Hand-written Digit Recognition Using Machine Learning

2023. 02. 09

S173102

PARK DOGNKYUN

Content

I. Overview of machine learning

II. Supervised learning

III. Hand-written digit recognition

  I. Minimum distance classifier

  II. Neural Network

  III. Principal Component Analysis (PCA)

  IV. Choosing best model

  V. Online demo (digit recognition web)

Machine Learning

- How to learn a machine
- training a computer to do a task (that humans do)

: Driving a vehicle, Translating from foreign language, Recognizing objects from photographs, etc.

Ex)

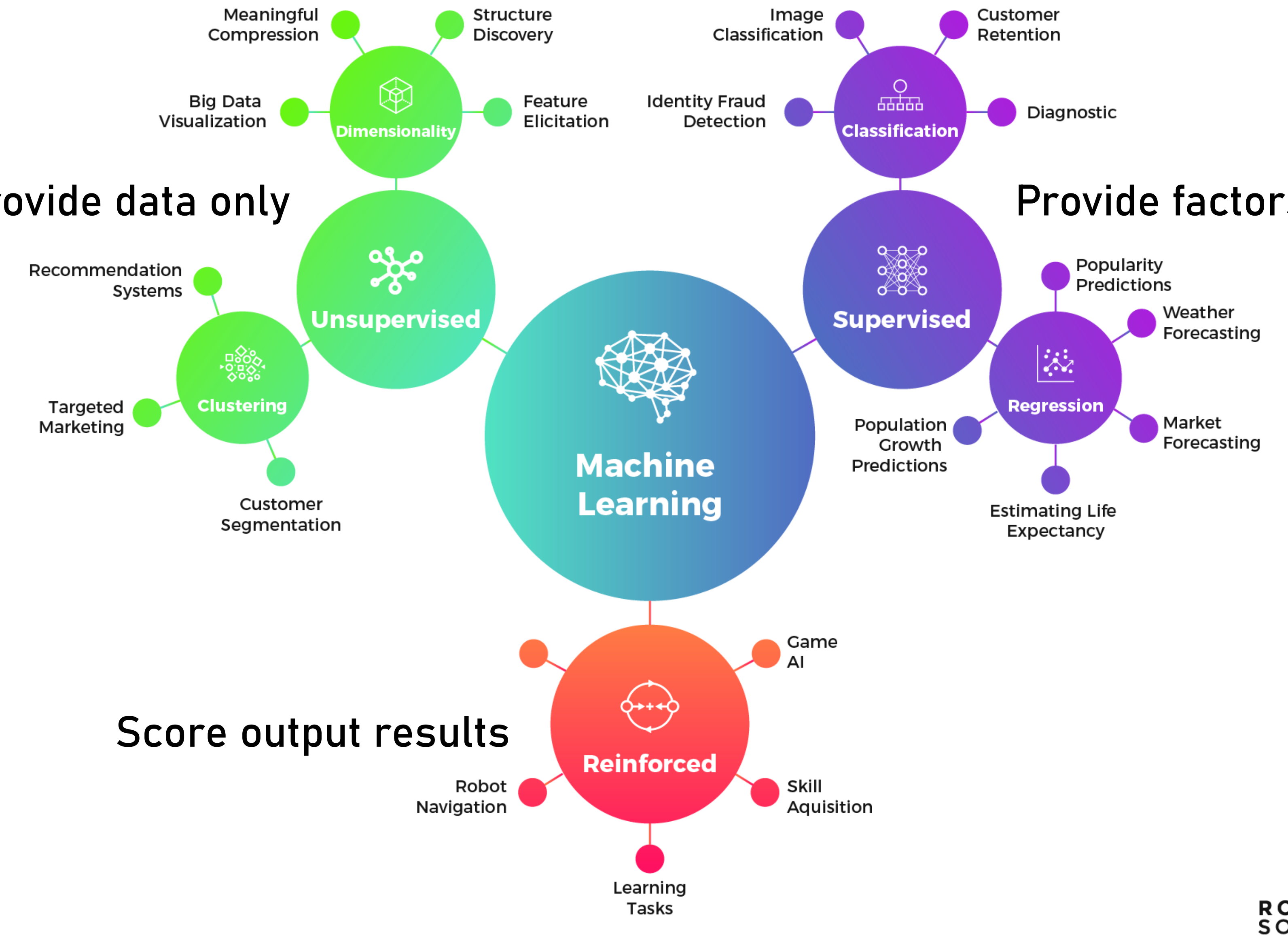
Digital voice assistants (Siri, Alexa), Chatbots(ChatGPT),

Go(碁) artificial intelligence, Autopilot, etc.

DIFFERENT TYPES OF MACHINE LEARNING

Provide data only

Provide factors and answers



Score output results

Machine Learning

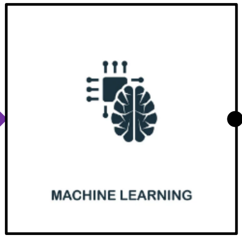
- Supervised Learning

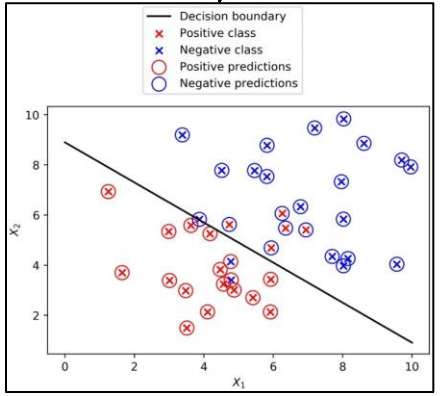
1. Input labeled data
2. Analyze data with predetermined factors
3. Creating a Decision Algorithm

Ex) Apple vs. Banana classification data

#	Data	Label	Shape	Color
1	🍎	Apple	Round	Red
2	🍏	Apple	Round	Green
3	🍌	Banana	Cylinder	Yellow
...	...	...	...	...

Factors





Machine Learning

- Supervised Learning

1. Input labeled data
2. Analyze data with predetermined factors
3. Creating a Decision Algorithm

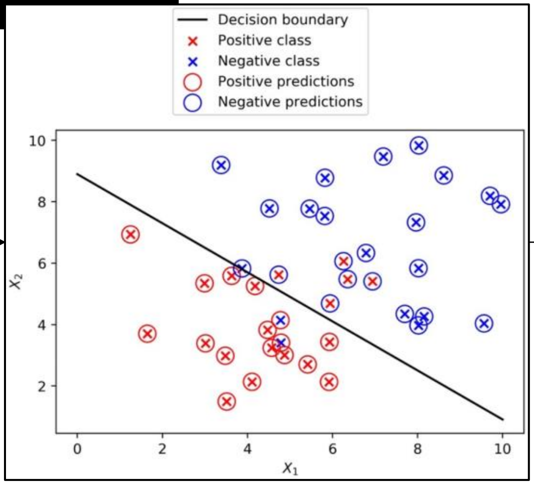
Ex) Apple vs. Banana classification data

New data OR Target data

Is this Apple? or Banana?

Use the factors of New or target data

'Cylinder' shape / 'Green' color



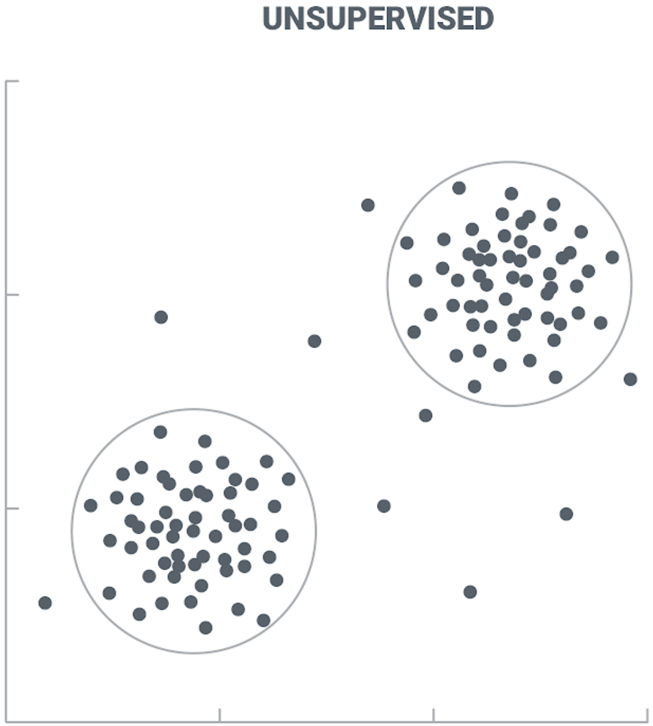
Banana

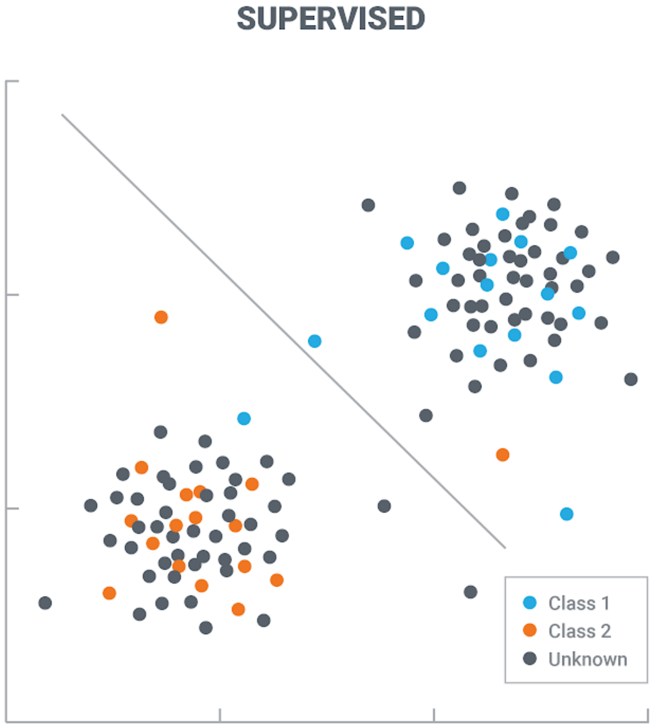
Machine Learning

- Unsupervised Learning

"Find out the rules and patterns on your own.“

Ex)

