Testing repeatability of distance between clusters in a UMAP space, given a changing input dataset

In this script, we compute multiple UMAP representations of a changing dataset of digits (0, 2-9). For each combination of these digits, we generate a separate UMAP representation.

We don't change any parameters for UMAP generation, except the input dataset (= combination of digits included).

Then, we compute the pairwise distances b/w the clusters of a given pair of digits, across the multiple UMAP representations. The UMAP representations where the given digit pair was not included were not considered.

For distance b/w clusters, the centroid of the cluster is computed as the geometric mean.

MNIST dataset:- https://en.wikipedia.org/wiki/MNIST_database

[LeCun et al., 1998a] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." Proceedings of the IEEE, 86(11):2278-2324, November 1998. [on-line version]

In [ ]:

Auxilliary set up

In [1]:
```
# Import packages

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.datasets import load_digits
import umap
import hdbscan
from scipy.spatial import distance
from scipy.stats import gmean, mode
from mpl_toolkits.axes_grid1 import make_axes_locatable
import sklearn.cluster as cluster
from itertools import permutations, combinations
from tqdm import tqdm
import warnings
```

```
/opt/anaconda3/envs/umap-proof/lib/python3.12/site-packages/tqdm/auto.py:2
1: TqdmWarning: IProgress not found. Please update jupyter and ipywidgets.
See https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

In [3]:

In [2]:
```
#### Auxilliary functions
```

In [3]:
```
def calc_gm(cluster):
    """ Caclulate the geometric mean of a cluster. """
    center_x = gmean(cluster[:,0])
    center_y = gmean(cluster[:,1])

    return np.array([center_x, center_y])
```

In [4]:
```
def collect_cluster(cluster_no):
    """ Collect all points in a cluster. """
    indices = np.where(labels == cluster_no)
    cluster_points = embedding[indices]

    return cluster_points
```

In [5]:
```
def calc_distance(cluster1, cluster2, cluster_gmeans):
    """ Calculate the euclidean distance between two clusters. """

    center1 = cluster_gmeans[cluster1]
    center2 = cluster_gmeans[cluster2]

    euc_dist = distance.euclidean(center1, center2)

    return euc_dist
```

In [6]:
```
def switch_labels(orig_labels, mnist):
    """
        Switch the labels of the clusters to the most common label in the

        Explanation:
        Normally, the clusters are labelled randomly. This function match
        It also makes all the labels in a cluster the same (= the most co

    """
    new_labels = np.zeros((orig_labels.shape))

    # The manually assigned labels (ground-truth)
    targets = mnist['target']

    sublabel = 1

    for k in np.arange(n_clusters):
        # Collect all the labels in a given cluster
        ind = np.where(orig_labels==k)
```

```
        # Find the ground truth label for the cluster
        t = mnist['target'][ind]
        # Find the mode of the ground truth labels in the cluster (to be
        cluster_mode = mode(t).mode

        # 2 clusters represent the same digit 1. In this case, we label o
        ## I have removed the digit 1 from the dataset, so this is not ne
        ### The + 1000 is just a placeholder while swapping the labels. C
        if cluster_mode==1:
            new_label = sublabel + 1000
            sublabel = 10
        else:
            new_label = cluster_mode + 1000
        # Switch to the new label
        new_labels[ind] = new_label

    return new_labels - 1000
```

In [ ]:

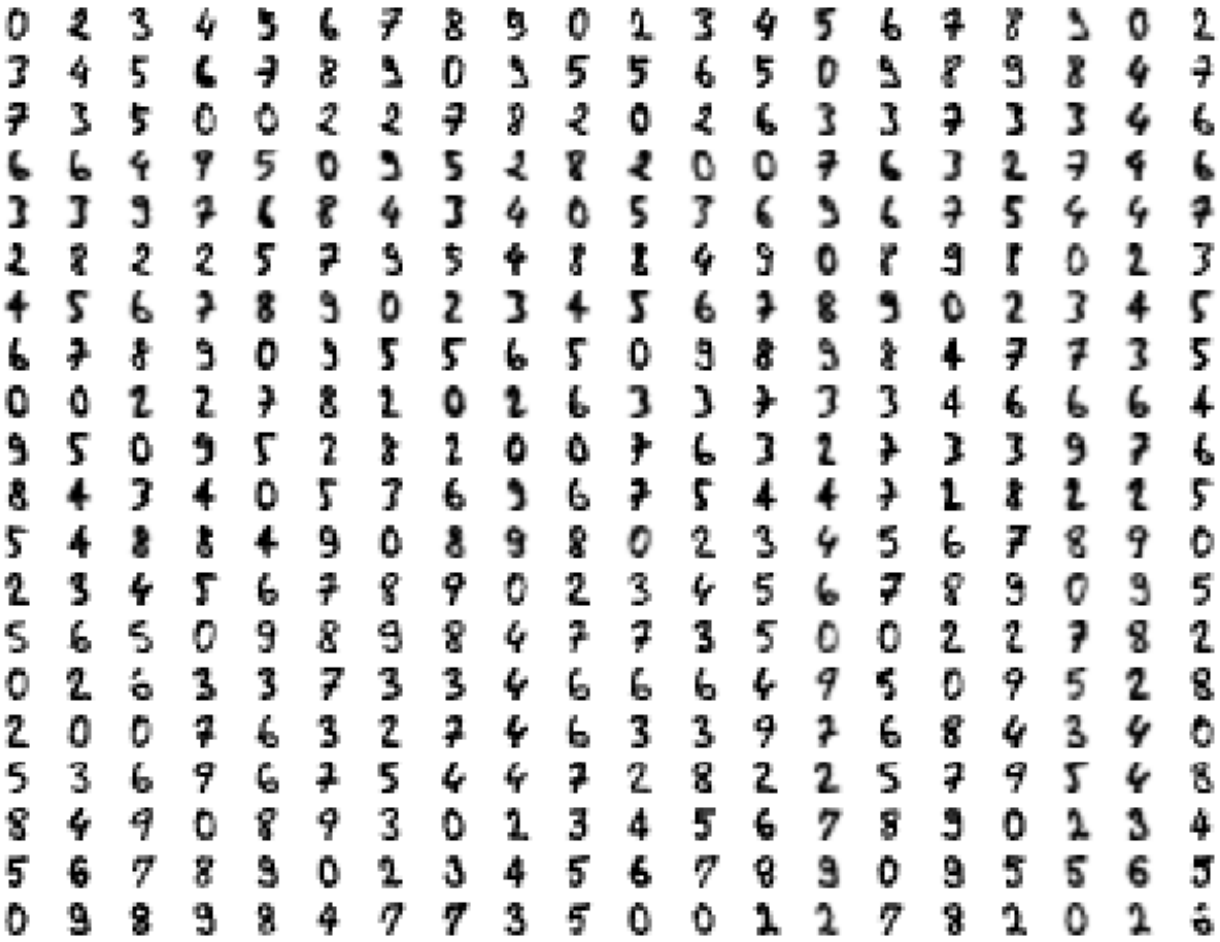
Setting the dataset up

In [7]:
```
### Load MNIST dataset of handwritten digits

mnist = load_digits()
mnist.keys()
```

Out[7]:
```
dict_keys(['data', 'target', 'frame', 'feature_names', 'target_names', '
images', 'DESCR'])
```

In [8]:
```
# Filter out the digit 1 from the mnist dataset and create a new dataset
mnist2 = {}
mnist2['data'] = mnist['data'][(mnist['target'] != 1)]
mnist2['target'] = mnist['target'][(mnist['target'] != 1)]
mnist2['frame'] = mnist['frame']
mnist2['feature_names'] = mnist['feature_names']
mnist2['target_names'] = mnist['target_names'][(mnist['target_names'] !=
mnist2['images'] = mnist['images'][(mnist['target'] != 1)]
mnist2['DESCR'] = mnist['DESCR']
```

In [9]:
```
#  Visualize a subset of the data
fig, ax_array = plt.subplots(20, 20)
axes = ax_array.flatten()
for i, ax in enumerate(axes):
    ax.imshow(mnist2['images'][i], cmap='gray_r')
plt.setp(axes, xticks=[], yticks=[], frame_on=False)
plt.tight_layout(h_pad=0.5, w_pad=0.01)
```



In [ ]:

In [ ]:

We generate multiple UMAP representations of the digits dataset

In [10]:
```
# Initialising a global seed for the random number generator
np.random.seed(0)
```

In [11]:
```
# Set up the path for saving the results
results_path = 'Figures/wo1_digits/exclusion/'

# Describing the included digits
all_digits = [0, 2, 3, 4, 5, 6, 7, 8, 9]
all_digits.sort()
n_all_digits = len(all_digits)
max_digit =  np.max(all_digits)
```

In [12]:
```
# Generate all possible pairs of digits with the given set
all_pair_combinations = list(combinations(all_digits, 2))
all_pair_combinations = np.array(all_pair_combinations)
n_pairs = len(all_pair_combinations)
print('All pair combinations:', all_pair_combinations)
```

```
All pair combinations: [[0 2]
 [0 3]
 [0 4]
 [0 5]
 [0 6]
 [0 7]
 [0 8]
 [0 9]
 [2 3]
 [2 4]
 [2 5]
 [2 6]
 [2 7]
 [2 8]
 [2 9]
 [3 4]
 [3 5]
 [3 6]
 [3 7]
 [3 8]
 [3 9]
 [4 5]
 [4 6]
 [4 7]
 [4 8]
 [4 9]
 [5 6]
 [5 7]
 [5 8]
 [5 9]
 [6 7]
 [6 8]
 [6 9]
 [7 8]
 [7 9]
 [8 9]]
```

In [13]:
```python
# Calculate all combinations of all_digits with minimum length 2
all_combinations = []
for r in range(2, len(all_digits) + 1):
    all_combinations.extend(combinations(all_digits, r))

n_combos = len(all_combinations)
# n_combos = len(combo_embedding['combination'])

# all_combinations = np.array(all_combinations)
print('All combinations:', all_combinations)
```

```
All combinations: [(0, 2), (0, 3), (0, 4), (0, 5), (0, 6), (0, 7), (0, 8),
(0, 9), (2, 3), (2, 4), (2, 5), (2, 6), (2, 7), (2, 8), (2, 9), (3, 4), (
3, 5), (3, 6), (3, 7), (3, 8), (3, 9), (4, 5), (4, 6), (4, 7), (4, 8), (4,
9), (5, 6), (5, 7), (5, 8), (5, 9), (6, 7), (6, 8), (6, 9), (7, 8), (7,
9), (8, 9), (0, 2, 3), (0, 2, 4), (0, 2, 5), (0, 2, 6), (0, 2, 7), (0, 2,
8), (0, 2, 9), (0, 3, 4), (0, 3, 5), (0, 3, 6), (0, 3, 7), (0, 3, 8), (0,
```

