ and feature extraction model $\mathcal{F}$. The clustering head $\mathcal{C}$ is a two layer MLP (multi-layer perceptron) which takes the embedding of some input data $e_i = \mathcal{F}(x_i, \theta_f)$, where θ_f are parameters of the feature model, and assigns a probability distribution p to weakly transformed input images among classes $\{p_i\}_{i=0}^{K-1}$ where $\mathbf{K}$ is the number of classes and $p_i = \mathcal{C}(e_i, \theta_c)$ with clustering head parameters θ_c. Let $\mathbf{P} \in \mathbb{R}^{N \times K}$ be a matrix containing class probabilities for each data point in the batch.

However, ground truth labels are needed in order to train MLP. To tackle this problem, Niu et al. [2022] propose that pseudo-label is assigned to each data point from the batch (concretely to its feature embedding). Niu et al. [2022] denote most confident samples for class k as

$$\mathcal{F}_k = \left\{ f_i | i \in \text{argtopk}\left(P_{:,k}, \frac{N}{K}\right) \right\}. \tag{1.2}$$

Then, cluster center for class k is

$$\gamma_k = \frac{K}{N} \sum_{f \in \mathcal{F}_k} f. \tag{1.3}$$

Then, $\frac{N}{K}$ nearest samples to $\gamma_k, \forall k = 1, 2, \cdots, K$ according to cosine similarity are found and denoted by χ^k and their pseudo-labels are set to $y_i^s = k$ where y_i^s is a label corresponding to data point $x_i \in \chi^k$. Therefore, batch containing pseudo-labelled images is defined by Niu et al. [2022] as

$$\chi^s = \left\{ (x_i, y_i^s) | \forall x_i \in \chi^k, k = 1, 2, \cdots, K \right\} \tag{1.4}$$

Next, according to labelled dataset χ^s the clustering head is optimized in the maximization step. This is done by minimizing the following loss from Niu et al. [2022]

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_{ce}(y_i^s, \text{softmax}(C(\mathcal{F}(\beta(x_i); \theta_{\mathcal{F}}^s); \theta_c))). \tag{1.5}$$

where $\mathcal{L}_{ce}$ is cross-entropy loss and β is strong augmentation.

Image augmentations

For this work the image augmentations from Niu et al. [2022] were used. For weak transformation random crop and random horizontal flip were used. For strong augmentation random horizontal flip, random crop and 4 random augmentations from the following: identity, auto contrast, equalize, rotate, solarize, color, contrast, brightness, sharpness, shear x, shear y, translate x, translate y and posterize.

1.4.2 KMeans

For comparison with the previously mentioned SPICE clustering this work also uses KMeans algorithm [Macqueen, 1967] in order to cluster images based on their representations from previously trained feature extraction model. KMeans firstly selects **K** cluster centers from the dataset and then iteratively assigns all data to a nearest cluster and updates the cluster centers until convergence or until a maximum number of iterations is reached.

1.5 t-SNE

t-SNE was introduced by van der Maaten and Hinton [2008] as a method of visualization of high-dimensional data into low-dimensional space while preserving local structure of data.

Let the input dataset be $\mathbf{X} \in \mathbb{R}^{n \times d}$ where n is the number of data points and d is the dimension of the high-dimensional feature space.

1.5.1 High-dimensional data representation

First, t-SNE converts Euclidean distances between data points into conditional probabilities which represent similarities between data points. Points that are close to each other are assigned high probability, whereas points that are far away from each other will have lower probability according to Gaussian probability density centered around each data point. Conditional probability that x_i picks x_j as its neighbour is given by van der Maaten and Hinton [2008] as

$$p_{j|i} = \frac{\exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-||x_i - x_k||^2/2\sigma_i^2)} \tag{1.6}$$

where σ_i is the variance of the Gaussian centered around x_i. Since this method models pairwise similarities $p_{i|i} = 0$. From this the joint probability distribution was redefined by van der Maaten and Hinton [2008] as

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n} \tag{1.7}$$

in order to battle outliers with high distance.

1.5.2 Low-dimensional data representation

For the low-dimensional mapping points $\mathbf{Y} \in \mathbb{R}^{n \times l}$ where l is the dimension of low dimensional space we are mapping to (usually 2 or 3 dimensions). For low-dimensional mapping, another probability distribution Q is considered and the joint probabilities can be computed as described by van der Maaten and Hinton [2008] as

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||y_k - y_l||^2)^{-1}} \tag{1.8}$$

where Student t-distribution with one degree of freedom is used. For measuring how much joint probabilities P and Q differ, authors use Kullbeck-Leibler divergence. The divergence is then minimized using gradient descent on cost function from van der Maaten and Hinton [2008]

$$C = \sum_i KL(P||Q) = \sum_i \sum_j p_{ij} \log \left(\frac{p_{ij}}{q_{ij}} \right). \tag{1.9}$$

The points $\mathbf{Y}$ in the low-dimensional space are initialized randomly from an isotropic Gaussian with small variance that is centered around the origin.

1.5.3 Variance

In order for the distribution on high-dimensional data to be computed, variance of the Gaussian is needed for each data point. The algorithm performs binary search over values of σ_i for each x_i so that it produces P_i with fixed perplexity specified by the user. P_i is conditional probability distribution over all data points given data point x_i. Perplexity is defined as

$$Perp(P_i) = 2^{H(P_i)} \tag{1.10}$$

where $H(P_i)$ is Shanon entropy defined as

$$H(P_i) = -\sum_j p_{j|i} \log(p_{j|i}). \tag{1.11}$$

2. Dataset

Dataset used in this work consists of web camera shots from CHMI (Czech Hydrometeorological Institute). The dataset contains pictures from 98 locations around Czech republic. Images were collected from the middle of March 2021 until the time of writing this work. The images in their original form have resolution 1600×1200 pixels and are RGB. In this work, some preprocessing was done in order to ease the hardware requirements for model training and also some models require certain type of preprocessing being done to the input. Only a small portion of images was used for training because of computational requirements and also certain locations had better view of the sky than others. Also, some filters had to be used in order to filter out images that were not suitable for training. In the next sections there are described the subsets of data used as training data and preprocessing done on each dataset.

Each of these datasets could have gone through additional preprocessing before entering model in experiments, however, preprocessing described in the next sections are common for all experiments and model-specific preprocessing is described for each experiment separately.

2.1 Prague+Brno dataset

First dataset that was used in this work is a combined dataset from Prague and Brno in order to get more training data rather than using only one location. Figure 2.1a and 2.1b show some example images from the dataset. It can be observed that in the pictures there are some objects that are not part of the sky, such as trees or houses. Also, there are watermarks and weather information present in every image. The method to get rid of this kind of noise is simply masking these areas with black pixels while preserving as much of the sky view as possible. Next, images are cropped to 1200×1200 pixels so they are square and from there are downscaled to 256×256 pixels for easier computational processing. An example of images after preprocessing are in Figure 2.1d and 2.1e show the final images after preprocessing. In this work, this dataset is going to be referred to as Prague+Brno dataset.

2.2 Holesov dataset

This location was chosen in particular because out of all locations in the whole dataset this one had the lowest horizon meaning that most of the picture consisted of the sky view. Example can be found in Figure 2.1c. Therefore, to minimize the noise in this dataset, the pictures were cropped in the middle section to get as big square image as possible while cropping out all the watermarks, land and weather information from the pictures. The resulting square image after cropping had resolution of 1000×1000 pixels and then was downscaled to 256×256 pixels as in Figure 2.1f. In this work, this dataset is going to be refered to as Holesov dataset.

(a) Brno (b) Prague (c) Holesov

(d) Brno (e) Prague (f) Holesov

Figure 2.1: Example pictures before (top) and after (bottom) preprocessing.

3. Experiments

This chapter goes through all the main experiments conducted in this work. It includes experiments done using the framework proposed in Niu et al. [2022], variations of this framework with alternative models and clustering algorithms and models that do not use anything from the aforementioned article.

3.1 Original SPICE pipeline

As the first experiment the pipeline from Niu et al. [2022] was used. Only feature extraction model and clustering head from the paper were employed for this experiment. The pipeline was trained separately on Prague+Brno and Holesov datasets.

Preprocessing

Before the training loop begins, the mean and standard deviation are computed for each colour band in each of the datasets. During training, the images are normalized to have mean equal to 0 and standard deviation equal to 1.

Hyperparameters

Image mean and standard deviation were calculated for each dataset separately and then used for the specific experiment. The training setup was the same as described in He et al. [2020] and the training parameters are included in Table 3.1. The hyperparameters for clustering head were used as the one for *ImageNet-10* [Deng et al., 2009] described in the SPICE repository with changes listed in Table 3.2. These hyperparameters were used for both preprocessed datasets. The choice of k is $4, 7, 10$ because we want to separate images into classes based on the cloud coverage and visually we could not identify more than 10 different classes in the data.

Parameter	Value
Number of clusters	10
Size after crop	256×256
Batch size	256
Learning rate	0.03
Weight decay	0.0001
Temperature	0.07
Queue size	65535

Table 3.1: Training parameters for MoCo.

Parameter	Value
Number of epochs	30
Crop size	256×256
Batch size	5000
Learning rate	0.005
Weight decay	0
Number of clusters	4,7,10

Table 3.2: Training parameters for SPICE.

3.2 MoCo with KMeans

For this experiment training of MoCo feature model is done in the same manner as in the original SPICE pipeline. Instead of using MLP as clustering head the KMeans algorithm is used to cluster the image embeddings. This is to compare the performance of SPICE clustering and other clustering methods. The number of classes set for KMeans is $4, 7, 10$. For training of the feature model the same two datasets were used as in the first experiment.

3.3 CAE with SPICE clustering

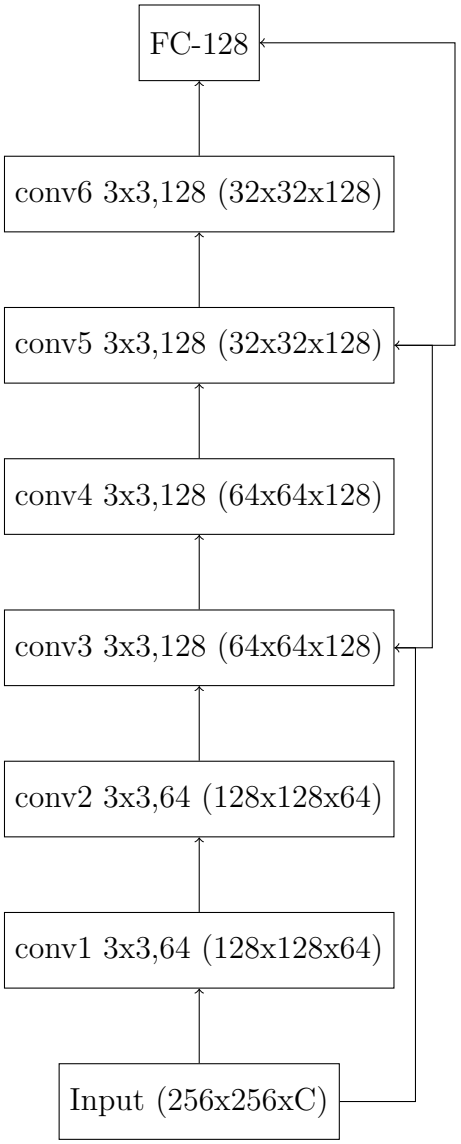


Figure 3.1: CAE encoder part.

This experiment tries to evaluate the performance of SPICE clustering head with a different feature extraction approach. The architecture of the CAE encoder is explained in Figure 3.1. It consists of 6 convolutional layers in encoder where every other layer, beginning with the second one, halves the resolution of the input

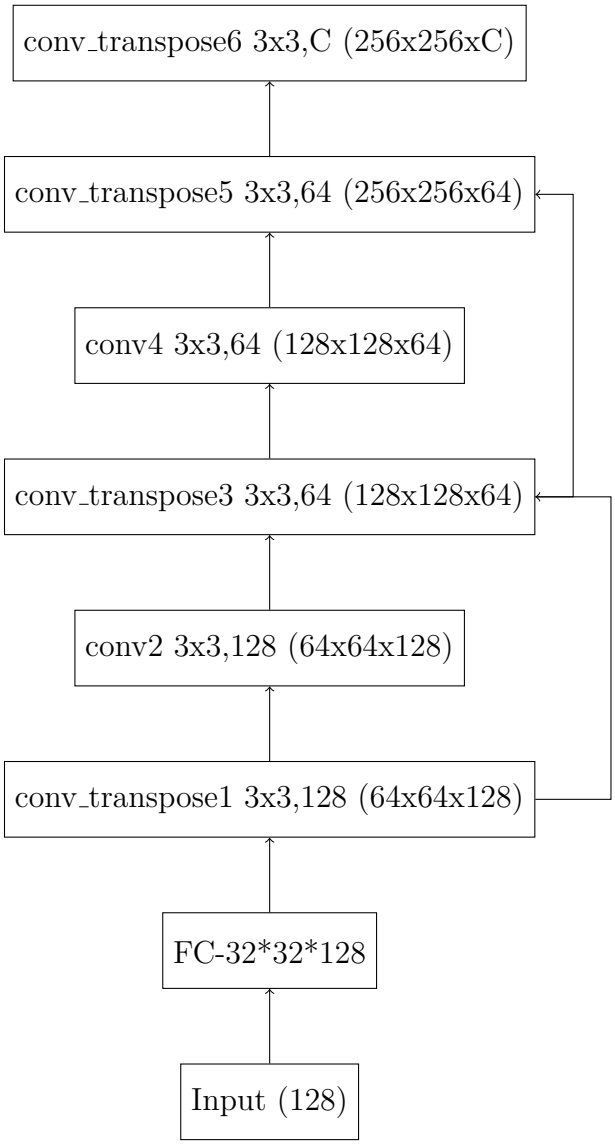


Figure 3.2: CAE decoder part.

image. In addition, the third layer also doubles the number of channels from 64 to 128. The first layer projects the number of channels into 64-dimensional space. Also, skip connections are used to skip between 1st and 2nd layer, 2nd layer and 3rd layer and so on. Prior to entering the model, the images are normalized to $[0, 1]$ range. Before entering into each layer but the first one, activation function *LeakyReLU* is applied. The decoder has the same architecture as encoder part and it is explained in Figure 3.2.

The encoder and decoder are trained at the same time. After the training is done, the decoder is not used anymore and only encoder is used to construct embeddings. The sizes of bottleneck used varied from 16 to 256.

The model is firstly trained on RGB images from both datasets. No colour modifications have been done to the dataset so it can be observed how colour data helps the model to learn meaningful representations. Then, the model is trained on grayscale images. Lastly, the model is trained on RGB images with additional edge data.

3.4 CAE with KMeans

This approach was chosen as a complete alternative which does not use any technology described in Niu et al. [2022]. The autoencoder was trained the same way as in the previous experiment and $k = 4, 7, 10$ for KMeans.

3.5 CAE on CIFAR10

To check if the proposed autoencoder architecture and training loop are capable of learning representations of general data the autoencoder is trained on *CIFAR10* dataset Krizhevsky et al. [2009] as reconstruction task. Then, PCA is performed on the resulting representations to see if any clusters were formed.

The hyperparameters for training the clustering head were the same as in the first experiment.

4. Results

The results shown from all experiments have various forms to analyse how the models perform and why they achieve such results. The results include image reconstructions from CAE, PCA and t-SNE visualizations of embeddings to get an idea of how the model perceives the data. Clusters were also evaluated using Silhouette coefficient from Rousseeuw [1987], ARI (Adjusted Rand Index) from Hubert and Arabie [1985] and manual visual inspection.

4.1 MoCo analysis

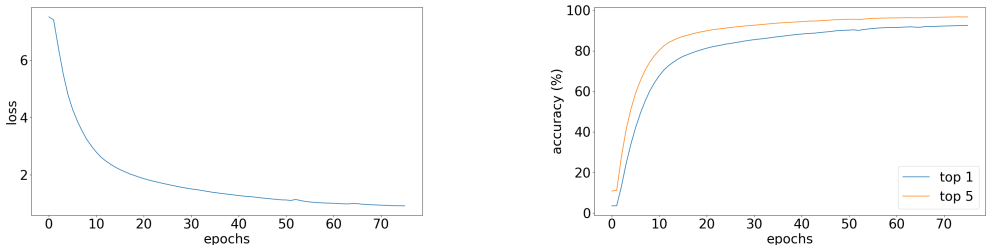
This section dives into image representations created by MoCo with embedding visualization using PCA and various clustering algorithms to find out if MoCo image embeddings capture features of the clouds that would be more suitable for clustering task.

4.1.1 MoCo training

MoCo was trained on both datasets separately and until the training loss converged. The training parameters were used as described in Chapter 3. The losses and accuracies were plotted on single run.

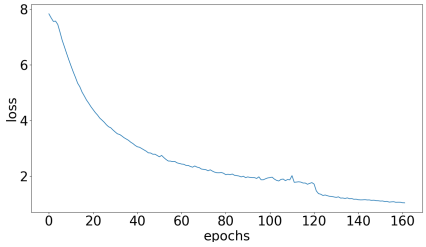
Prague+Brno

MoCo was trained for 75 epochs until convergence of the loss function plotted in Figure 4.1a. Moreover, the top 1 and top 5 accuracies were calculated, where in top 5 accuracy the model was very close to 100 % and in top 1 accuracy the model got around above 90 % accuracy. Accuracies were computed such that when query produced by query encoder has the smallest angle with its positive key produced by key encoder compared to the rest of keys that are present in the queue, then it is considered as correct assignment. The plotted graphs can be found in Figure 4.1b.