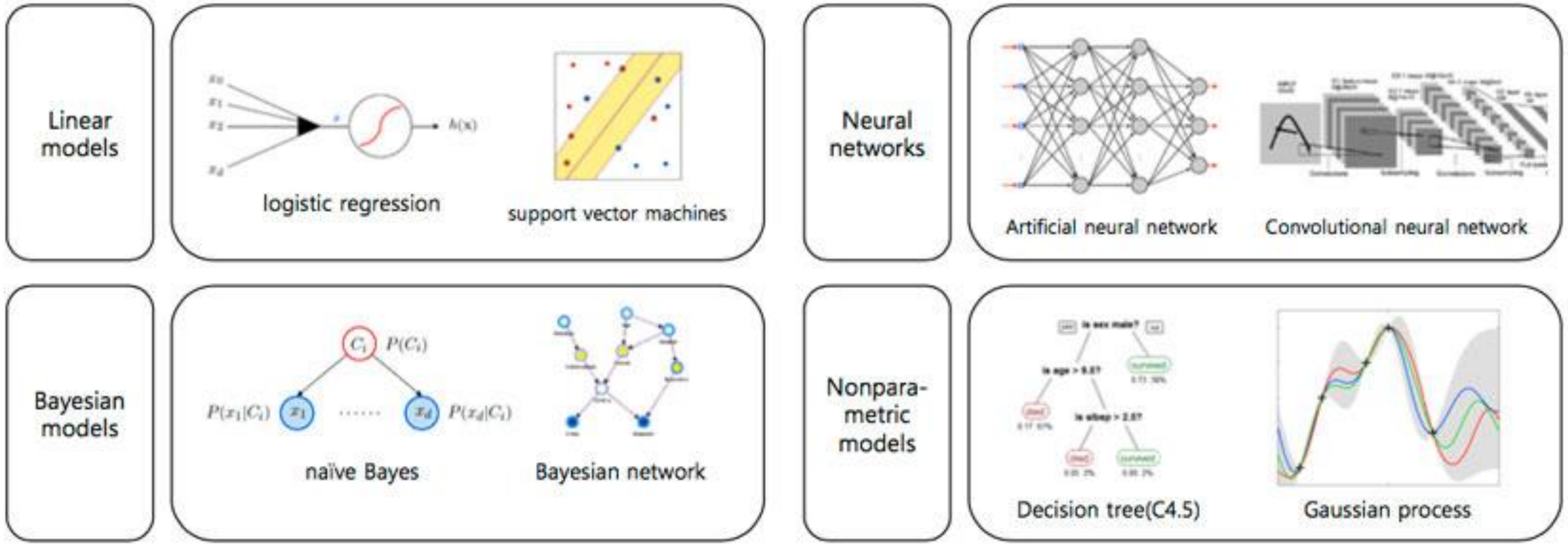




Machine Learning

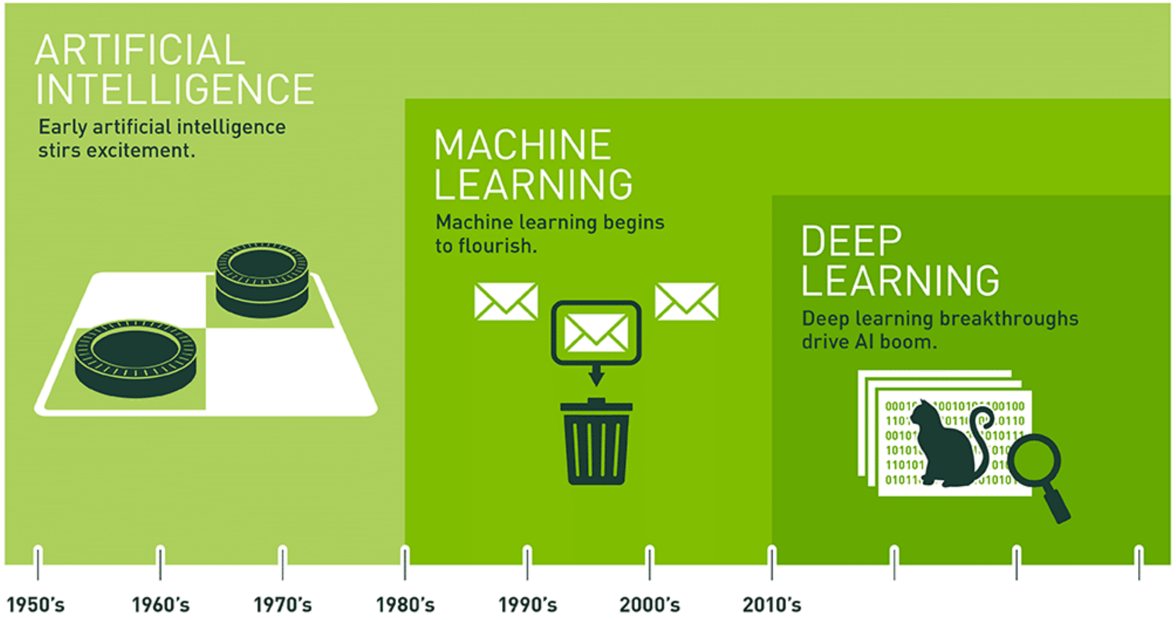
- Learning models

Just as people have different study methods for each subject, machine learning has different efficient learning methods depending on the situation.



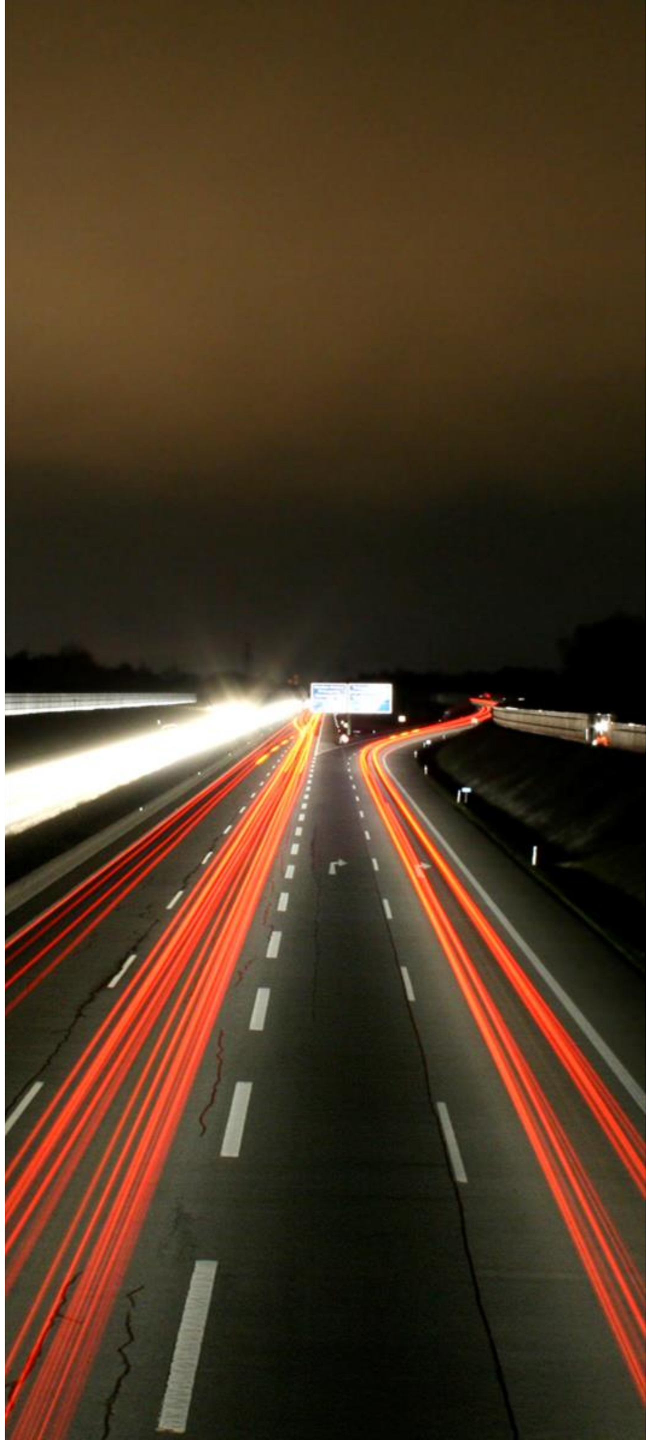
Deep Learning

- Machine Learning : The learning method of a machine
- Neural Network Model : One of the learning models
- Deep Learning : Techniques for using neural network models with many hidden layers



Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

Hand-written Digit recognition



Hand-written Digit recognition

- Classification Problem

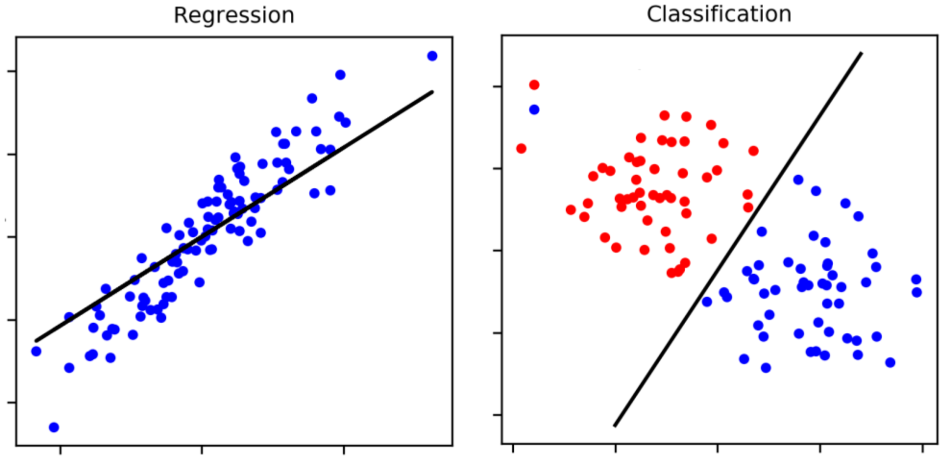
Machine learning classification problems are those which require the given data set to be classified in two or more categories.

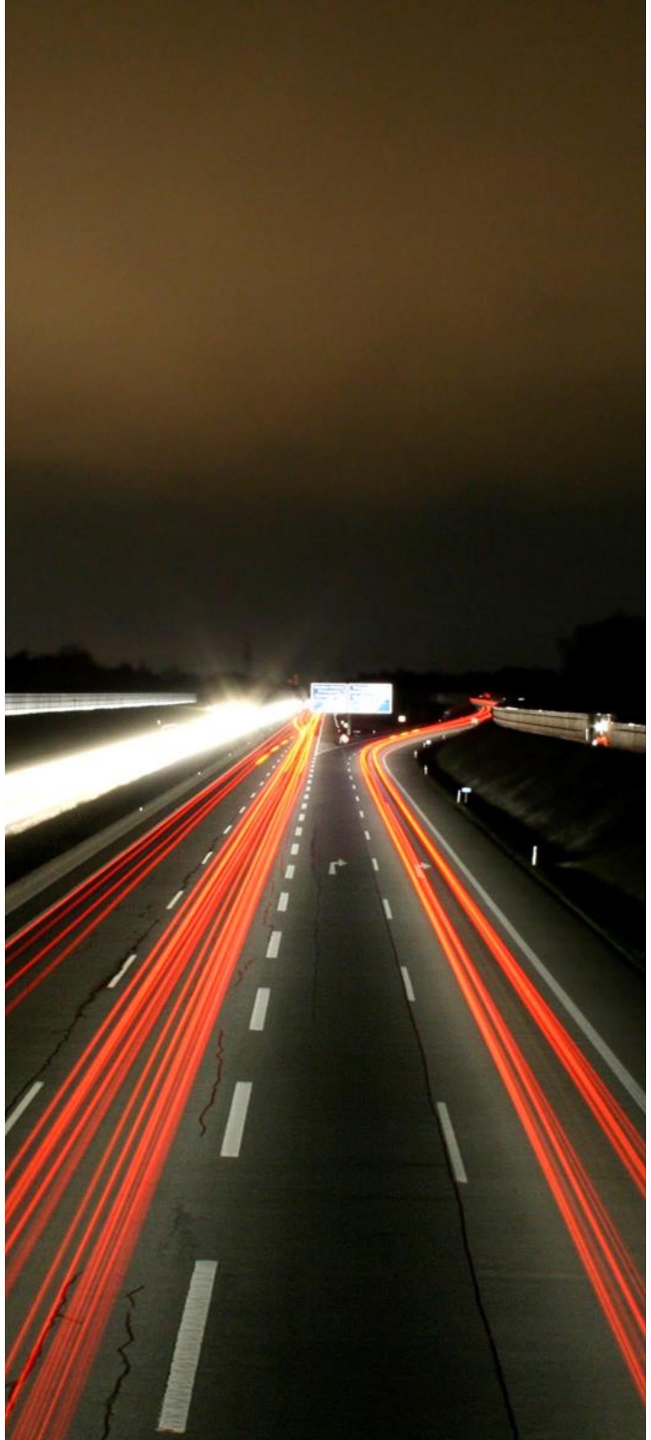
Classification model : Classification ← Categorical data

Ex) SVM(Support Vector Machine), Decision Tree, etc.

Cf. Regression model : Prediction ← Continuous data

Ex) Linear/ Non-linear Regression, Logistics Regression, etc.