```
3, 9), (0, 4, 5), (0, 4, 6), (0, 4, 7), (0, 4, 8), (0, 4, 9), (0, 5, 6), (
0, 5, 7), (0, 5, 8), (0, 5, 9), (0, 6, 7), (0, 6, 8), (0, 6, 9), (0, 7,
8), (0, 7, 9), (0, 8, 9), (2, 3, 4), (2, 3, 5), (2, 3, 6), (2, 3, 7), (2,
3, 8), (2, 3, 9), (2, 4, 5), (2, 4, 6), (2, 4, 7), (2, 4, 8), (2, 4, 9), (
2, 5, 6), (2, 5, 7), (2, 5, 8), (2, 5, 9), (2, 6, 7), (2, 6, 8), (2, 6,
9), (2, 7, 8), (2, 7, 9), (2, 8, 9), (3, 4, 5), (3, 4, 6), (3, 4, 7), (3,
4, 8), (3, 4, 9), (3, 5, 6), (3, 5, 7), (3, 5, 8), (3, 5, 9), (3, 6, 7), (
3, 6, 8), (3, 6, 9), (3, 7, 8), (3, 7, 9), (3, 8, 9), (4, 5, 6), (4, 5,
7), (4, 5, 8), (4, 5, 9), (4, 6, 7), (4, 6, 8), (4, 6, 9), (4, 7, 8), (4,
7, 9), (4, 8, 9), (5, 6, 7), (5, 6, 8), (5, 6, 9), (5, 7, 8), (5, 7, 9), (
5, 8, 9), (6, 7, 8), (6, 7, 9), (6, 8, 9), (7, 8, 9), (0, 2, 3, 4), (0, 2,
3, 5), (0, 2, 3, 6), (0, 2, 3, 7), (0, 2, 3, 8), (0, 2, 3, 9), (0, 2, 4,
5), (0, 2, 4, 6), (0, 2, 4, 7), (0, 2, 4, 8), (0, 2, 4, 9), (0, 2, 5, 6),
(0, 2, 5, 7), (0, 2, 5, 8), (0, 2, 5, 9), (0, 2, 6, 7), (0, 2, 6, 8), (0,
2, 6, 9), (0, 2, 7, 8), (0, 2, 7, 9), (0, 2, 8, 9), (0, 3, 4, 5), (0, 3,
4, 6), (0, 3, 4, 7), (0, 3, 4, 8), (0, 3, 4, 9), (0, 3, 5, 6), (0, 3, 5,
7), (0, 3, 5, 8), (0, 3, 5, 9), (0, 3, 6, 7), (0, 3, 6, 8), (0, 3, 6, 9),
(0, 3, 7, 8), (0, 3, 7, 9), (0, 3, 8, 9), (0, 4, 5, 6), (0, 4, 5, 7), (0,
4, 5, 8), (0, 4, 5, 9), (0, 4, 6, 7), (0, 4, 6, 8), (0, 4, 6, 9), (0, 4,
7, 8), (0, 4, 7, 9), (0, 4, 8, 9), (0, 5, 6, 7), (0, 5, 6, 8), (0, 5, 6,
9), (0, 5, 7, 8), (0, 5, 7, 9), (0, 5, 8, 9), (0, 6, 7, 8), (0, 6, 7, 9),
(0, 6, 8, 9), (0, 7, 8, 9), (2, 3, 4, 5), (2, 3, 4, 6), (2, 3, 4, 7), (2,
3, 4, 8), (2, 3, 4, 9), (2, 3, 5, 6), (2, 3, 5, 7), (2, 3, 5, 8), (2, 3,
5, 9), (2, 3, 6, 7), (2, 3, 6, 8), (2, 3, 6, 9), (2, 3, 7, 8), (2, 3, 7,
9), (2, 3, 8, 9), (2, 4, 5, 6), (2, 4, 5, 7), (2, 4, 5, 8), (2, 4, 5, 9),
(2, 4, 6, 7), (2, 4, 6, 8), (2, 4, 6, 9), (2, 4, 7, 8), (2, 4, 7, 9), (2,
4, 8, 9), (2, 5, 6, 7), (2, 5, 6, 8), (2, 5, 6, 9), (2, 5, 7, 8), (2, 5,
7, 9), (2, 5, 8, 9), (2, 6, 7, 8), (2, 6, 7, 9), (2, 6, 8, 9), (2, 7, 8,
9), (3, 4, 5, 6), (3, 4, 5, 7), (3, 4, 5, 8), (3, 4, 5, 9), (3, 4, 6, 7),
(3, 4, 6, 8), (3, 4, 6, 9), (3, 4, 7, 8), (3, 4, 7, 9), (3, 4, 8, 9), (3,
5, 6, 7), (3, 5, 6, 8), (3, 5, 6, 9), (3, 5, 7, 8), (3, 5, 7, 9), (3, 5,
8, 9), (3, 6, 7, 8), (3, 6, 7, 9), (3, 6, 8, 9), (3, 7, 8, 9), (4, 5, 6,
7), (4, 5, 6, 8), (4, 5, 6, 9), (4, 5, 7, 8), (4, 5, 7, 9), (4, 5, 8, 9),
(4, 6, 7, 8), (4, 6, 7, 9), (4, 6, 8, 9), (4, 7, 8, 9), (5, 6, 7, 8), (5,
6, 7, 9), (5, 6, 8, 9), (5, 7, 8, 9), (6, 7, 8, 9), (0, 2, 3, 4, 5), (0,
2, 3, 4, 6), (0, 2, 3, 4, 7), (0, 2, 3, 4, 8), (0, 2, 3, 4, 9), (0, 2, 3,
5, 6), (0, 2, 3, 5, 7), (0, 2, 3, 5, 8), (0, 2, 3, 5, 9), (0, 2, 3, 6, 7),
(0, 2, 3, 6, 8), (0, 2, 3, 6, 9), (0, 2, 3, 7, 8), (0, 2, 3, 7, 9), (0, 2,
3, 8, 9), (0, 2, 4, 5, 6), (0, 2, 4, 5, 7), (0, 2, 4, 5, 8), (0, 2, 4, 5,
9), (0, 2, 4, 6, 7), (0, 2, 4, 6, 8), (0, 2, 4, 6, 9), (0, 2, 4, 7, 8), (
0, 2, 4, 7, 9), (0, 2, 4, 8, 9), (0, 2, 5, 6, 7), (0, 2, 5, 6, 8), (0, 2,
5, 6, 9), (0, 2, 5, 7, 8), (0, 2, 5, 7, 9), (0, 2, 5, 8, 9), (0, 2, 6, 7,
8), (0, 2, 6, 7, 9), (0, 2, 6, 8, 9), (0, 2, 7, 8, 9), (0, 3, 4, 5, 6), (
0, 3, 4, 5, 7), (0, 3, 4, 5, 8), (0, 3, 4, 5, 9), (0, 3, 4, 6, 7), (0, 3,
4, 6, 8), (0, 3, 4, 6, 9), (0, 3, 4, 7, 8), (0, 3, 4, 7, 9), (0, 3, 4, 8,
9), (0, 3, 5, 6, 7), (0, 3, 5, 6, 8), (0, 3, 5, 6, 9), (0, 3, 5, 7, 8), (
0, 3, 5, 7, 9), (0, 3, 5, 8, 9), (0, 3, 6, 7, 8), (0, 3, 6, 7, 9), (0, 3,
6, 8, 9), (0, 3, 7, 8, 9), (0, 4, 5, 6, 7), (0, 4, 5, 6, 8), (0, 4, 5, 6,
9), (0, 4, 5, 7, 8), (0, 4, 5, 7, 9), (0, 4, 5, 8, 9), (0, 4, 6, 7, 8), (
0, 4, 6, 7, 9), (0, 4, 6, 8, 9), (0, 4, 7, 8, 9), (0, 5, 6, 7, 8), (0, 5,
6, 7, 9), (0, 5, 6, 8, 9), (0, 5, 7, 8, 9), (0, 6, 7, 8, 9), (2, 3, 4, 5,
6), (2, 3, 4, 5, 7), (2, 3, 4, 5, 8), (2, 3, 4, 5, 9), (2, 3, 4, 6, 7), (
2, 3, 4, 6, 8), (2, 3, 4, 6, 9), (2, 3, 4, 7, 8), (2, 3, 4, 7, 9), (2, 3,
```

```
4, 8, 9), (2, 3, 5, 6, 7), (2, 3, 5, 6, 8), (2, 3, 5, 6, 9), (2, 3, 5, 7,
8), (2, 3, 5, 7, 9), (2, 3, 5, 8, 9), (2, 3, 6, 7, 8), (2, 3, 6, 7, 9), (
2, 3, 6, 8, 9), (2, 3, 7, 8, 9), (2, 4, 5, 6, 7), (2, 4, 5, 6, 8), (2, 4,
5, 6, 9), (2, 4, 5, 7, 8), (2, 4, 5, 7, 9), (2, 4, 5, 8, 9), (2, 4, 6, 7,
8), (2, 4, 6, 7, 9), (2, 4, 6, 8, 9), (2, 4, 7, 8, 9), (2, 5, 6, 7, 8), (
2, 5, 6, 7, 9), (2, 5, 6, 8, 9), (2, 5, 7, 8, 9), (2, 6, 7, 8, 9), (3, 4,
5, 6, 7), (3, 4, 5, 6, 8), (3, 4, 5, 6, 9), (3, 4, 5, 7, 8), (3, 4, 5, 7,
9), (3, 4, 5, 8, 9), (3, 4, 6, 7, 8), (3, 4, 6, 7, 9), (3, 4, 6, 8, 9), (
3, 4, 7, 8, 9), (3, 5, 6, 7, 8), (3, 5, 6, 7, 9), (3, 5, 6, 8, 9), (3, 5,
7, 8, 9), (3, 6, 7, 8, 9), (4, 5, 6, 7, 8), (4, 5, 6, 7, 9), (4, 5, 6, 8,
9), (4, 5, 7, 8, 9), (4, 6, 7, 8, 9), (5, 6, 7, 8, 9), (0, 2, 3, 4, 5, 6),
(0, 2, 3, 4, 5, 7), (0, 2, 3, 4, 5, 8), (0, 2, 3, 4, 5, 9), (0, 2, 3, 4,
6, 7), (0, 2, 3, 4, 6, 8), (0, 2, 3, 4, 6, 9), (0, 2, 3, 4, 7, 8), (0, 2,
3, 4, 7, 9), (0, 2, 3, 4, 8, 9), (0, 2, 3, 5, 6, 7), (0, 2, 3, 5, 6, 8), (
0, 2, 3, 5, 6, 9), (0, 2, 3, 5, 7, 8), (0, 2, 3, 5, 7, 9), (0, 2, 3, 5, 8,
9), (0, 2, 3, 6, 7, 8), (0, 2, 3, 6, 7, 9), (0, 2, 3, 6, 8, 9), (0, 2, 3,
7, 8, 9), (0, 2, 4, 5, 6, 7), (0, 2, 4, 5, 6, 8), (0, 2, 4, 5, 6, 9), (0,
2, 4, 5, 7, 8), (0, 2, 4, 5, 7, 9), (0, 2, 4, 5, 8, 9), (0, 2, 4, 6, 7,
8), (0, 2, 4, 6, 7, 9), (0, 2, 4, 6, 8, 9), (0, 2, 4, 7, 8, 9), (0, 2, 5,
6, 7, 8), (0, 2, 5, 6, 7, 9), (0, 2, 5, 6, 8, 9), (0, 2, 5, 7, 8, 9), (0,
2, 6, 7, 8, 9), (0, 3, 4, 5, 6, 7), (0, 3, 4, 5, 6, 8), (0, 3, 4, 5, 6,
9), (0, 3, 4, 5, 7, 8), (0, 3, 4, 5, 7, 9), (0, 3, 4, 5, 8, 9), (0, 3, 4,
6, 7, 8), (0, 3, 4, 6, 7, 9), (0, 3, 4, 6, 8, 9), (0, 3, 4, 7, 8, 9), (0,
3, 5, 6, 7, 8), (0, 3, 5, 6, 7, 9), (0, 3, 5, 6, 8, 9), (0, 3, 5, 7, 8,
9), (0, 3, 6, 7, 8, 9), (0, 4, 5, 6, 7, 8), (0, 4, 5, 6, 7, 9), (0, 4, 5,
6, 8, 9), (0, 4, 5, 7, 8, 9), (0, 4, 6, 7, 8, 9), (0, 5, 6, 7, 8, 9), (2,
3, 4, 5, 6, 7), (2, 3, 4, 5, 6, 8), (2, 3, 4, 5, 6, 9), (2, 3, 4, 5, 7,
8), (2, 3, 4, 5, 7, 9), (2, 3, 4, 5, 8, 9), (2, 3, 4, 6, 7, 8), (2, 3, 4,
6, 7, 9), (2, 3, 4, 6, 8, 9), (2, 3, 4, 7, 8, 9), (2, 3, 5, 6, 7, 8), (2,
3, 5, 6, 7, 9), (2, 3, 5, 6, 8, 9), (2, 3, 5, 7, 8, 9), (2, 3, 6, 7, 8,
9), (2, 4, 5, 6, 7, 8), (2, 4, 5, 6, 7, 9), (2, 4, 5, 6, 8, 9), (2, 4, 5,
7, 8, 9), (2, 4, 6, 7, 8, 9), (2, 5, 6, 7, 8, 9), (3, 4, 5, 6, 7, 8), (3,
4, 5, 6, 7, 9), (3, 4, 5, 6, 8, 9), (3, 4, 5, 7, 8, 9), (3, 4, 6, 7, 8,
9), (3, 5, 6, 7, 8, 9), (4, 5, 6, 7, 8, 9), (0, 2, 3, 4, 5, 6, 7), (0, 2,
3, 4, 5, 6, 8), (0, 2, 3, 4, 5, 6, 9), (0, 2, 3, 4, 5, 7, 8), (0, 2, 3, 4,
5, 7, 9), (0, 2, 3, 4, 5, 8, 9), (0, 2, 3, 4, 6, 7, 8), (0, 2, 3, 4, 6, 7,
9), (0, 2, 3, 4, 6, 8, 9), (0, 2, 3, 4, 7, 8, 9), (0, 2, 3, 5, 6, 7, 8), (
0, 2, 3, 5, 6, 7, 9), (0, 2, 3, 5, 6, 8, 9), (0, 2, 3, 5, 7, 8, 9), (0, 2,
3, 6, 7, 8, 9), (0, 2, 4, 5, 6, 7, 8), (0, 2, 4, 5, 6, 7, 9), (0, 2, 4, 5,
6, 8, 9), (0, 2, 4, 5, 7, 8, 9), (0, 2, 4, 6, 7, 8, 9), (0, 2, 5, 6, 7, 8,
9), (0, 3, 4, 5, 6, 7, 8), (0, 3, 4, 5, 6, 7, 9), (0, 3, 4, 5, 6, 8, 9), (
0, 3, 4, 5, 7, 8, 9), (0, 3, 4, 6, 7, 8, 9), (0, 3, 5, 6, 7, 8, 9), (0, 4,
5, 6, 7, 8, 9), (2, 3, 4, 5, 6, 7, 8), (2, 3, 4, 5, 6, 7, 9), (2, 3, 4, 5,
6, 8, 9), (2, 3, 4, 5, 7, 8, 9), (2, 3, 4, 6, 7, 8, 9), (2, 3, 5, 6, 7, 8,
9), (2, 4, 5, 6, 7, 8, 9), (3, 4, 5, 6, 7, 8, 9), (0, 2, 3, 4, 5, 6, 7,
8), (0, 2, 3, 4, 5, 6, 7, 9), (0, 2, 3, 4, 5, 6, 8, 9), (0, 2, 3, 4, 5, 7,
8, 9), (0, 2, 3, 4, 6, 7, 8, 9), (0, 2, 3, 5, 6, 7, 8, 9), (0, 2, 4, 5, 6,
7, 8, 9), (0, 3, 4, 5, 6, 7, 8, 9), (2, 3, 4, 5, 6, 7, 8, 9), (0, 2, 3, 4,
5, 6, 7, 8, 9)]
```

In [14]:
```
# Make a container to store the details of the UMAP representation of eac
combo_embedding = {
    'combination': [],
    'cluster_gmeans': [],
```