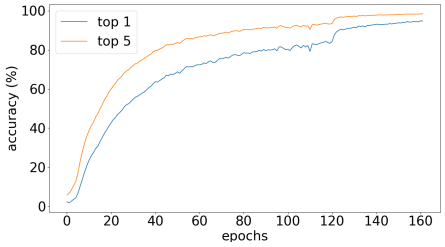


(a) Loss. (b) Accuracy.

Figure 4.1: Training performance of MoCo on Prague+Brno dataset.

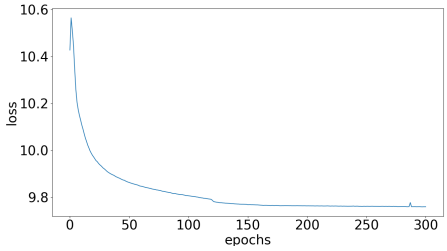


(a) Loss.

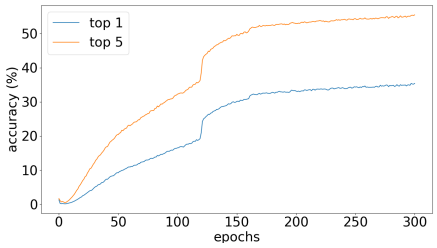


(b) Accuracy.

Figure 4.2: Training performance of MoCo on Holesov dataset.



(a) Loss.



(b) Accuracy.

Figure 4.3: Training performance of MoCo on Holesov dataset with $\tau = 0.7$.

Holesov

Similar training performance was observed in the case of Holesov dataset. In this case the model had less data because only one location was used and also night images were filtered out. The model ran for 160 epochs for the loss function to converge as can be seen in Figure 4.2a. Similarly to Prague+Brno dataset, the training top 1 accuracy was a bit above 90% and top 5 accuracy reached very close to 100% as can be observed in Figure 4.2b.

The images for the original SPICE were kept in RGB and mean and variance for each colour band was computed for each dataset to be used as normalization during training.

Temperature parameter

According to Hinton et al. [2015] temperature parameter τ used in MoCo training in softmax in Figure 1.1 produces softer probability distribution when higher values of τ are used. Therefore, in effort to separate different types of clouds in the representation space temperature was set to $\tau = 0.7$ instead of the original $\tau = 0.07$. This resulted in model converging too quickly without learning meaningful representation. The loss function did not lower as much as for the case for lower τ which can be observed in Figure 4.3a. Accuracies were also much lower, where top 1 accuracy reached only 35% and top 5 accuracy reached almost 60% in Figure 4.3b.

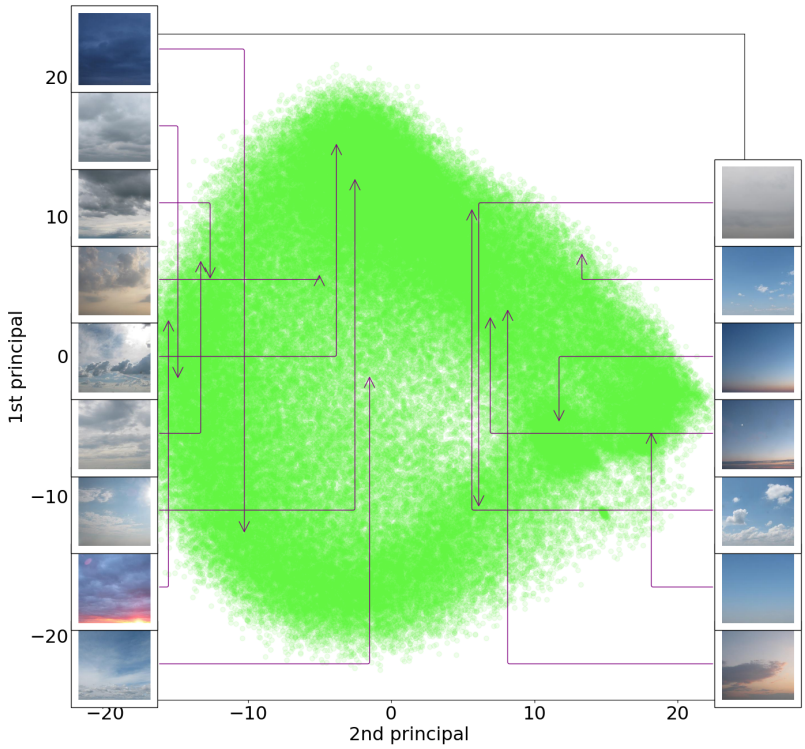


Figure 4.4: PCA on Holesov dataset for MoCo.

4.1.2 PCA on MoCo

This subsection contains PCA visualizations of the embeddings of the last layer before fully connected layer of ResNet50. The visualizations also contain image examples to further explain the feature space and colours separate points from different locations. Some figures were also projected with smaller subsets of data for better clarity of graphs.

Holesov

PCA on the whole training dataset for Holesov reveals that 2 or 3 clusters can be found in the feature representation. Figure 4.4 shows that similar images are indeed closer to each other in representation space. Another thing to note from the visualization is that even images that have different clouds in them appear closer in the visualization, most like due to their colour similarity. Lastly, brighter images appear higher along y-axis and dimmer images appear lower along y-axis.

Prague+Brno

For Prague+Brno dataset it can be observed from Figure 4.5 that only one smaller cluster and one big clusters were formed for both locations. Analysis of the smaller cluster reveals that the smaller cluster for both locations contains night images and the bigger cluster contains daytime images. The visualization also reveals that the model also learned to distinguish between the locations because for

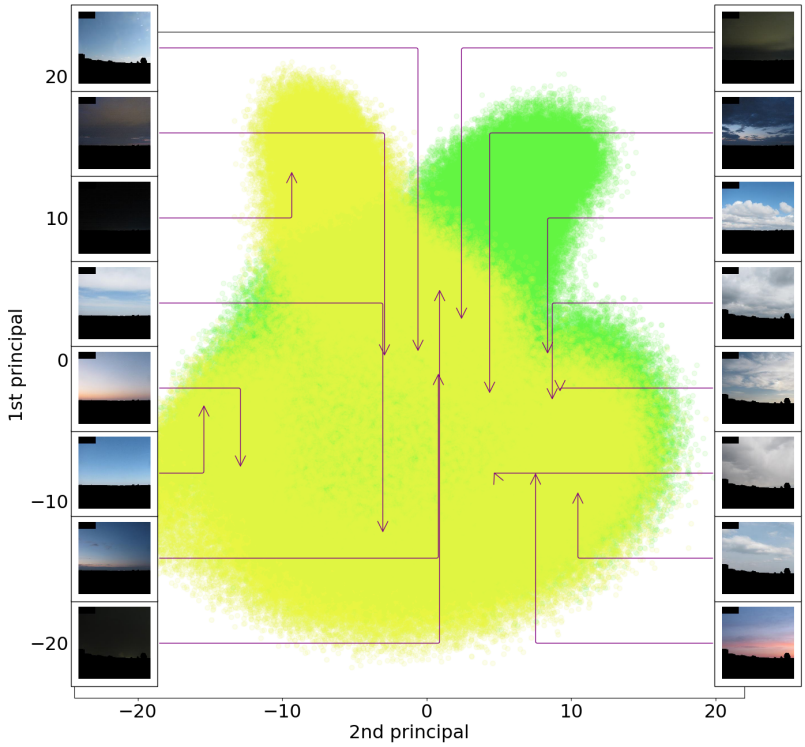


Figure 4.5: PCA on Prague+Brno dataset for MoCo.

example the visualized daytime image of clear blue sky appeared in a completely different place after PCA projection for Prague and for Brno. Similarly as for Holesov dataset it can be observed that images containing different clouds are closer to each other when their overall colour is also similar, therefore image containing lots of smaller gray clouds is close to an image containing a big gray cloud.

4.1.3 t-SNE on MoCo

This subsection contains t-SNE visualizations of the feature space computed from the last layer before fully connected layer in ResNet50 just like in the case of PCA. Visualizations also contain example images and colours of points distinguish between locations in the same graph.

Holesov

In the case of Holesov dataset no visible clusters were formed in the case of t-SNE in Figure 4.6. Some smaller pseudo-clusters were formed on the outside of the big cluster but those are likely outliers like darker images. Images that contain sky fully covered in clouds appear close to each other on the bottom right part of the visualization. On the other hand, images containing blue sky with partial cloud coverage appeared further away from each other compared to PCA in Figure 4.4 where they appeared closer to each other. Similarly, images containing sky fully

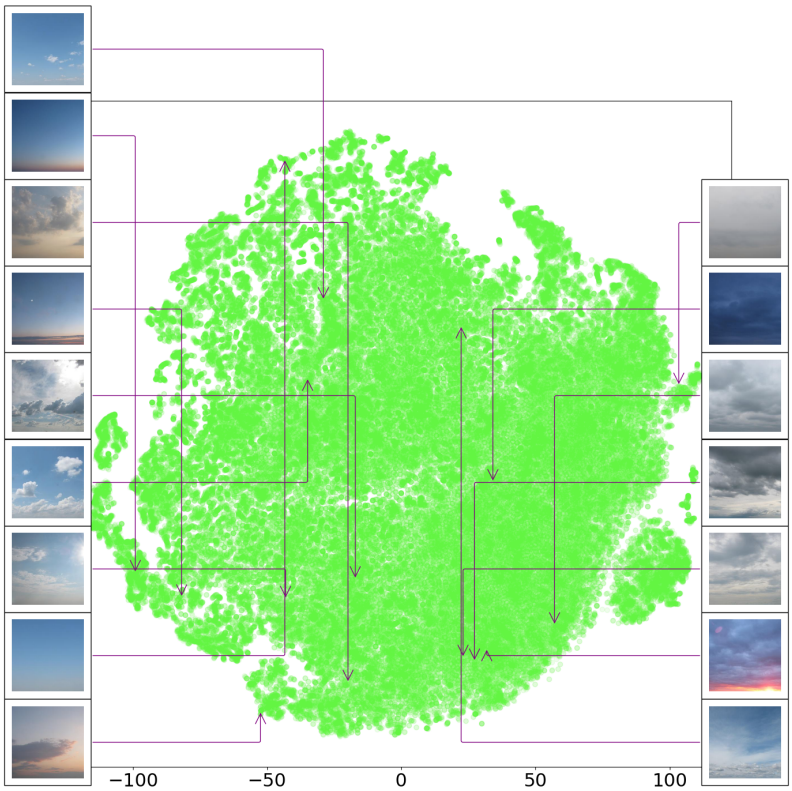


Figure 4.6: t-SNE on Holesov dataset for MoCo.

covered in clouds appear close to each other in PCA.

Prague+Brno

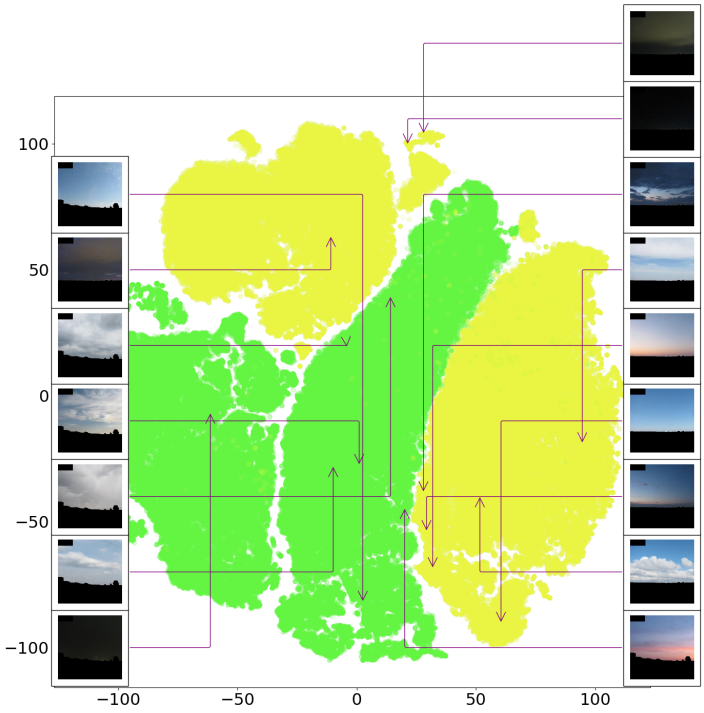
The case for Prague+Brno is more interesting as some clusters were formed in Figure 4.7. However, the model almost perfectly separated each location into two clusters. The clusters for Prague are separated by a big cluster of mostly images from Brno and there is an additional cluster next to the smaller Prague cluster. Moreover, the two clusters for each location contain daytime and night images respectively.

4.1.4 Nearest neighbours

For this test 5 distinct example images were taken and top 100 images according to cosine similarity were found for each query image. Example images are clear blue sky, one big fluffy cloud in the sky, one big gray rainy cloud covering whole sky, sunset image, blue sky with small fluffy clouds. The top 100 images were visually evaluated by their relevance to query.

Holesov

Blue sky with small fluffy clouds: 3 of the 100 nearest neighbours were irrelevant to the query. **Sunsent image**: Only 10 of 100 images was found



Yellow is Prague and green is Brno.

Figure 4.7: t-SNE on Prague+Brno dataset for MoCo.

irrelevant containing clear blue sky. **One big fluffy cloud in blue sky**: 9 out of 100 images were found irrelevant. Most of the irrelevant images showed a big fluffly cloud covering the whole sky and some of them showed a big fluffy rainy cloud covering the whole sky. **One big gray rainy cloud covering whole sky**: All of the images were relevant. **Clear blue sky**: All of the 100 closest images were relevant. Examples of query and its 5 closest neighbours can be found in Figure 4.8 and outlier histograms are found in Figure 4.9.

Prague+Brno

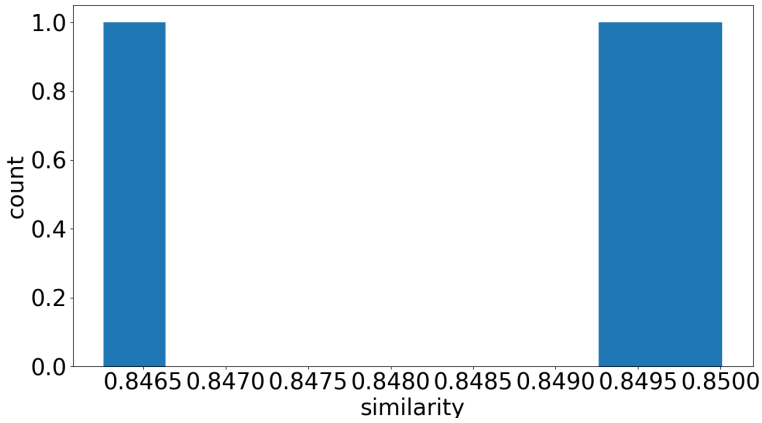
Blue sky with small fluffy clouds: 3 out of 100 were irrelevant. **Sunsent image**: All of the images were relevant. **One big fluffy cloud in blue sky**: 39 out of 100 images were found irrelevant, most of which were big gray clouds covering the whole sky. **One big gray rainy cloud covering whole sky**: All of the images were relevant. **Clear blue sky**: All of the images were relevant. Examples of query images and its 5 nearest neighbours are in Figure 4.10 and histogram of outliers is in Figure 4.11. One thing to note is that MoCo was consistent on this dataset with searching closest neighbours by having images from the same location close to each other, therefore all the results for nearest neighbours were from the same location.

4.1.5 Discussion

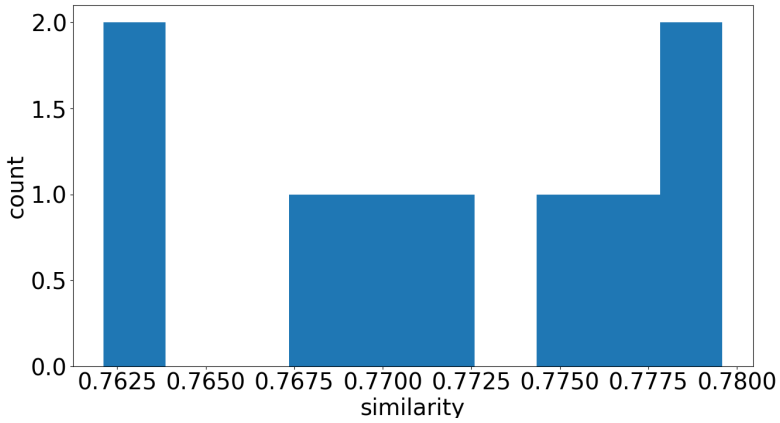
MoCo feature extraction model shows promising results when it comes to grouping similar images into representation space. Searching for nearest neighbours

The leftmost image in each row is query image and 5 images to the right are the nearest neighbours, leftmost being the closest. Queries are in the following order: sunset, blue sky with fluffy clouds, clear sky, one gray cloud, one big fluffy cloud.

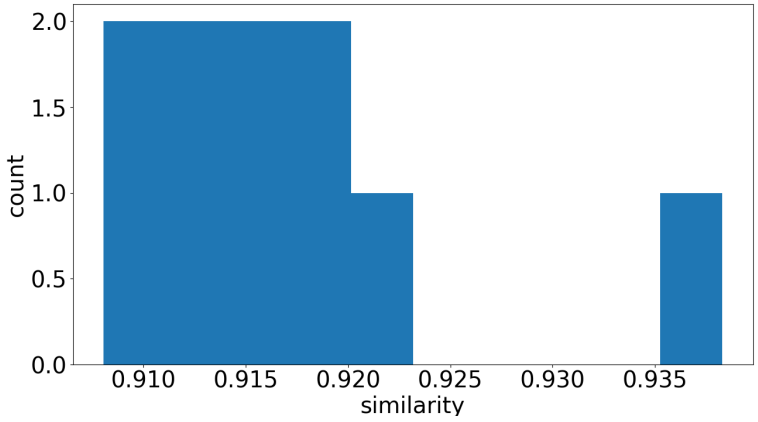
Figure 4.8: Queries and their 5 nearest neighbours for MoCo on Holesov dataset.