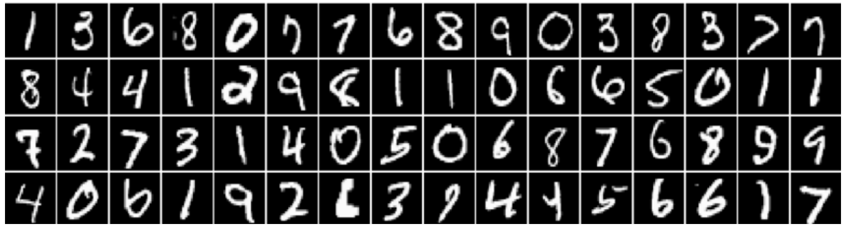


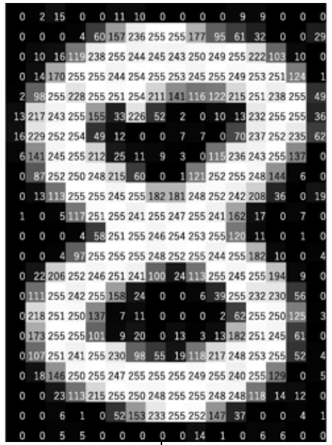
Hand-written Digit recognition

- Data Explanation

MNIST (Modified National Institute of Standards and Technology)

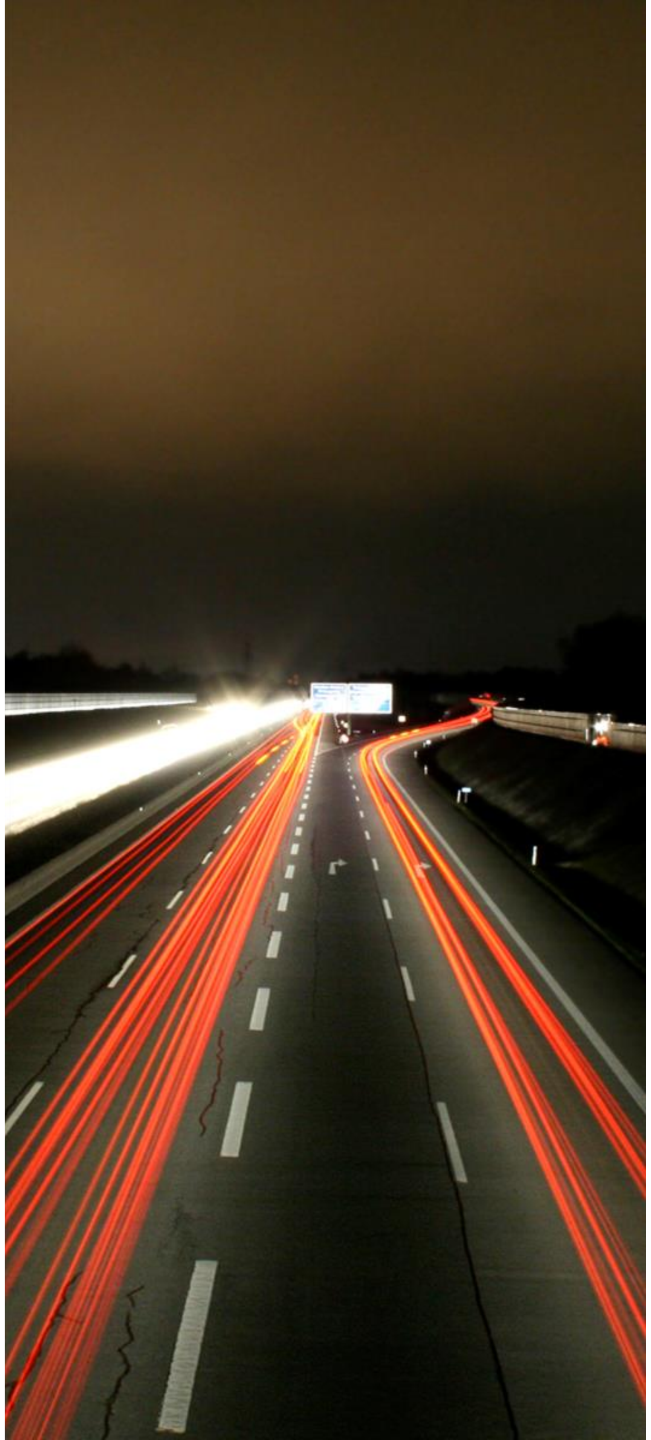
digit image data set (42000 images of digits)





0	2	15	0	0	11	10	0	0	0	0	9	9	0	0	0
0	0	0	4	60	157	236	255	255	177	95	61	32	0	0	29
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	49
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0
0	87	252	250	248	215	60	0	1	121	252	255	248	144	6	0
0	13	113	255	255	245	255	182	181	248	252	242	208	36	0	19
1	0	5	117	251	255	241	255	247	255	241	162	17	0	7	0
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4
0	22	206	252	246	251	241	100	24	113	255	245	255	194	9	0
0	111	255	242	255	158	24	0	0	6	39	255	232	230	56	0
0	218	251	250	137	7	11	0	0	0	2	62	255	250	125	3
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4
0	18	146	250	255	247	255	255	255	249	255	240	255	129	0	5
0	0	23	113	215	255	250	248	255	255	248	248	118	14	12	0
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1
0	0	5	5	0	0	0	0	0	14	1	0	6	6	0	0

	1	2	3	4	…	784	785
#	Label	pixel0	pixel1	pixel2	…	pixel782	Pixel783
36471	8	0	2	15	…	0	0



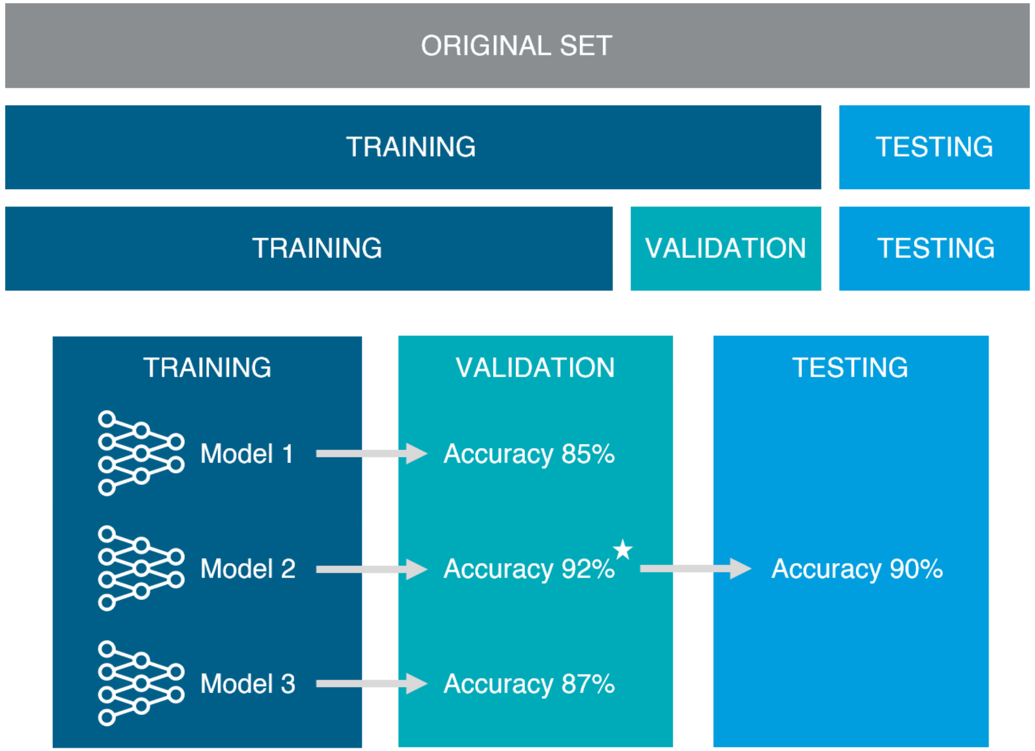
Hand-written Digit recognition

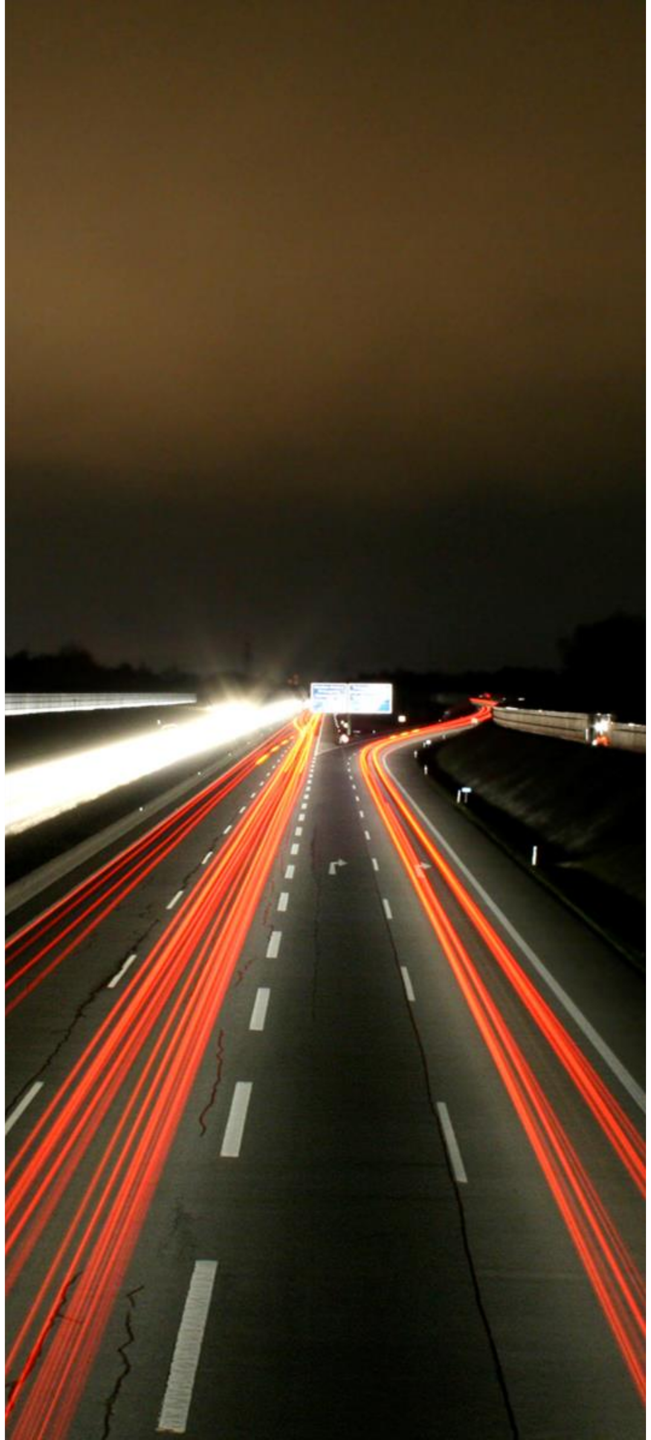
- Training set / Test set

Training set : Data set used to train the model

Test set : Used to measure the expected performance of the model.

* Validation set : Data set that is applied to each of the different models to measure performance and is used to select the final model.