```
    'distance_matrix': []
}
```

In [15]:
```
# Using a fixed seed for all UMAP projections
rseed = np.random.randint(0,1e9)

# Run the dimension reduction and clustering for each combination of digi
for n_combo, combo in tqdm(enumerate(all_combinations)):
    print('n_combo #: ', n_combo)
    print('combo : ', np.array(combo))

    # Sort the digits in the combination
    curr_digits = np.array(combo)
    curr_digits.sort()

    # Filter mnist2 dataset to only have curr_digits
    indices = np.where(np.isin(mnist2['target'], curr_digits))

    mnist2_curr = {}
    mnist2_curr['data'] = mnist2['data'][indices]
    mnist2_curr['target'] = mnist2['target'][indices]
    mnist2_curr['images'] = mnist2['images'][indices]

    # Create a UMAP embedding of the data
    reducer = umap.UMAP(random_state=rseed)
    reducer.fit(mnist2_curr['data'])
    embedding = reducer.transform(mnist2_curr['data'])

    # Normalise the embedding (VERIFY: Is normalisation needed?)
    embedding = (embedding-embedding.min())
    embedding = embedding/embedding.max()

    # the number of clusters = the number of digits in the combination
    n_clusters = len(curr_digits)

    # Cluster the UMAP embedding
    labels = cluster.KMeans(n_clusters=n_clusters).fit_predict(embedding)
    # # labels = hdbscan.HDBSCAN(min_samples=10, min_cluster_size=50).fit



    # Cleaning
    ## Matching the generated labels to the digit it represents
    new_labels = switch_labels(labels, mnist2_curr)

    # Find the centroid = geometric mean of each cluster
    cluster_gmeans  = np.zeros((n_clusters, 2))
    cluster_gmeans_dict = {}
    for nc in np.arange(n_clusters):
        cluster_points = collect_cluster(nc)

        cluster_gmeans[nc] = calc_gm(cluster_points)
        cluster_gmeans_dict[curr_digits[nc]] = cluster_gmeans[nc]
```

```
    # Make a plot of the UMAP embedding and clusters
    fig = plt.figure()
    plt.scatter(embedding[:, 0], embedding[:, 1], c=new_labels, cmap='tab

    plt.gca().set_aspect('equal', 'datalim')
    # plt.colorbar(boundaries=np.arange(new_labels.min()+2, new_labels.ma
    plt.colorbar(boundaries=np.arange(new_labels.min(), new_labels.max()+

    plt.scatter(cluster_gmeans[:,0], cluster_gmeans[:,1], c='black', s=10

    plt.title('UMAP projection of the Digits dataset', fontsize=14)

    plt.savefig(results_path + 'UMAPs/n' + str(n_combo) + '_combo' + str(

    plt.close()

    # Create a distance matrix for the pairwise distances in the given UM
    distance_matrix = np.zeros((max_digit+2, max_digit+2))

    # Find distance between all pairs of clusters
    for digit1 in all_digits:
        for digit2 in all_digits:
            if digit1 < digit2:
                if digit1 in curr_digits and digit2 in curr_digits:

                    dist = calc_distance(digit1, digit2, cluster_gmeans_d

                    distance_matrix[digit1, digit2] = dist
                    # distance_matrix[digit1, digit2] = 0

    # Save the details of the UMAP embedding
    combo_embedding['combination'].append(curr_digits)
    combo_embedding['cluster_gmeans'].append(cluster_gmeans_dict)
    combo_embedding['distance_matrix'].append(distance_matrix)

    # BEGIN: Suppress warnings
    warnings.filterwarnings('ignore')
    # END:
```

```
0it [00:00, ?it/s]
n_combo #:  0
combo :  [0 2]
/opt/anaconda3/envs/umap-proof/lib/python3.12/site-packages/umap/umap_.py:
1945: UserWarning: n_jobs value 1 overridden to 1 by setting random_state.
Use no seed for parallelism.
  warn(f"n_jobs value {self.n_jobs} overridden to 1 by setting random_stat
e. Use no seed for parallelism.")
1it [00:06,  6.18s/it]
n_combo #:  1
combo :  [0 3]
2it [00:07,  3.36s/it]
```

```
n_combo #:  2
combo :  [0 4]
3it [00:08,  2.42s/it]
n_combo #:  3
combo :  [0 5]
4it [00:10,  2.02s/it]
n_combo #:  4
combo :  [0 6]
5it [00:11,  1.77s/it]
n_combo #:  5
combo :  [0 7]
6it [00:12,  1.61s/it]
n_combo #:  6
combo :  [0 8]
7it [00:14,  1.49s/it]
n_combo #:  7
combo :  [0 9]
8it [00:15,  1.41s/it]
n_combo #:  8
combo :  [2 3]
9it [00:17,  1.48s/it]
n_combo #:  9
combo :  [2 4]
10it [00:18,  1.40s/it]
n_combo #:  10
combo :  [2 5]
11it [00:19,  1.35s/it]
n_combo #:  11
combo :  [2 6]
12it [00:20,  1.34s/it]
n_combo #:  12
combo :  [2 7]
13it [00:22,  1.31s/it]
n_combo #:  13
combo :  [2 8]
14it [00:23,  1.30s/it]
n_combo #:  14
combo :  [2 9]
15it [00:24,  1.27s/it]
n_combo #:  15
combo :  [3 4]
16it [00:25,  1.29s/it]
n_combo #:  16
combo :  [3 5]
17it [00:27,  1.42s/it]
n_combo #:  17
combo :  [3 6]
18it [00:28,  1.37s/it]
```

```
n_combo #:  18
combo :  [3 7]
19it [00:30,  1.34s/it]
n_combo #:  19
combo :  [3 8]
20it [00:31,  1.32s/it]
n_combo #:  20
combo :  [3 9]
21it [00:32,  1.31s/it]
n_combo #:  21
combo :  [4 5]
22it [00:33,  1.29s/it]
n_combo #:  22
combo :  [4 6]
23it [00:35,  1.26s/it]
n_combo #:  23
combo :  [4 7]
24it [00:36,  1.27s/it]
n_combo #:  24
combo :  [4 8]
25it [00:37,  1.29s/it]
n_combo #:  25
combo :  [4 9]
26it [00:39,  1.44s/it]
n_combo #:  26
combo :  [5 6]
27it [00:40,  1.39s/it]
n_combo #:  27
combo :  [5 7]
28it [00:42,  1.36s/it]
n_combo #:  28
combo :  [5 8]
29it [00:43,  1.36s/it]
n_combo #:  29
combo :  [5 9]
30it [00:44,  1.38s/it]
n_combo #:  30
combo :  [6 7]
31it [00:46,  1.40s/it]
n_combo #:  31
combo :  [6 8]
32it [00:47,  1.36s/it]
n_combo #:  32
combo :  [6 9]
33it [00:48,  1.33s/it]
n_combo #:  33
combo :  [7 8]
34it [00:50,  1.37s/it]
```

```
n_combo #:  34
combo :  [7 9]
35it [00:51,  1.44s/it]
n_combo #:  35
combo :  [8 9]
36it [00:53,  1.58s/it]
n_combo #:  36
combo :  [0 2 3]
37it [00:55,  1.62s/it]
n_combo #:  37
combo :  [0 2 4]
38it [00:57,  1.76s/it]
n_combo #:  38
combo :  [0 2 5]
39it [00:59,  1.74s/it]
n_combo #:  39
combo :  [0 2 6]
40it [01:00,  1.72s/it]
n_combo #:  40
combo :  [0 2 7]
41it [01:02,  1.69s/it]
n_combo #:  41
combo :  [0 2 8]
42it [01:04,  1.67s/it]
n_combo #:  42
combo :  [0 2 9]
43it [01:05,  1.67s/it]
n_combo #:  43
combo :  [0 3 4]
44it [01:07,  1.69s/it]
n_combo #:  44
combo :  [0 3 5]
45it [01:09,  1.68s/it]
n_combo #:  45
combo :  [0 3 6]
46it [01:11,  1.89s/it]
n_combo #:  46
combo :  [0 3 7]
47it [01:13,  2.01s/it]
n_combo #:  47
combo :  [0 3 8]
48it [01:16,  2.07s/it]
n_combo #:  48
combo :  [0 3 9]
49it [01:18,  2.03s/it]
n_combo #:  49
combo :  [0 4 5]
50it [01:19,  1.97s/it]
```

```
n_combo #:  50
combo :  [0 4 6]
51it [01:21,  1.98s/it]
n_combo #:  51
combo :  [0 4 7]
52it [01:23,  1.92s/it]
n_combo #:  52
combo :  [0 4 8]
53it [01:25,  1.86s/it]
n_combo #:  53
combo :  [0 4 9]
54it [01:27,  1.85s/it]
n_combo #:  54
combo :  [0 5 6]
55it [01:29,  2.12s/it]
n_combo #:  55
combo :  [0 5 7]
56it [01:31,  2.00s/it]
n_combo #:  56
combo :  [0 5 8]
57it [01:33,  1.89s/it]
n_combo #:  57
combo :  [0 5 9]
58it [01:34,  1.82s/it]
n_combo #:  58
combo :  [0 6 7]
59it [01:36,  1.79s/it]
n_combo #:  59
combo :  [0 6 8]
60it [01:38,  1.78s/it]
n_combo #:  60
combo :  [0 6 9]
61it [01:40,  1.82s/it]
n_combo #:  61
combo :  [0 7 8]
62it [01:42,  1.86s/it]
n_combo #:  62
combo :  [0 7 9]
63it [01:44,  1.95s/it]
n_combo #:  63
combo :  [0 8 9]
64it [01:46,  1.92s/it]
n_combo #:  64
combo :  [2 3 4]
65it [01:47,  1.84s/it]
n_combo #:  65
combo :  [2 3 5]
66it [01:50,  1.98s/it]
```

