Statistical machine learning final report: Classification of handwritten digits

Adam Wolkowycki
University of Southern Denmark
adwol21@student.sdu.dk

May 31, 2023

1 Introduction

Handwritten digits classification is a classic problem in the field of machine learning and pattern recognition. With the widespread use of digital devices, the demand for automated recognition of handwritten digits has increased significantly. In the recent years, statistical machine learning techniques have been used extensively to address this problem. In this report, I present the results of my study on the classification of handwritten digits using Support Vector Machines (SVM) and Artificial Neural Networks (ANN). My choice can be justified by their proven ability to obtain high accuracy results in the image recognition problem. The goal is to determine which algorithm performs the best on the dataset in terms of accuracy and computation time.

2 Data preprocessing and augmentation

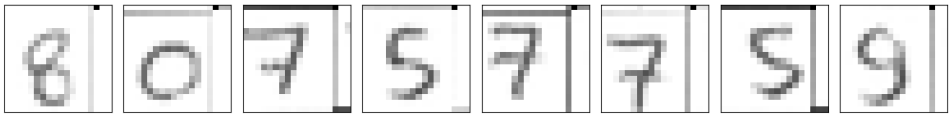
Data processing is a critical step that involves preparing the data for analysis and modeling. In this project I used custom dataset that consists of 24000 grayscale images, in the size of 18×18 pixels, handwritten by 12 students. I distinguished two cases: all-persons-in dataset, where we have data from all individuals in train and test set and disjunct set that splits the data according to the individuals. The splitting ratio was 9:1. One should notice, the raw dataset has some noise caused by inaccurate scanning, which can affect the performance of machine learning algorithms. Therefore, I implemented several preprocessing steps to remove the noise and improve quality of it.

2.1 Normalization

The dataset contains grayscale images of handwritten digits, with pixel values ranging from 0 to 255. To make the data more amenable, the pixel values had been simply normalized by dividing them by 255.

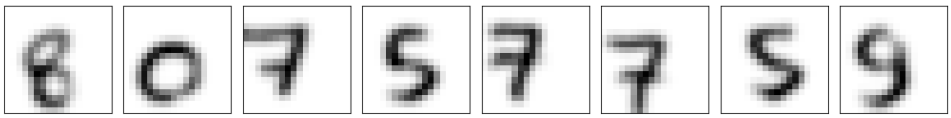
2.2 Noise removal

Due to a nature of the data collection process, the images in the dataset may contain noise, such as artifacts from the scanning process (fig. 1a) or variations in handwriting

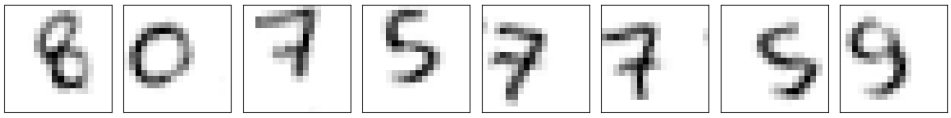


(a) Raw data with artifacts from the scanning process.

(b) Images denoised with Connected Components algorithm.



(c) Images blurred with Gaussian filter.



(d) Data augmented by applying random rotation and shift.

Figure 1: Data preprocessing and augmentation.

styles. To remove them, I used Connected Components algorithm (fig. 1b) that identifies and removes small, isolated regions of pixels in the image that are unlikely to be part of a digit.

2.3 Image smoothing

To further reduce noise and improve the quality of the images, to each image had been applied Gaussian filter with sigma value of 0.7 (fig. 1c). This smoothing technique helped to blur out any small fluctuations in pixel values that might not be relevant for classification.

2.4 Random rotation and shift

To increase the size and variability of the dataset, I performed random rotations and shifts on each image (fig. 1d). This was done by randomly rotating the image within an angle range of 15 degrees and shifting it horizontally and vertically within a distance range of 3 pixels in both directions. This helped to introduce variability in the dataset and prevent overfitting by ensuring that the model does not learn to rely on specific features of the training images.