Hand-written Digit recognition

- Training set / Test set

We split the data into training set(75%) and test set(25%)

Split Data

```
[6]: ratio = 25 # Split ratio(Test Sets Ratio)[%]
     ratio = ratio/100
```

(test set)ratio = 25%

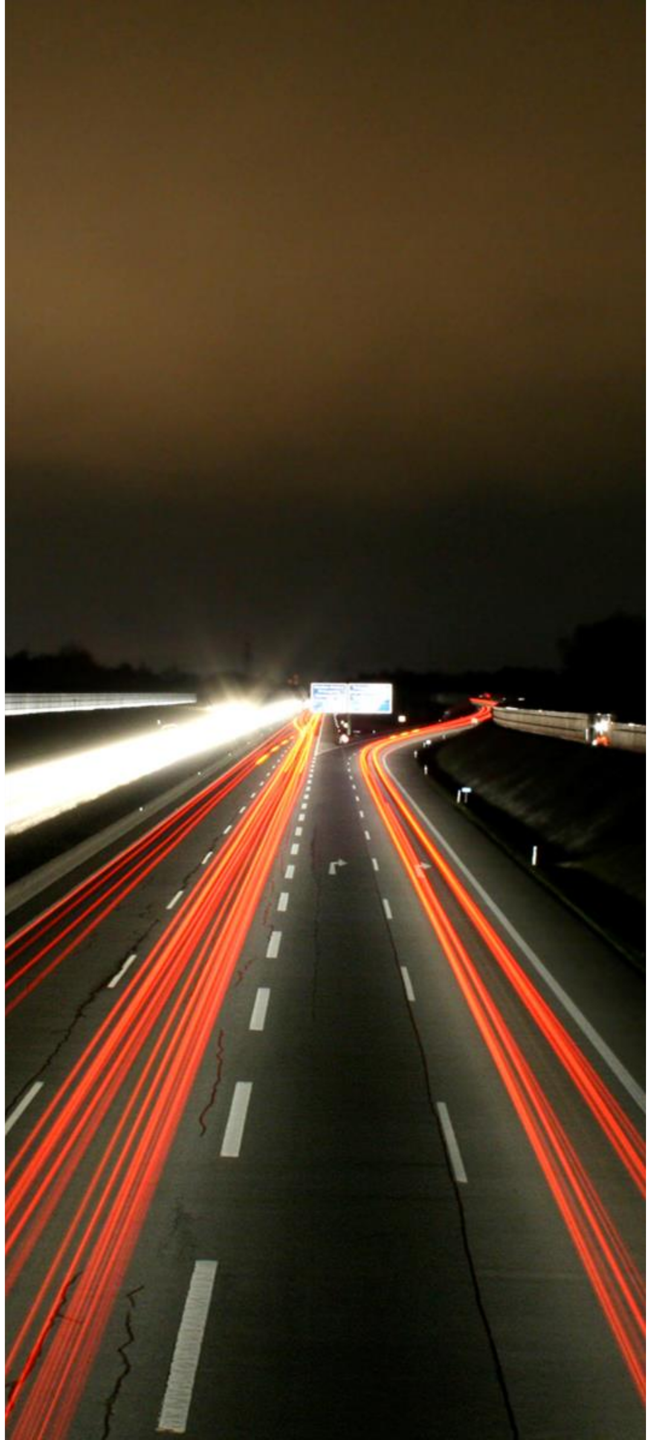
```
[7]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = ratio, shuffle=True, random_state=99)
```

```
[8]: print(y_train.value_counts().sort_index())
     print(y_train.count())
     train_No = y_train.count()
     print(y_test.value_counts().sort_index())
     print(y_test.count())
     test_No = y_test.count()
```

```
0    3104
1    3498
2    3108
3    3230
4    3085
5    2871
6    3086
7    3326
8    3077
9    3115
Name: label, dtype: int64
31500
0    1028
1    1186
2    1069
3    1121
4     987
5     924
6    1051
7    1075
8     986
9    1073
Name: label, dtype: int64
10500
```

Number of training set data = 31500

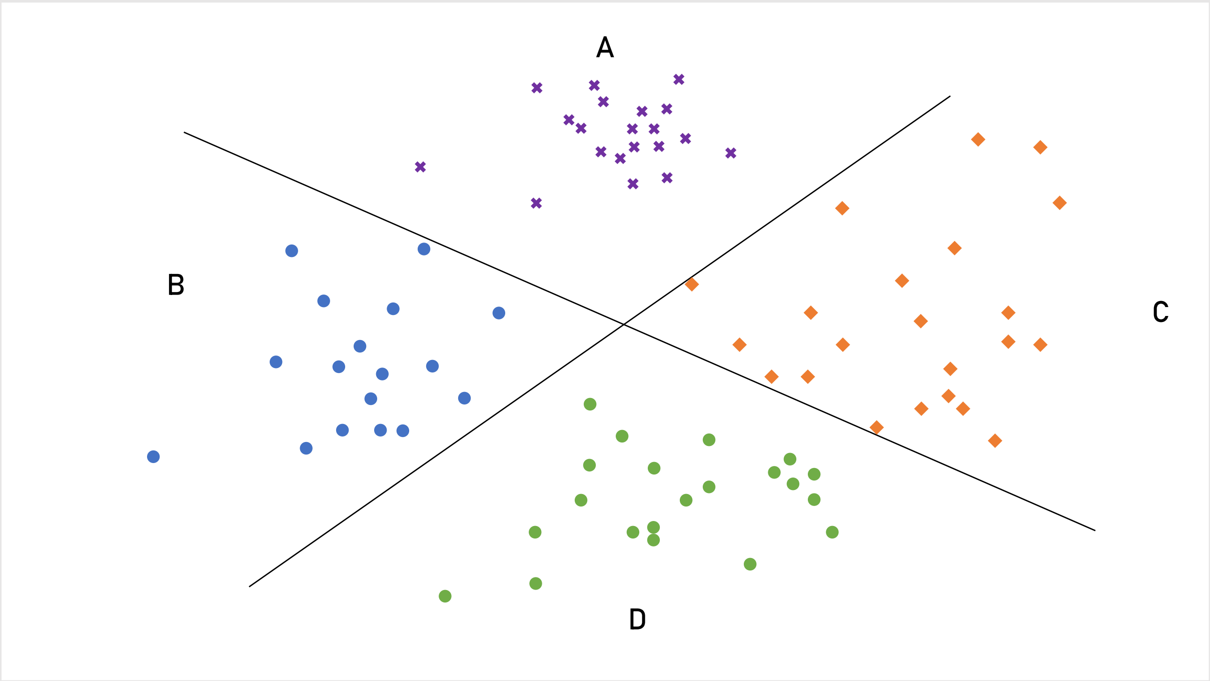
Number of test set data = 10500

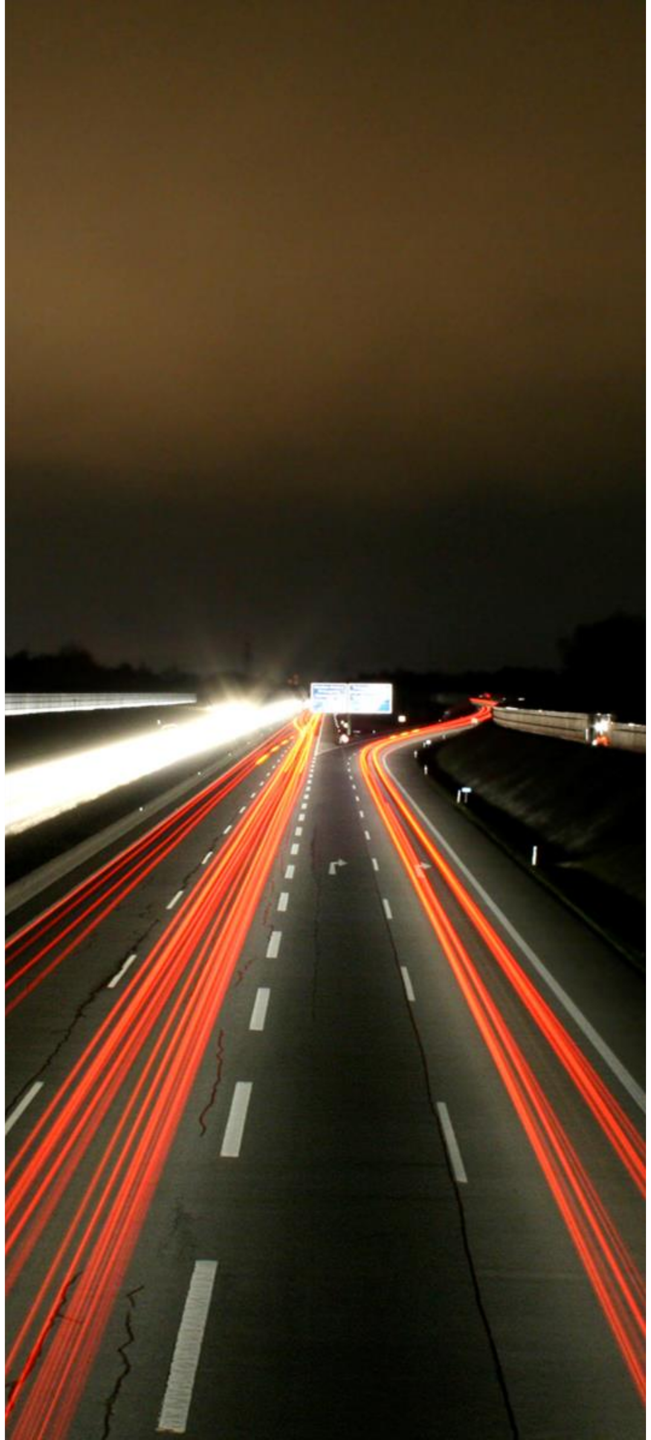


Hand-written Digit recognition

- Minimum Distance Classifier Algorithm

1. Select random centroids-data of each group
2. Calculate each distance from the new data or target data
3. Determine the data by the centroid at the minimum distance