(a) Blue sky with small fluffy clouds.



(b) One big fluffy cloud.

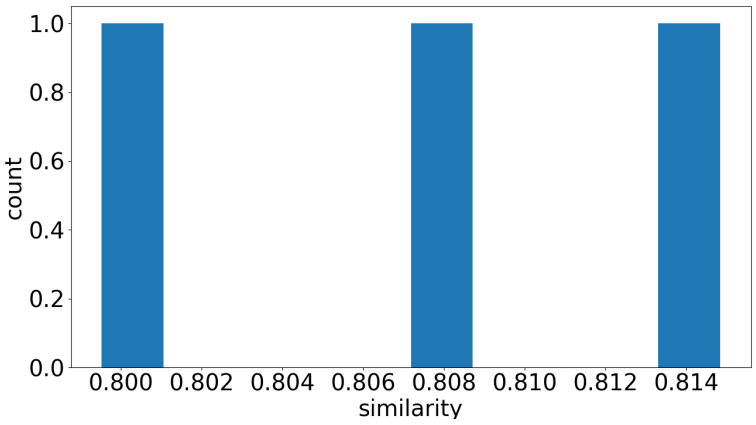


(c) Sunset.

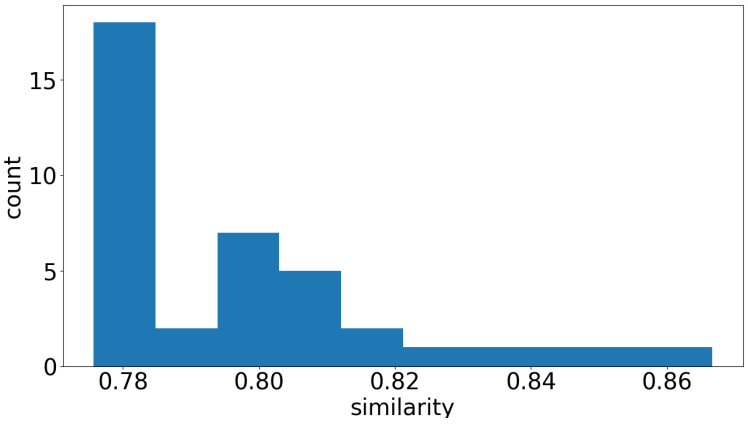
Figure 4.9: Outlier histograms for MoCo on Holesov dataset.

The leftmost image in each row is query image and 5 images to the right are the nearest neighbours, leftmost being the closest. Queries are in the following order: sunset, blue sky with fluffy clouds, clear sky, one gray cloud, one big fluffy cloud.

Figure 4.10: Queries and their 5 nearest neighbours for MoCo on Prague+Brno dataset.

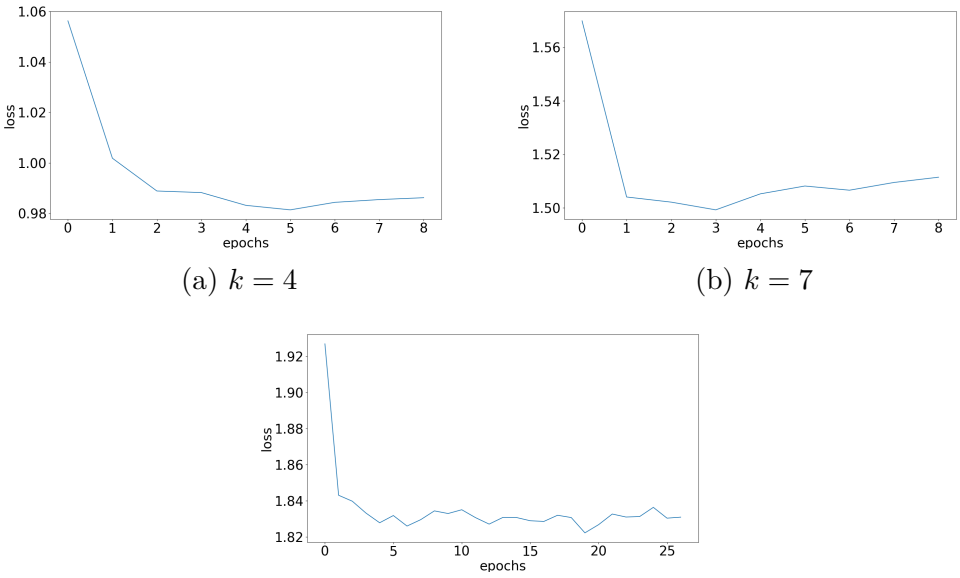


(a) Blue sky with small fluffy clouds.



(b) One big fluffy cloud.

Figure 4.11: Outlier histograms for MoCo on Prague+Brno dataset.



(a) $k = 4$ (b) $k = 7$

(c) $k = 10$

Figure 4.12: Training loss for MoCo+SPICE on Holesov dataset.

showed consistent results with the cloud coverage on top 100 nearest neighbours with only minimal number of irrelevant images found. PCA and t-SNE also show this by putting similar images close to each other in the visualizations. However, we recognize that the visualizations did not show the formation of visible clusters which might cause problems with clustering these features.

4.2 Original SPICE pipeline

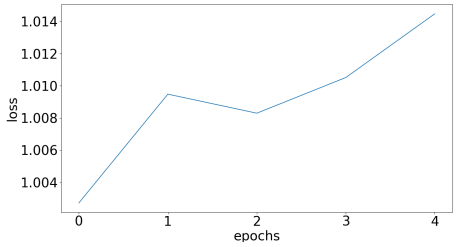
The original SPICE pipeline was trained as described in Niu et al. [2022]. Feature extraction model MoCo was trained first on both datasets separately and its training performance were described in previous section. After the models for both datasets were trained, clustering heads for both feature extraction models were trained on top of frozen feature extraction models. Lastly, the clustering heads were used to infer classes for the whole training dataset and the resulting clusters were analyzed visually and using Silhouette score.

4.2.1 SPICE Training

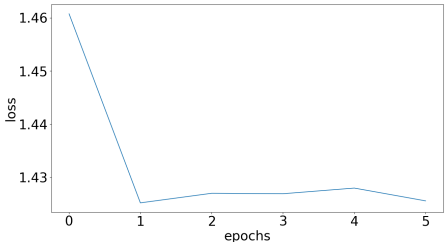
After MoCo was trained, its parameters were frozen and the last fully-connected layer was removed. Therefore, the output of the last convolutional layer of size 2048 was taken as feature representation. This servers as an input to 2-layer MLP which serves as clustering head for clustering the image representations. The training losses were plotted on single run.

Training of SPICE for Holesov was stable and the loss usually converged after 5 iterations which can be observed in Figure 4.12.

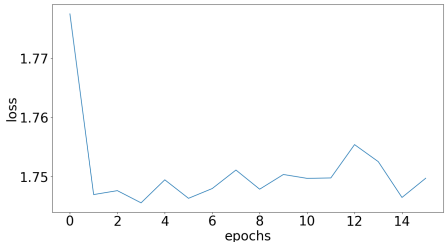
In case of Prague+Brno dataset training of SPICE was a lot more unstable where for $k = 4$ the loss did not converge at all and for the rest the loss jumped between values. Training performance can be found in Figure 4.13.



(a) $k = 4$



(b) $k = 7$



(c) $k = 10$

Figure 4.13: Training loss for MoCo+SPICE on Prague+Brno dataset.

4.2.2 Clustering evaluation

After SPICE training the feature extraction model and SPICE clustering head were used to infer cluster labels for each image from the training dataset. For both MoCo and SPICE the model from last training epochs were taken. For each sample the predicted most probable class was taken as a result of the clustering.

Silhouette score

Clustering head for Holesov dataset was trained with the same hyperparameters but with different number of classes to classify to. Then, the resulting clusterings were taken and their Silhouette score was computed in Table 4.1. The score shows very poor clustering performance where the score does not exceed 0.25 indicating that the clusters were poorly separated.

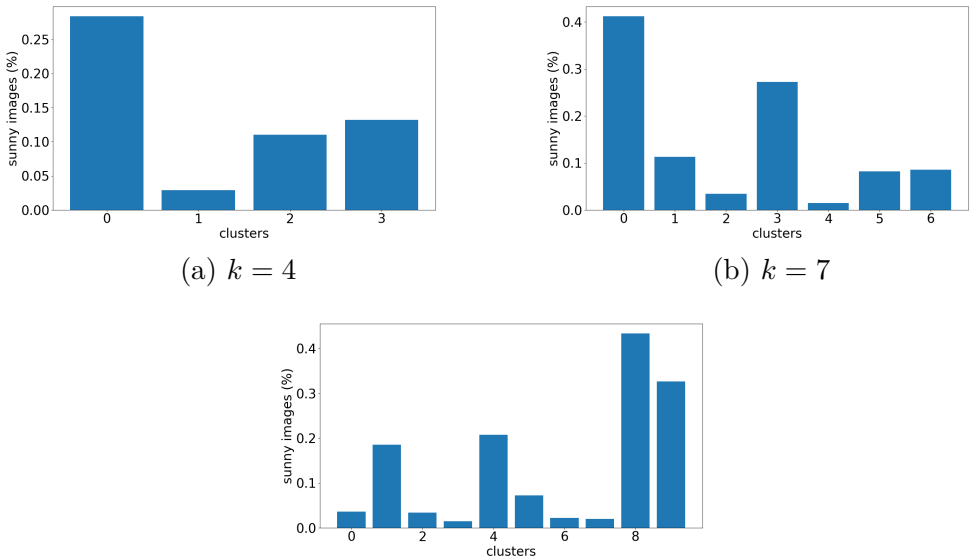
k	Silhouette coefficient
4	0.0414
7	0.045
10	0.05

Table 4.1: Silhouette score for SPICE on Holesov dataset.

k	Silhouette coefficient
4	0.016
7	0.009
10	0.052

Table 4.2: Silhouette score for SPICE on Prague+Brno dataset.

In case of Prague+Brno dataset similar performance was observed. Surprisingly, even though t-SNE showed at least 4 clusters, the Silhouette score for 4 clusters was only 0.016 indicating worse clustering performance than the same model trained on Holesov dataset. Computed scores are included in Table 4.2. Again, the higher k, the higher Silhouette coefficient, however, even for $k = 10$ the coefficient is very low.



(a) $k = 4$ (b) $k = 7$

(c) $k = 10$

Figure 4.14: Sunny image distributions for MoCo+SPICE on Holesov dataset.

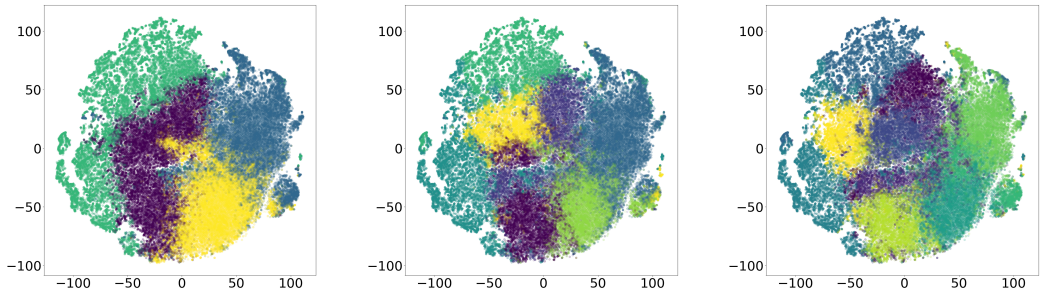
Visual inspection

Furthermore visual inspection revealed that for each k in models trained for Holesov dataset there was at least one cluster containing mostly images with sky fully covered in one gray cloud. Also, some clusters showed to prefer images containing the Sun and some preferred images without the Sun. Therefore, simple Sun detector was implemented by thresholding pixels to only the ones with values $(255, 255, 255)$ and using Hough Circle Detector Duda and Hart [1972] from OpenCV to detect if the high brightness pixels for a circle. If they do, the image is flagged as sunny. For $k = 4$ one cluster seemed to favor sunny images the most with 30% of images flagged as sunny. For $k = 7$ two clusters favored sunny images with one of them containing 40% and the second one containing 30% of sunny images. In case of $k = 10$ one of the clusters contained 45% and one with 35% of sunny images. Sunny image distributions can be found in Figure 4.14.

Visually most interpretable clusters were found with $k = 7$. Clusters 0 and 6 contained mostly images of white fluffy clouds in blue sky. Cluster 1 tends to prefer clouds with smoother texture. In clusters 2 and 5 mostly big gray rainy clouds were found. Cluster 3 preferred images from morning and evening containing the Sun and mostly clear sky, although many outliers were also present. Cluster 4 contained mostly images with clear or almost clear blue sky. Examples of these images obtained by random sampling each cluster can be found in Figure 4.15.

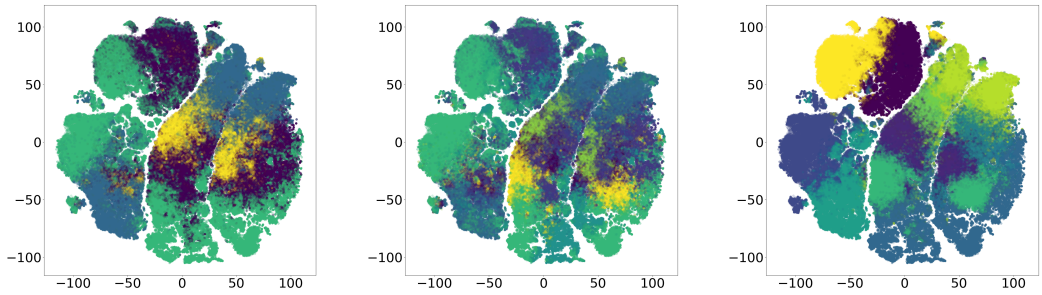
In the case of Prague+Brno dataset, clusters with similar characteristics as for Holesov dataset were observed. However, all clusters contained outliers in the form of completely dark image from night. This suggests that the model did not learn find the cluster containing dark images and did not separate the from the others, which will be discussed further in later section.

Figure 4.15: Examples of clustering results for MoCo+SPICE on Holesov for $k = 7$. Each class has 5 examples on rows, the clusters are ordered from 0 to 6 in top-down order.



(a) $k = 4$ (b) $k = 7$ (c) $k = 10$

Figure 4.16: Clustering of MoCo+SPICE on Holesov visualized using tSNE.



(a) $k = 4$ (b) $k = 7$ (c) $k = 10$

Figure 4.17: Clustering of MoCo+SPICE on Prague+Brno visualized using tSNE.

PCA and t-SNE

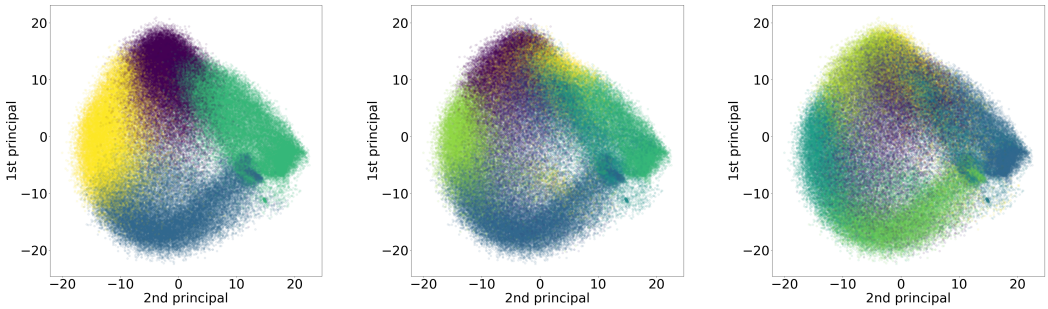
The clustering of the embedding space is visualized in Figures 4.16 and 4.17 for t-SNE and in Figures 4.18 and 4.19 for PCA. Figure 4.16a shows that the two visual clusters formed in t-SNE were split into two sub-clusters by SPICE. For higher number of clusters SPICE splits the clusters even more.

Figure 4.17 shows that for Prague+Brno dataset SPICE did not separate the locations even though they formed clusters in tSNE.

Figure 4.18a shows that the the embedding space was symmetrically separated but for higher cluster numbers the clustering stops being so symmetric.

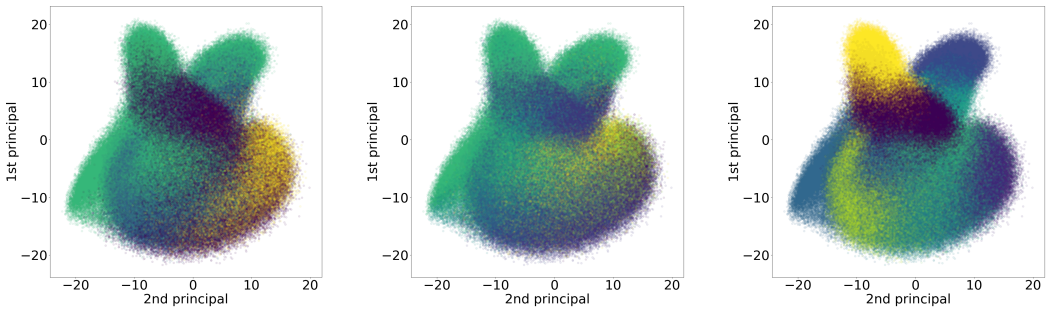
4.2.3 Discussion

Clustering using SPICE algorithm had very low Silhouette coefficient meaning that the clusters were poorly separated. PCA and t-SNE show that the clusters were overlapping and were not clearly separated. However, higher number of clusters had the tendency to have higher values of the coefficient which could mean that in order to achieve better clustering the number of clusters should be significantly higher. This might lead to better precision of the model, where we would have more clusters capturing the same amount of cloud coverage, however, fewer outliers would be present in those clusters. On the other hand, looking at t-SNE of both datasets we can see that for each number of clusters there were



(a) $k = 4$ (b) $k = 7$ (c) $k = 10$

Figure 4.18: Clustering of MoCo+SPICE on Holesov visualized using PCA.



