# EE4-68 Pattern Recognition (2018-2019): Coursework 2

Martin Ferianc, Alexander Montgomerie-Corcoran
{mf2915, am9215}@ic.ac.org, CID:{00984924, 01052454}

## Abstract

*This work investigates a re-identification problem involving a dataset of pedestrians first presented by Li et al. [1] in the CUHK03 dataset. The baseline approach explores a k-Nearest Neighbours (k-NN) classifier, employing Euclidean and Manhattan distance and k-Means algorithms. This work further explores dimensionality reduction techniques, neighbourhood component analysis (NCA) and neural networks for learning a distance metric that improves the baseline k-NN approach. The best result was achieved by using NCA with an accuracy of 49% and 0.46 mAP at rank 1.*

## 1. Introduction

The task in this work is to perform re-identification of a person between camera views by matching a person from one camera view to the same person from a set of images in another camera view. To do this, a metric to describe similarity is needed to compare these images, which is used to find the *nearest neighbour*. The goal is to learn a function, $d$, or a transformation, $\Sigma$. Thus, for any pairs of objects the distance of the objects from the same category should be smaller, while the points from dissimilar category are further apart.

### 1.1. Dataset

The problem is presented in the CUHK03 dataset [1] which contains 14096 images of 1360 pedestrians in total. The data is further split into *training*, *query* and *gallery* sets with respective sizes of 7368, 1400 and 5328 samples. All images have been processed by ResNet-50 [2], which produces a set of $N = 2048$ features for each image. These are not correlated with respect to the other data, judging by correlation matrices shown in Figs. 4, 5 and 6. There are approximately 8-10 images per person recorded from 2 different cameras. The number of images per camera per person is not equal among all the classes, potentially leading to a bias towards certain individuals.

## 2. Baseline Approaches

### 2.1. k-Nearest Neighbours

k-NN is a non-parametric method that can be used for classification or regression. In case of classification, the input $x_{input}$, is compared to all previously supplied data, $x_i$ ; $\forall\, i \in [0, M]$, based on a distance metric. The most basic metrics are Euclidean (Eq. 1) and Manhattan (Eq. 2) distance:

$$d(x_{input}, x_i)_{Euclidean} = \sum_{i=0}^{M} \sqrt{(x_i - x_{input})^2} \quad (1)$$

$$d(x_{input}, x_i)_{Manhattan} = \sum_{i=0}^{M} |x_i - x_{input}| \quad (2)$$

where $M$ is the number of features per $x_i$ or $x_{input}$. Each $x_i$ is ordered with respect to its distance from $x_{input}$.

The output is the label corresponding to the majority in $k$ smallest distances with respect to the input feature vector.

The baseline approach does not attempt to extract particular features, parameters or augment the data in any way. The choice of metric has the largest effect, where Manhattan distance may be preferable to Euclidean distance in case of a highly dimensional data [3], such as in this problem.

For this dataset, each image from the *query* set, $x_{input}$ with camera id $cam_{input}$ and label $id_{input}$, is compared to each image in a subset of the *gallery* set. This subset can be described as a set of *gallery* images, $x_i$, without the sample with the same label and the same camera id as the compared *query* image,

$$x_i \in \{gallery \cap (\overline{cam_i = cam_{input} \cap id_i = id_{input}})\}$$

The results for $k \in [1, 10]$ or @rank1...10, for Euclidean and Manhattan baseline approaches are in Fig. 1 and Table 1. The rank 1 accuracy for Euclidean distance is **47%**.

We note that, as $k$ is increased, the probability of occurrence of the correct label among the $k$ closest results increases, yet the mean average precision (mAP) decreases, as seen in Fig. 2. An explanation for the decreasing mAP is that as $k$ is increased, so is the probability of introducing false positives. As there are limited images of the same id, the proportion of incorrect ids increases. This in turn causes the vote to become more influenced by incorrect ids. Note, that we did not see any significant difference in results between the Euclidean and Manhattan distance metrics.

### 2.2. k-Means

k-Means is an unsupervised learning method that aims to partition $D$ observations into $k$ clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. For this problem, the $k$ clusters are chosen to represent the number of unique individuals in the *gallery* set.

Upon training, the classifier assembles a list of labels which belong to all points corresponding to the closest cluster centre with respect to the test sample. The predicted label is then decided based on majority voting of the labels in the list. The k-Means classifier was able to achieve **72.3%** accuracy which is mainly due to keeping the samples with the same camera id and label in the gallery dataset upon which the classifier was trained.