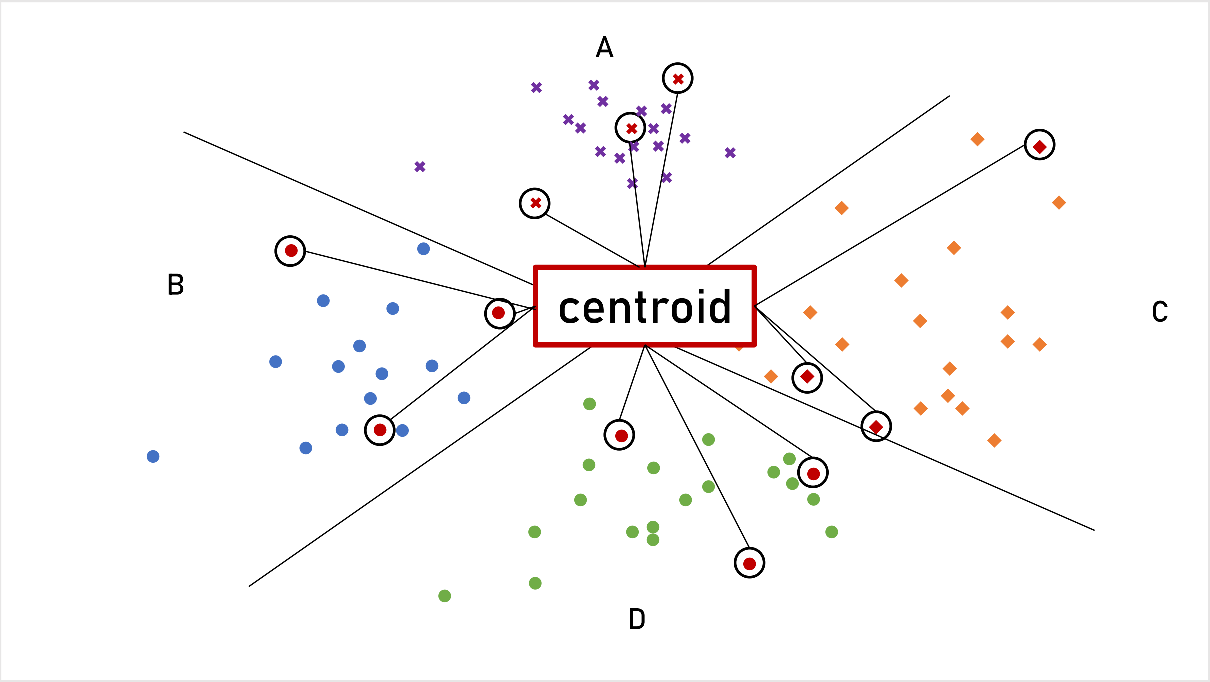


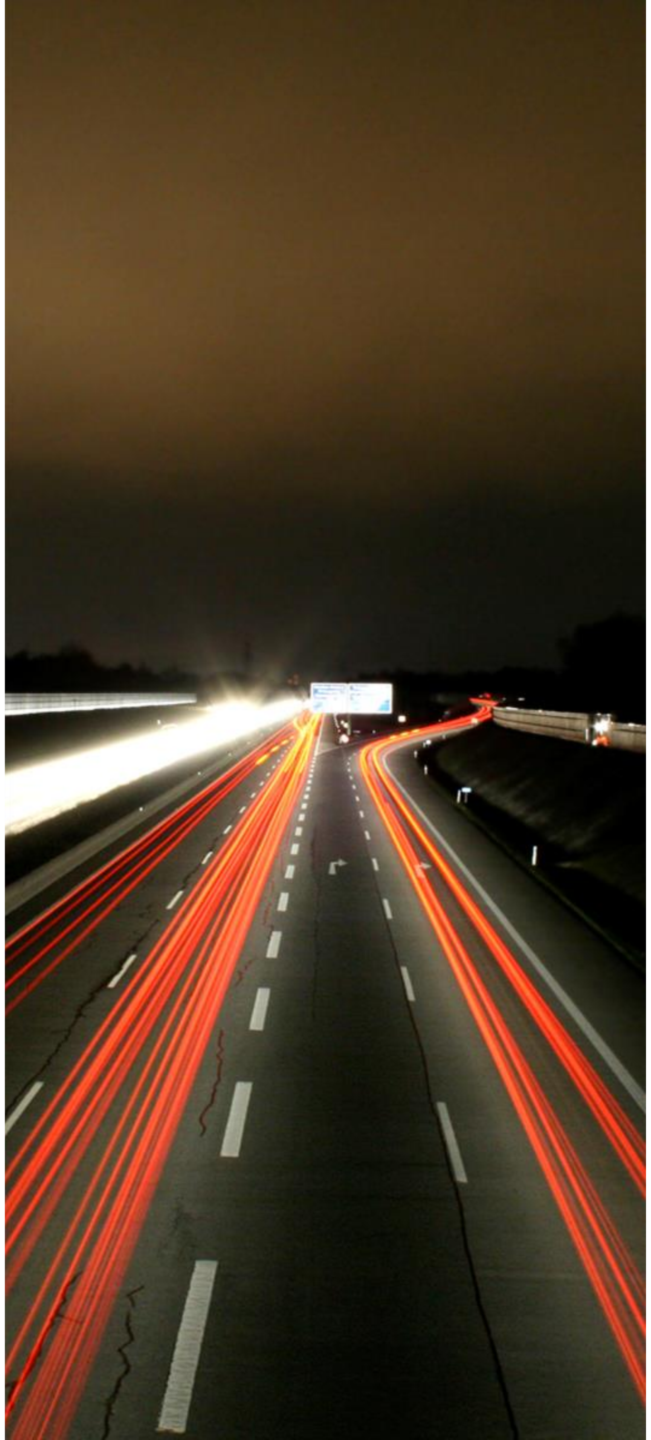


Hand-written Digit recognition

- Minimum Distance Classifier Algorithm

1. Select random centroids-data of each group
2. Calculate each distance from the new data or target data
3. Determine the data by the centroid at the minimum distance

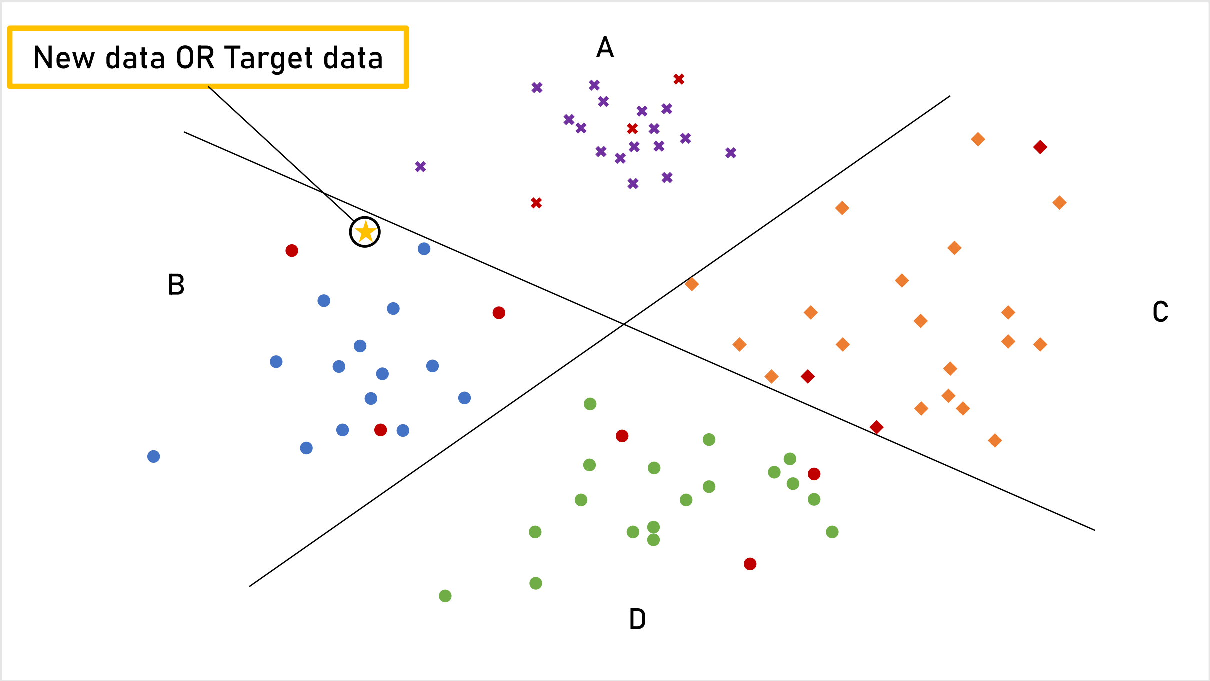


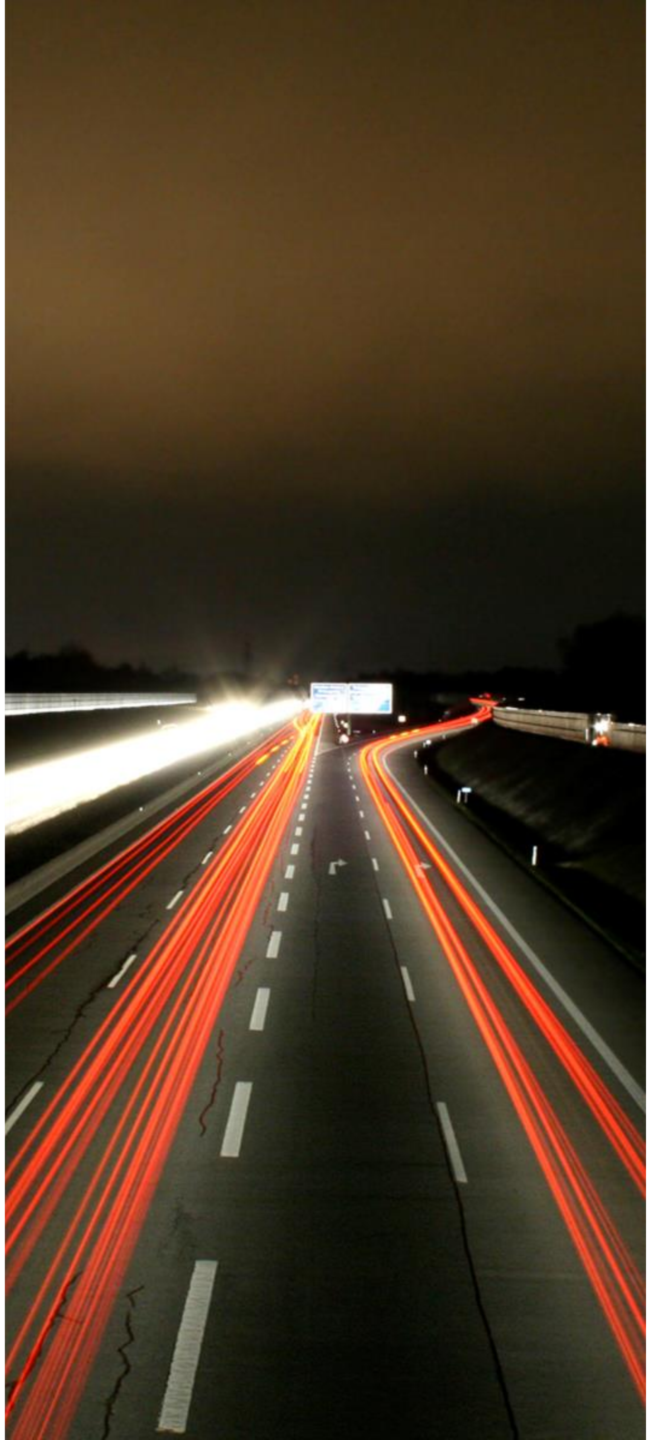


Hand-written Digit recognition

- Minimum Distance Classifier Algorithm

1. Select random centroids-data of each group
2. Calculate each distance from the new data or target data
3. Determine the data by the centroid at the minimum distance