(a) $k = 4$ (b) $k = 7$ (c) $k = 10$

Figure 4.19: Clustering of MoCo+SPICE on Prague+Brno visualized using PCA.

the same sections of the representation space that ended up in the same cluster implying that there must be some inner separation of the visual clusters formed in t-SNE.

4.3 MoCo+KMeans

For this experiment, pre-trained feature models from previous experiment were used and KMeans was fit as clustering algorithm on features extracted from input images.

4.3.1 KMeans training

KMeans was trained with *k-means++* cluster initialization setting. Tables 4.3 and 4.4 shows number of epochs the model was trained for and intertia on single run. Inertia is computed by measuring the distance between each data point and its corresponding cluster center, squaring this distance and summing these distances across all clusters.

Result \ k	4	7	10
Inertia	75048	69106	65400
Epochs	45	46	47

Table 4.3: Training results for KMeans with MoCo on Holesov dataset.

Result \ k	4	7	10
Inertia	307341	288324	276738
Epochs	24	35	116

Table 4.4: Training results for KMeans with MoCo on Prague+Brno dataset.

4.3.2 Clustering evaluation

Clustering result was evaluated in similar manner to previous experiment and the results of KMeans were compared to SPICE clustering results.

Silhouette score

KMeans trained on the same features from MoCo pre-trained in the previous experiment was run for different values of k to see how the algorithm clusters the representation space. KMeans achieved better Silhouette score than SPICE for clustering for both datasets which can be observed in Tables 4.5 and 4.6.

Visual inspection

For Holesov dataset and $k = 7$ had the most interesting results. Clusters 0 and 3 contained images of big gray rainy cloud covering the whole sky. Custer 1 contained many images of clear blue sky. Clusters 2 and 4 contained images with blue sky and fluffy clouds. Cluster 5 preferred clouds with softer texture and cluster 6 contained mostly images from early morning or late evening where the sky contained orange and yellow colours from the Sun. Examples of these clusters can be found in Figure 4.20.

k	Silhouette coefficient
4	0.056
7	0.075
10	0.083

Table 4.5: Silhouette score for KMeans with MoCo on Holesov dataset.

k	Silhouette coefficient
4	0.065
7	0.067
10	0.065

Table 4.6: Silhouette score for KMeans with MoCo on Prague+Brno dataset.

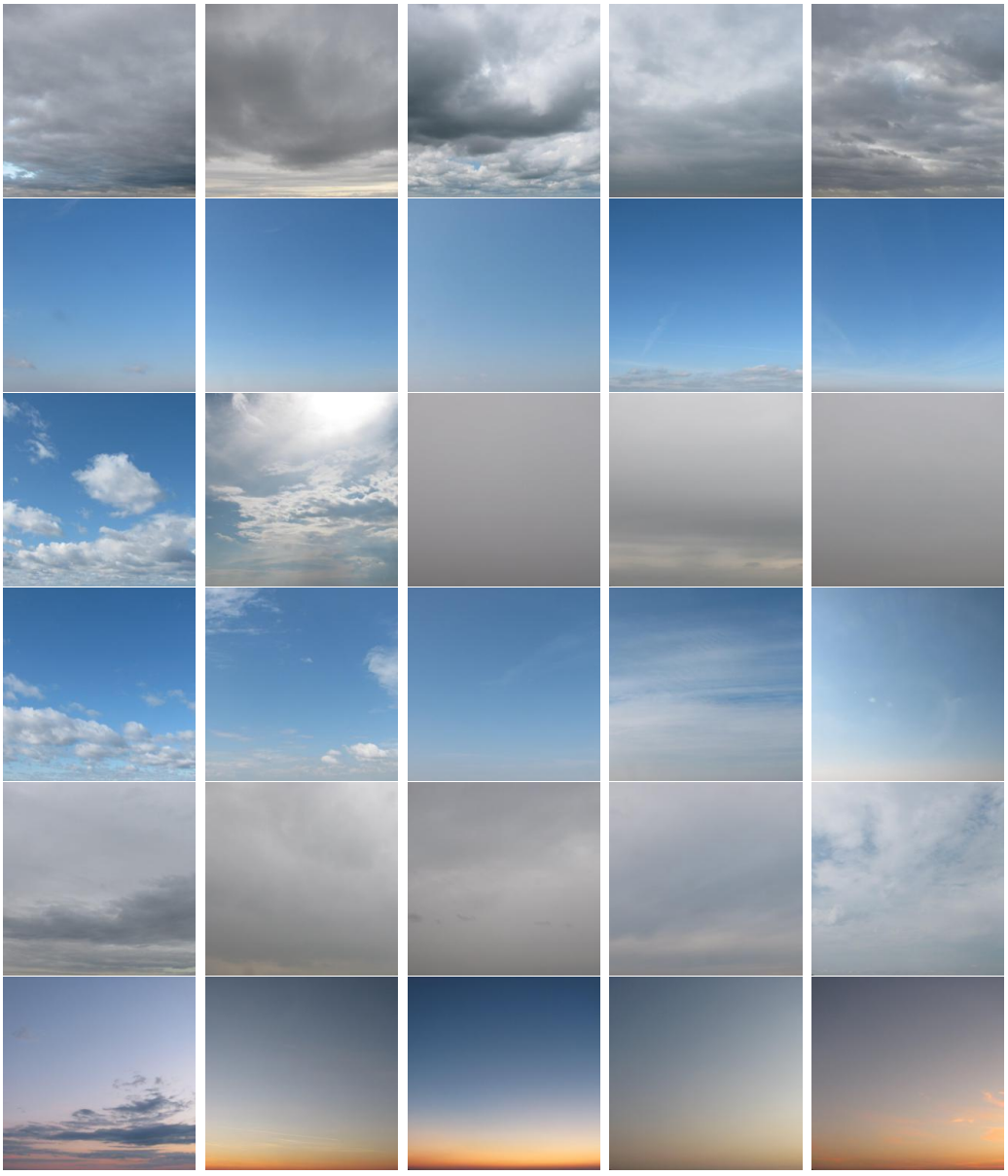
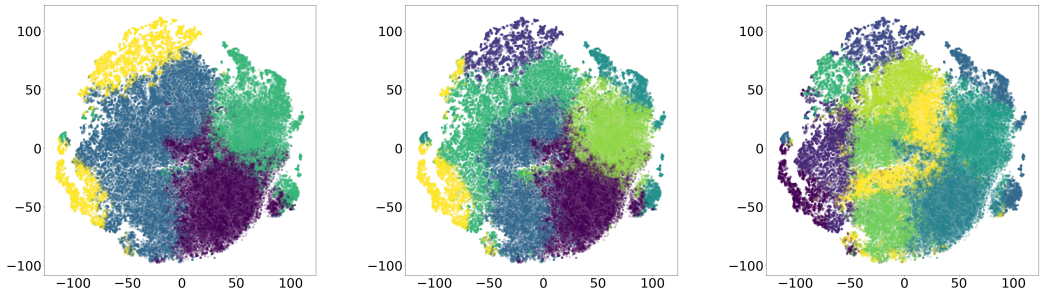
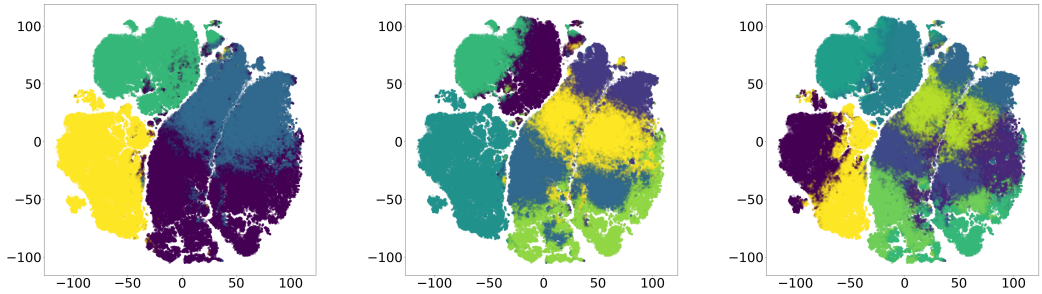


Figure 4.20: Examples of clustering results for MoCo+KMeans on Holesov for $k = 7$. Each class has 5 examples on rows, the clusters are ordered from 0 to 6 in top-down order.



(a) $k = 4$ (b) $k = 7$ (c) $k = 10$

Figure 4.21: MoCo+KMeans clustering of embedding space of Holesov visualized using tSNE.



(a) $k = 4$ (b) $k = 7$ (c) $k = 10$

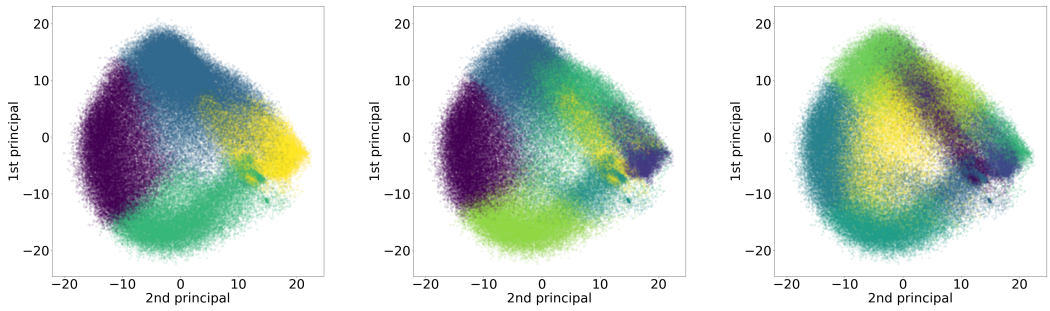
Figure 4.22: MoCo+KMeans clustering of embedding space of Prague+Brno visualized using tSNE.

PCA and t-SNE

Surprisingly and similarly to SPICE clustering KMeans did not distinguish between different locations for Prague and Brno whose clusters are visible in t-SNE in Figure 4.22. On the other hand, for $k = 4$ it can be observed from Figure 4.24 that the two outlier-like clusters were separated for all k but the data points in the main cluster were merged in many clusters.

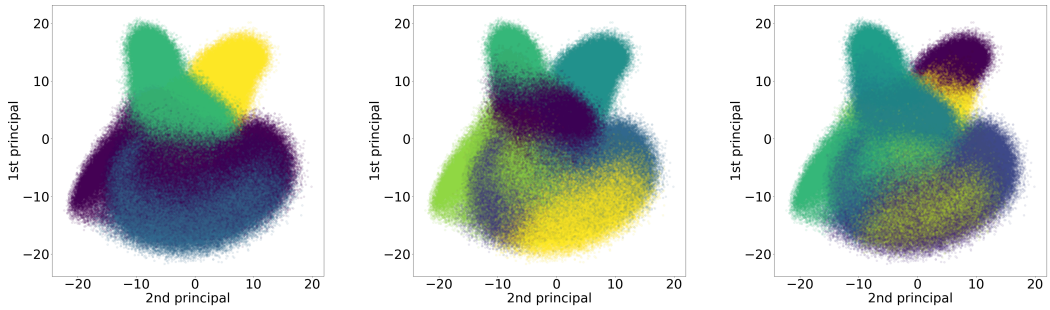
4.3.3 Discussion

KMeans clustering achieved a bit better Silhouette score compared to SPICE clustering. Similarly to SPICE, increasing the number of clusters lead to higher score at least for Holesov dataset. Visual inspection identified clusters that had visual interpretation on the amount of cloud coverage, however, many outliers were also present in the identified clusters. PCA and t-SNE reveal similar cluster distribution as in the case of SPICE clustering. Also, for each number of clusters the clusters visualized divided the representation space in a similar manner as did SPICE clustering.



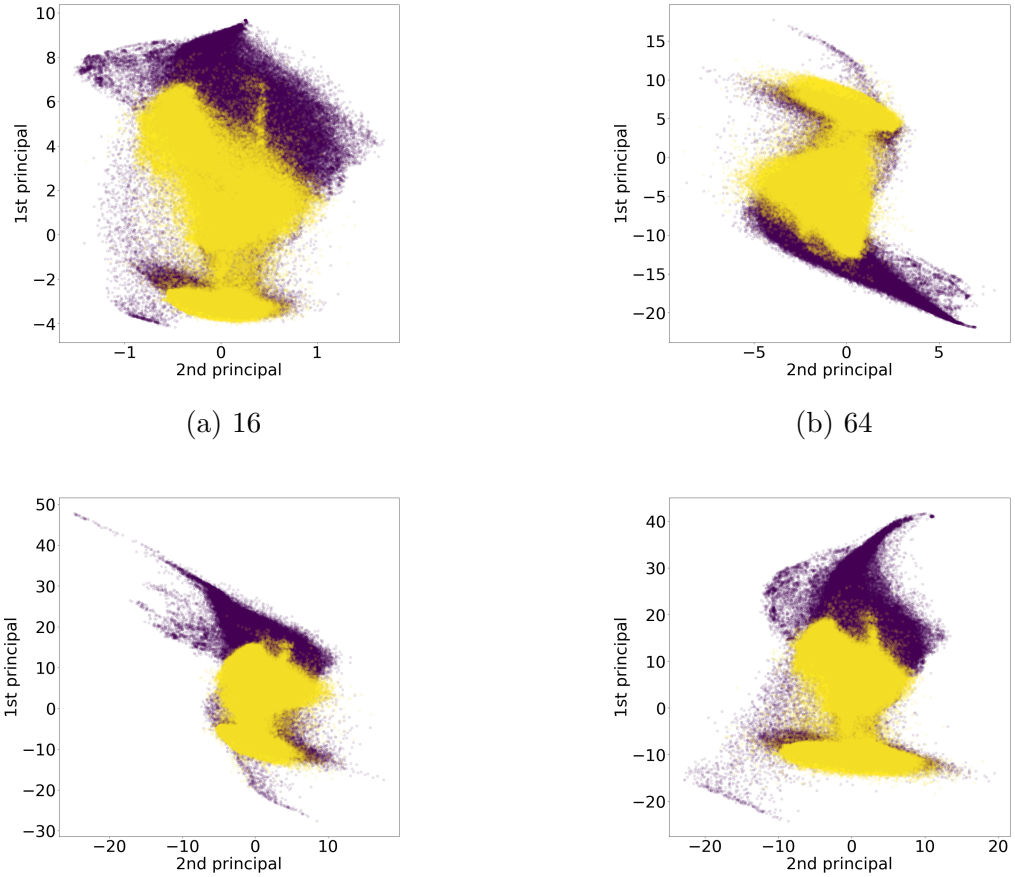
(a) $k = 4$ (b) $k = 7$ (c) $k = 10$

Figure 4.23: MoCo+KMeans clustering of embedding space of Holesov visualized using PCA.



(a) $k = 4$ (b) $k = 7$ (c) $k = 10$

Figure 4.24: MoCo+KMeans clustering of embedding space of Prague+Brno visualized using PCA.



(a) 16

(b) 64

(c) 128

(d) 256

Purple is Brno, yellow is Prague.

Figure 4.25: Comparison of bottlenecks for CAE on Prague+Brno.

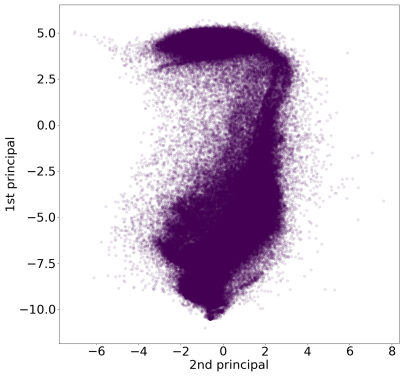
4.4 CAE analysis

This section focuses on CAE and its properties when it comes to image reconstruction and clustering. The image representations were visualized using PCA along with sample images and their nearest neighbours to better understand what the model focuses on when embedding an image.

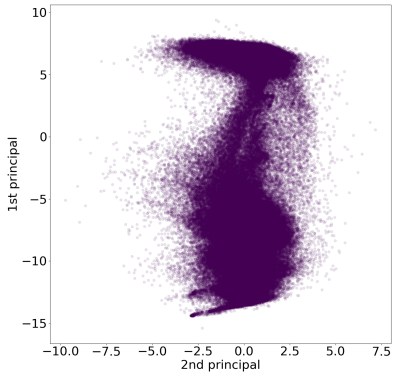
4.4.1 Grayscale images

CAE has shown to focus on the mask as it is unique for each location in and also it takes a big portion of the picture in Prague+Brno dataset. Therefore, the training loss was modified to only consider pixels that are outside of the mask for Prague+Brno dataset. However, this change did not help the autoencoder to distinguish between the clouds in the sky. Figure 4.25 shows PCA on bottleneck of sizes 16, 64, 128 and 256. It can be observed that Prague and Brno merged together in the PCA but but still no obvious clusters were formed. In visualization in Figure 4.25b and 4.25c the shape formed by data points is the same but just rotated.