## 3. Improved approach

The baseline approaches do not utilise the training data and as such are not able to improve the discriminative properties of the features presented to the classifier.

We propose to investigate feature extraction and augmentation, as well as learning a metric $d$ based on the available training data. In addition to choosing the right metric, a substantial challenge is to leverage the hyperparameters and avoid over-fitting with respect to the training data.

A new fusion rule is also proposed for combining the outputs for $k \geq 2$. This improved rule implements a non-linear weighting through a Gaussian function that scales the vote, favouring points that are perceived as closer to the test image.

### 3.1. Cosine Similarity

A simple improvement that can be done to the baseline approach prior to looking at the training data is to normalise the magnitude of features in the samples. By normalising the features to a magnitude between 0 and 1, a bias towards features of large overall magnitude is avoided.

Given that the magnitude is now irrelevant, we attempted to use cosine similarity to calculate the angle between the samples, rather than distance. The classifier achieved an improvement of accuracy **47.53%** in comparison to the baseline approach for rank 1. Results can be seen in Table 1 or in Figs. 1 & 7.

### 3.2. Mahalanobis Distance

Previous metrics assumed an equal weighting of features given. However, by using training data, a transformation $\Sigma$ can be learnt which improves the separation between data of different labels and clusters similar data together. This transformation can be applied to the features and the Euclidean distance can be found between these features, as described in Eq. 3 [4]:

$$d(A, B) = \sqrt{(A - B)'\Sigma(A - B)} \tag{3}$$

Methods for obtaining $\Sigma$ are evaluated in the following sections. The first approach aims to discern the largest fea-

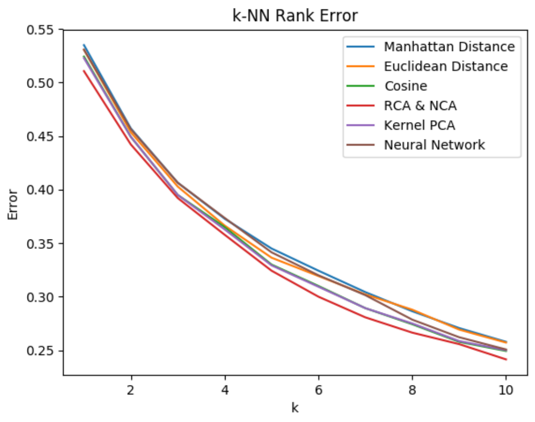


Figure 1: Rank errors with respect to different approaches.

tures of the data, and the subsequent section finds a transformation which aims to find the most important features for classification.

#### 3.2.1 Kernel Principal Component Analysis

Principal component analysis methods are initially investigated. Kernel Principal Component Analysis (KPCA) is applied to the dataset to extract the most significant features of the data, reducing the dimensionality of the data and hence improving the generalisation of the classifier. A Gaussian Radial Basis Function is used to project the features to a higher (infinite) dimensional space, in order to expose a greater separation between features which are deemed to be non-linear.

A matrix $W$ is learned, which consists of the eigenvectors with largest eigenvalues in the kernel space. This can be used in Eq. 3 such that $\Sigma = W'W$.

By reducing the number of features from 2048 to 500, this classifier achieved **48%** accuracy for rank 1, which is a 2% improvement to the baseline approach. The reduction in features also lead to four times faster evaluation time as well as four times lower memory requirements for storing the samples.

The drawback of the KPCA method is that the features extracted do not incorporate information about the data's labels, and as such, do not lead to discriminative features. These features are not interpretable as they exist within an infinite dimensional space.

#### 3.2.2 Relevant Component Analysis and Neighbourhood Components Analysis

To improve on the KPCA method, supervised methods of feature extraction were employed.