2.5 Dimensionality reduction

The raw images in the dataset have a resolution of 18×18 pixels, resulting in 324 features per image. To reduce the dimensionality of the data and speed up the training process, I

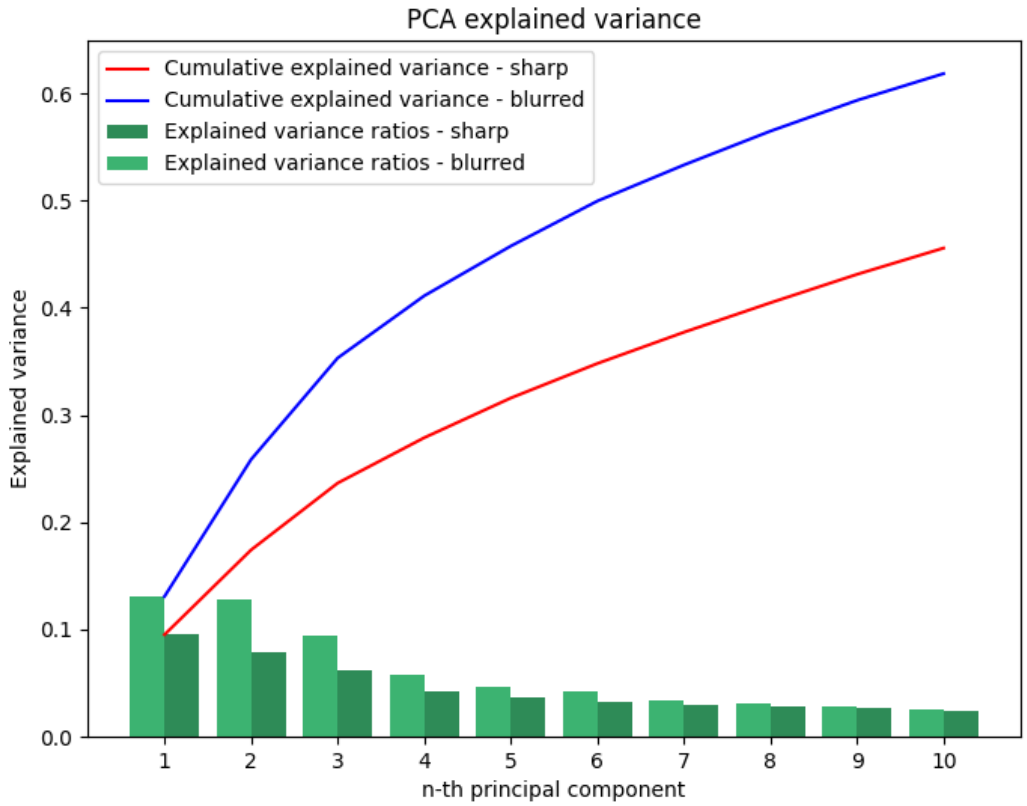


Figure 2: Variance ratios of sharp (dark green) and blurred (light green) datasets explained by their first 10 principal components followed by their cumulative sums (sharp - red, blurred - blue).

used Principal Component Analysis (PCA) to transform the 324 features into a smaller set of principal components that capture most of the variance in the data. I retained first 94 principal components of dataset with blurred images that explained 99% of its variance. This feature representation captures the most important patterns in the data and provides more compact and efficient input format. It is noteworthy that performing the same operation on dataset of the images that were not blurred, requires almost twice more principal components to explain its variance (fig 2).

3 Support Vector Machines

The SVM algorithm works by finding the hyperplane that maximizes the margin between two classes. Because there could be many of them that might classify the data, one reasonable choice is to find the best hyperplane that represents the largest separation, or margin, between the two classes. In the case of multi-class classification, multiple hyperplanes are trained to separate each class from the others.

The SVM classifier was trained on the preprocessed training set using scikit-learn's "SVC" class. I used a radial basis function (RBF) kernel to train it, which is a popular choice for image classification tasks. The regularization parameter (C) was set to 1.