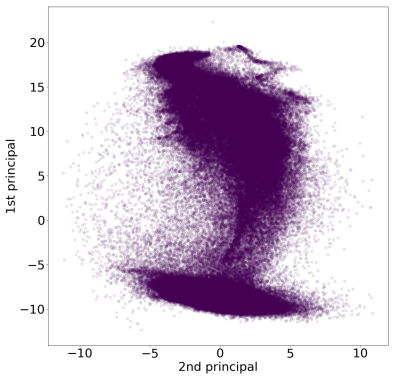
On the dataset from Holesov the result is a lot more consistent across bot-



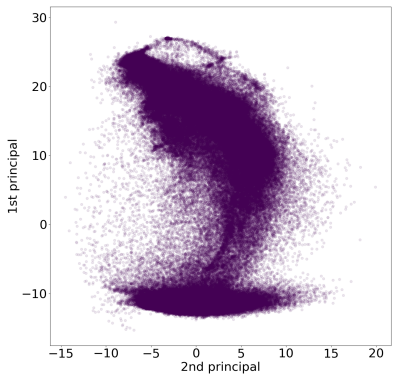
(a) 32



(b) 64

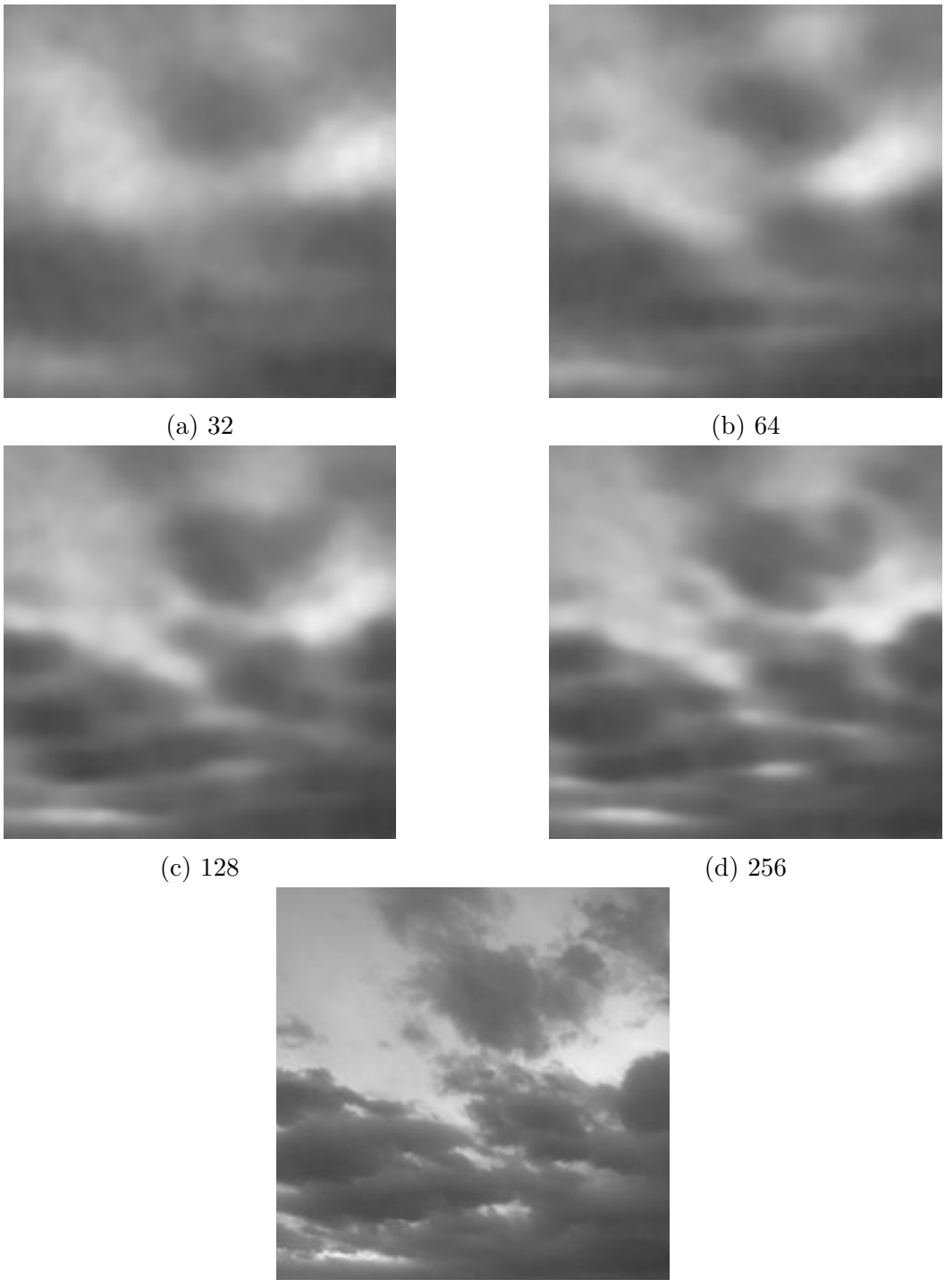


(c) 128



(d) 256

Figure 4.26: Comparison of bottlenecks for autoencoder on Holesov.



(a) 32

(b) 64

(c) 128

(d) 256

(e) Original

Figure 4.27: Comparison of the final reconstructions for autoencoder on Holesov dataset.

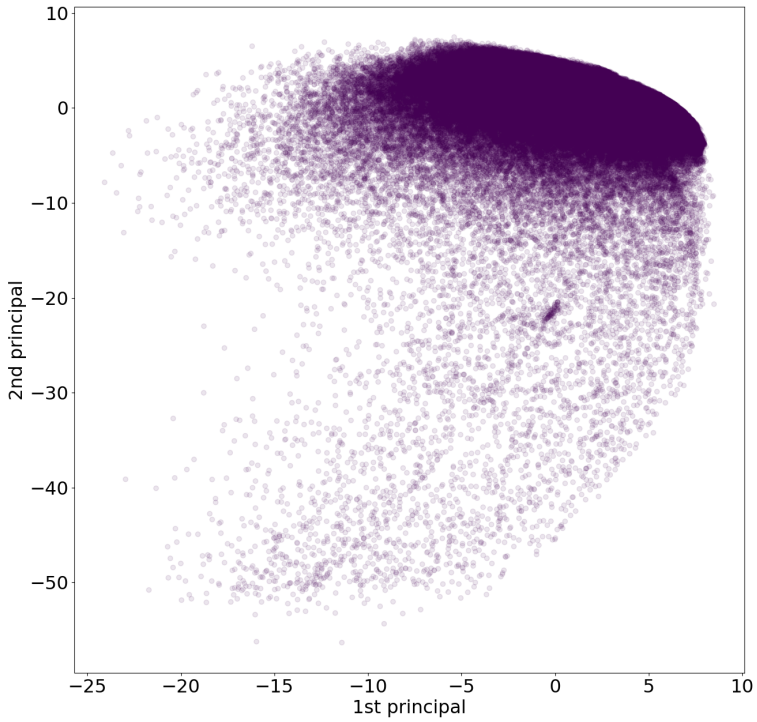


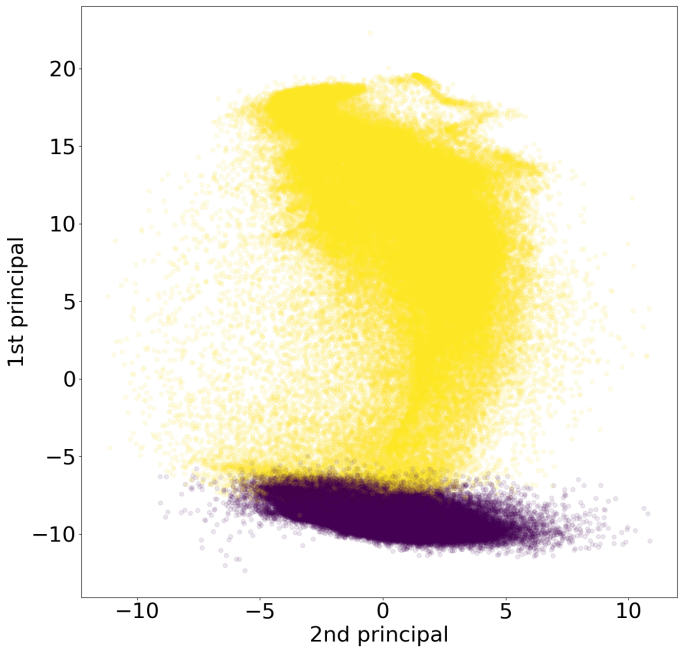
Figure 4.28: PCA of BW Holesov filtered dataset on autoencoder.

tlenecks. The shape the data points form is very similar for all bottleneck sizes disregarding rotation. Figure 4.26 shows comparison of these visualizations. It can be observed that most points are concentrated into one cluster but there is also a significant number of outliers which are far away from the cluster. One thing that images laying outside of the cluster had in common was that they all contained visually high contrast images typically ones from evening when the sky is still visible but the sun is about to set. Therefore, the mean brightness of these images was calculated and they were filtered out in preprocessing step and the autoencoder was retrained on the new filtered dataset. Resulting PCA shows only one big cluster of daytime images, other than that no visible clusters were formed in Figure 4.28. The final image reconstructions can be found in Figure 4.27.

4.4.2 RGB images

RGB images are supposed to supply the model with additional information about the clouds located in the pictures by providing colour data instead of just providing the brightness levels for each pixel. However, even this change did not help the model to learn additional information that would help the model to distinguish the clouds in the sky. Figure 4.29 shows PCA on the training dataset. Again, it can be observed that darker pictures (indicated in yellow) form a huge outlier-like cluster and all daytime images (purple colour) form one big cluster.

Daytime image cluster can be inspected more by applying stricter filter to



Yellow points are images with average brightness under a threshold, the rest is purple.

Figure 4.29: PCA of RGB Holesov dataset on autoencoder.

the dataset based on image brightness. Therefore, images from between 08:00:00 and 22:00:00 were kept only if their average grayscale brightness was at least 130. The model was later re-trained on this filtered data and the resulting PCA can be found in Figure 4.30 only for bottleneck of size 128 in both cases. The additional filtering and re-training did not help the model to form any visible clusters and all the images still form the same one big cluster.

4.4.3 Representations of images from certain timespan

The cloud coverage in one day may vary as well as it may vary throughout multiple days. To understand the image embeddings more, PCA was done only on 5 consequent days and the images projected in PCA were inspected in Figure 4.31. From the visualization it can be observed that similar images appear close to each other. Therefore, the autoencoder learns a representation that is good for image reconstruction but is not suitable for clustering of the images. Images that are similar in colour end up close to each other in the latent space which does not necessarily mean that there is the same type of cloud in the picture.

4.4.4 Additional edge data

To battle the effect that was discussed in the previous subsection the autoencoder is once again trained on the filtered Holesov dataset but this time a new channel containing edge data is added to the input of the model. Therefore, the size

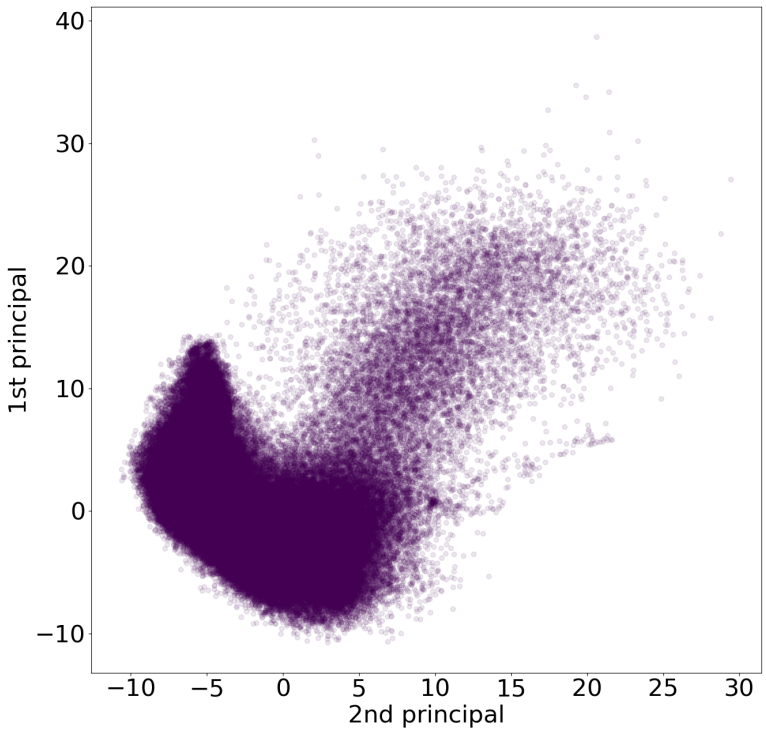


Figure 4.30: PCA on filtered RGB Holesov dataset.

of the input is $(N, 4, 256, 256)$ where N is the batch size and there are 3 RGB channels and one edge channel. The edge channel consists of output of applying Sobel edge detector to the image as discussed in Kanopoulos et al. [1988]. The resulting magnitudes can either be filtered by a threshold, where small magnitudes are completely forgotten and magnitudes bigger than the threshold are amplified, or they can be all normalized and considered as an alternative image to the input image. In this work, the magnitudes were normalized into interval $[0, 1]$ and used as an additional input channel of an image.

CAE with bottleneck 128 was then trained for 50 iterations with training performance over single run plotted in Figure 4.33 and the resulting reconstructions of the input image and its edges can be found in Figure 4.32. The image reconstruction lost some detail but the clouds are still quite visible in the image. Edge reconstruction has lower sharpness than original but still the cloud shapes were recognized by the model.

PCA in Figure 4.34 was computed on the whole dataset but the projections were made only on images from the first 5 days. The visualization reveals that some similar looking clouds were in fact close together in the projection.

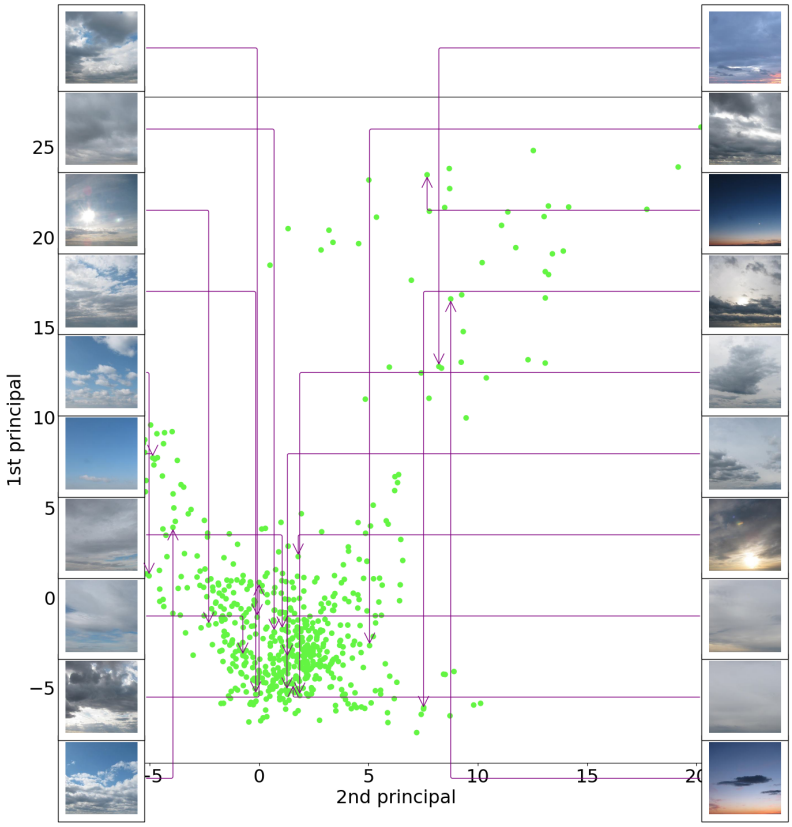
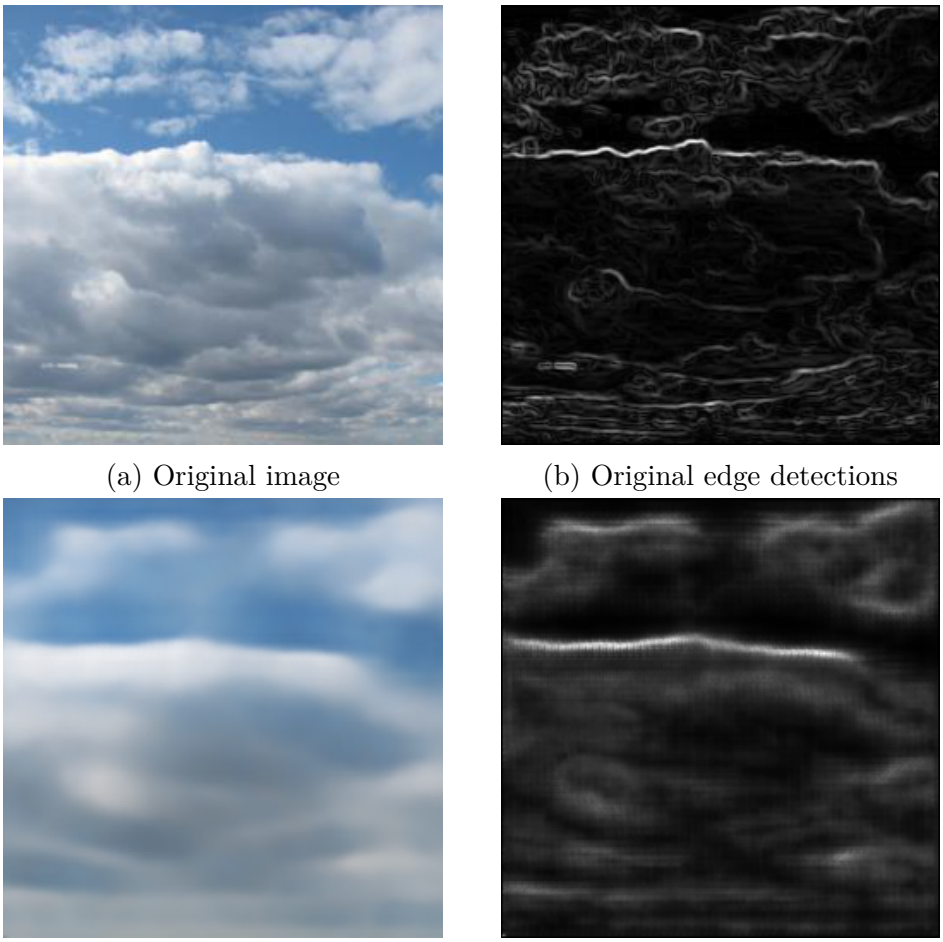


Figure 4.31: PCA of CAE on 5 consecutive days from RGB filtered Holesov dataset.