A new transformation, $V$, is sought such that the separation of samples with the same label is decreased, and

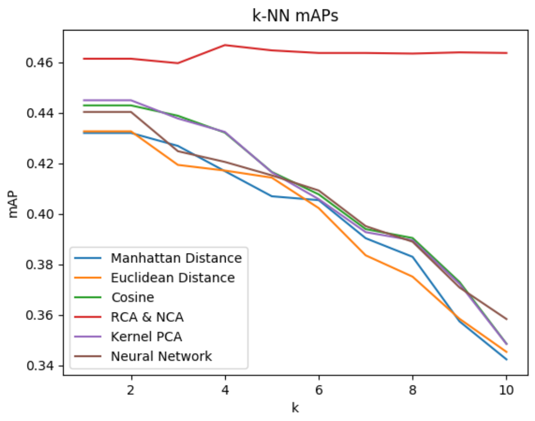


Figure 2: Mean average precision.

separation of dissimilar labels is increased. Neighbourhood Component Analysis [5] (NCA) presents a viable method of learning this transformation. By evaluating the distance on transformed features, the NCA transformation iteratively optimises the expected leave-one-out classification error on the training data using the neighbour selection rule.

Initially, a probability of a point $i$ being a neighbour of $j$ is defined in Eq. 4 as the softmax function on the Mahalanobis distance between $i$ and $j$:

$$p_{ij} = \frac{exp(-||Vx_i - Vx_j||^2)}{\sum_{k \neq i} exp(-||Vx_i - Vx_k||^2)} \tag{4}$$

The probability that the neighbour $j$ is correctly classified (member of class $C_j$) is defined in in Eq. 5:

$$f(V) = \sum_i \sum_{j \in C_j} p_{ij} \tag{5}$$

As we wish to maximise this probability of correct classification, a $V$ is found which is able to maximise $f(V)$. A minimum can be found using the gradient described in Eq. 6:

$$\frac{\delta f}{\delta V} = 2V \sum_i (\sum_k p_{ik} x_{ik} x'_{ik} - \frac{\sum_{j \in C_i} p_{ij} x_{ij} x'_{ij}}{p_{ij}}) \tag{6}$$

where $x_{ij}$ is a training point. The problem cannot be analytically solved in closed form and hence a gradient decent method is used (Limited-memory BFGS [6]).

The resultant matrix, $V$, can be used in the Mahalanobis distance metric, described in Eq. 3. This matrix has large dimensionality ($V \in \mathbb{R}^{N \times N}$), which has the potential to cause over-fitting to the training set, and also causes large execution times when testing.

To reduce over-fitting and execution times, a transformation is applied to the data to reduce dimensionality before NCA. The method we employ is PCA-RCA [7], which first selects the $h$ largest components enclosing the largest variance using PCA, and then applies RCA to select the features based on their relevancy to the labels of the samples.

The RCA transformation amplifies relevant variability and suppresses irrelevant variability of the samples with respect to their labels, as described in [7]. In this way, we are able to extract features most important for clustering of identities.

We combined NCA with PCA-RCA to obtain an improved distance metric. Having performed PCA-RCA on the training set, NCA was trained on these transformed features. In this sense, NCA is finding an optimised version of RCA by using gradient descent methods to iteratively improve on the initial transformation provided by PCA-RCA.

Having obtained the matrix $U$ from PCA-RCA, and subsequently $V$ from NCA, a new matrix for $\Sigma$ can be used with Eq. 3, described as $\Sigma = (VU)'(VU)$.

This method resulted in **49%** accuracy at rank 1 with 250 components provided by PCA-RCA. It also provided consistent mAP across ranks, which suggest the metric is confident in its separation of inputs. The main disadvantage of using this methods is its computational cost in finding the optimal $V$, which we attempted to balance by firstly applying PCA-RCA on the data.

## 3.3. Neural Network Distance Metric

The principal shortcoming of traditional metric learning based methods is that the feature representation of the data and the metric are learned disjointly. The previous approaches only considered learning a transformation of the features by using $\Sigma$. To improve on the previous methods, a distance metric $d$ together with augmenting and weighting the features is learnt by training a neural network. As the features have been generated using a neural network, there is motivation to classify using similar principals.

### 3.3.1 Training and Validation

As the task of the network is to give a distance based on the similarity of labels, a training set is formed based on this principle. We define disjoint training and validation sets of input pairs with 8:2 ratio formed from the original *training* set, along with penalties associated with each of the pairs in these sets[1]. These penalties are normalised to being $0$ for pairs of the same label, and $1$ otherwise, describing the distance between these pairs.