One of the drawbacks of the SMVs is their computational complexity that can be

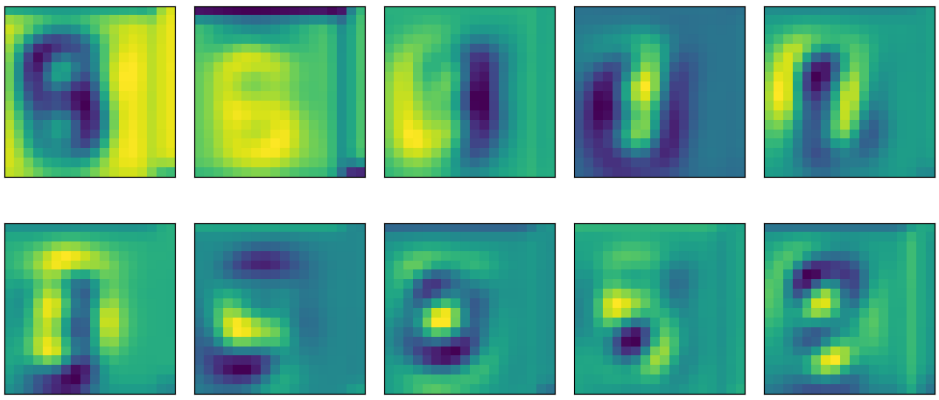


Figure 3: First 10 PCA eigenloadings of blurred dataset.

significant, especially when dealing with large datasets. Training SVMs involves solving a quadratic optimization problem, which can be time-consuming for datasets with a large number of samples or features.

It can even get worse when we decide to apply cross validation with SVM. It is a technique that assess the performance and generalization ability of a model. It involves dividing the available data into multiple subsets or "folds." The model is trained on a subset of the data and evaluated on the remaining fold. This process is repeated multiple times, with each fold serving as both a training and evaluation set. The results are then averaged to provide an estimate of the performance, helping to mitigate issues such as overfitting and providing a more robust evaluation of the model's capabilities. Applying cross validation with SVM in my case did not change the accuracy of the classifier in any significant way.

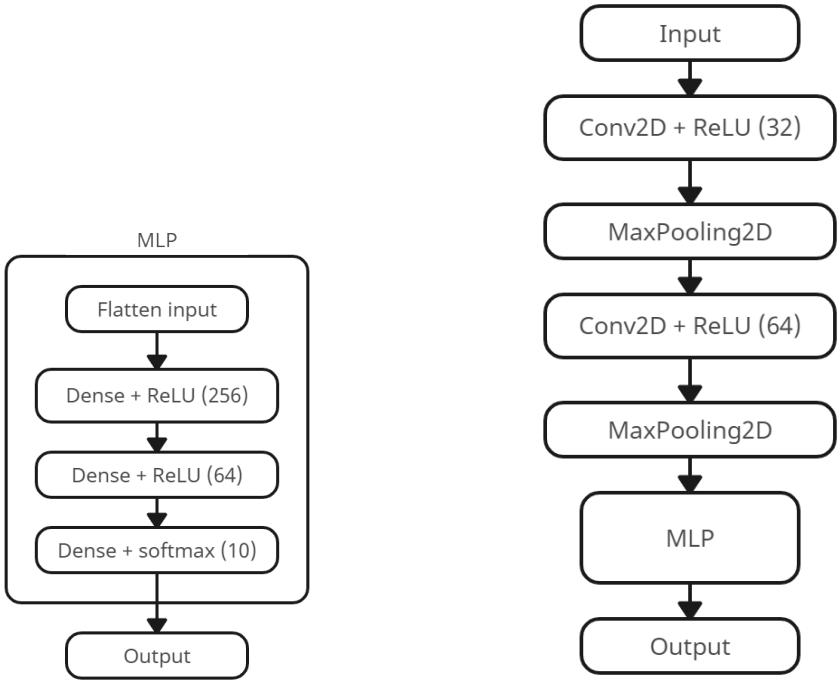
4 Artificial Neural Networks

The ANN algorithm works by building a network of interconnected nodes (neurons) that learn to extract features from the input data and make predictions. In the case of multi-class classification, the output layer of the network has multiple nodes, one for each class, and the network is trained to predict the probability distribution over these classes. In this report I am focusing of two neural network architectures: Multi-Layer Perceptron (MLP) and Convolutional Neural Network (CNN).