(a) Original image (b) Original edge detections

(c) Reconstructed image (d) Reconstructed edge detections

Figure 4.32: Image reconstruction of autoencoder on RGB Holesov dataset.

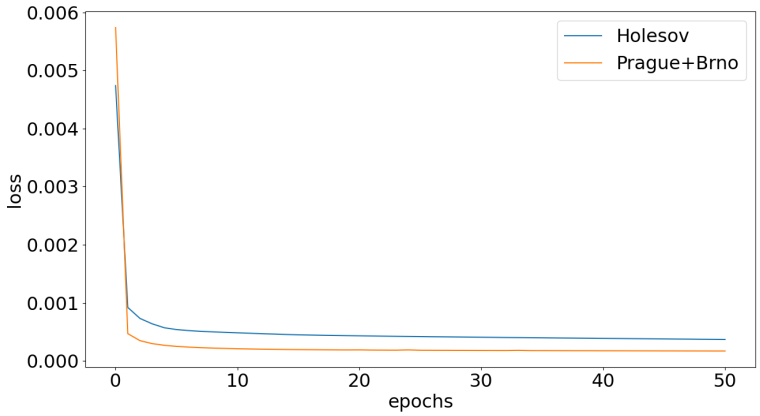


Figure 4.33: CAE training loss on RGB Holesov with edge data.

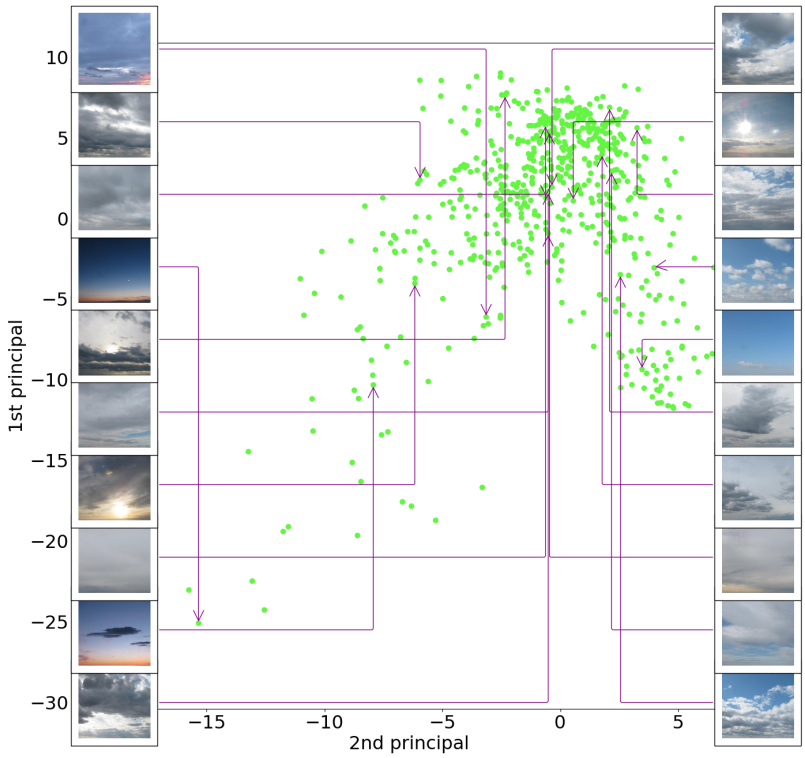


Figure 4.34: PCA of autoencoder embeddings of the first 5 days.

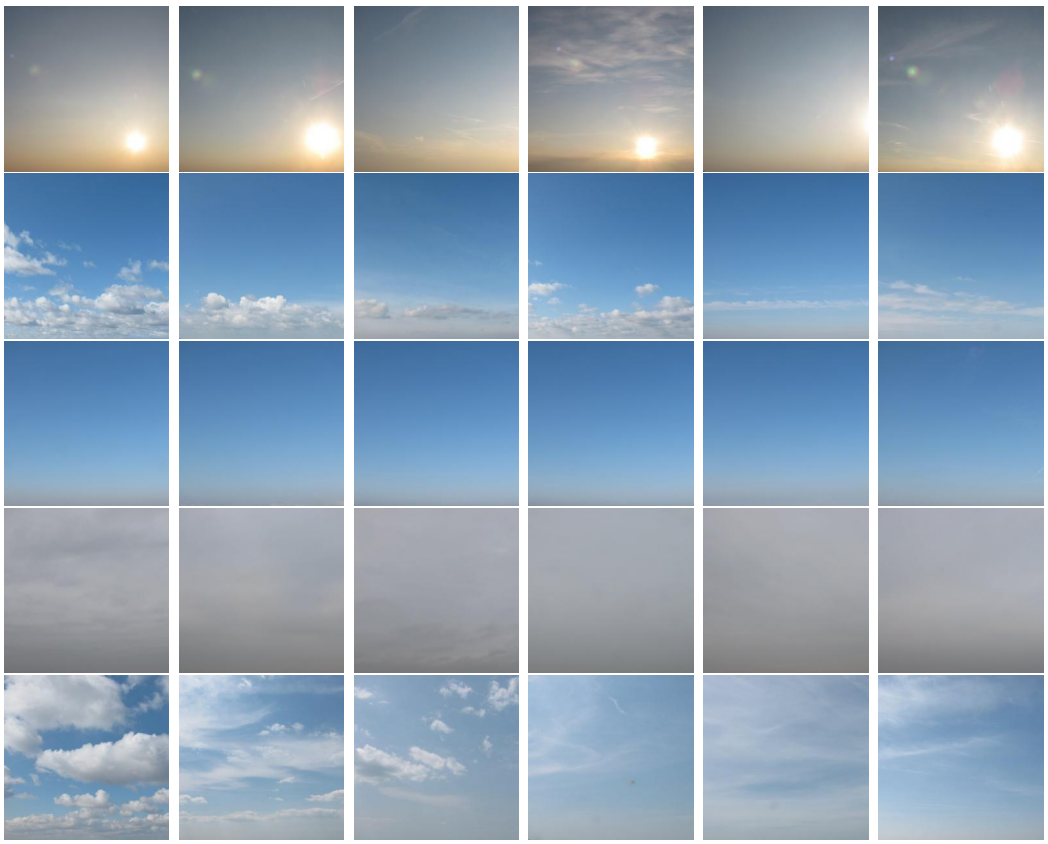
4.4.5 Nearest neighbours

Holesov

Blue sky with small fluffy clouds: 5 out of 100 were irrelevant. **Sunsent image**: 16 out of 100 images contained small soft clouds. **One big fluffy cloud in blue sky**: 16 out of 100 images were irrelevant, they were mostly images containing one big gray rainy cloud and no fluffy clouds. **One big gray rainy cloud covering whole sky**: All of the images were relevant. **Clear blue sky**: All images were relevant. Examples of query and its 5 closest neighbours can be found in Figure 4.35 and outiler histograms can be found in Figure 4.36.

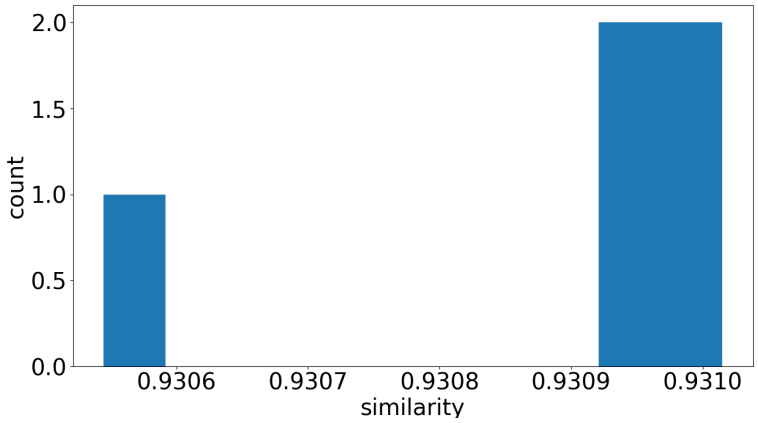
Prauge+Brno

Blue sky with small fluffy clouds: 22 out of 100 were irrelevant most of which contained images with softer clouds in a blue sky. **Sunsent image**: 21 out of 100 were irrelevant. Interestingly, the visual colour distribution of the images was similar for all 100 images. **One big fluffy cloud in blue sky**: 31 out of 100 images were irrelevant, mostly containing soft clouds covering the whole sky. **One big gray rainy cloud covering whole sky**: All of the images were relevant. **Clear blue sky**: All of the images were relevant. Examples can be found in Figure 4.37 and outlier histograms can be found in Figure 4.38. Similarly as for MoCo, CAE was consistent with searching closest neighbours for a particular location where all the results were from the same location.

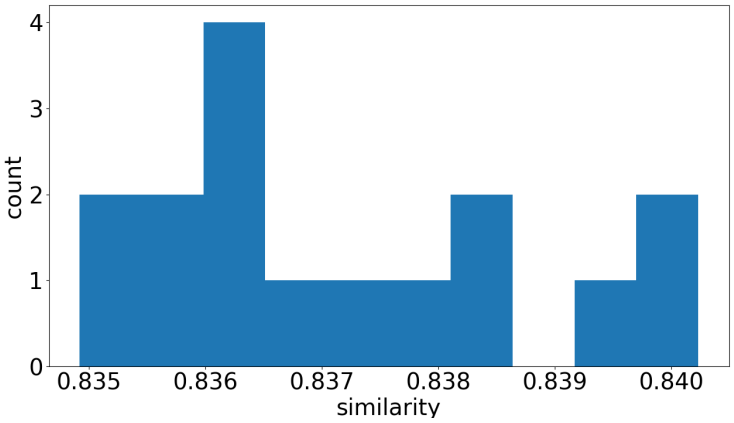


The leftmost image in each row is query image and 5 images to the right are the nearest neighbours, leftmost being the closest. Queries are in the following order: sunset, blue sky with fluffy clouds, clear sky, one gray cloud, one big fluffy cloud.

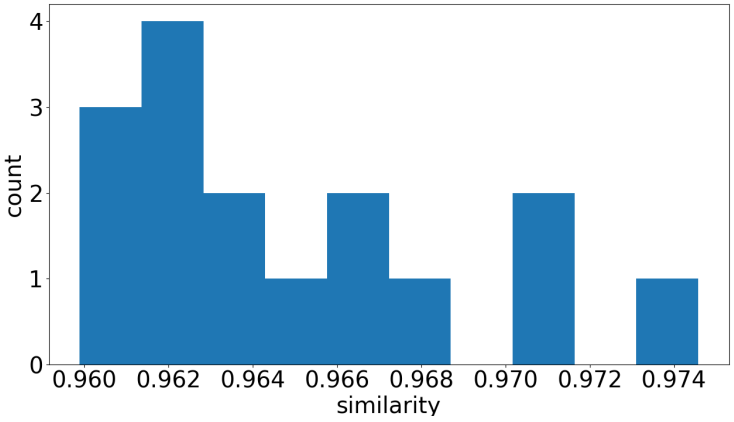
Figure 4.35: Queries and their 5 nearest neighbours for CAE on Holesov dataset.



(a) Blue sky with small fluffy clouds.



(b) One big fluffy cloud.

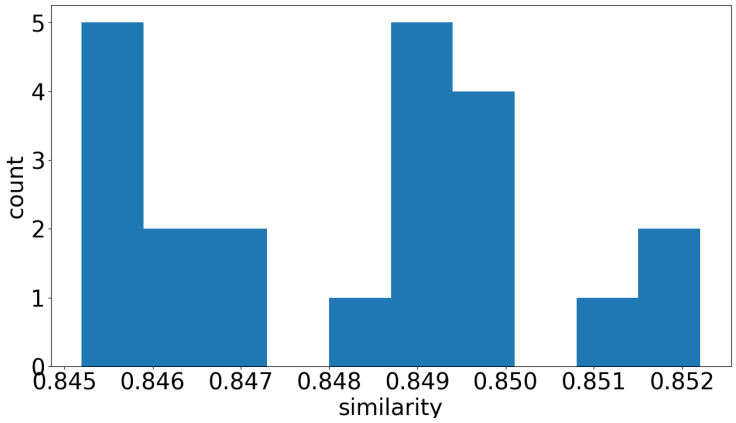


(c) Sunset.

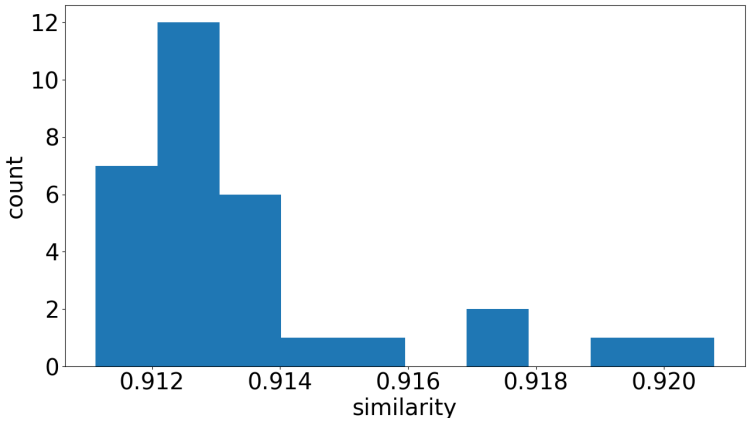
Figure 4.36: Outlier histograms for CAE on Holesov dataset.

The leftmost image in each row is query image and 5 images to the right are the nearest neighbours, leftmost being the closest. Queries are in the following order: sunset, blue sky with fluffy clouds, clear sky, one gray cloud, one big fluffy cloud.

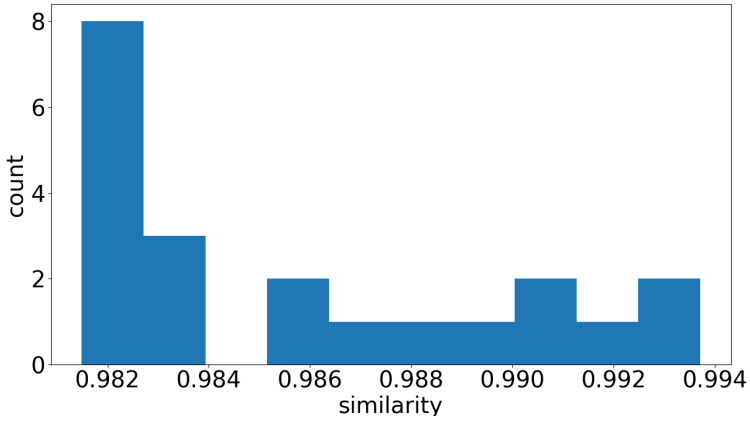
Figure 4.37: Queries and their 5 nearest neighbours for CAE on Prague+Brno dataset.



(a) Blue sky with small fluffy clouds.



(b) One big fluffy cloud.



(c) Sunset.

Figure 4.38: Outlier histograms for CAE on Prague+Brno dataset.

4.4.6 Discussion

CAE has shown to pay close attention to brightness of images and in case of RGB images also to their colour distribution. These different factors were projected into PCA as one cluster containing mainly daytime images and outliers outside of this big cluster containing evening or nighttime images.

Decreasing the bottleneck from 256 down to 16 has shown do not impact the distribution of data in the representation space as the shape of PCA projection is similar across all bottlenecks. Increasing the size of bottleneck has proven to result in higher quality reconstructions, however, achieving high quality of image reconstruction is not our goal.

Using RGB images has proven to be better choice instead of grayscale images because the model got more data to work with. Also adding edge data as another channel helped the model to find similar images. Therefore, in the next experiments, CAE is trained on RGB datasets with edge data.

The model did not perform as well as MoCo for nearest neighbours, for top 100 nearest neighbours many images were flagged as irrelevant and in some cases the model has completely mistaken images that were foundationally different.

4.5 CAE+SPICE

In the following experiments, the CAE was frozen as feature extraction model and SPICE clustering was trained on top of these frozen features. The training parameters for SPICE were used the same as for previous experiments where SPICE clustering was used.

4.5.1 SPICE Training

In the case of training SPICE on top of CAE features the training was very unstable and in case of $k = 7$ the training loss diverged. Training performance over single run can be found in Figure 4.39. In case for Prague+Brno dataset the loss diverged in all cases and therefore was not evaluated.

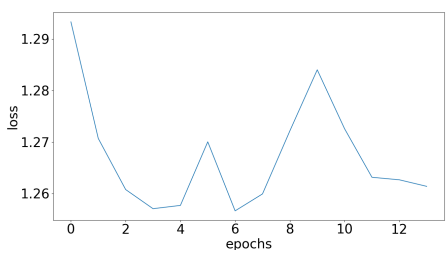
4.5.2 Clustering evaluation

Silhouette score

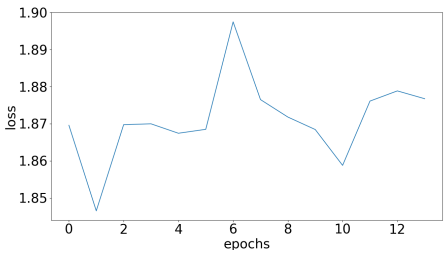
Silhouette score for this setup ended up still very low, with the highest score being 0.235 for $k = 4$ in Holesov dataset. With higher numbers of clusters the score lowers slightly. Overall, the values are higher than in previous experiments, however, they are not directly comaprable as different feature model was used for feature extraction. The results are in Table 4.7 only for Holesov dataset.