```
n_combo #:  66
combo :  [2 3 6]
67it [01:52,  1.94s/it]
n_combo #:  67
combo :  [2 3 7]
68it [01:54,  2.00s/it]
n_combo #:  68
combo :  [2 3 8]
69it [01:56,  2.05s/it]
n_combo #:  69
combo :  [2 3 9]
70it [01:58,  2.09s/it]
n_combo #:  70
combo :  [2 4 5]
71it [02:01,  2.25s/it]
n_combo #:  71
combo :  [2 4 6]
72it [02:03,  2.29s/it]
n_combo #:  72
combo :  [2 4 7]
73it [02:06,  2.35s/it]
n_combo #:  73
combo :  [2 4 8]
74it [02:08,  2.34s/it]
n_combo #:  74
combo :  [2 4 9]
75it [02:10,  2.22s/it]
n_combo #:  75
combo :  [2 5 6]
76it [02:12,  2.09s/it]
n_combo #:  76
combo :  [2 5 7]
77it [02:14,  2.15s/it]
n_combo #:  77
combo :  [2 5 8]
78it [02:16,  2.03s/it]
n_combo #:  78
combo :  [2 5 9]
79it [02:17,  1.93s/it]
n_combo #:  79
combo :  [2 6 7]
80it [02:19,  1.83s/it]
n_combo #:  80
combo :  [2 6 8]
81it [02:21,  1.78s/it]
n_combo #:  81
combo :  [2 6 9]
82it [02:22,  1.77s/it]
```

```
n_combo #:  82
combo :  [2 7 8]
83it [02:24,  1.73s/it]
n_combo #:  83
combo :  [2 7 9]
84it [02:26,  1.72s/it]
n_combo #:  84
combo :  [2 8 9]
85it [02:27,  1.72s/it]
n_combo #:  85
combo :  [3 4 5]
86it [02:29,  1.70s/it]
n_combo #:  86
combo :  [3 4 6]
87it [02:31,  1.69s/it]
n_combo #:  87
combo :  [3 4 7]
88it [02:33,  1.91s/it]
n_combo #:  88
combo :  [3 4 8]
89it [02:35,  1.84s/it]
n_combo #:  89
combo :  [3 4 9]
90it [02:37,  1.79s/it]
n_combo #:  90
combo :  [3 5 6]
91it [02:38,  1.77s/it]
n_combo #:  91
combo :  [3 5 7]
92it [02:40,  1.74s/it]
n_combo #:  92
combo :  [3 5 8]
93it [02:42,  1.72s/it]
n_combo #:  93
combo :  [3 5 9]
94it [02:43,  1.72s/it]
n_combo #:  94
combo :  [3 6 7]
95it [02:45,  1.69s/it]
n_combo #:  95
combo :  [3 6 8]
96it [02:47,  1.68s/it]
n_combo #:  96
combo :  [3 6 9]
97it [02:48,  1.71s/it]
n_combo #:  97
combo :  [3 7 8]
98it [02:50,  1.73s/it]
```

```
n_combo #:  98
combo :  [3 7 9]
99it [02:52,  1.77s/it]
n_combo #:  99
combo :  [3 8 9]
100it [02:55,  1.99s/it]
n_combo #:  100
combo :  [4 5 6]
101it [02:56,  1.96s/it]
n_combo #:  101
combo :  [4 5 7]
102it [02:58,  1.89s/it]
n_combo #:  102
combo :  [4 5 8]
103it [03:00,  1.81s/it]
n_combo #:  103
combo :  [4 5 9]
104it [03:01,  1.76s/it]
n_combo #:  104
combo :  [4 6 7]
105it [03:03,  1.74s/it]
n_combo #:  105
combo :  [4 6 8]
106it [03:05,  1.71s/it]
n_combo #:  106
combo :  [4 6 9]
107it [03:07,  1.71s/it]
n_combo #:  107
combo :  [4 7 8]
108it [03:08,  1.70s/it]
n_combo #:  108
combo :  [4 7 9]
109it [03:10,  1.67s/it]
n_combo #:  109
combo :  [4 8 9]
110it [03:11,  1.65s/it]
n_combo #:  110
combo :  [5 6 7]
111it [03:13,  1.66s/it]
n_combo #:  111
combo :  [5 6 8]
112it [03:15,  1.65s/it]
n_combo #:  112
combo :  [5 6 9]
113it [03:17,  1.92s/it]
n_combo #:  113
combo :  [5 7 8]
114it [03:19,  1.85s/it]
```

```
n_combo #:  114
combo :  [5 7 9]
115it [03:21,  1.84s/it]
n_combo #:  115
combo :  [5 8 9]
116it [03:22,  1.81s/it]
n_combo #:  116
combo :  [6 7 8]
117it [03:24,  1.79s/it]
n_combo #:  117
combo :  [6 7 9]
118it [03:26,  1.77s/it]
n_combo #:  118
combo :  [6 8 9]
119it [03:28,  1.85s/it]
n_combo #:  119
combo :  [7 8 9]
120it [03:30,  1.82s/it]
n_combo #:  120
combo :  [0 2 3 4]
121it [03:32,  1.93s/it]
n_combo #:  121
combo :  [0 2 3 5]
122it [03:34,  2.01s/it]
n_combo #:  122
combo :  [0 2 3 6]
123it [03:36,  2.01s/it]
n_combo #:  123
combo :  [0 2 3 7]
124it [03:38,  2.02s/it]
n_combo #:  124
combo :  [0 2 3 8]
125it [03:40,  2.02s/it]
n_combo #:  125
combo :  [0 2 3 9]
126it [03:43,  2.26s/it]
n_combo #:  126
combo :  [0 2 4 5]
127it [03:45,  2.20s/it]
n_combo #:  127
combo :  [0 2 4 6]
128it [03:47,  2.15s/it]
n_combo #:  128
combo :  [0 2 4 7]
129it [03:49,  2.15s/it]
n_combo #:  129
combo :  [0 2 4 8]
130it [03:51,  2.12s/it]
```

```
n_combo #:  130
combo :  [0 2 4 9]
131it [03:53,  2.10s/it]
n_combo #:  131
combo :  [0 2 5 6]
132it [03:55,  2.09s/it]
n_combo #:  132
combo :  [0 2 5 7]
133it [03:57,  2.08s/it]
n_combo #:  133
combo :  [0 2 5 8]
134it [04:00,  2.07s/it]
n_combo #:  134
combo :  [0 2 5 9]
135it [04:02,  2.06s/it]
n_combo #:  135
combo :  [0 2 6 7]
136it [04:04,  2.09s/it]
n_combo #:  136
combo :  [0 2 6 8]
137it [04:06,  2.07s/it]
n_combo #:  137
combo :  [0 2 6 9]
138it [04:09,  2.45s/it]
n_combo #:  138
combo :  [0 2 7 8]
139it [04:12,  2.53s/it]
n_combo #:  139
combo :  [0 2 7 9]
140it [04:15,  2.86s/it]
n_combo #:  140
combo :  [0 2 8 9]
141it [04:18,  2.89s/it]
n_combo #:  141
combo :  [0 3 4 5]
142it [04:21,  2.79s/it]
n_combo #:  142
combo :  [0 3 4 6]
143it [04:23,  2.63s/it]
n_combo #:  143
combo :  [0 3 4 7]
144it [04:25,  2.45s/it]
n_combo #:  144
combo :  [0 3 4 8]
145it [04:27,  2.35s/it]
n_combo #:  145
combo :  [0 3 4 9]
146it [04:29,  2.26s/it]
```

```
n_combo #:  146
combo :  [0 3 5 6]
147it [04:32,  2.22s/it]
n_combo #:  147
combo :  [0 3 5 7]
148it [04:34,  2.18s/it]
n_combo #:  148
combo :  [0 3 5 8]
149it [04:36,  2.16s/it]
n_combo #:  149
combo :  [0 3 5 9]
150it [04:38,  2.14s/it]
n_combo #:  150
combo :  [0 3 6 7]
151it [04:40,  2.12s/it]
n_combo #:  151
combo :  [0 3 6 8]
152it [04:42,  2.15s/it]
n_combo #:  152
combo :  [0 3 6 9]
153it [04:44,  2.13s/it]
n_combo #:  153
combo :  [0 3 7 8]
154it [04:47,  2.35s/it]
n_combo #:  154
combo :  [0 3 7 9]
155it [04:49,  2.28s/it]
n_combo #:  155
combo :  [0 3 8 9]
156it [04:51,  2.23s/it]
n_combo #:  156
combo :  [0 4 5 6]
157it [04:53,  2.18s/it]
n_combo #:  157
combo :  [0 4 5 7]
158it [04:55,  2.15s/it]
n_combo #:  158
combo :  [0 4 5 8]
159it [04:57,  2.12s/it]
n_combo #:  159
combo :  [0 4 5 9]
160it [05:00,  2.12s/it]
n_combo #:  160
combo :  [0 4 6 7]
161it [05:02,  2.10s/it]
n_combo #:  161
combo :  [0 4 6 8]
162it [05:04,  2.11s/it]
```

```
n_combo #:  162
combo :  [0 4 6 9]
163it [05:06,  2.13s/it]
n_combo #:  163
combo :  [0 4 7 8]
164it [05:08,  2.13s/it]
n_combo #:  164
combo :  [0 4 7 9]
165it [05:10,  2.14s/it]
n_combo #:  165
combo :  [0 4 8 9]
166it [05:12,  2.11s/it]
n_combo #:  166
combo :  [0 5 6 7]
167it [05:14,  2.10s/it]
n_combo #:  167
combo :  [0 5 6 8]
168it [05:17,  2.37s/it]
n_combo #:  168
combo :  [0 5 6 9]
169it [05:19,  2.28s/it]
n_combo #:  169
combo :  [0 5 7 8]
170it [05:21,  2.22s/it]
n_combo #:  170
combo :  [0 5 7 9]
171it [05:24,  2.21s/it]
n_combo #:  171
combo :  [0 5 8 9]
172it [05:26,  2.17s/it]
n_combo #:  172
combo :  [0 6 7 8]
173it [05:28,  2.16s/it]
n_combo #:  173
combo :  [0 6 7 9]
174it [05:30,  2.20s/it]
n_combo #:  174
combo :  [0 6 8 9]
175it [05:33,  2.26s/it]
n_combo #:  175
combo :  [0 7 8 9]
176it [05:35,  2.27s/it]
n_combo #:  176
combo :  [2 3 4 5]
177it [05:37,  2.27s/it]
n_combo #:  177
combo :  [2 3 4 6]
178it [05:39,  2.27s/it]
```

```
n_combo #:  178
combo :  [2 3 4 7]
179it [05:42,  2.29s/it]
n_combo #:  179
combo :  [2 3 4 8]
180it [05:44,  2.25s/it]
n_combo #:  180
combo :  [2 3 4 9]
181it [05:46,  2.24s/it]
n_combo #:  181
combo :  [2 3 5 6]
182it [05:48,  2.21s/it]
n_combo #:  182
combo :  [2 3 5 7]
183it [05:51,  2.49s/it]
n_combo #:  183
combo :  [2 3 5 8]
184it [05:54,  2.41s/it]
n_combo #:  184
combo :  [2 3 5 9]
185it [05:56,  2.38s/it]
n_combo #:  185
combo :  [2 3 6 7]
186it [05:58,  2.36s/it]
n_combo #:  186
combo :  [2 3 6 8]
187it [06:01,  2.36s/it]
n_combo #:  187
combo :  [2 3 6 9]
188it [06:03,  2.38s/it]
n_combo #:  188
combo :  [2 3 7 8]
189it [06:05,  2.38s/it]
n_combo #:  189
combo :  [2 3 7 9]
190it [06:08,  2.38s/it]
n_combo #:  190
combo :  [2 3 8 9]
191it [06:10,  2.32s/it]
n_combo #:  191
combo :  [2 4 5 6]
192it [06:12,  2.25s/it]
n_combo #:  192
combo :  [2 4 5 7]
193it [06:14,  2.22s/it]
n_combo #:  193
combo :  [2 4 5 8]
194it [06:16,  2.17s/it]
```

```
n_combo #:  194
combo :  [2 4 5 9]
195it [06:18,  2.14s/it]
n_combo #:  195
combo :  [2 4 6 7]
196it [06:20,  2.11s/it]
n_combo #:  196
combo :  [2 4 6 8]
197it [06:22,  2.10s/it]
n_combo #:  197
combo :  [2 4 6 9]
198it [06:25,  2.13s/it]
n_combo #:  198
combo :  [2 4 7 8]
199it [06:27,  2.28s/it]
n_combo #:  199
combo :  [2 4 7 9]
200it [06:31,  2.65s/it]
n_combo #:  200
combo :  [2 4 8 9]
201it [06:33,  2.57s/it]
n_combo #:  201
combo :  [2 5 6 7]
202it [06:36,  2.57s/it]
n_combo #:  202
combo :  [2 5 6 8]
203it [06:38,  2.61s/it]
n_combo #:  203
combo :  [2 5 6 9]
204it [06:41,  2.61s/it]
n_combo #:  204
combo :  [2 5 7 8]
205it [06:44,  2.55s/it]
n_combo #:  205
combo :  [2 5 7 9]
206it [06:46,  2.48s/it]
n_combo #:  206
combo :  [2 5 8 9]
207it [06:48,  2.34s/it]
n_combo #:  207
combo :  [2 6 7 8]
208it [06:50,  2.25s/it]
n_combo #:  208
combo :  [2 6 7 9]
209it [06:52,  2.19s/it]
n_combo #:  209
combo :  [2 6 8 9]
210it [06:54,  2.15s/it]
```