$l_{sim}$ similar pairs and $l_{diff}$ dissimilar pairs are presented to the network during training. To avoid potential bias towards similar/disimilar pairs, an equal number of each is chosen such that $l_{sim} = l_{diff}$. A validation set is also used to monitor over-fitting and only parameters which reduce the validation loss are kept. The validation set represents

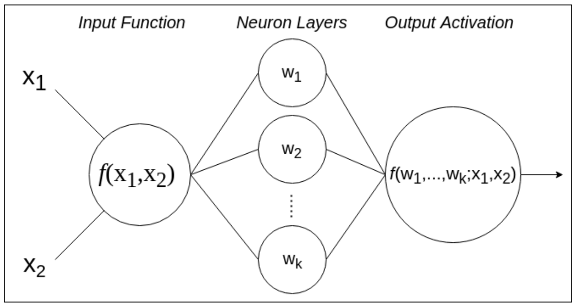
[1]The formation of the dataset is further explained in Appendix B

the test set more closely by containing more dissimilar pairs than similar ($l_{sim} < l_{diff}$).

### 3.3.2 Architecture

Fig. 3 describes the network that is investigated. The network takes two input features and produces an output symbolising the distance between them. This method incorporates principles seen in Siamese Networks [8], except that the inputs have already been processed to extract features, therefore there is only a need to implement the decision layers [9].

Figure 3: Distance Neural Network.



Our decision network has 3 stages: *Input Process, Neural Network, Output Activation.*

The input process applies a function which combines the two input vectors to give a representation of the pair. Potential representations are a concatenation of pairs, absolute difference, covariance between pairs and others. In our method, we take the absolute difference of the inputs:

$$f(\mathbf{x_1}, \mathbf{x_2}) = |\mathbf{x_1} - \mathbf{x_2}| \tag{7}$$

where $\mathbf{x_1}$ and $\mathbf{x_2}$ are inputs to the network. This function produces the difference in features between each input, and the absolute ensures that the model is not biased as to the ordering of $\mathbf{x_1}$ and $\mathbf{x_2}$. Compared to concatenation and covariance, this method results in less dimensions at the input to the neural network layer, reducing the number of parameters in the network and a likeliness to over-fit.

The output of the previous stage is passed to a neural network of $k$ nodes. In this stage, all outputs of the input function are weighted and linearly combined within each node of the network. A *relu* activation function is applied to the output of each node. The function at each node can be expressed as (Eq. 8):

$$node_i \;=\; relu(< w_i, f(\mathbf{x_1}, \mathbf{x_2}) >) \tag{8}$$

This stage extracts the *most important* differences between the input features. The operation of each node draws parallels to the methods seen in section 3.2, where we are learning coefficients based on the difference in the input to describe the distance between them.

The output of the network is expressed as a single node with a sigmoid function applied at the output. A sigmoid function was chosen to give a non-linear representation of the measure of dissimilarity between 0 and 1.

The network which achieved the best result (**47%** accuracy and 0.44 mAP)using 200 nodes. It was trained by using a binary cross-entropy loss function together with N-ADAM optimiser [10]. Note, that we added L2 penalties, batch normalisation [11] and dropout after the neural network to avoid over-fitting to the training data. Similar to section 3.2, this method has a large computational cost during training as well as at test time, due to the large number of parameters. The fact that most of parameters are wrapped inside the hidden layers of the network makes it hard to interpret the model, and essentially transforms this metric into a *black box*.

## 4. Conclusion

In conclusion, we presented the baseline method and demonstrated several gradual improvements through data pre-processing, augmentation, feature reduction and even a neural-network based distance metric.

As seen in Fig. 1, the best result: **49%** accuracy and 0.46 mAP at rank 1 was achieved by combining RCA and NCA, which is a 3% improvement in comparison to the baseline method. Looking closer at confusion matrices in Figs. 8, 9, 10, 11, 12 and 13 we did not see a bias towards a particular individual. We note, that as we increased $k$ the precision as well as accuracy decreases for all the methods except PCA-RCA and NCA.

The presented approaches all demonstrate an improvement in accuracy in comparison to the baseline methods, except neural network, however they are only marginal. We suspect that the complexity of our approach, especially in the architecture of neural network, was lower than the complexity of the problem and thus, our methods were not discriminative enough with respect to the camera views. Also, as the data has been heavily preprocessed, there is very little improvement that can be made on its discriminative properties.