4.1 Multi-Layer Perceptron

Multi-Layer Perceptron (MLP), also known as a feedforward network is the fundamental neural net structure that features multiple layers of connected neurons. This model takes one-dimentional vector of signals as an input that implies in my implementation I had to flatten the images into them. Each of these vectors has the length of 324. the first fully connected layer has 256 neurons, and is followed by the second, which has 64 neurons. The output layer has 10 neurons with softmax activation function, representing the probabilities of the input image belonging to each of the 10 possible classes (fig. 4a).

The model is compiled using Adam optimizer, which is a popular gradient-based



(a) MLP. (b) CNN.

Figure 4: Architecture of neural network models.

optimization algorithm, and the sparse categorical cross-entropy loss function that is commonly used for multiclass classification problems where the classes are mutually exclusive. The accuracy metric is used to evaluate the performance of the model during training and testing.

4.2 Convolutional Neural Network

Convolutional Neural Networks (CNNs) are a type of neural network that are particularly well-suited for image classification tasks. They consist of multiple layers of filters that learn to extract features from the input data, such as edges, corners, and other patterns, at increasingly abstract levels of representation. I have implemented CNN model starting with the input layer that takes two-dimentional images as an input. The next layer is convolutional with 32 filters of size 3×3 and a ReLU activation function. This is followed by a max pooling layer with a pool size of 2×2, which reduces the spatial dimensions of the feature maps by a factor of 2. The second convolutional layer has 64 filters of size 3×3 and a ReLU activation function, followed by another max pooling layer with a pool size of 2×2. The resulting feature maps are then flattened into a one-dimensional array, which is fed into several dense layers that have been designed in the exact same way as Multi-Layer Perceptron (fig. 4b).

The model is compiled using Adam optimizer and the sparse categorical cross-entropy loss function. The accuracy metric is used to evaluate the performance of the model during training and testing. Unlike an MLP, a CNN takes into account the spatial structure of the input images. This is achieved through the use of convolutional layers, which apply filters to small regions of the input image and extract local features. Pooling layers are then used to reduce the spatial dimensions of the feature maps and capture

the most salient features.

5 Results

The table (tab. 1) presents results measured in the terms of accuracy. The scores indicate that all models achieved relatively high accuracy in classifying handwritten digits. The SVM model on PCA-reduced dataset demonstrated competitive performance, particularly on disjunct dataset. MLP exhibited slightly better performance on the training set for both datasets, but its accuracy on the test set was slightly lower compared to the SVM models. CNN outperformed all other models, achieving the highest accuracy scores on both datasets. The trick that made it possible was training the neural networks on the dataset, which has been enlarged ten times (240000 images) using augmentation techniques listed in the second chapter.

Table 1: Performance comparison

	All-Persons-In		Disjunct	
	Train	Test	Train	Test
SVM	0.9440	0.9121	0.9403	0.9032
SVM + PCA	0.9519	0.9196	0.9501	0.9125
MLP	0.9630	0.9146	0.9547	0.9063
CNN	0.9939	0.9858	0.9938	0.9840

Speaking of the computation time (tab. 2), the results indicate variations in the times among the different models. The SVM model had relatively shorter fitting times compared to the other models. The preprocessing with PCA significantly reduced the computation time of it. The MLP model had a longer fitting time due to the training of 200 epochs, but its prediction time was minimal. The CNN model required the longest computation time among all models due to its deeper architecture and training for 30 epochs.