```
n_combo #:  210
combo :  [2 7 8 9]
211it [06:56,  2.12s/it]
n_combo #:  211
combo :  [3 4 5 6]
212it [06:58,  2.13s/it]
n_combo #:  212
combo :  [3 4 5 7]
213it [07:00,  2.10s/it]
n_combo #:  213
combo :  [3 4 5 8]
214it [07:02,  2.09s/it]
n_combo #:  214
combo :  [3 4 5 9]
215it [07:04,  2.08s/it]
n_combo #:  215
combo :  [3 4 6 7]
216it [07:06,  2.08s/it]
n_combo #:  216
combo :  [3 4 6 8]
217it [07:09,  2.37s/it]
n_combo #:  217
combo :  [3 4 6 9]
218it [07:12,  2.28s/it]
n_combo #:  218
combo :  [3 4 7 8]
219it [07:14,  2.21s/it]
n_combo #:  219
combo :  [3 4 7 9]
220it [07:16,  2.17s/it]
n_combo #:  220
combo :  [3 4 8 9]
221it [07:18,  2.17s/it]
n_combo #:  221
combo :  [3 5 6 7]
222it [07:20,  2.14s/it]
n_combo #:  222
combo :  [3 5 6 8]
223it [07:22,  2.12s/it]
n_combo #:  223
combo :  [3 5 6 9]
224it [07:24,  2.11s/it]
n_combo #:  224
combo :  [3 5 7 8]
225it [07:26,  2.10s/it]
n_combo #:  225
combo :  [3 5 7 9]
226it [07:28,  2.09s/it]
```

```
n_combo #:  226
combo :  [3 5 8 9]
227it [07:30,  2.06s/it]
n_combo #:  227
combo :  [3 6 7 8]
228it [07:32,  2.05s/it]
n_combo #:  228
combo :  [3 6 7 9]
229it [07:34,  2.07s/it]
n_combo #:  229
combo :  [3 6 8 9]
230it [07:37,  2.11s/it]
n_combo #:  230
combo :  [3 7 8 9]
231it [07:39,  2.11s/it]
n_combo #:  231
combo :  [4 5 6 7]
232it [07:41,  2.12s/it]
n_combo #:  232
combo :  [4 5 6 8]
233it [07:43,  2.12s/it]
n_combo #:  233
combo :  [4 5 6 9]
234it [07:46,  2.44s/it]
n_combo #:  234
combo :  [4 5 7 8]
235it [07:48,  2.32s/it]
n_combo #:  235
combo :  [4 5 7 9]
236it [07:50,  2.24s/it]
n_combo #:  236
combo :  [4 5 8 9]
237it [07:52,  2.18s/it]
n_combo #:  237
combo :  [4 6 7 8]
238it [07:54,  2.14s/it]
n_combo #:  238
combo :  [4 6 7 9]
239it [07:56,  2.11s/it]
n_combo #:  239
combo :  [4 6 8 9]
240it [07:58,  2.09s/it]
n_combo #:  240
combo :  [4 7 8 9]
241it [08:00,  2.07s/it]
n_combo #:  241
combo :  [5 6 7 8]
242it [08:03,  2.10s/it]
```

```
n_combo #:  242
combo :  [5 6 7 9]
243it [08:05,  2.09s/it]
n_combo #:  243
combo :  [5 6 8 9]
244it [08:07,  2.08s/it]
n_combo #:  244
combo :  [5 7 8 9]
245it [08:09,  2.07s/it]
n_combo #:  245
combo :  [6 7 8 9]
246it [08:11,  2.06s/it]
n_combo #:  246
combo :  [0 2 3 4 5]
247it [08:13,  2.24s/it]
n_combo #:  247
combo :  [0 2 3 4 6]
248it [08:16,  2.33s/it]
n_combo #:  248
combo :  [0 2 3 4 7]
249it [08:18,  2.38s/it]
n_combo #:  249
combo :  [0 2 3 4 8]
250it [08:21,  2.42s/it]
n_combo #:  250
combo :  [0 2 3 4 9]
251it [08:24,  2.48s/it]
n_combo #:  251
combo :  [0 2 3 5 6]
252it [08:27,  2.81s/it]
n_combo #:  252
combo :  [0 2 3 5 7]
253it [08:30,  2.74s/it]
n_combo #:  253
combo :  [0 2 3 5 8]
254it [08:32,  2.69s/it]
n_combo #:  254
combo :  [0 2 3 5 9]
255it [08:35,  2.67s/it]
n_combo #:  255
combo :  [0 2 3 6 7]
256it [08:38,  2.64s/it]
n_combo #:  256
combo :  [0 2 3 6 8]
257it [08:40,  2.60s/it]
n_combo #:  257
combo :  [0 2 3 6 9]
258it [08:43,  2.58s/it]
```

```
n_combo #:  258
combo :  [0 2 3 7 8]
259it [08:45,  2.59s/it]
n_combo #:  259
combo :  [0 2 3 7 9]
260it [08:48,  2.57s/it]
n_combo #:  260
combo :  [0 2 3 8 9]
261it [08:50,  2.56s/it]
n_combo #:  261
combo :  [0 2 4 5 6]
262it [08:53,  2.54s/it]
n_combo #:  262
combo :  [0 2 4 5 7]
263it [08:55,  2.54s/it]
n_combo #:  263
combo :  [0 2 4 5 8]
264it [08:58,  2.55s/it]
n_combo #:  264
combo :  [0 2 4 5 9]
265it [09:01,  2.61s/it]
n_combo #:  265
combo :  [0 2 4 6 7]
266it [09:03,  2.65s/it]
n_combo #:  266
combo :  [0 2 4 6 8]
267it [09:06,  2.64s/it]
n_combo #:  267
combo :  [0 2 4 6 9]
268it [09:09,  2.67s/it]
n_combo #:  268
combo :  [0 2 4 7 8]
269it [09:11,  2.66s/it]
n_combo #:  269
combo :  [0 2 4 7 9]
270it [09:14,  2.63s/it]
n_combo #:  270
combo :  [0 2 4 8 9]
271it [09:18,  2.98s/it]
n_combo #:  271
combo :  [0 2 5 6 7]
272it [09:20,  2.85s/it]
n_combo #:  272
combo :  [0 2 5 6 8]
273it [09:23,  2.77s/it]
n_combo #:  273
combo :  [0 2 5 6 9]
274it [09:25,  2.72s/it]
```

```
n_combo #:  274
combo :  [0 2 5 7 8]
275it [09:28,  2.80s/it]
n_combo #:  275
combo :  [0 2 5 7 9]
276it [09:31,  2.83s/it]
n_combo #:  276
combo :  [0 2 5 8 9]
277it [09:34,  2.82s/it]
n_combo #:  277
combo :  [0 2 6 7 8]
278it [09:37,  2.81s/it]
n_combo #:  278
combo :  [0 2 6 7 9]
279it [09:39,  2.74s/it]
n_combo #:  279
combo :  [0 2 6 8 9]
280it [09:42,  2.71s/it]
n_combo #:  280
combo :  [0 2 7 8 9]
281it [09:45,  2.75s/it]
n_combo #:  281
combo :  [0 3 4 5 6]
282it [09:48,  2.69s/it]
n_combo #:  282
combo :  [0 3 4 5 7]
283it [09:50,  2.64s/it]
n_combo #:  283
combo :  [0 3 4 5 8]
284it [09:53,  2.61s/it]
n_combo #:  284
combo :  [0 3 4 5 9]
285it [09:55,  2.61s/it]
n_combo #:  285
combo :  [0 3 4 6 7]
286it [09:58,  2.73s/it]
n_combo #:  286
combo :  [0 3 4 6 8]
287it [10:02,  2.94s/it]
n_combo #:  287
combo :  [0 3 4 6 9]
288it [10:04,  2.89s/it]
n_combo #:  288
combo :  [0 3 4 7 8]
289it [10:08,  2.97s/it]
n_combo #:  289
combo :  [0 3 4 7 9]
290it [10:10,  2.96s/it]
```

```
n_combo #:  290
combo :  [0 3 4 8 9]
291it [10:15,  3.28s/it]
n_combo #:  291
combo :  [0 3 5 6 7]
292it [10:17,  3.06s/it]
n_combo #:  292
combo :  [0 3 5 6 8]
293it [10:20,  2.89s/it]
n_combo #:  293
combo :  [0 3 5 6 9]
294it [10:22,  2.76s/it]
n_combo #:  294
combo :  [0 3 5 7 8]
295it [10:24,  2.68s/it]
n_combo #:  295
combo :  [0 3 5 7 9]
296it [10:27,  2.66s/it]
n_combo #:  296
combo :  [0 3 5 8 9]
297it [10:30,  2.63s/it]
n_combo #:  297
combo :  [0 3 6 7 8]
298it [10:32,  2.61s/it]
n_combo #:  298
combo :  [0 3 6 7 9]
299it [10:35,  2.67s/it]
n_combo #:  299
combo :  [0 3 6 8 9]
300it [10:38,  2.66s/it]
n_combo #:  300
combo :  [0 3 7 8 9]
301it [10:40,  2.68s/it]
n_combo #:  301
combo :  [0 4 5 6 7]
302it [10:43,  2.67s/it]
n_combo #:  302
combo :  [0 4 5 6 8]
303it [10:46,  2.69s/it]
n_combo #:  303
combo :  [0 4 5 6 9]
304it [10:49,  2.70s/it]
n_combo #:  304
combo :  [0 4 5 7 8]
305it [10:51,  2.65s/it]
n_combo #:  305
combo :  [0 4 5 7 9]
306it [10:54,  2.63s/it]
```

```
n_combo #:  306
combo :  [0 4 5 8 9]
307it [10:56,  2.63s/it]
n_combo #:  307
combo :  [0 4 6 7 8]
308it [10:59,  2.67s/it]
n_combo #:  308
combo :  [0 4 6 7 9]
309it [11:02,  2.71s/it]
n_combo #:  309
combo :  [0 4 6 8 9]
310it [11:05,  2.76s/it]
n_combo #:  310
combo :  [0 4 7 8 9]
311it [11:07,  2.77s/it]
n_combo #:  311
combo :  [0 5 6 7 8]
312it [11:11,  3.12s/it]
n_combo #:  312
combo :  [0 5 6 7 9]
313it [11:14,  2.93s/it]
n_combo #:  313
combo :  [0 5 6 8 9]
314it [11:16,  2.81s/it]
n_combo #:  314
combo :  [0 5 7 8 9]
315it [11:19,  2.75s/it]
n_combo #:  315
combo :  [0 6 7 8 9]
316it [11:22,  2.67s/it]
n_combo #:  316
combo :  [2 3 4 5 6]
317it [11:24,  2.63s/it]
n_combo #:  317
combo :  [2 3 4 5 7]
318it [11:27,  2.59s/it]
n_combo #:  318
combo :  [2 3 4 5 8]
319it [11:29,  2.59s/it]
n_combo #:  319
combo :  [2 3 4 5 9]
320it [11:32,  2.57s/it]
n_combo #:  320
combo :  [2 3 4 6 7]
321it [11:34,  2.57s/it]
n_combo #:  321
combo :  [2 3 4 6 8]
322it [11:37,  2.58s/it]
```