Visual inspection

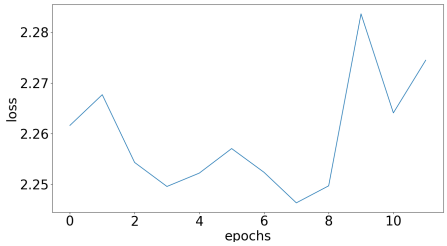
From visual inspection of this setup for all k parameters there was no obvious systematic clustering of the images. Each of the clusters appeared to contain images of various nature ranging from clear blue sky to fully covered sky by clouds for both datasets.



(a) $k = 4$



(b) $k = 7$

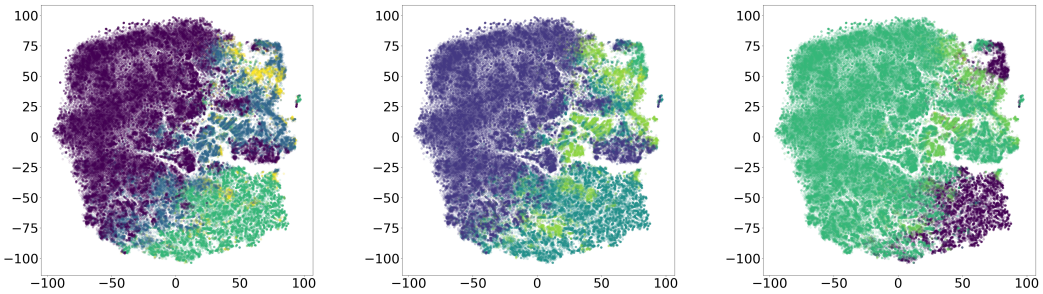


(c) $k = 10$

Figure 4.39: Training loss for SPICE with pretrained CAE on Holesov dataset.

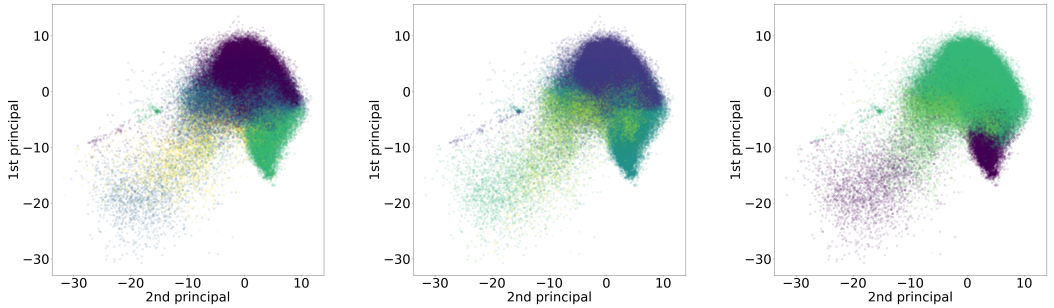
k	Silhouette coefficient
4	0.235
7	0.22
10	0.221

Table 4.7: Silhouette score for CAE+SPICE on Holesov dataset.



(a) $k = 4$ (b) $k = 7$ (c) $k = 10$

Figure 4.40: CAE+SPICE clustering of embedding space of Holesov visualized using tSNE.



(a) $k = 4$ (b) $k = 7$ (c) $k = 10$

Figure 4.41: CAE+SPICE clustering of embedding space of Holesov visualized using PCA.

PCA and tSNE

Both PCA in Figure 4.41 and t-SNE in Figure 4.40 show that SPICE clustered most of the images into one cluster and only a small amount of images into the rest of classes, therefore the clusters contain disproportional amount of images.

4.5.3 Discussion

SPICE training was difficult in this experiment setup. The training process was very unstable and in case of Prague+Brno dataset the loss did not converge at all. The reason for this is that SPICE uses data augmentation during its training process and it relies on the fact that the feature extraction model is invariant to this kind of data augmentation. MoCo was trained in this sense to be invariant towards data augmentation, however, this is not the case for CAE making it hard for the model to converge.

Silhouette score for Holesov dataset was quite high, however, this is due to most of the images being assigned to one cluster. For higher number of clusters most of the images ended up in 4 of the clusters, therefore PCA and t-SNE visualizations look very similar for all of the number of clusters and because of this reason they only contain 4 colours.

From visual inspection it was also obvious that most of the images appeared in one cluster and therefore the clusters contained a lot of variance in images.

4.6 CAE+KMeans

4.6.1 KMeans training

KMeans clustering was applied to features extracted from CAE which was pre-trained in the previous experiments. KMeans model had very high inertia after training, however, the reason for this is that there are many outliers outside of the main cluster whose distances are very high. Each of the models was trained under 300 iterations which is the maximum number of iterations set.

Result \ k	4	7	10
Inertia	14492326	11831869	10708672
Epochs	24	26	60

Table 4.8: Training results for KMeans with CAE on Holesov dataset.

4.6.2 Clustering evaluation

Silhouette score

Silhouette score in this setup was overall a little bit lower than the score for SPICE clustering, however, from the results of visual inspection in the next section and from PCA and tSNE visualizations the clustering is more interpretable with KMeans model than it is with SPICE clustering. However, the scores are still very low with values not even reaching 0.25. Similarly as for the previous experiment the scores are higher than for MoCo clustering experiments but the results are not directly comparable because different feature extraction model was used.

Visual inspection

In this setup CAE seemed to focus more on the brightness of an image and its colour composition rather than the objects contained in the images. This was especially visible for $k = 7$ where clusters 0, 4 and 5 contained daytime images, cluster 1 contained evening or morning images, cluster 2 contained images with

Result \ k	4	7	10
Inertia	116766416	82800200	69142840
Epochs	12	72	15

Table 4.9: Training results for KMeans with CAE on Prague+Brno dataset.

k	Silhouette coefficient
4	0.219
7	0.172
10	0.174

Table 4.10: Silhouette score for CAE+KMeans on Holesov dataset.

k	Silhouette coefficient
4	0.448
7	0.283
10	0.278

Table 4.11: Silhouette score for CAE+KMeans on Prague+Brno dataset.

gray rainy cloud and 6 contained similar images but from evenings and mornings. Cluster 3 contained mostly late evening images. Examples can be found in Figure 4.42.

PCA and t-SNE

From PCA in Figures 4.45 and 4.46 and t-SNE in Figures 4.43 and 4.44 better partitioning of representation space can be observed by the clustering algorithm. The outliers outside of the main cluster usually ended up in a separate cluster which resulted in one or multiple clusters containing mostly evening or morning images. The rest of the clusters split the main visual cluster into multiple clusters.

4.6.3 Discussion

Clustering with KMeans achieved significantly better results than clustering with SPICE. Silhouette score was significantly higher for Prague+Brno, especially for $k = 4$, and the score for Holesov was similar to SPICE score.

From visual inspection it was obvious that the model focuses more on brightness and colour distributions of the images instead of focusing on the objects in the images. Clusters were formed mostly according to brightness of the images where daytime and nighttime images got separated into separate clusters.

This fact is more obvious from t-SNE of Prague+Brno where the night clusters ended up in one or more clusters separate from all of the others.

4.7 CAE+KMeans on CIFAR10

In order to get some reference output on more diverse data AE was trained on *CIFAR10* dataset Krizhevsky, Nair, and Hinton [2009] and clustered using KMeans. This way it can be compared how the model behaves on more diverse data and on the cloud dataset.

4.7.1 PCA and t-SNE

Figure 4.47 shows representation space of CAE on *CIFAR10* dataset. No visible clusters were formed in the visualization and additionally it can be observed in Figure 4.47a that the ground truth is scattered all over the visualization and therefore similar images are not necessarily close to each other in the representation space. Similar observations apply for t-SNE in Figure 4.48.

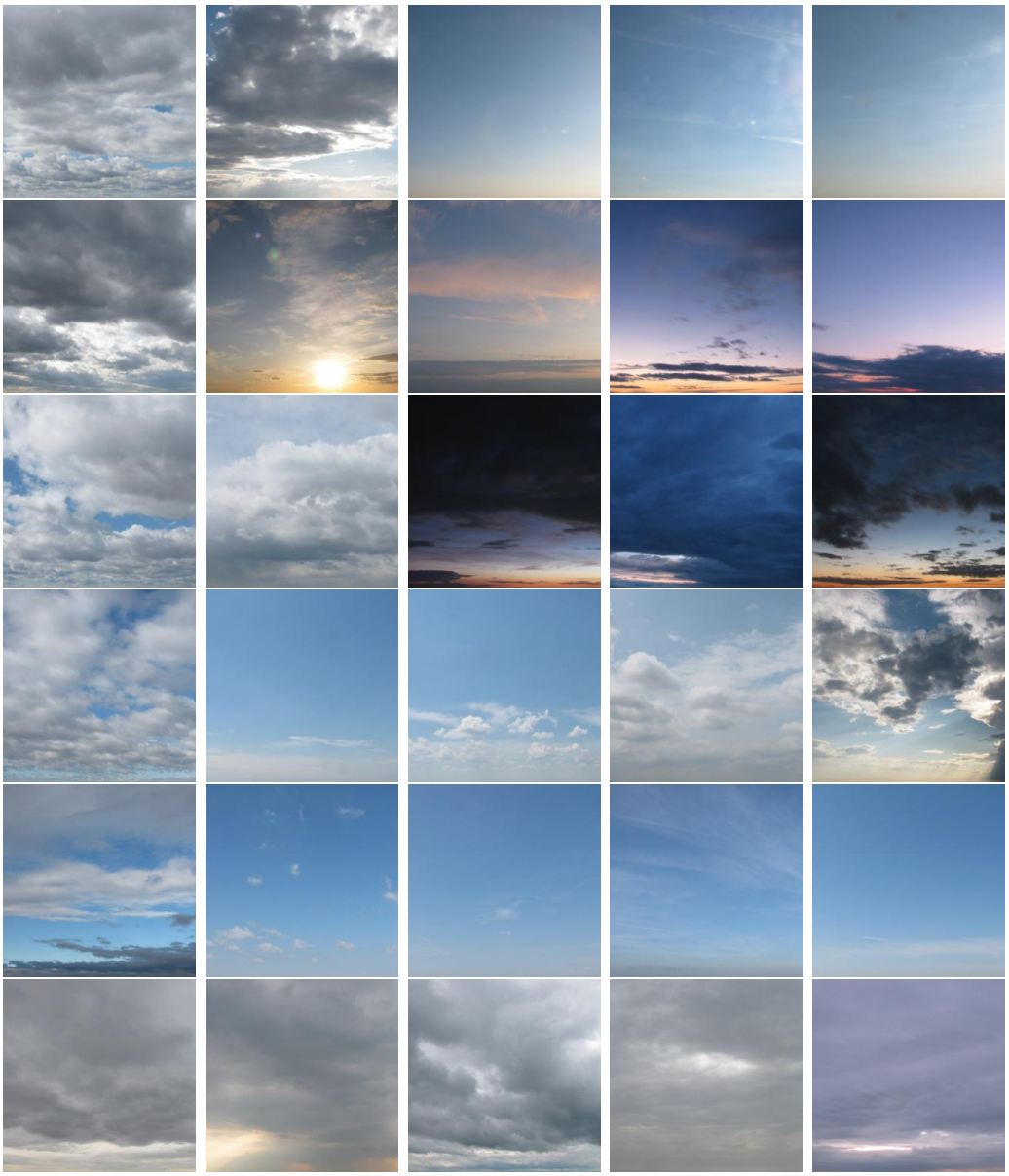
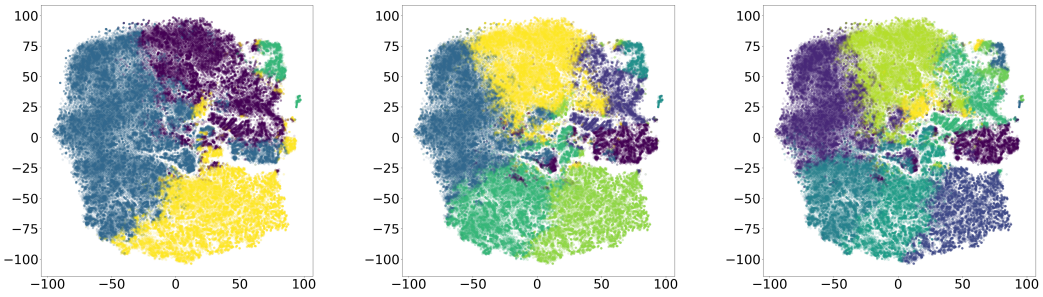
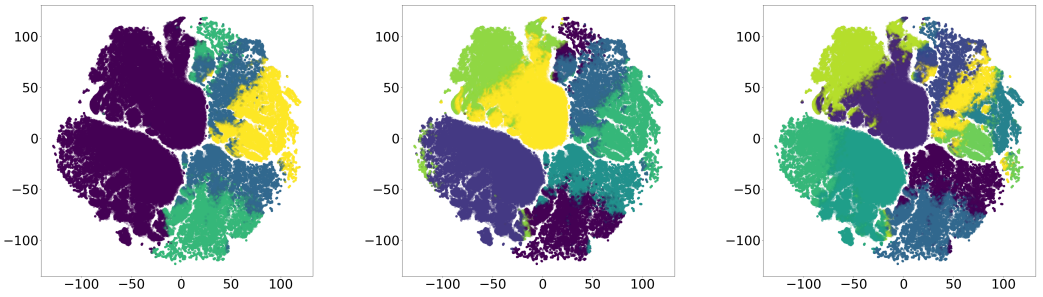


Figure 4.42: Examples of clustering results for CAE+KMeans on Holesov for $k = 7$. Each class has 5 examples on rows, the clusters are ordered from 0 to 6 in top-down order.



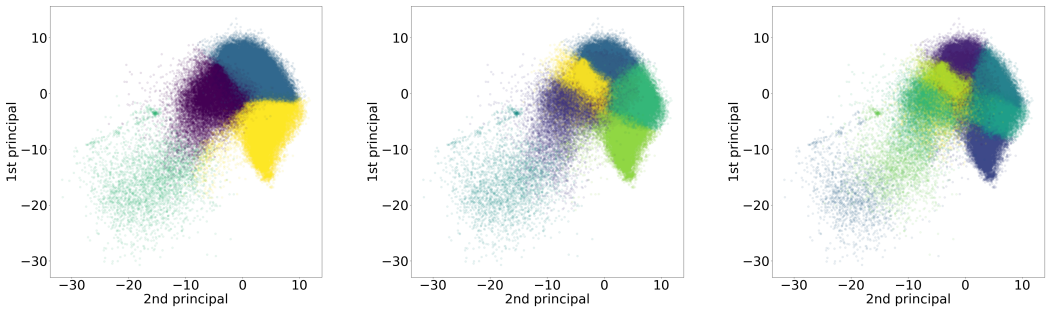
(a) $k = 4$ (b) $k = 7$ (c) $k = 10$

Figure 4.43: CAE+KMeans clustering of embedding space of Holesov visualized using tSNE.



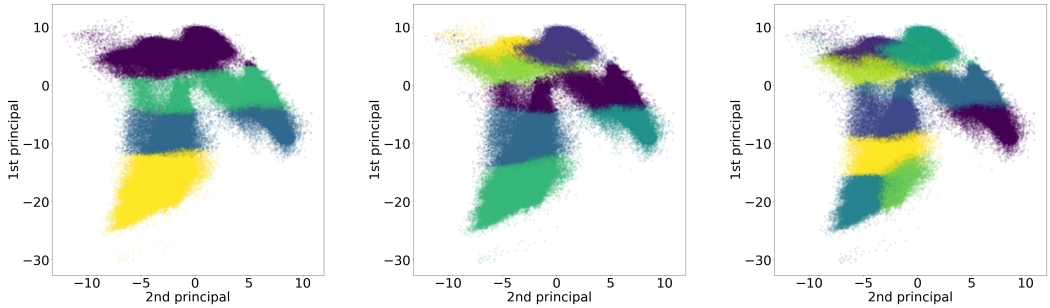
(a) $k = 4$ (b) $k = 7$ (c) $k = 10$

Figure 4.44: CAE+KMeans clustering of embedding space of Prague+Brno visualized using tSNE.



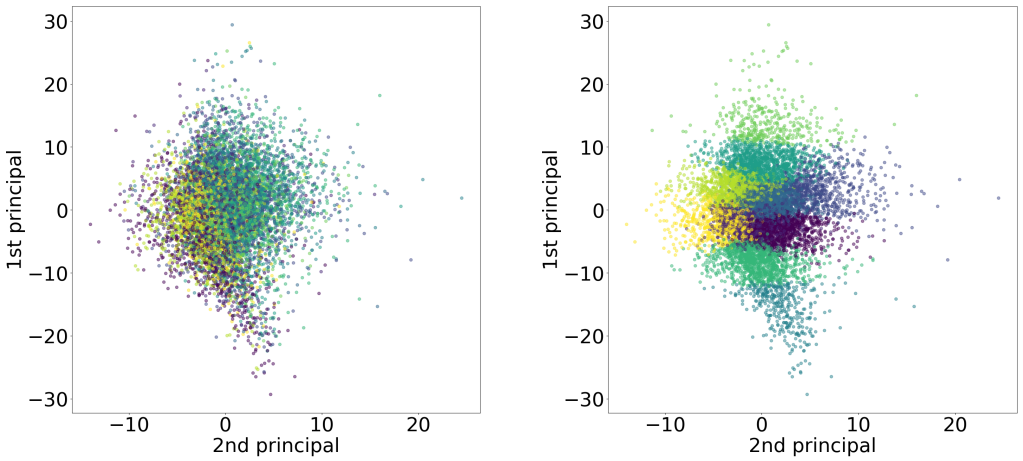
(a) $k = 4$ (b) $k = 7$ (c) $k = 10$

Figure 4.45: CAE+KMeans clustering of embedding space of Holesov visualized using PCA.