Table 2: Computation time comparison

	Time [s]		
	Fitting	Prediction	Total
SVM	68.86	11.37	80.23
SVM + PCA	38.09	6.43	44.52
MLP (200 epochs)	195.65	0.20	195.85
CNN (30 epochs)	262.92	0.39	263.31

6 Conclusions

The results show that all models were able to achieve accuracy around 91% on the task of digit classification except CNN network that obtained superior accuracy around 98.5% on both datasets. The augmented dataset together with CNN's ability to capture spatial information through its convolutional layers were the reasons that contributed to its performance. CNNs are generally more effective than the other methods for image

classification tasks, especially when dealing with complex images with multiple objects or intricate spatial patterns. However, they can be more computationally expensive, sometimes return misclassifications (fig. 5), and require larger amounts of training data.

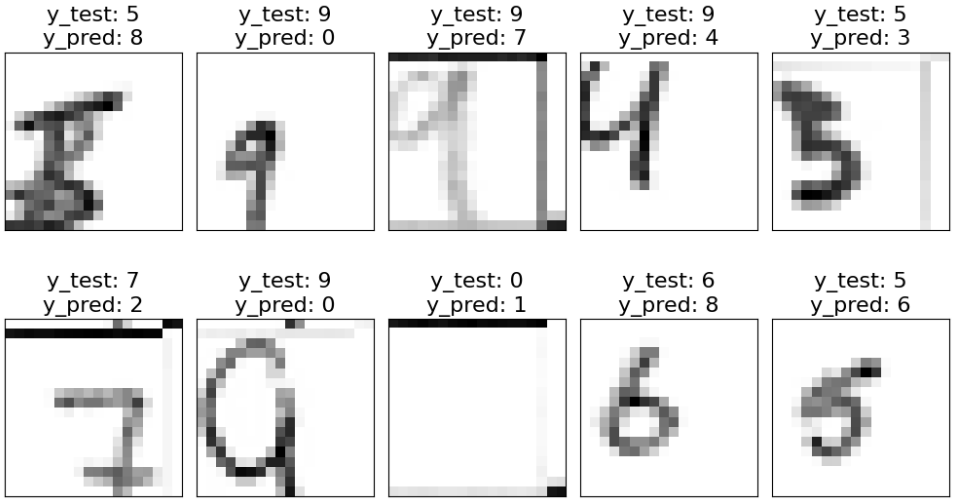


Figure 5: Images misclassified by CNN.

The research has also brought insights into various data preprocessing and augmentation techniques, including normalization, noise removal, image smoothing, random rotation and shift, and dimensionality reduction using Principal Component Analysis (PCA), to improve the performance of the models. My experiments showed that these techniques were effective in improving the accuracy of the models.

Overall, my study demonstrates the effectiveness of applied techniques in the task of handwritten digit classification. While all three models performed well, the CNN was the most accurate, indicating the importance of accounting for the spatial structure of the input images in image classification tasks.

7 Future work

More advanced data augmentation techniques such as elastic deformation and contrast adjustment could be explored. These techniques could further increase the diversity and robustness of the training dataset and improve the performance of the machine learning models.

The avenue worth considering is the utilization of ensemble methods, such as bagging, with multiple CNNs. By applying bagging to them, the diversity of the models can be leveraged to further enhance the accuracy and reliability of the classification results. Each CNN within the ensemble can be trained with e.g. different kernel sizes that would contribute to increase the final accuracy.

Another interesting direction for future work is the use of transformer models that have shown remarkable performance in various natural language processing (NLP) and computer vision tasks. Transformers are a type of deep learning model that uses self-attention mechanisms to capture dependencies between input data points. They have achieved state-of-the-art results in various NLP tasks, such as language translation and text classification, and are increasingly being applied to computer vision tasks as well.

One potential advantage of using transformer models for is their ability to capture long-range dependencies between input pixels. This could be particularly useful for recognizing complex patterns and structures in handwritten digits that are difficult to capture using traditional machine learning methods.