```
n_combo #:  322
combo :  [2 3 4 6 9]
323it [11:39,  2.55s/it]
n_combo #:  323
combo :  [2 3 4 7 8]
324it [11:42,  2.53s/it]
n_combo #:  324
combo :  [2 3 4 7 9]
325it [11:44,  2.50s/it]
n_combo #:  325
combo :  [2 3 4 8 9]
326it [11:47,  2.50s/it]
n_combo #:  326
combo :  [2 3 5 6 7]
327it [11:49,  2.54s/it]
n_combo #:  327
combo :  [2 3 5 6 8]
328it [11:52,  2.58s/it]
n_combo #:  328
combo :  [2 3 5 6 9]
329it [11:55,  2.61s/it]
n_combo #:  329
combo :  [2 3 5 7 8]
330it [11:57,  2.60s/it]
n_combo #:  330
combo :  [2 3 5 7 9]
331it [12:00,  2.62s/it]
n_combo #:  331
combo :  [2 3 5 8 9]
332it [12:02,  2.59s/it]
n_combo #:  332
combo :  [2 3 6 7 8]
333it [12:05,  2.58s/it]
n_combo #:  333
combo :  [2 3 6 7 9]
334it [12:09,  2.99s/it]
n_combo #:  334
combo :  [2 3 6 8 9]
335it [12:12,  2.85s/it]
n_combo #:  335
combo :  [2 3 7 8 9]
336it [12:14,  2.81s/it]
n_combo #:  336
combo :  [2 4 5 6 7]
337it [12:17,  2.76s/it]
n_combo #:  337
combo :  [2 4 5 6 8]
338it [12:20,  2.74s/it]
```

```
n_combo #:  338
combo :  [2 4 5 6 9]
339it [12:22,  2.67s/it]
n_combo #:  339
combo :  [2 4 5 7 8]
340it [12:25,  2.63s/it]
n_combo #:  340
combo :  [2 4 5 7 9]
341it [12:27,  2.58s/it]
n_combo #:  341
combo :  [2 4 5 8 9]
342it [12:30,  2.54s/it]
n_combo #:  342
combo :  [2 4 6 7 8]
343it [12:32,  2.63s/it]
n_combo #:  343
combo :  [2 4 6 7 9]
344it [12:35,  2.73s/it]
n_combo #:  344
combo :  [2 4 6 8 9]
345it [12:38,  2.76s/it]
n_combo #:  345
combo :  [2 4 7 8 9]
346it [12:41,  2.80s/it]
n_combo #:  346
combo :  [2 5 6 7 8]
347it [12:44,  2.80s/it]
n_combo #:  347
combo :  [2 5 6 7 9]
348it [12:47,  2.86s/it]
n_combo #:  348
combo :  [2 5 6 8 9]
349it [12:50,  2.95s/it]
n_combo #:  349
combo :  [2 5 7 8 9]
350it [12:53,  2.92s/it]
n_combo #:  350
combo :  [2 6 7 8 9]
351it [12:56,  2.88s/it]
n_combo #:  351
combo :  [3 4 5 6 7]
352it [12:59,  2.88s/it]
n_combo #:  352
combo :  [3 4 5 6 8]
353it [13:01,  2.84s/it]
n_combo #:  353
combo :  [3 4 5 6 9]
354it [13:04,  2.82s/it]
```

```
n_combo #:  354
combo :  [3 4 5 7 8]
355it [13:07,  2.75s/it]
n_combo #:  355
combo :  [3 4 5 7 9]
356it [13:11,  3.18s/it]
n_combo #:  356
combo :  [3 4 5 8 9]
357it [13:13,  2.98s/it]
n_combo #:  357
combo :  [3 4 6 7 8]
358it [13:16,  2.87s/it]
n_combo #:  358
combo :  [3 4 6 7 9]
359it [13:18,  2.77s/it]
n_combo #:  359
combo :  [3 4 6 8 9]
360it [13:21,  2.69s/it]
n_combo #:  360
combo :  [3 4 7 8 9]
361it [13:23,  2.63s/it]
n_combo #:  361
combo :  [3 5 6 7 8]
362it [13:26,  2.63s/it]
n_combo #:  362
combo :  [3 5 6 7 9]
363it [13:29,  2.65s/it]
n_combo #:  363
combo :  [3 5 6 8 9]
364it [13:31,  2.64s/it]
n_combo #:  364
combo :  [3 5 7 8 9]
365it [13:34,  2.64s/it]
n_combo #:  365
combo :  [3 6 7 8 9]
366it [13:37,  2.67s/it]
n_combo #:  366
combo :  [4 5 6 7 8]
367it [13:40,  2.70s/it]
n_combo #:  367
combo :  [4 5 6 7 9]
368it [13:42,  2.69s/it]
n_combo #:  368
combo :  [4 5 6 8 9]
369it [13:45,  2.68s/it]
n_combo #:  369
combo :  [4 5 7 8 9]
370it [13:48,  2.66s/it]
```

```
n_combo #:  370
combo :  [4 6 7 8 9]
371it [13:50,  2.63s/it]
n_combo #:  371
combo :  [5 6 7 8 9]
372it [13:53,  2.58s/it]
n_combo #:  372
combo :  [0 2 3 4 5 6]
373it [13:55,  2.68s/it]
n_combo #:  373
combo :  [0 2 3 4 5 7]
374it [13:58,  2.75s/it]
n_combo #:  374
combo :  [0 2 3 4 5 8]
375it [14:01,  2.83s/it]
n_combo #:  375
combo :  [0 2 3 4 5 9]
376it [14:05,  2.99s/it]
n_combo #:  376
combo :  [0 2 3 4 6 7]
377it [14:08,  3.12s/it]
n_combo #:  377
combo :  [0 2 3 4 6 8]
378it [14:12,  3.19s/it]
n_combo #:  378
combo :  [0 2 3 4 6 9]
379it [14:15,  3.20s/it]
n_combo #:  379
combo :  [0 2 3 4 7 8]
380it [14:19,  3.65s/it]
n_combo #:  380
combo :  [0 2 3 4 7 9]
381it [14:22,  3.46s/it]
n_combo #:  381
combo :  [0 2 3 4 8 9]
382it [14:25,  3.32s/it]
n_combo #:  382
combo :  [0 2 3 5 6 7]
383it [14:29,  3.26s/it]
n_combo #:  383
combo :  [0 2 3 5 6 8]
384it [14:32,  3.16s/it]
n_combo #:  384
combo :  [0 2 3 5 6 9]
385it [14:35,  3.11s/it]
n_combo #:  385
combo :  [0 2 3 5 7 8]
386it [14:38,  3.17s/it]
```

```
n_combo #:  386
combo :  [0 2 3 5 7 9]
387it [14:41,  3.18s/it]
n_combo #:  387
combo :  [0 2 3 5 8 9]
388it [14:44,  3.14s/it]
n_combo #:  388
combo :  [0 2 3 6 7 8]
389it [14:47,  3.12s/it]
n_combo #:  389
combo :  [0 2 3 6 7 9]
390it [14:50,  3.07s/it]
n_combo #:  390
combo :  [0 2 3 6 8 9]
391it [14:53,  3.06s/it]
n_combo #:  391
combo :  [0 2 3 7 8 9]
392it [14:57,  3.23s/it]
n_combo #:  392
combo :  [0 2 4 5 6 7]
393it [15:01,  3.54s/it]
n_combo #:  393
combo :  [0 2 4 5 6 8]
394it [15:06,  3.86s/it]
n_combo #:  394
combo :  [0 2 4 5 6 9]
395it [15:10,  3.90s/it]
n_combo #:  395
combo :  [0 2 4 5 7 8]
396it [15:13,  3.69s/it]
n_combo #:  396
combo :  [0 2 4 5 7 9]
397it [15:16,  3.48s/it]
n_combo #:  397
combo :  [0 2 4 5 8 9]
398it [15:19,  3.33s/it]
n_combo #:  398
combo :  [0 2 4 6 7 8]
399it [15:22,  3.20s/it]
n_combo #:  399
combo :  [0 2 4 6 7 9]
400it [15:25,  3.13s/it]
n_combo #:  400
combo :  [0 2 4 6 8 9]
401it [15:28,  3.08s/it]
n_combo #:  401
combo :  [0 2 4 7 8 9]
402it [15:31,  3.07s/it]
```

```
n_combo #:  402
combo :  [0 2 5 6 7 8]
403it [15:34,  3.03s/it]
n_combo #:  403
combo :  [0 2 5 6 7 9]
404it [15:37,  3.02s/it]
n_combo #:  404
combo :  [0 2 5 6 8 9]
405it [15:42,  3.63s/it]
n_combo #:  405
combo :  [0 2 5 7 8 9]
406it [15:45,  3.53s/it]
n_combo #:  406
combo :  [0 2 6 7 8 9]
407it [15:48,  3.45s/it]
n_combo #:  407
combo :  [0 3 4 5 6 7]
408it [15:52,  3.44s/it]
n_combo #:  408
combo :  [0 3 4 5 6 8]
409it [15:55,  3.44s/it]
n_combo #:  409
combo :  [0 3 4 5 6 9]
410it [15:59,  3.51s/it]
n_combo #:  410
combo :  [0 3 4 5 7 8]
411it [16:02,  3.50s/it]
n_combo #:  411
combo :  [0 3 4 5 7 9]
412it [16:06,  3.49s/it]
n_combo #:  412
combo :  [0 3 4 5 8 9]
413it [16:11,  4.14s/it]
n_combo #:  413
combo :  [0 3 4 6 7 8]
414it [16:15,  4.11s/it]
n_combo #:  414
combo :  [0 3 4 6 7 9]
415it [16:19,  3.95s/it]
n_combo #:  415
combo :  [0 3 4 6 8 9]
416it [16:22,  3.74s/it]
n_combo #:  416
combo :  [0 3 4 7 8 9]
417it [16:25,  3.54s/it]
n_combo #:  417
combo :  [0 3 5 6 7 8]
418it [16:29,  3.46s/it]
```

```
n_combo #:  418
combo :  [0 3 5 6 7 9]
419it [16:32,  3.47s/it]
n_combo #:  419
combo :  [0 3 5 6 8 9]
420it [16:36,  3.60s/it]
n_combo #:  420
combo :  [0 3 5 7 8 9]
421it [16:40,  3.73s/it]
n_combo #:  421
combo :  [0 3 6 7 8 9]
422it [16:44,  3.73s/it]
n_combo #:  422
combo :  [0 4 5 6 7 8]
423it [16:47,  3.60s/it]
n_combo #:  423
combo :  [0 4 5 6 7 9]
424it [16:50,  3.43s/it]
n_combo #:  424
combo :  [0 4 5 6 8 9]
425it [16:53,  3.28s/it]
n_combo #:  425
combo :  [0 4 5 7 8 9]
426it [16:56,  3.21s/it]
n_combo #:  426
combo :  [0 4 6 7 8 9]
427it [16:59,  3.14s/it]
n_combo #:  427
combo :  [0 5 6 7 8 9]
428it [17:02,  3.10s/it]
n_combo #:  428
combo :  [2 3 4 5 6 7]
429it [17:05,  3.08s/it]
n_combo #:  429
combo :  [2 3 4 5 6 8]
430it [17:08,  3.03s/it]
n_combo #:  430
combo :  [2 3 4 5 6 9]
431it [17:12,  3.47s/it]
n_combo #:  431
combo :  [2 3 4 5 7 8]
432it [17:16,  3.44s/it]
n_combo #:  432
combo :  [2 3 4 5 7 9]
433it [17:19,  3.43s/it]
n_combo #:  433
combo :  [2 3 4 5 8 9]
434it [17:22,  3.29s/it]
```

```
n_combo #:  434
combo :  [2 3 4 6 7 8]
435it [17:25,  3.22s/it]
n_combo #:  435
combo :  [2 3 4 6 7 9]
436it [17:28,  3.13s/it]
n_combo #:  436
combo :  [2 3 4 6 8 9]
437it [17:31,  3.07s/it]
n_combo #:  437
combo :  [2 3 4 7 8 9]
438it [17:34,  3.00s/it]
n_combo #:  438
combo :  [2 3 5 6 7 8]
439it [17:37,  2.97s/it]
n_combo #:  439
combo :  [2 3 5 6 7 9]
440it [17:40,  2.95s/it]
n_combo #:  440
combo :  [2 3 5 6 8 9]
441it [17:43,  2.95s/it]
n_combo #:  441
combo :  [2 3 5 7 8 9]
442it [17:46,  2.92s/it]
n_combo #:  442
combo :  [2 3 6 7 8 9]
443it [17:49,  3.03s/it]
n_combo #:  443
combo :  [2 4 5 6 7 8]
444it [17:53,  3.27s/it]
n_combo #:  444
combo :  [2 4 5 6 7 9]
445it [17:56,  3.27s/it]
n_combo #:  445
combo :  [2 4 5 6 8 9]
446it [17:59,  3.17s/it]
n_combo #:  446
combo :  [2 4 5 7 8 9]
447it [18:02,  3.17s/it]
n_combo #:  447
combo :  [2 4 6 7 8 9]
448it [18:05,  3.11s/it]
n_combo #:  448
combo :  [2 5 6 7 8 9]
449it [18:08,  3.10s/it]
n_combo #:  449
combo :  [3 4 5 6 7 8]
450it [18:11,  3.07s/it]
```