From the observed results, if we increased the complexity into multiple stages such as in combining RCA and PCA, we were able to achieve an improvement in accuracy. It was suggested by the recent published results [12] to explore gradual non-linear hierarchical metric learning in form of a neural network. In this potentially future work, a certain portion of the neural network would be intuitively learning a feature transformation corresponding to one metric and a following layer would correspond to learning another metric and thus gradually developing a transformation that can discriminate between samples from different classes.

## References

[1] W. Li, R. Zhao, T. Xiao, and X. Wang, "Deepreid: Deep filter pairing neural network for person re-identification," in *CVPR*, 2014.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[3] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional space," in *Database Theory*, (Berlin, Heidelberg), pp. 420–434, Springer Berlin Heidelberg, 2001.

[4] P. C. Mahalanobis, "On the generalized distance in statistics," *Proceedings of the National Institute of Sciences (Calcutta)*, vol. 2, pp. 49–55, 1936.

[5] J. Goldberger, G. E. Hinton, S. T. Roweis, and R. R. Salakhutdinov, "Neighbourhood components analysis," in *Advances in Neural Information Processing Systems 17*, pp. 513–520, MIT Press, 2005.

[6] D. C. LIU and J. NOCEDAL, "On the limited memory BFGS method for large scale optimization," *Math. Programming*, vol. 45, no. 3, (Ser. B), pp. 503–528, 1989.

[7] N. Shental, T. Hertz, D. Weinshall, and M. Pavel, "Adjustment learning and relevant component analysis," in *Computer Vision ECCV 2002: 7th European Conference on Computer Vision Copenhagen, Denmark, May 2831, 2002 Proceedings, Part IV*, vol. 2353 of *Lecture Notes in Computer Science*, pp. 776–790, Berlin, Heidelberg: Springer Berlin Heidelberg, 2002.

[8] J. M. Bromley, I. Guyon, Y. LeCun, E. Sackinger, and R. Shah, "Signature verification using a siamese time delay neural network," in *7th Annual Neural Information Processing Systems Conference*, pp. 737–744, Morgan Kaufmann Publishers, 1994. Advances in Neural Information Processing Systems.

[9] S. Zagoruyko and N. Komodakis, "Learning to compare image patches via convolutional neural networks," *CoRR*, vol. abs/1504.03641, 2015.

[10] S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, vol. abs/1609.04747, 2016.

[11] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pp. 448–456, JMLR.org, 2015.

[12] W. Li, J. Huo, Y. Shi, Y. Gao, L. Wang, and J. Luo, "Online deep metric learning," *CoRR*, vol. abs/1805.05510, 2018.

## A. Dataset Characteristics

The correlation matrices were computed out of 100 randomly chosen samples per each partition of the dataset.

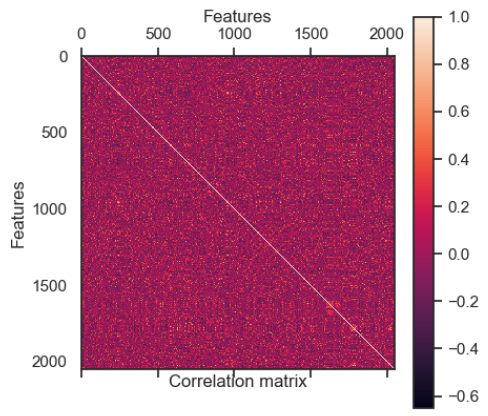


Figure 4: Correlation matrix for training data.

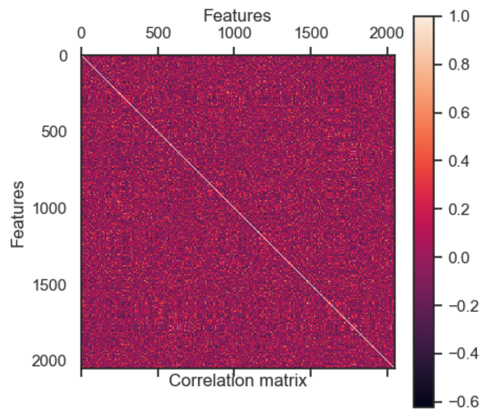


Figure 5: Correlation matrix for gallery data.