8 Appendix

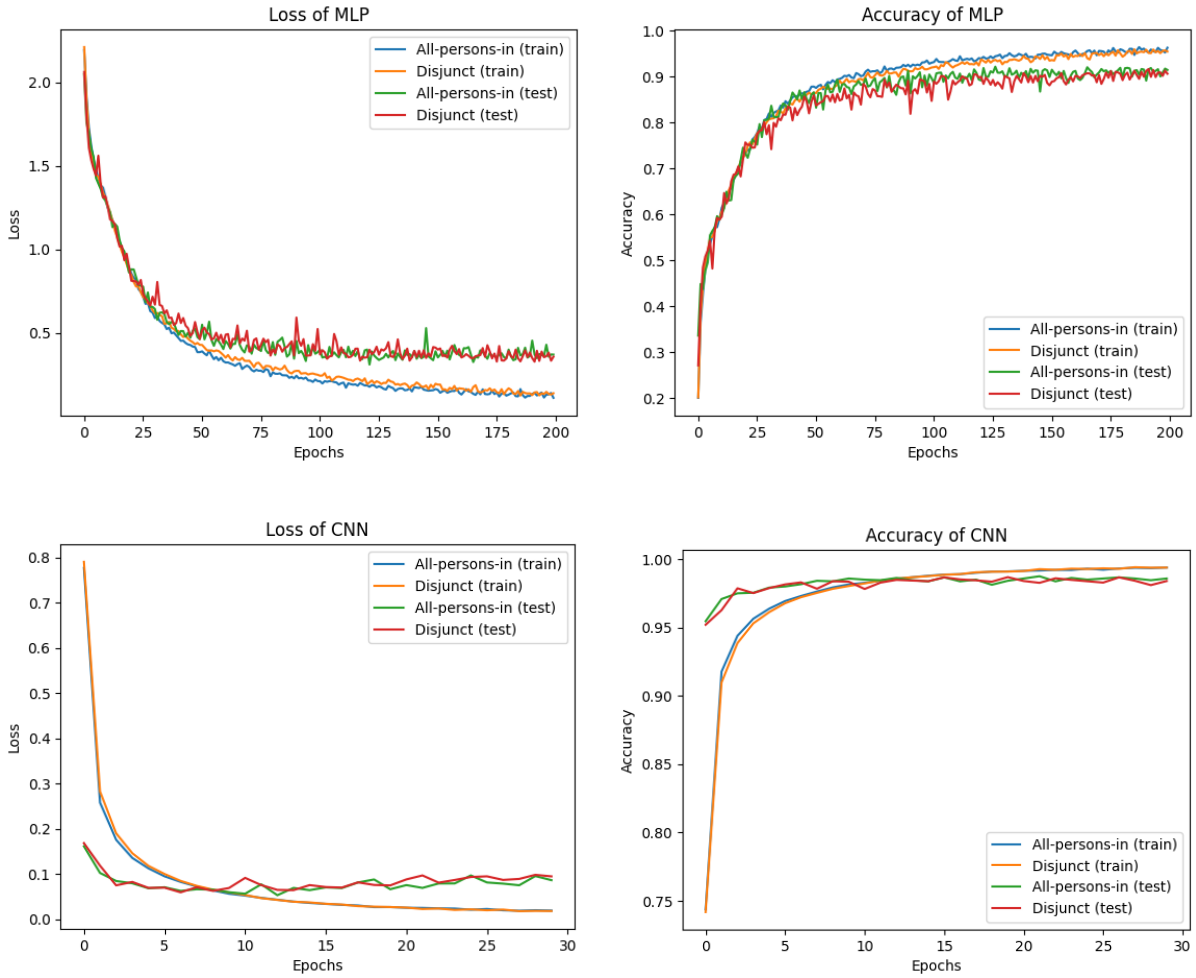


Figure 6: Performance of the neural networks.