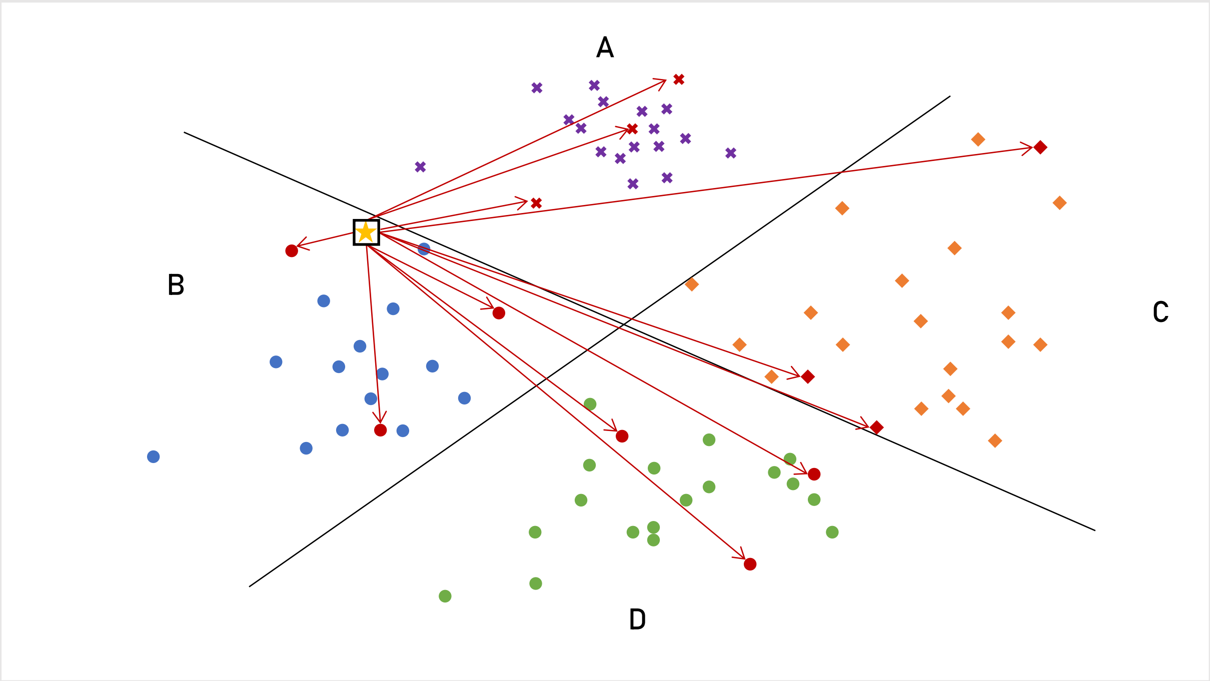




Hand-written Digit recognition

- Minimum Distance Classifier Algorithm

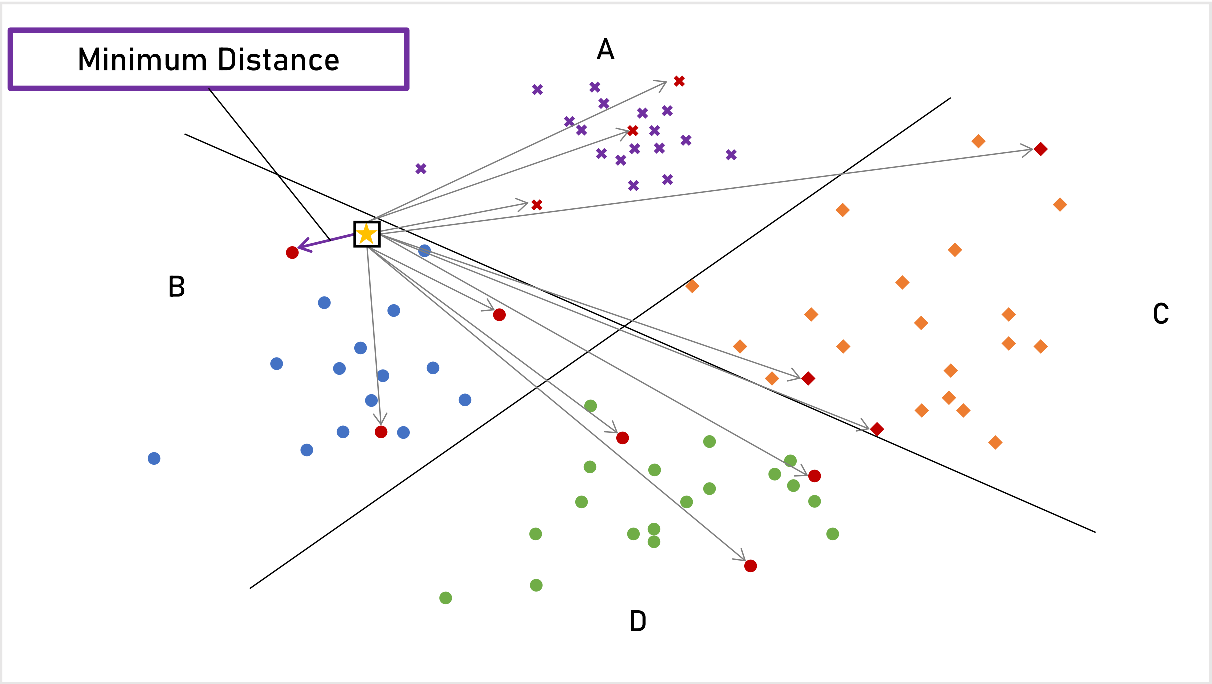
1. Select random centroids-data of each group
2. Calculate each distance from the new data or target data
3. Determine the data by the centroid at the minimum distance

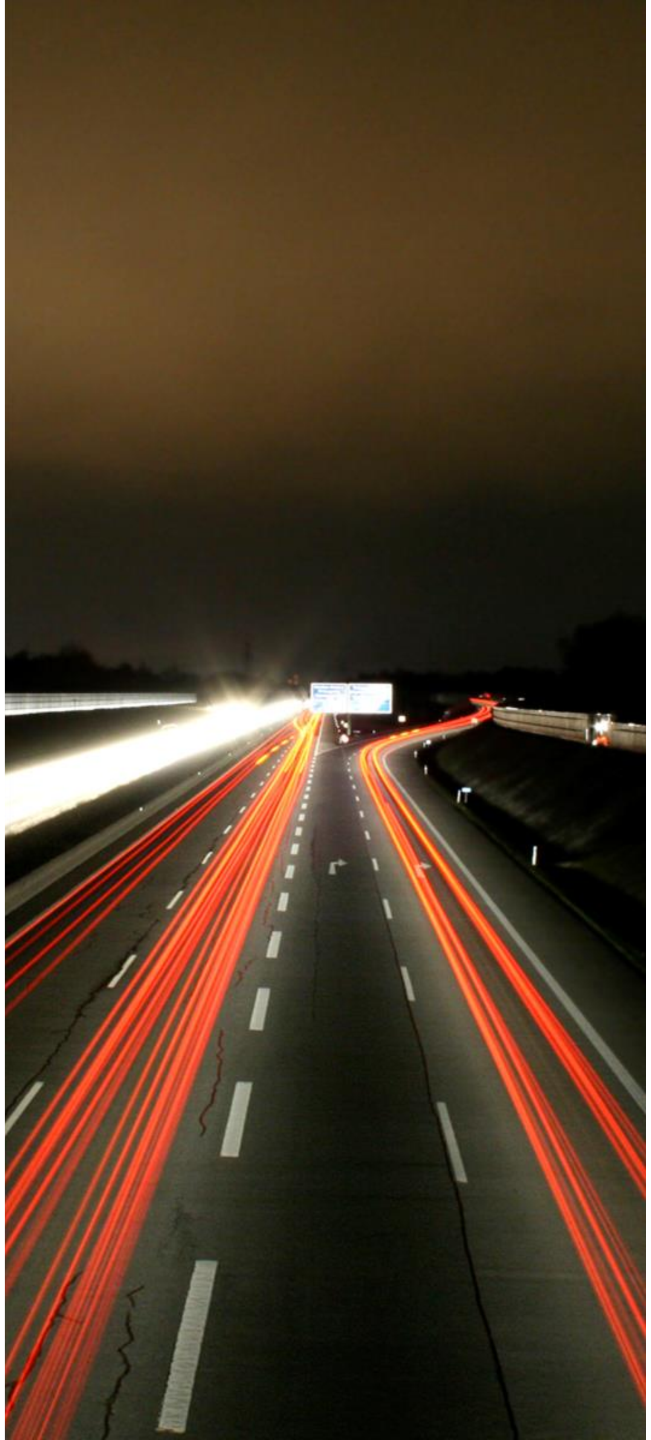


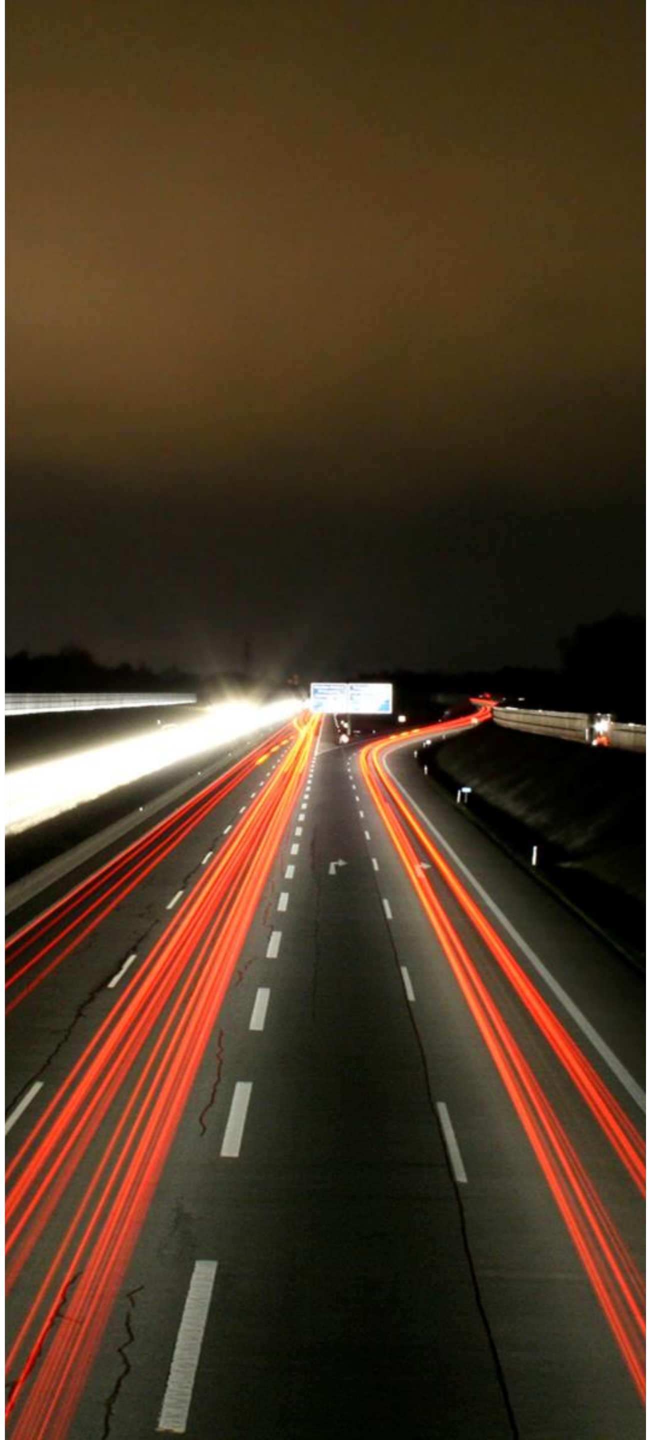
Hand-written Digit recognition

- Minimum Distance Classifier Algorithm

1. Select random centroids-data of each group
2. Calculate each distance from the new data or target data
3. Determine the data by the centroid at the minimum distance