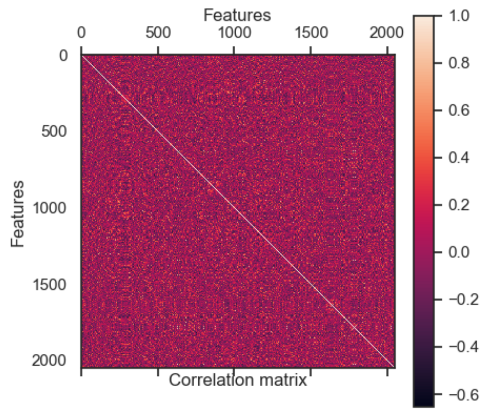


Figure 6: Correlation matrix for query data.

## B. Dataset Generation

For the desired size of the training or validation data set, we selected a sample at random and paired it with other samples. In general, we paired the sample to 10 other samples out of which 6 were samples corresponding to a different label, 3 were samples corresponding to the same label and 1 sample was an identity. The algorithm kept track of previously randomly chosen samples to avoid duplicates. Based on the labels of the samples in a pair, we assigned a penalty to the pair: 1 if they had different labels and 0 if they shared the same label, including identities. This procedure was repeated until the size of the desired data set was reached. Then the dataset was reshuffled to achieve consistency in mini-batch gradient descent.

## C. Results

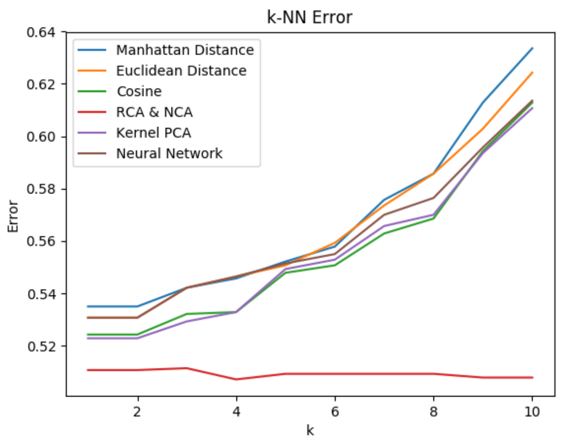


Figure 7: Errors with respect to voting.

## D. Confusion Matrices

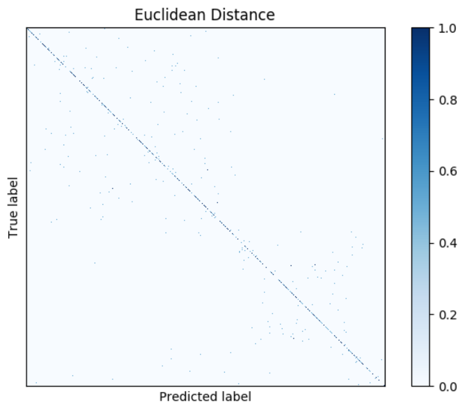


Figure 8: Confusion matrix for Euclidean distance.

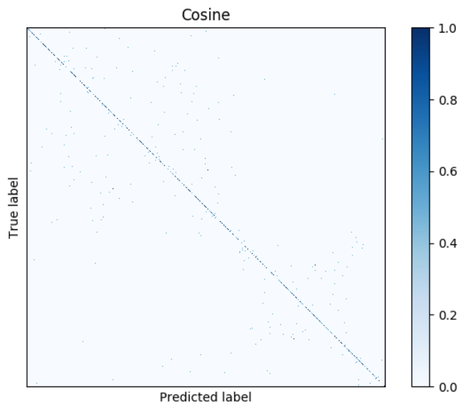


Figure 10: Confusion matrix for Cosine distance.

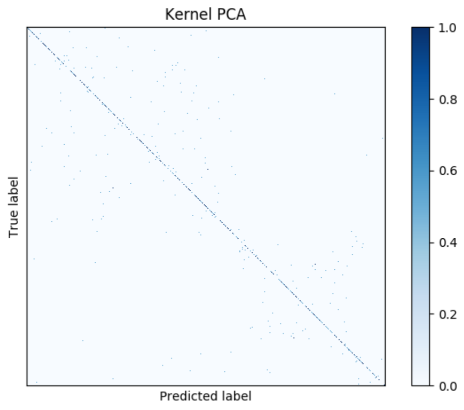


Figure 11: Confusion matrix for Kernel PCA.