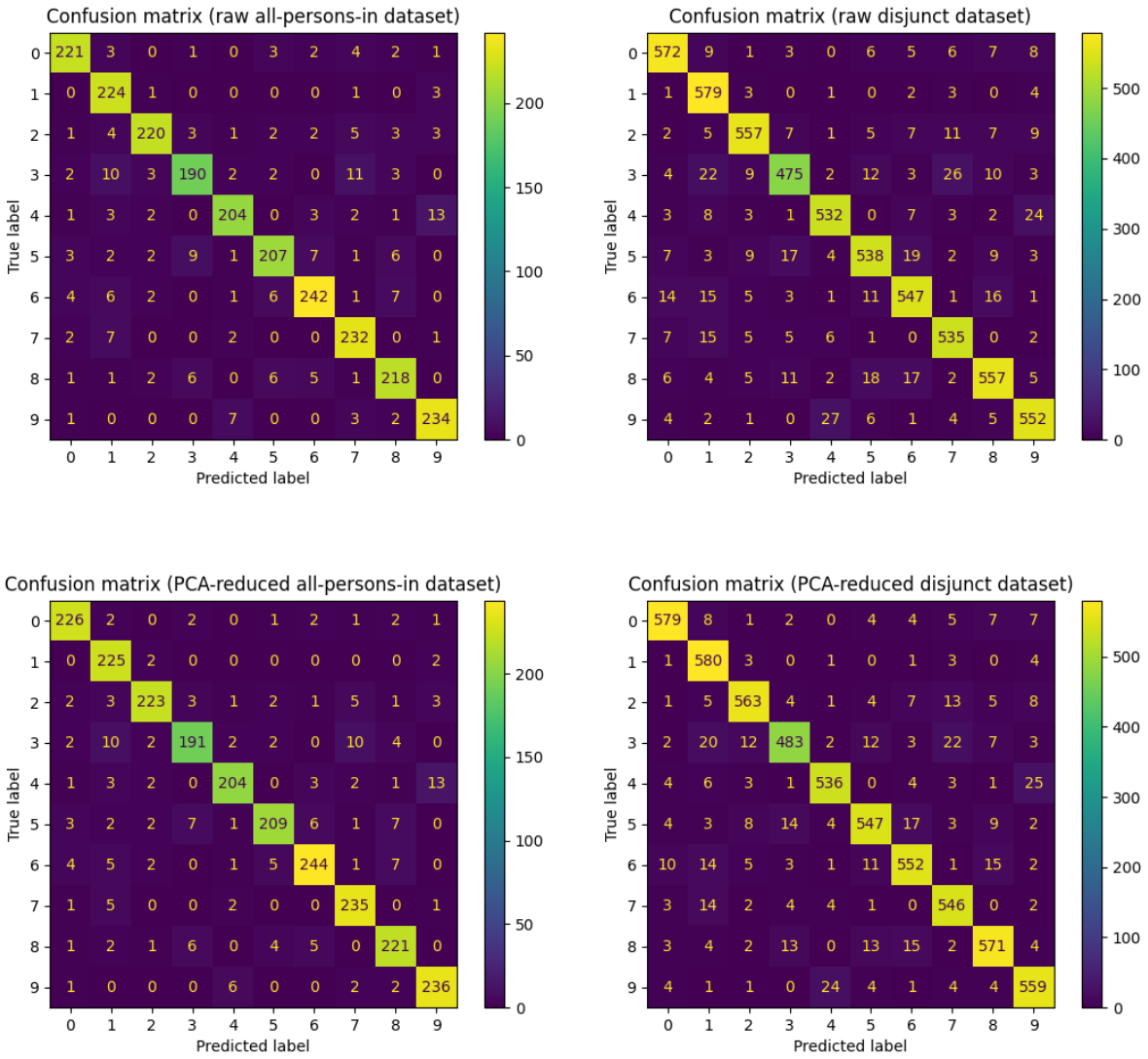


Figure 7: Confusion matrices of SVM.