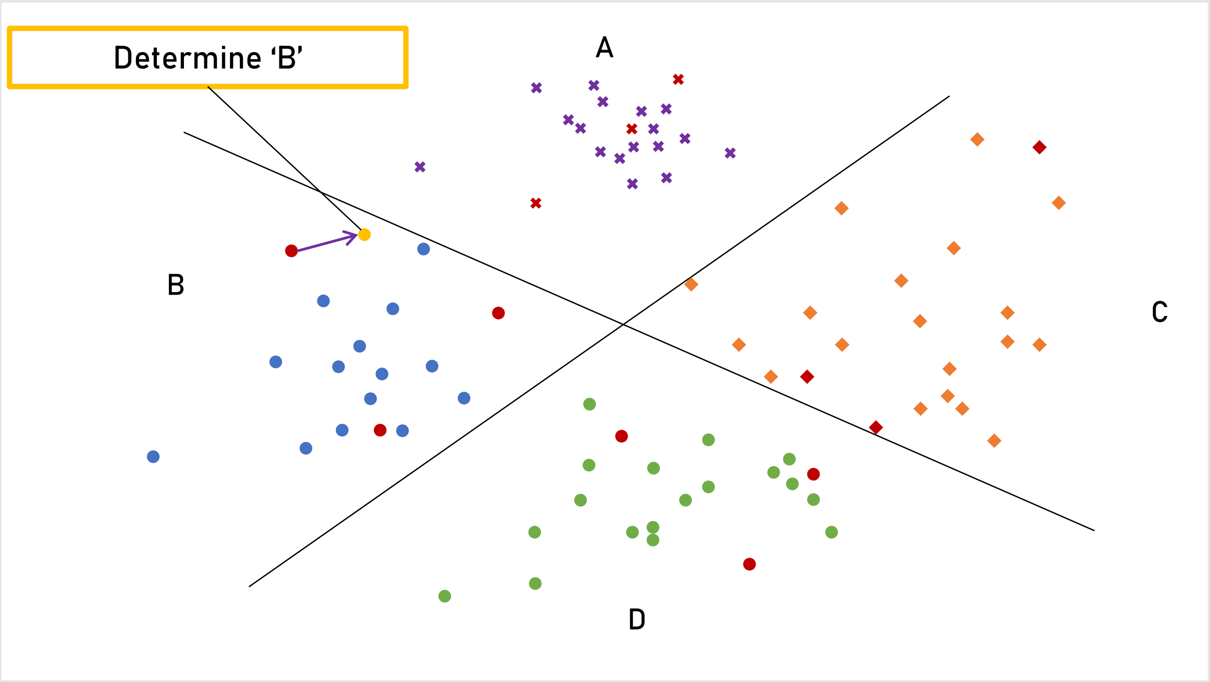


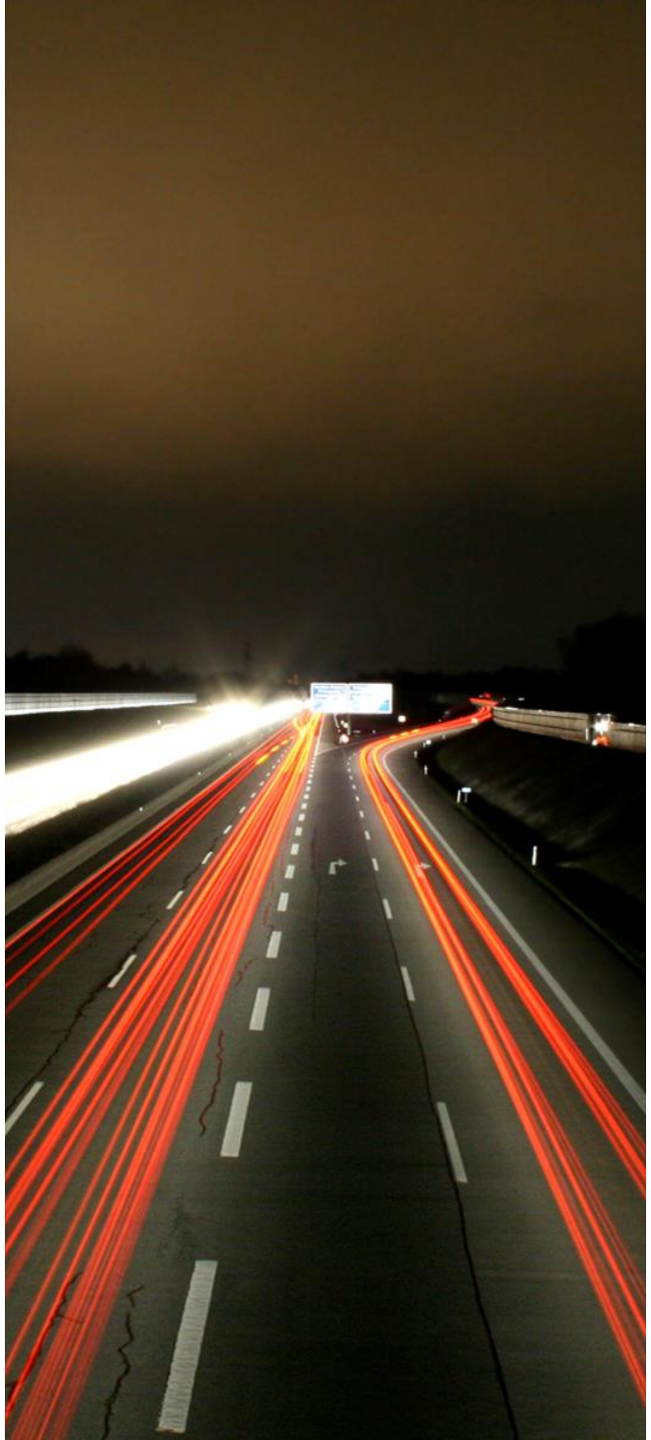


Hand-written Digit recognition

- Minimum Distance Classifier Algorithm

1. Select random centroids-data of each group
2. Calculate each distance from the new data or target data
3. Determine the data by the centroid at the minimum distance





Hand-written Digit recognition

- Minimum Distance Classifier Algorithm

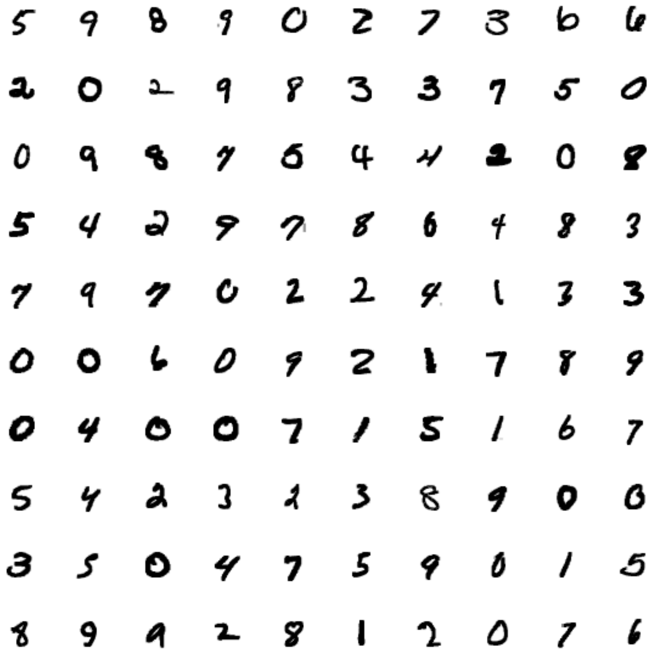
We select 50-random centroids-data of each group

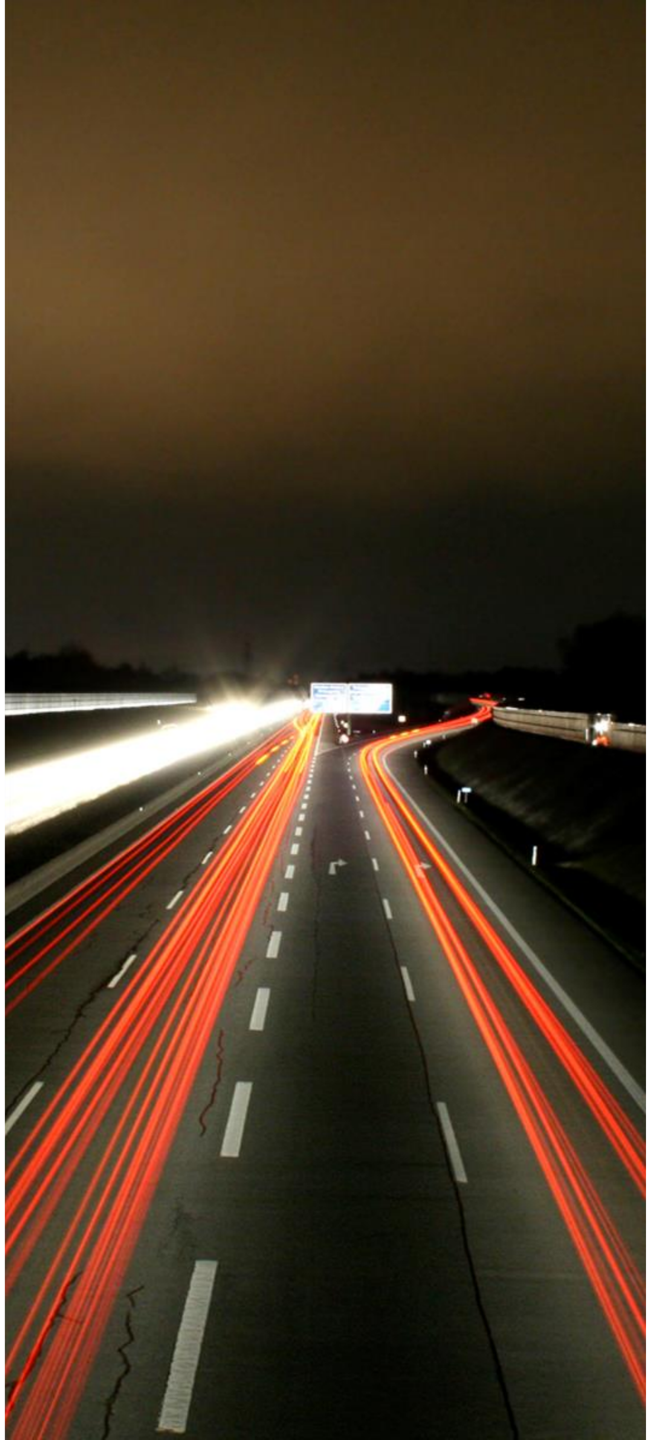
(And Visualize 100-random samples of cluster centroids)

Cluster Centroid Visualization

[11]:
```
rdsample = pd.DataFrame([])
rdsample_list = rd.sample(list(X_t),100) # 100 random samples of cluster centroid
rdsample = X.iloc[rdsample_list,:]
Visualization(X, rdsample)
```

[11]: Error displaying widget: model not found





Hand-written Digit recognition

- Minimum Distance Classifier Algorithm; Result

Prediction

Test Set

```
[12]: %%time
distances_test = np.array([]);
minidx_test = np.array([]);
m = len(X_t);
n = len(X_test);
p_max = int(n / 500);
range_op1 = 0; # for j range
range_op2 = 0; # for j range
for p in range(0,p_max):
    range_op1 = 500 * p
    range_op2 = 500 * (p+1)
    # Calculating distances
    for j in range(range_op1, range_op2):
        replica = [X_test.iloc[j,1:] for k in range(0,m)]
        replica = np.array(replica)
        d = (X_t - replica)*(X_t - replica) # Calculate MSE
        d_sum = d.sum(axis=1) # d rowsums
        minidx_test = np.append(minidx_test, np.argmin(d_sum)).astype(int) # index of min value on rowsums
        distances_test = np.append(distances_test,d_sum,axis=0).astype(int) # store d_sum value

print('Complete.')
```

```
Complete.
Wall time: 23min 58s
```

Learning Time : 23min 58sec

```
[13]: result_test = minidx_test // 50
zerocounts_test1 = result_test - y_test
zerocounts_test2 = len(zerocounts_test1[zerocounts_test1==0])
accuracy_test = (zerocounts_test2 / len(X_test)) * 100
print('accuracy(test_set):{:.2f}%'.format(accuracy_test))
```

```
accuracy(test_set):94.94%
```

Accuracy:94.94%

It takes a long time, and the accuracy is low.
So, we consider different model

```
[14]: result_table0 = pd.DataFrame([])
result_table0 = pd.concat([result_table0,pd.DataFrame(y_test).value_counts().sort_index()]).astype(int)
result_table0 = pd.concat([result_table0,pd.DataFrame(result_test).value_counts().sort_index()],axis=1).astype(int)
result_table0.index = ['0','1','2','3','4','5','6','7','8','9']
result_table0.columns = ['Prediction','Actual']
result_table0.transpose()
```

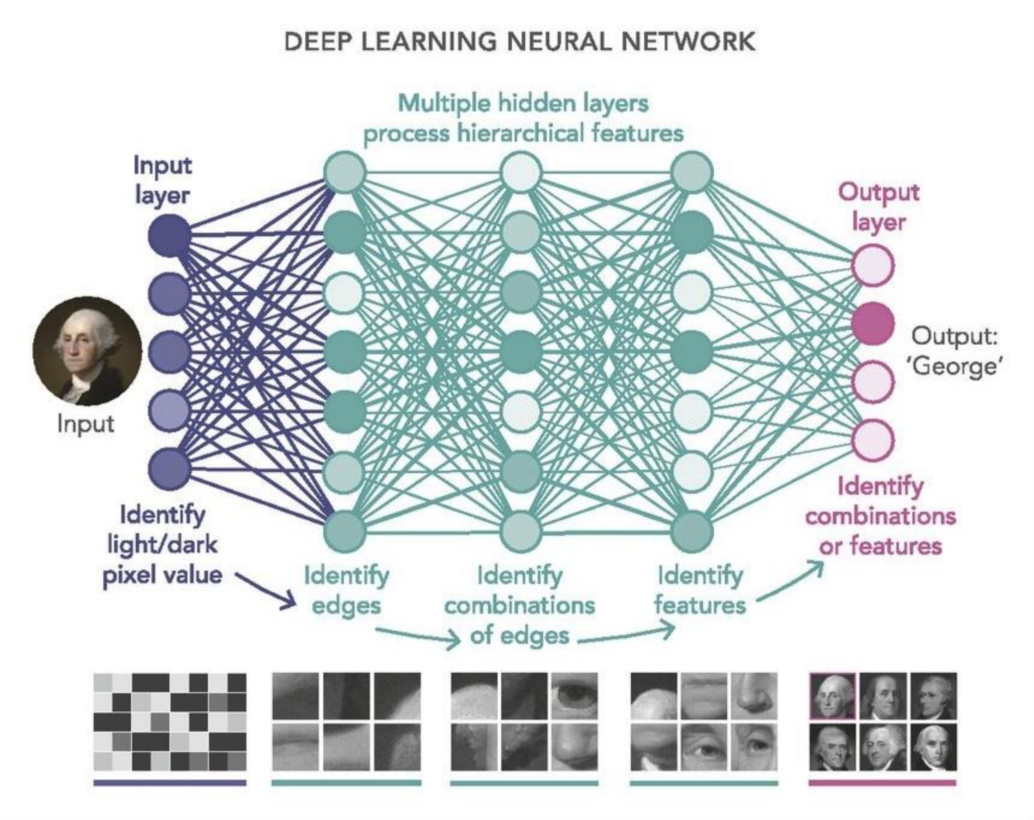
[14]:

	0	1	2	3	4	5	6	7	8	9
Prediction	1028	1186	1069	1121	987	924	1051	1075	986	1073
Actual	1028	1209	1055	1103	975	935	1069	1073	968	1085

Hand-written Digit recognition

- Neural Network

A neural network is composed of artificial neurons or nodes.