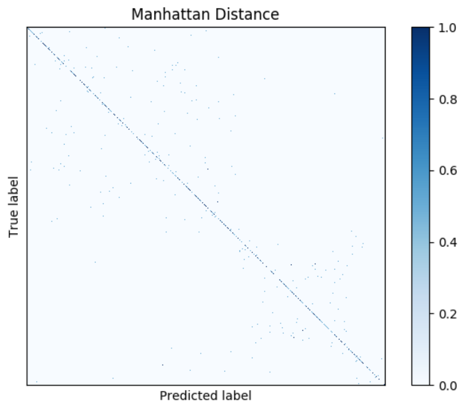


Figure 9: Confusion matrix for Manhattan distance.

| **Method** | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **k** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Euclidean Distance | **mAP** | 0.433 | 0.433 | 0.419 | 0.417 | 0.414 | 0.402 | 0.384 | 0.375 | 0.358 | 0.345 |
| | **Rank $k$ Error** [%] | 53.1 | 45.4 | 40.3 | 36.6 | 33.6 | 31.9 | 30.2 | 28.8 | 26.9 | 25.7 |
| Manhattan Distance | **mAP** | 0.432 | 0.432 | 0.427 | 0.417 | 0.407 | 0.405 | 0.390 | 0.383 | 0.357 | 0.342 |
| | **Rank $k$ Error** [%] | 53.5 | 45.7 | 40.6 | 37.3 | 34.5 | 32.4 | 30.4 | 28.7 | 27.1 | 25.8 |
| Cosine Distance | **mAP** | 0.443 | 0.443 | 0.439 | 0.432 | 0.417 | 0.408 | 0.393 | 0.390 | 0.373 | 0.348 |
| | **Rank $k$ Error** [%] | 52.4 | 44.9 | 39.5 | 36.5 | 33.0 | 31.0 | 28.9 | 27.4 | 25.8 | 24.9 |
| Kernel PCA | **mAP** | 0.445 | 0.445 | 0.438 | 0.432 | 0.416 | 0.406 | 0.393 | 0.389 | 0.372 | 0.348 |
| | **Rank $k$ Error** [%] | 52.3 | 44.9 | 39.5 | 36.3 | 32.9 | 30.9 | 28.9 | 27.5 | 25.9 | 25.0 |
| RCA & NCA | **mAP** | 0.461 | 0.461 | 0.460 | 0.467 | 0.465 | 0.464 | 0.464 | 0.464 | 0.464 | 0.464 |
| | **Rank $k$ Error** [%] | 51.1 | 44.2 | 39.2 | 35.8 | 32.4 | 30.0 | 28.1 | 26.6 | 25.6 | 24.1 |
| Neural Network | **mAP** | 0.440 | 0.440 | 0.425 | 0.421 | 0.415 | 0.409 | 0.395 | 0.389 | 0.371 | 0.358 |
| | **Rank $k$ Error** [%] | 53.1 | 45.6 | 40.6 | 37.4 | 34.1 | 32.0 | 30.1 | 27.9 | 26.2 | 25.1 |

Table 1: k-NN Identification Results: Method corresponds to the distance metric used or the feature extraction, mAP means mean average precision. Rank $k$ Error corresponds to the respective error at the rank $k$.

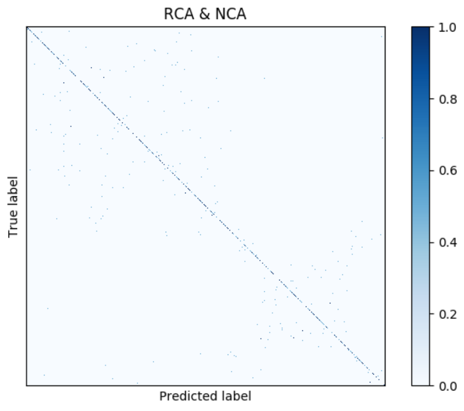


Figure 12: Confusion matrix for RCA & NCA.

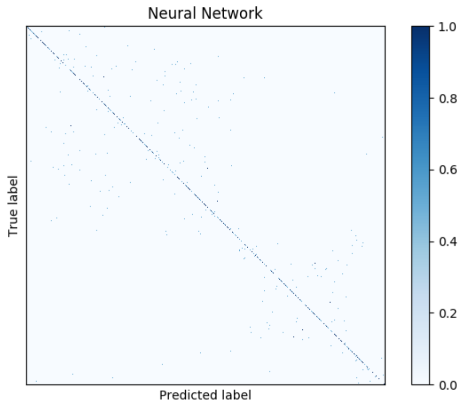


Figure 13: Confusion matrix for neural network.