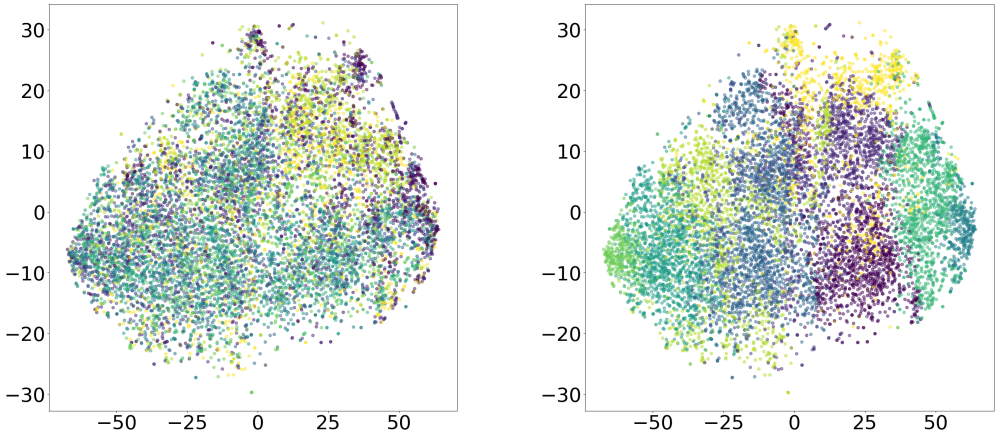
(a) $k = 4$ (b) $k = 7$ (c) $k = 10$

Figure 4.46: CAE+KMeans clustering of embedding space of Prague+Brno visualized using PCA.



(a) Ground truth. (b) Predictions.

Figure 4.47: PCA of CAE on CIFAR10



(a) Ground truth. (b) Predictions.

Figure 4.48: t-SNE of CAE on CIFAR10

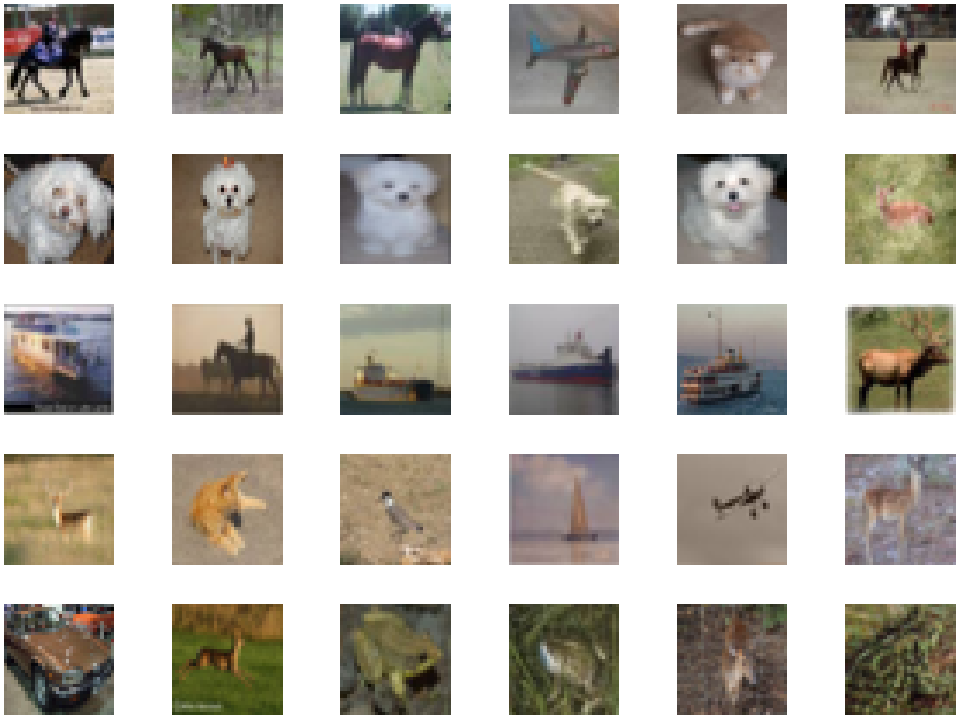


Figure 4.49: Sample images and their 5 nearest neighbours in feature space for CIFAR10 autoencoder. Leftmost image is original and the 5 images on the right are its nearest neighbours.

4.7.2 KMeans

KMeans run for 227 iterations until it converged, unsurprisingly after seeing the results from PCA, the resulting clusters were very inhomogeneous and the clustering appeared almost random. CIFAR10 is a labeled dataset, therefore metrics can be calculated for clustering evaluation.

ARI (Adjusted Rand Index)

Chosen metric for this case was ARI which is a metric that evaluates the agreement between two clustering results. It considers all pairs of samples and counts the pairs that are assigned to the same or different cluster. ARI has range $[-1, 1]$ where

- -1 is worse than random,
- 0 is agreement that is not better than random,
- 1 is a perfect agreement between clusters.

In this case, ARI of KMeans on frozen feature model was 0.043 meaning that the clustering was practically random.

4.7.3 Nearest neighbours

Figure 4.49 shows some sample images and their nearest neighbours according to their cosine similarity. For example, images of horses (first row), poodles (second

row) and ships (third row) were put very close to each other in the latent space whereas pictures of deers (second to last row) and cars (last row) were close to pictures not containing those objects. This is likely due to pictures having similar colour distributions, for example poodles are white and take up almost whole image, whereas horses are darker colour with green background.

4.7.4 Discussion

CAE+KMeans on *CIFAR10* has shown poor performance for clustering task. CAE could find some similar object using nearest neighbours but the representations were not suitable enough for clustering task because ARI was very close to 0 implying the clustering was no better than random clustering.

5. Implementation

This chapter describes the technical requirements needed for running the code used in this work and a short manual describing the code architecture.

5.1 Implementation requirements

Implementation was done using Python 3.9. for its popularity in machine learning. The libraries used in this work are the following:

- **Numpy 1.23.5.** For mathematical operations.
- **OpenCV 4.9.0.80** For implemented CV algorithms.
- **Pillow 9.4.0** Image operations.
- **Pytorch 2.0.0** Main machine learning framework.
- **Pytorch-CUDA 11.8** Cuda support for PyTorch.
- **Scikit-learn 1.2.2** Implemented machine learning algorithms.
- **Seaborn 0.12.2** Data visualization.
- **Torchvision 0.15.0** Image manipulation within PyTorch.

On hardware side the code need to run on a machine with CUDA support and at least $20GB$ of VRAM but optimally at least $50GB$ of VRAM.

5.2 Architecture

The root of the project contains several folders and files. There is *README.md* file which describes how to run the script in more detail. Then, folder **SPICE/** contains repository from https://github.com/niuchuangnn/SPICE/tree/main (accessed on 2023-03-01) with SPICE framework implementation [Niu et al., 2022]. Scripts were modified in order to adjust to new dataset with clouds and to be able to cluster features from CAE. Furthermore, **SPICE/configs** contains configuration files for scripts, **SPICE/moco** contains models and datasets for MoCo, **SPICE/spice** contains models and datasets for SPICE clustering and **SPICE/tools** contains scripts for running original SPICE framework.

Folder **autoencoders/** contains CAE and dataset implementation and scripts for training and testing CAE.

Folder **visualization/** contains scripts for generating plots and PCA and t-SNE visualizations.

5.3 Scripts

In this section main scripts required for training and their parameters are described.

5.3.1 Dataset preparation

Dataset is prepared using **cloud_converter.py**. Parameters for this script are:

- `--original-path` Path to root of dataset of images in their original form. File structure is `root_of_dataset/location/date_of_images/img.jpg`.
- `--save-path` Path where to save preprocessed images.
- `--filter-dark` Whether to filter out images whose brightness to lower than threshold.
- `--threshold` Brightness threshold.
- `--mask` Mask out images using supplied mask. Only available for Prague and Brno.

5.3.2 MoCo training

For MoCo training we use **SPICE/tools/train_moco.py**. It takes the following arguments:

- `--data_type` Type of dataset used. In this case we use "clouds".
- `--data` Path to root folder of the dataset.
- `--all` 0 if only train data is used, 1 if all data is used.
- `--img_size` Size of image after random crop.
- `--save_folder` Path to folder to save the model.
- `--save_freq` Frequency of saving the model.
- `--arch` The backbone architecture to be used. In our case it is "resnet50"
- `--workers` Number of worker processes for loading data.
- `--epochs` Number of epochs to train for.
- `--start-epoch` Number of epoch to start training from when resuming.
- `--batch-size` Size of batch.
- `--lr` Learning rate.
- `--schedule` Learning rate scheduler to drop learning rate by $10x$.
- `--momentum` Momentum for momentum update.
- `--wd` Weight decay.
- `--print-freq` Frequency of printing of training performance.
- `--resume` Path to the latest checkpoint.

- `--world-size` Number of nodes for distributed training.
- `--rank` Node rank for distributed training.
- `--dist-url` URL for distributed training.
- `--dist-backend` Backend for distributed training.
- `--seed` Random seed for training initialization.
- `--gpu` ID of GPU to be used.
- `--multiprocessing-distributed` Turns of distributed multiprocessing.
- `--moco-dim` Dimension of MoCo training space.
- `--moco-k` Size of queue for negative keys.
- `--moco-m` Momentum used in momentum update of key encoder.
- `--moco-t` Softmax temperature.
- `--mlp` Use MLP head on top of backbone instead of one fully connected layer.
- `--aug-plus` Use additional data augmentations.
- `--cos` Use cosine learning rate scheduler.

5.3.3 Feature extraction from MoCo

For feature extraction we use **SPICE/tools/pre_compute_embedding.py**. Only parameter for this script is `--config-file` where a configuration file from **SPICE/configs/clouds** is supplied. More information about configuration file can be found in **README.md**.

5.3.4 Training of CAE

For CAE training we use **autoencoders/train_autoencoder.py**. Parameters that need to be supplied to this script are:

- `--channels` Number of input channels for CAE.
- `--base-channels` Base number of channels after expansion.
- `--latent-dim` Size of bottleneck.
- `--batch-size` Batch size.
- `--epochs` Number of epochs to train for.
- `--result` Path to store result.
- `--dataset` Path to root folder of dataset.

- `--edge` Add edge detection as another input channel.
- `--mask` Use mask on the input images to mask out unnecessary parts of image.

5.3.5 Feature extraction from CAE

For feature extraction from CAE we use **autoencoders/embedding.py**. Parameters for this script are the following:

- `--channels` Number of input channels for autoencoder.
- `--latent-dimension` Size of bottleneck.
- `--dataset` Path to root folder of dataset.
- `--model` Path to saved model.
- `--result` Path to save the result.
- `--batch-size` Batch size.
- `--edge` Include edge detection data as another input channel.
- `--mask` Use mask for input images. Only Prague and Brno are supported.

5.3.6 Training of SPICE clustering

To train SPICE we use script **SPICE/tools/train_self_v2.py**. Parameter `--config-file` takes the path to a configuration file located in **SPICE/configs/clouds** that are prepared for each dataset and feature extraction model. For description of configuration file read **README.md** in the attached code.

5.3.7 Training of KMeans

Training of KMeans is performed using script **kmeans_cluster.py**. Parameters for this script are:

- `--embeddings` Path to exported embeddings from feature extraction model.
- `--result` Path to save trained model and predictions.
- `--k` Number of clusters.

5.3.8 Image scores for SPICE

To extract embeddings for MoCo we use **SPICE/tools/local_consistency.py**. First parameter for this script is `--config-file` which is configuration file from **SPICE/configs/clouds** and was prepared for each dataset. For more information about configuration file refer to **README.md**. Second parameter is `--embedding` which is path to file containing pre-computed image features from feature extraction model.

5.3.9 Clustering of images

Final clustering is performed using **classificator.py**. It takes the following arguments:

- `--scores-path` Path to file containing class probabilities for each clustered sample.
- `--predictions-path` Path to file containing class predictions for each clustered sample.
- `--data-path` Path to dataset that is being clustered.
- `--save-path` Path for clustered images to be saved.
- `--save-images` Turns on saving images into folders in `--save-path` based on their clusters.
- `--max-images` Maximum number of images per cluster.

5.4 Workflow

The high-level overview of the workflow from training to clustering is the following:

1. Train feature extraction model (CAE/MoCo).
2. Extract features from the feature extraction model into a file using corresponding script.
3. Train clustering head on top of these features (SPICE/KMeans).
4. (Only for SPICE) Save scores of images.
5. Save clustered images into folders corresponding to their clusters.

Conclusion

In this work, we classified images of clouds using original SPICE framework and then swapped its components to see their impact on the result of classification.

For feature extraction model we used MoCo and CAE, where MoCo showed more promising results in the extracted representations. Similar images were close in the representation space and also the clusters formed after classification similar parts of embedding space to be in the same cluster regardless of method used. CAE tends to focus more on other image factors such as their brightness and colour distribution. However, PCA and t-SNE of representations extracted by both CAE and MoCo did not show any clusters forming besides clusters separating different image locations or the time of day they were taken.

For clustering we used SPICE clustering and KMeans. Silhouette coefficient revealed better clustering performance of KMeans on features extracted by both MoCo and CAE regardless of dataset. Visually, the setup MoCo+SPICE had the most interpretable clusters out of all setups. However, these clusters also contained a lot of outliers where the model would mistake clouds for a sky of certain colour.

Therefore, the system could be used as querying system, where MoCo would be used as feature extraction model and user would manually search for a query image and the system would return a specified number of similar images as results. In order to achieve the original goal a better feature extraction method needs to be used because the methods used in this work did not fully extract all the characteristics needed in order to perform classification task.

Future works

While we experimented with various feature extraction and clustering methods in this work some things remain unexplored:

1. **Different number of clusters**: The experiments in this work only used 4, 7 and 10 number of clusters, however, higher number of clusters showed higher Silhouette coefficient in our experiments. Therefore, higher number of clusters could yield more clusters containing the same type of images but with fewer outliers.

2. **Different feature extraction method**: Different feature extraction could result in better representation of images in the feature space which would be more suitable for clustering. Good results on clouds specifically were achieved by Caron et al. [2021] by using ViT for feature extraction.

3. **Transfer learning**: Authors of Dematties et al. [2023] achieved good results with very similar setup to our but using a labelled dataset. Therefore, the model could be pre-trained on the labelled dataset and then the trained model could be used on our dataset for inference.

4. **Utilizing pre-trained models**: Pre-trained models such as ResNet50 pre-trained on *ImageNet* could be used for feature extraction, therefore frozen features from ResNet would be clustered.

Bibliography

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jegou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9630–9640, 2021. doi: 10.1109/ICCV48922.2021.00951.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/chen20j.html`.

Dario Dematties, Bhupendra A. Raut, Seongha Park, Robert C. Jackson, Sean Shahkarami, Yongho Kim, Rajesh Sankaran, Pete Beckman, Scott M. Collis, and Nicola Ferrier. Let's unleash the network judgment: A self-supervised approach for cloud image analysis. *Artificial Intelligence for the Earth Systems*, 2(2):220063, 2023. doi: 10.1175/AIES-D-22-0063.1. URL `https://journals.ametsoc.org/view/journals/aies/2/2/AIES-D-22-0063.1.xml`.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=YicbFdNTTy`.

Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, jan 1972. ISSN 0001-0782. doi: 10.1145/361237.361242. URL `https://doi.org/10.1145/361237.361242`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9726–9735, 2020. doi: 10.1109/CVPR42600.2020.00975.

G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. doi: 10.

1126/science.1127647. URL https://www.science.org/doi/abs/10.1126/science.1127647.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015. URL http://arxiv.org/abs/1503.02531.

Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, Dec 1985. ISSN 1432-1343. doi: 10.1007/BF01908075. URL https://doi.org/10.1007/BF01908075.

N. Kanopoulos, N. Vasanthavada, and R.L. Baker. Design of an image edge detection filter using the sobel operator. *IEEE Journal of Solid-State Circuits*, 23(2):358–367, 1988. doi: 10.1109/4.996.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). Technical report, University of Toronto, 2009. URL https://www.cs.toronto.edu/~kriz/cifar.html.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Qi Lv, Qian Li, Kai Chen, Yao Lu, and Liwen Wang. Classification of ground-based cloud images by contrastive self-supervised learning. *Remote Sensing*, 14(22), 2022. ISSN 2072-4292. doi: 10.3390/rs14225821. URL https://www.mdpi.com/2072-4292/14/22/5821.

J. Macqueen. Some methods for classification and analysis of multivariate observations. In *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

Andrzej Maćkiewicz and Waldemar Ratajczak. Principal components analysis (pca). *Computers & Geosciences*, 19(3):303–342, 1993. ISSN 0098-3004. doi: https://doi.org/10.1016/0098-3004(93)90090-R. URL https://www.sciencedirect.com/science/article/pii/009830049390090R.

Chuang Niu, Hongming Shan, and Ge Wang. Spice: Semantic pseudo-labeling for image clustering. *IEEE Transactions on Image Processing*, 31:7264–7278, 2022. ISSN 1941-0042. doi: 10.1109/tip.2022.3221290. URL http://dx.doi.org/10.1109/TIP.2022.3221290.

Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987. ISSN 0377-0427. doi: https://doi.org/10.1016/0377-0427(87)90125-7. URL https://www.sciencedirect.com/science/article/pii/0377042787901257.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL http://jmlr.org/papers/v9/vandermaaten08a.html.

List of Figures

List of Tables

List of Abbreviations

CAE - convolutional autoencoder
AE - autoencoder
CV - computer vision
CHMI - Czech hydrometeorological Institute
RGB - red, green, blue

A. Attachments

A.1 Code

The code used for this work. It also includes a copy of SPICE repository with modified scripts and CAE implementation.

A.2 Trained models

Trained models for each of the main experiments are included.

A.3 Test images

A small subset of data which can be used for clustering.