```
n_combo #:  450
combo :  [3 4 5 6 7 9]
451it [18:14,  3.14s/it]
n_combo #:  451
combo :  [3 4 5 6 8 9]
452it [18:17,  3.10s/it]
n_combo #:  452
combo :  [3 4 5 7 8 9]
453it [18:20,  3.08s/it]
n_combo #:  453
combo :  [3 4 6 7 8 9]
454it [18:25,  3.38s/it]
n_combo #:  454
combo :  [3 5 6 7 8 9]
455it [18:28,  3.34s/it]
n_combo #:  455
combo :  [4 5 6 7 8 9]
456it [18:31,  3.26s/it]
n_combo #:  456
combo :  [0 2 3 4 5 6 7]
457it [18:37,  4.04s/it]
n_combo #:  457
combo :  [0 2 3 4 5 6 8]
458it [18:42,  4.35s/it]
n_combo #:  458
combo :  [0 2 3 4 5 6 9]
459it [18:45,  4.15s/it]
n_combo #:  459
combo :  [0 2 3 4 5 7 8]
460it [18:49,  4.01s/it]
n_combo #:  460
combo :  [0 2 3 4 5 7 9]
461it [18:53,  3.93s/it]
n_combo #:  461
combo :  [0 2 3 4 5 8 9]
462it [18:57,  3.87s/it]
n_combo #:  462
combo :  [0 2 3 4 6 7 8]
463it [19:00,  3.78s/it]
n_combo #:  463
combo :  [0 2 3 4 6 7 9]
464it [19:04,  3.82s/it]
n_combo #:  464
combo :  [0 2 3 4 6 8 9]
465it [19:08,  3.95s/it]
n_combo #:  465
combo :  [0 2 3 4 7 8 9]
466it [19:13,  4.28s/it]
```

```
n_combo #:  466
combo :  [0 2 3 5 6 7 8]
467it [19:20,  4.87s/it]
n_combo #:  467
combo :  [0 2 3 5 6 7 9]
468it [19:26,  5.37s/it]
n_combo #:  468
combo :  [0 2 3 5 6 8 9]
469it [19:32,  5.38s/it]
n_combo #:  469
combo :  [0 2 3 5 7 8 9]
470it [19:37,  5.32s/it]
n_combo #:  470
combo :  [0 2 3 6 7 8 9]
471it [19:41,  5.05s/it]
n_combo #:  471
combo :  [0 2 4 5 6 7 8]
472it [19:45,  4.73s/it]
n_combo #:  472
combo :  [0 2 4 5 6 7 9]
473it [19:49,  4.41s/it]
n_combo #:  473
combo :  [0 2 4 5 6 8 9]
474it [19:53,  4.19s/it]
n_combo #:  474
combo :  [0 2 4 5 7 8 9]
475it [19:56,  4.02s/it]
n_combo #:  475
combo :  [0 2 4 6 7 8 9]
476it [20:00,  3.96s/it]
n_combo #:  476
combo :  [0 2 5 6 7 8 9]
477it [20:04,  4.10s/it]
n_combo #:  477
combo :  [0 3 4 5 6 7 8]
478it [20:08,  4.02s/it]
n_combo #:  478
combo :  [0 3 4 5 6 7 9]
479it [20:12,  3.98s/it]
n_combo #:  479
combo :  [0 3 4 5 6 8 9]
480it [20:16,  3.95s/it]
n_combo #:  480
combo :  [0 3 4 5 7 8 9]
481it [20:20,  3.88s/it]
n_combo #:  481
combo :  [0 3 4 6 7 8 9]
482it [20:23,  3.82s/it]
```

```
n_combo #:  482
combo :  [0 3 5 6 7 8 9]
483it [20:27,  3.84s/it]
n_combo #:  483
combo :  [0 4 5 6 7 8 9]
484it [20:31,  3.88s/it]
n_combo #:  484
combo :  [2 3 4 5 6 7 8]
485it [20:37,  4.38s/it]
n_combo #:  485
combo :  [2 3 4 5 6 7 9]
486it [20:41,  4.19s/it]
n_combo #:  486
combo :  [2 3 4 5 6 8 9]
487it [20:44,  4.01s/it]
n_combo #:  487
combo :  [2 3 4 5 7 8 9]
488it [20:48,  3.87s/it]
n_combo #:  488
combo :  [2 3 4 6 7 8 9]
489it [20:51,  3.83s/it]
n_combo #:  489
combo :  [2 3 5 6 7 8 9]
490it [20:55,  3.78s/it]
n_combo #:  490
combo :  [2 4 5 6 7 8 9]
491it [20:59,  3.73s/it]
n_combo #:  491
combo :  [3 4 5 6 7 8 9]
492it [21:02,  3.67s/it]
n_combo #:  492
combo :  [0 2 3 4 5 6 7 8]
493it [21:07,  3.88s/it]
n_combo #:  493
combo :  [0 2 3 4 5 6 7 9]
494it [21:11,  3.99s/it]
n_combo #:  494
combo :  [0 2 3 4 5 6 8 9]
495it [21:15,  4.06s/it]
n_combo #:  495
combo :  [0 2 3 4 5 7 8 9]
496it [21:19,  4.08s/it]
n_combo #:  496
combo :  [0 2 3 4 6 7 8 9]
497it [21:23,  4.12s/it]
n_combo #:  497
combo :  [0 2 3 5 6 7 8 9]
498it [21:28,  4.13s/it]
```

```
n_combo #:  498
combo :  [0 2 4 5 6 7 8 9]
499it [21:32,  4.13s/it]
n_combo #:  499
combo :  [0 3 4 5 6 7 8 9]
500it [21:36,  4.17s/it]
n_combo #:  500
combo :  [2 3 4 5 6 7 8 9]
501it [21:40,  4.26s/it]
n_combo #:  501
combo :  [0 2 3 4 5 6 7 8 9]
502it [21:46,  2.60s/it]
502it [21:46,  2.60s/it]
```

In [16]:
```
# Store the combination matrix in a csv file
combo_df = pd.DataFrame.from_dict(combo_embedding)
combo_df.to_csv(results_path + 'combo_dataframe.csv', index=True)
```

In [17]:
```
# Making a readable dictionary of the combo_embeddings to later store in
readable_distance_dict = {
    'combination': [],
    'pair': [],
    'distance': []
}

# For every combination, for every test pair, store the distance
for nc, combo in enumerate(combo_embedding['combination']):
    for pair in all_pair_combinations:
        i = pair[0]
        j = pair[1]

        if i<j:
            dist = combo_embedding['distance_matrix'][nc][i, j]
            readable_distance_dict['combination'].append(combo)
            readable_distance_dict['pair'].append(pair)
            readable_distance_dict['distance'].append(dist)


# Store the distance matrix in a csv file
distance_df = pd.DataFrame.from_dict(readable_distance_dict)
distance_df.to_csv(results_path + 'distance_dataframe.csv', index=True)
```

In [ ]:
```

```

Analyses

Plotting the pairwise distance b/w a given pair across the several combinations

In [18]:
```
# Plot distances from all combinations for each pair of digit clusters
```

```
fig, ax = plt.subplots(1, figsize=(20, 4))
x_ticklabels = []

n_combos = len(combo_embedding['combination'])


# Iterate over all pairs of digits
for n_pair, test_pair in enumerate(all_pair_combinations):
    i = test_pair[0]
    j = test_pair[1]

    if i<j:
        # Fetch the pairwise digit for all combinations
        distances = [combo_embedding['distance_matrix'][combo_index][i, j

        # Discard distances = 0 => pair is not present in combo
        valid_distances = [d for d in distances if d != 0]
        n_valid_distances = len(valid_distances)

        # combo_indices = [combo_index for combo_index in np.arange(n_com


        print('test_pair: ', test_pair)
        print('n_valid_distances: ', n_valid_distances)

        # Plot the pairwise distances at a given x-coordinate
        ax.scatter(np.zeros((n_valid_distances))+n_pair, valid_distances,
        # im=ax.scatter(np.zeros((n_valid_distances))+n_pair, valid_dista

        # Add a boxplot of the distances of the test pair in different co
        ax.boxplot(valid_distances,
                patch_artist=False,  # fill with color
                positions = [n_pair],
                boxprops=dict(color='lightgrey'),
                whiskerprops=dict(color='lightgrey'),
                showfliers = False,
                medianprops = dict(color='black')
                )

        # Add the x-ticklabels
        tl = str(i) + '-' + str(j)
        x_ticklabels.append(tl)


# Plot formatting
ax.set_xticklabels(x_ticklabels, rotation=45);
ax.set_ylabel('Euclidean distance\nb/w test pair in diff combinations')
ax.set_xlabel('Test pair')

ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
# ax.spines['bottom'].set_visible(False)
# ax.spines['left'].set_visible(False)
```

```
ax.set_ylim(-.1, 1.5)
```

```
test_pair:  [0 2]
n_valid_distances:  128
test_pair:  [0 3]
n_valid_distances:  128
test_pair:  [0 4]
n_valid_distances:  128
test_pair:  [0 5]
n_valid_distances:  128
test_pair:  [0 6]
n_valid_distances:  128
test_pair:  [0 7]
n_valid_distances:  128
test_pair:  [0 8]
n_valid_distances:  128
test_pair:  [0 9]
n_valid_distances:  128
test_pair:  [2 3]
n_valid_distances:  128
test_pair:  [2 4]
n_valid_distances:  128
test_pair:  [2 5]
n_valid_distances:  128
test_pair:  [2 6]
n_valid_distances:  128
test_pair:  [2 7]
n_valid_distances:  128
test_pair:  [2 8]
n_valid_distances:  128
test_pair:  [2 9]
n_valid_distances:  128
test_pair:  [3 4]
n_valid_distances:  128
test_pair:  [3 5]
n_valid_distances:  128
test_pair:  [3 6]
n_valid_distances:  128
test_pair:  [3 7]
n_valid_distances:  128
test_pair:  [3 8]
n_valid_distances:  128
test_pair:  [3 9]
n_valid_distances:  128
test_pair:  [4 5]
n_valid_distances:  128
test_pair:  [4 6]
n_valid_distances:  128
test_pair:  [4 7]
n_valid_distances:  128
test_pair:  [4 8]
n_valid_distances:  128
test_pair:  [4 9]
```