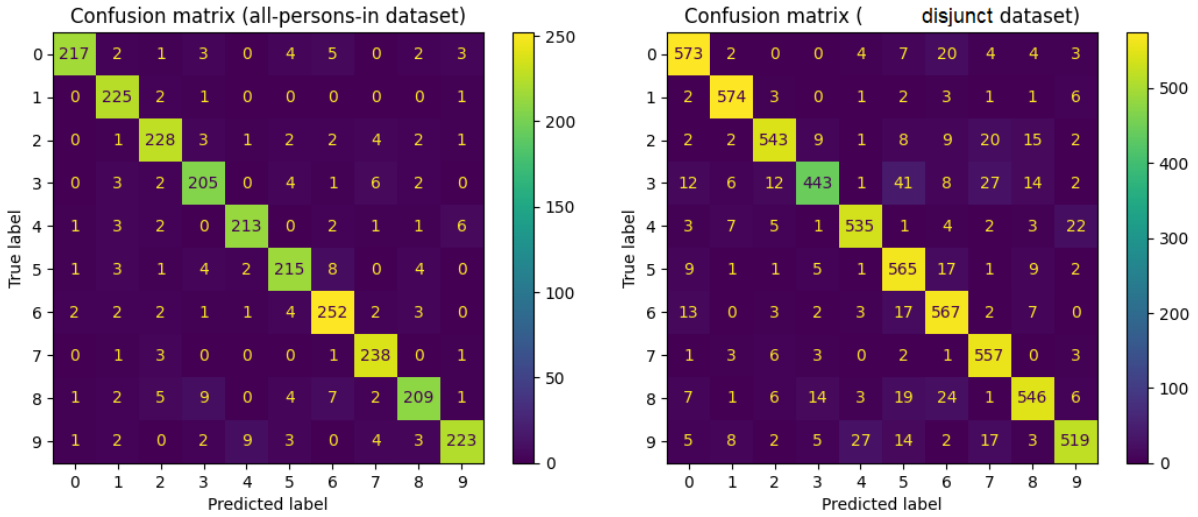


Figure 8: Confusion matrices of MLP.

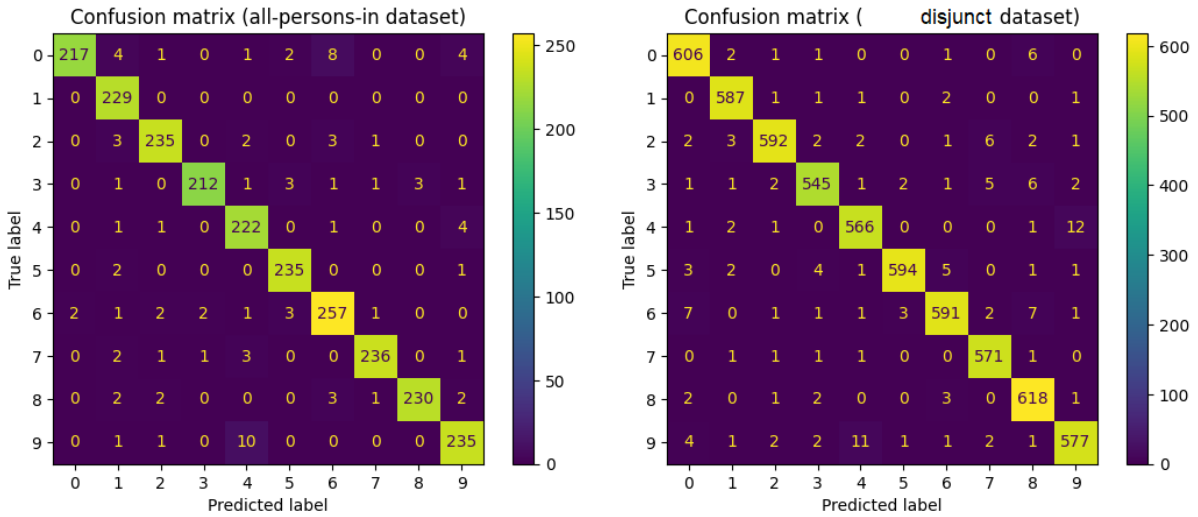


Figure 9: Confusion matrices of CNN.