```
n_valid_distances:  128
test_pair:  [5 6]
n_valid_distances:  128
test_pair:  [5 7]
n_valid_distances:  128
test_pair:  [5 8]
n_valid_distances:  128
test_pair:  [5 9]
n_valid_distances:  128
test_pair:  [6 7]
n_valid_distances:  128
test_pair:  [6 8]
n_valid_distances:  128
test_pair:  [6 9]
n_valid_distances:  128
test_pair:  [7 8]
n_valid_distances:  128
test_pair:  [7 9]
n_valid_distances:  128
test_pair:  [8 9]
n_valid_distances:  128
```

Out[18]: (-0.1, 1.5)

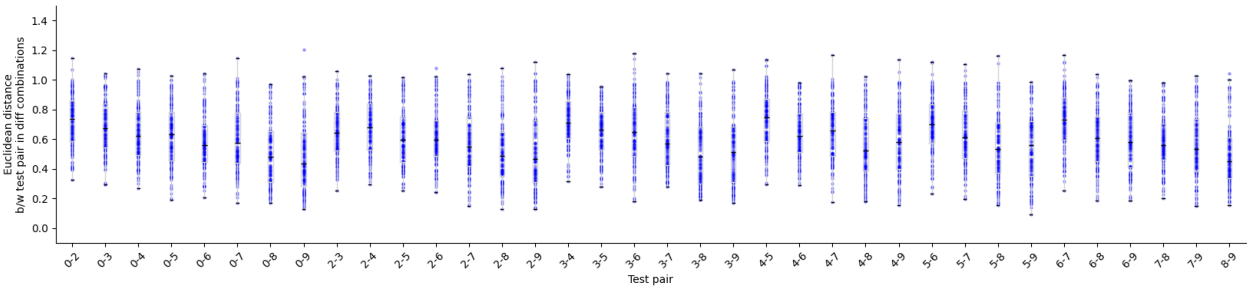


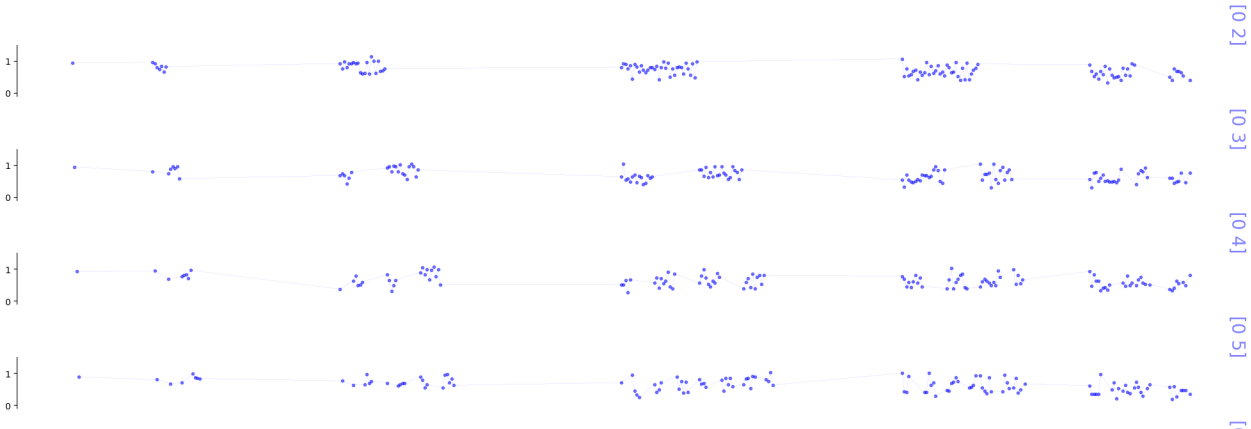
Fig: The x axis indicates the pair of digits being tested. The blue point shows the euclidean distance b/w the clusters of the test pair, in the UMAP representation of one combination. The box plot describes the distribution of pairwise distances for a given test pair across all valid combinations.

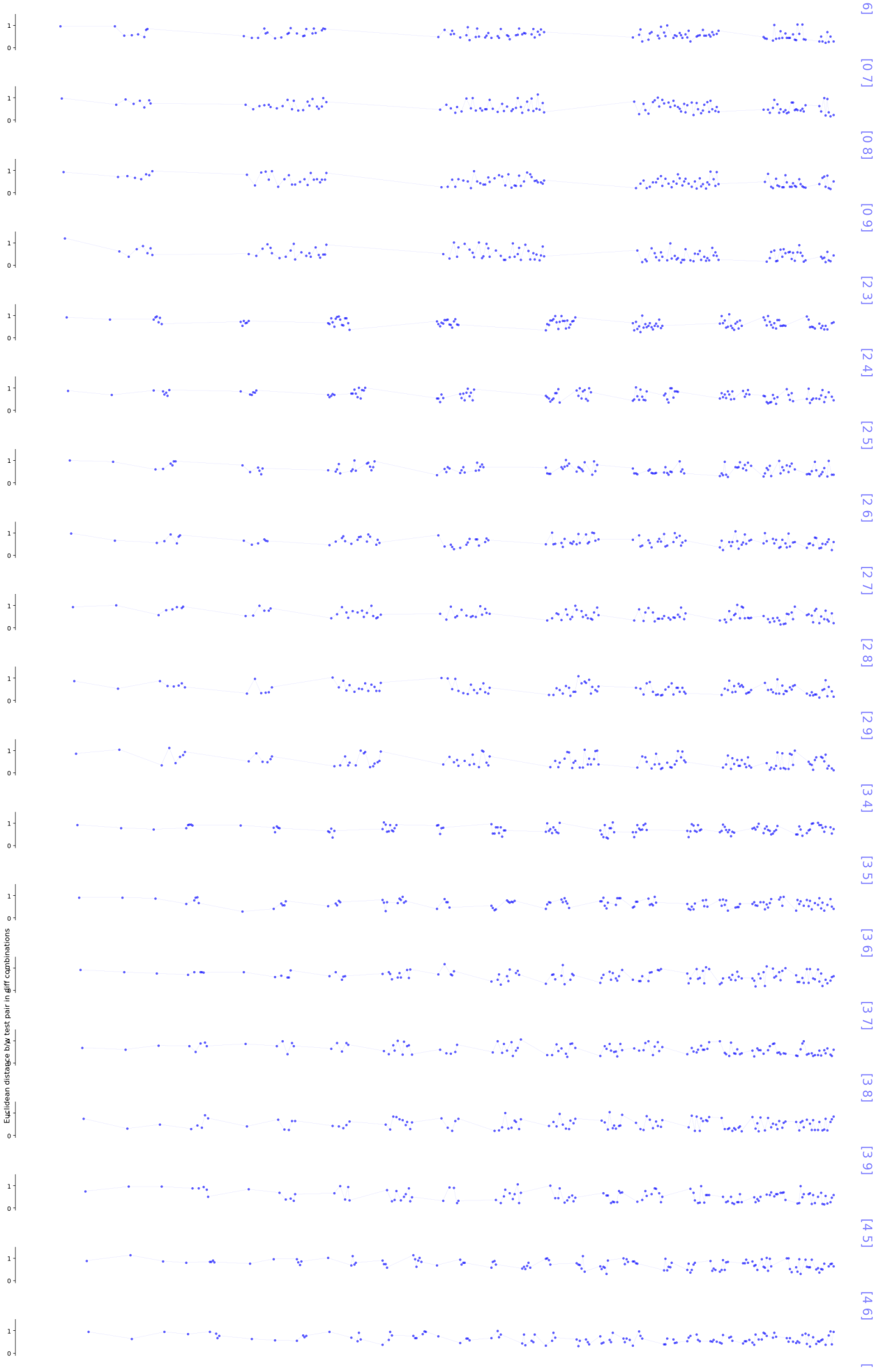
In [19]:
```
# Saving the figure
fig.savefig(results_path + 'pairwise_distances_across_pairs.png', bbox_in
```

In [22]:
```
```

In [31]:
```
# Plot distances for each pair of digit clusters across all combinations

fig, ax = plt.subplots(n_pairs, figsize=(20, 60), sharex=True)
x_ticklabels = []

# Iterate over all pairs of digits
for n_pair, test_pair in enumerate(all_pair_combinations):
    i = test_pair[0]
    j = test_pair[1]
```

```
    if i<j:
        # Fetch the pairwise digit for all combinations
        distances = np.array([combo_embedding['distance_matrix'][combo_in

        # Make x axis with the combinations
        x_combo  =  np.arange(n_combos)

        # Discard where distances = 0 => pair is not present in combo
        non_zero_indices = np.where(distances != 0)
        x_combo = x_combo[non_zero_indices]
        distances = distances[non_zero_indices]

        # Plot the pairwise distances across all combinations for a given
        ax[n_pair].plot(x_combo, distances, alpha=0.5, lw=0.1, label=str(


        # tl = str(i) + '-' + str(j)


    # Plot formatting
    ax[n_pair].set_title(str(test_pair), loc='right', color='blue', alpha

    ax[n_pair].set_ylim(-.1, 1.5)
    ax[n_pair].set_xticks([])
    ax[n_pair].tick_params(axis='x', which='both', bottom=False)

    ax[n_pair].spines['top'].set_visible(False)
    ax[n_pair].spines['right'].set_visible(False)
    ax[n_pair].spines['bottom'].set_visible(False)

if n_pair == n_pairs-1:
    x_ticklabels = combo_embedding['combination']
    ax[n_pair].spines['bottom'].set_visible(True)
    ax[n_pair].tick_params(axis='x', which='both', bottom=True)
    ax[n_pair].set_xticks(np.arange(n_combos)[::20])
    ax[n_pair].set_xticklabels(x_ticklabels[::20], rotation=45, ha="right

fig.supylabel('Euclidean distance b/w test pair in diff combinations')
fig.supxlabel('Combination')

fig.tight_layout()
```




Euclidean distance b/w test pair in diff combinations
[6]
[0 7]
[0 8]
[0 9]
[2 3]
[2 4]
[2 5]
[2 6]
[2 7]
[2 8]
[2 9]
[3 4]
[3 5]
[3 6]
[3 7]
[3 8]
[3 9]
[4 5]
[4 6]

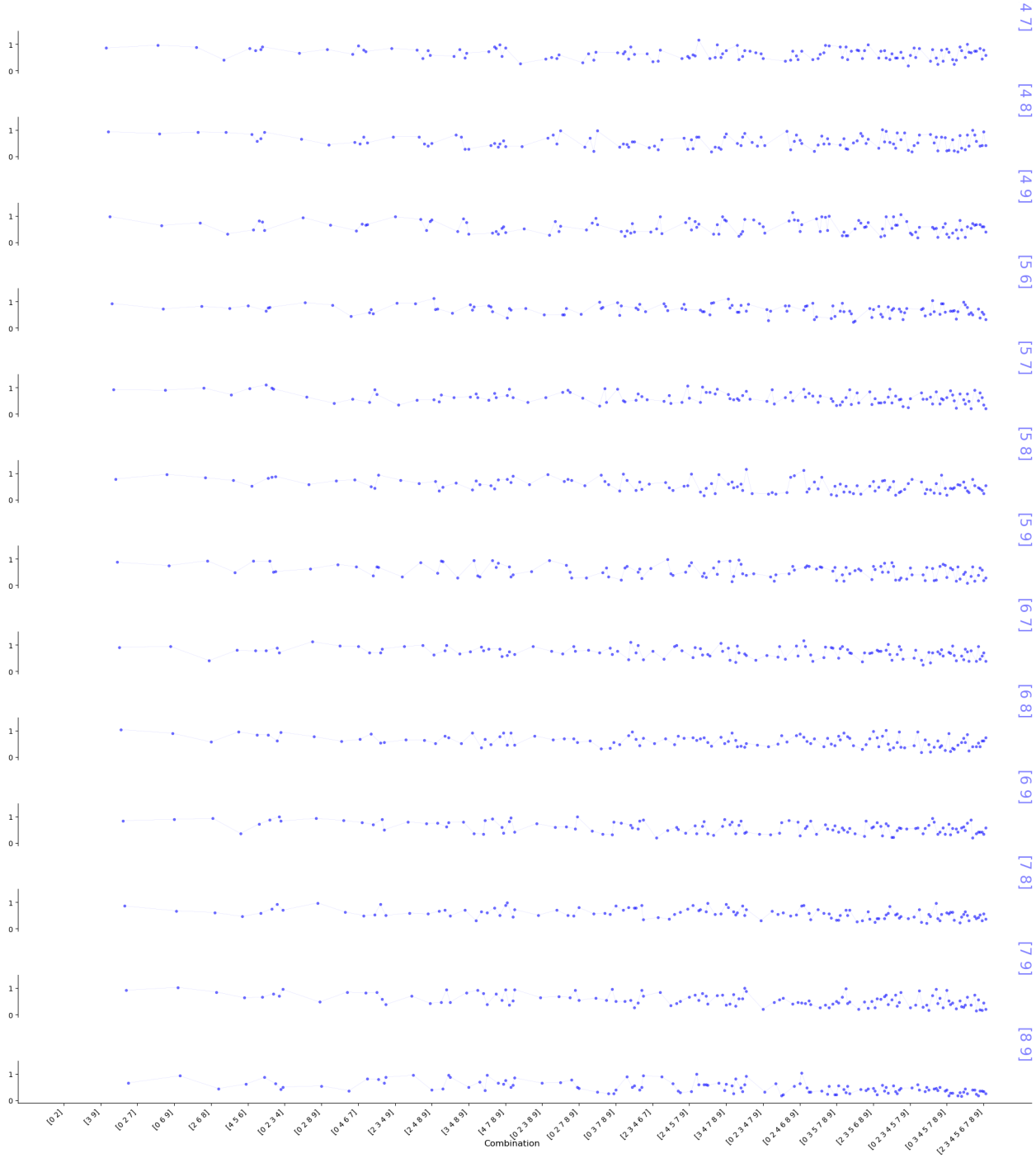


Fig: The x axis indicates the combination of digits being used to make the UMAP representation. The tested pair of digits is written in blue on the right edge of each subplot. The blue point shows the euclidean distance b/w the clusters of the test pair in the UMAP representation of the indicated combination. The blue line connects the pairwise distances of the same test pair in diff combinations.

In [32]:
```
# Saving the figure
fig.savefig('Figures/wo1_digits/exclusion/pairwise_distances_across_combo
```

In [ ]:
```

```