Hand-written Digit recognition

- Neural Network

- Weight(s)

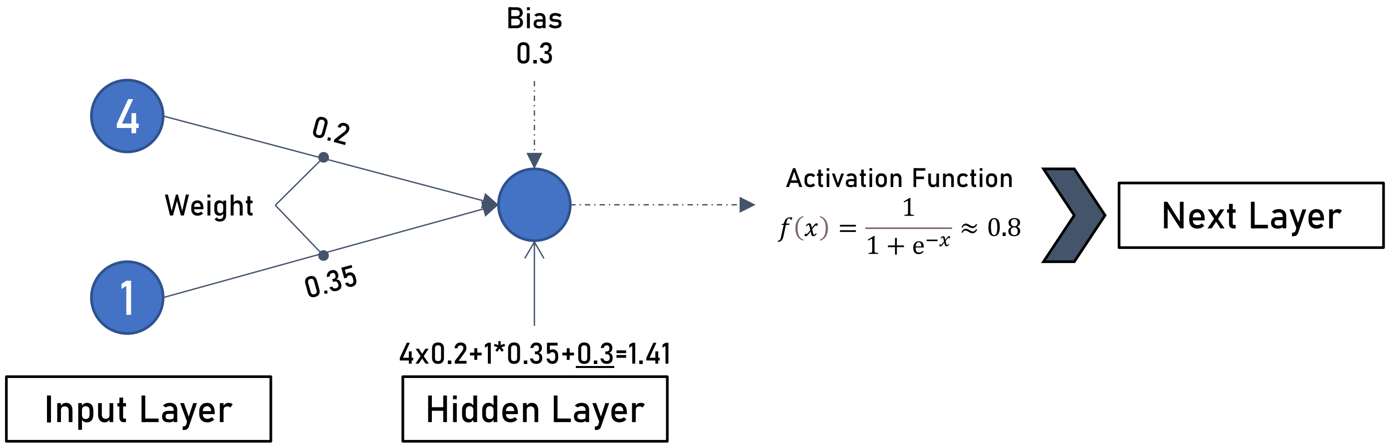
Parameters that control the importance of each input signal to the result.

- Bias(es)

Parameters that regulate the activation conditions of neurons.

- Activation function

A function that determines the transmission of a weighted sum of the inputs.



Hand-written Digit recognition

- Neural Network Model1

```
# network parameters
input_size = X_trainN.shape[1]
batch_size = 128
hidden_units = 256
dropout = 0.25
```

```
[29]: # this model is a 3-layer MLP with sigmoid and dropout after each layer
model_1 = tf.keras.models.Sequential()
model_1.add(layers.Dense(hidden_units, input_dim=input_size))
model_1.add(layers.Activation('sigmoid'))
model_1.add(layers.Dropout(dropout))
model_1.add(layers.Dense(hidden_units))
model_1.add(layers.Activation('sigmoid'))
model_1.add(layers.Dropout(dropout))
model_1.add(layers.Dense(num_labels))
model_1.add(layers.Activation('softmax'))
model_1.summary()

# loss function for one-hot vector using adam optimizer
model_1.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

```
Model: "sequential_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_33 (Dense)             (None, 256)

activation_33 (Activation)   (None, 256)

dropout_22 (Dropout)         (None, 256)

dense_34 (Dense)             (None, 256)

activation_34 (Activation)   (None, 256)

dropout_23 (Dropout)         (None, 256)

dense_35 (Dense)             (None, 10)

activation_35 (Activation)   (None, 10)                0
=================================================================
Total params: 269,322
Trainable params: 269,322
Non-trainable params: 0
_________________________________________________________________
```

Input layer 784

Hidden layer 256

Hidden layer 256

Output layer 10

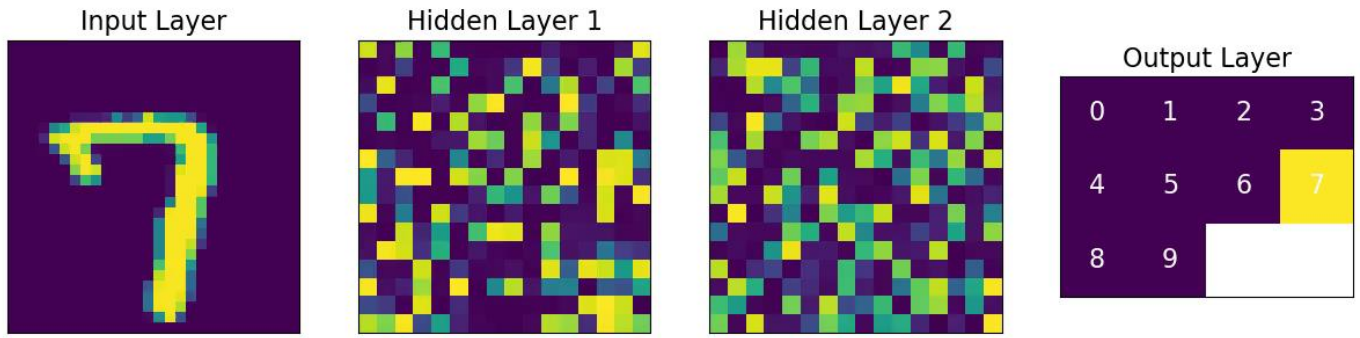
```
Epoch 1/20
247/247 [==============================] - 1s 3ms/step - loss: 0.9207 - accuracy: 0.7196
Epoch 2/20
247/247 [==============================] - 1s 3ms/step - loss: 0.3531 - accuracy: 0.8956
Epoch 3/20
247/247 [==============================] - 1s 3ms/step - loss: 0.2818 - accuracy: 0.9156
Epoch 4/20
247/247 [==============================] - 1s 4ms/step - loss: 0.2397 - accuracy: 0.9282
Epoch 5/20
247/247 [==============================] - 1s 4ms/step - loss: 0.2093 - accuracy: 0.9383
Epoch 6/20
247/247 [==============================] - 1s 5ms/step - loss: 0.1858 - accuracy: 0.9434
Epoch 7/20
247/247 [==============================] - 1s 4ms/step - loss: 0.1605 - accuracy: 0.9521
Epoch 8/20
247/247 [==============================] - 1s 4ms/step - loss: 0.1442 - accuracy: 0.9551
Epoch 9/20
247/247 [==============================] - 1s 4ms/step - loss: 0.1326 - accuracy: 0.9588
Epoch 10/20
247/247 [==============================] - 1s 4ms/step - loss: 0.1174 - accuracy: 0.9633
Epoch 11/20
247/247 [==============================] - 1s 3ms/step - loss: 0.1107 - accuracy: 0.9651
Epoch 12/20
247/247 [==============================] - 1s 3ms/step - loss: 0.1007 - accuracy: 0.9687
Epoch 13/20
247/247 [==============================] - 1s 3ms/step - loss: 0.0929 - accuracy: 0.9705
Epoch 14/20
247/247 [==============================] - 1s 3ms/step - loss: 0.0857 - accuracy: 0.9733
Epoch 15/20
247/247 [==============================] - 1s 3ms/step - loss: 0.0794 - accuracy: 0.9750
Epoch 16/20
247/247 [==============================] - 1s 4ms/step - loss: 0.0710 - accuracy: 0.9779
Epoch 17/20
247/247 [==============================] - 1s 4ms/step - loss: 0.0683 - accuracy: 0.9781
Epoch 18/20
247/247 [==============================] - 1s 3ms/step - loss: 0.0631 - accuracy: 0.9785
Epoch 19/20
247/247 [==============================] - 1s 4ms/step - loss: 0.0583 - accuracy: 0.9816
Epoch 20/20
247/247 [==============================] - 1s 5ms/step - loss: 0.0520 - accuracy: 0.9837
83/83 [==============================] - 0s 2ms/step - loss: 0.0948 - accuracy: 0.9721

Test accuracy: 97.2%
```

Accuracy : 97.2%

Learning Time : 20.73sec

Hand-written Digit recognition

- Neural Network Model1



Hand-written Digit recognition

- Neural Network Model2 (Convolution Neural Network)

- Convolution layer

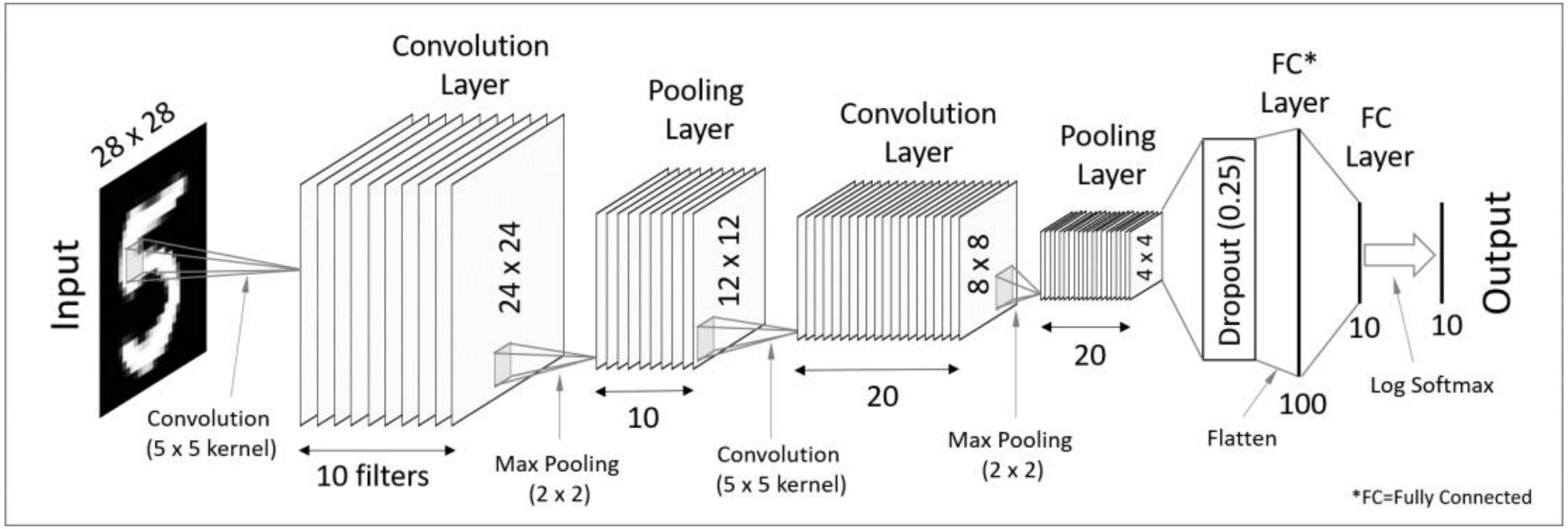
Layer that determines the characteristics of an image with multiple filters.

- Pooling layer

Layer that enhance the characteristics of the image.

- FC(Fully Connected) layer

Layer that classifies images based on extracted data.



Hand-written Digit recognition

- Neural Network Model2 (Convolution Neural Network)

```
[8]: model_2 = models.Sequential()
model_2.add(layers.Conv2D(10, (5, 5), activation='relu', input_shape=(28, 28, 1)))
model_2.add(layers.MaxPooling2D((2, 2)))
model_2.add(layers.Conv2D(20, (5, 5), activation='relu'))
model_2.add(layers.MaxPooling2D((2, 2)))
model_2.add(layers.Dropout(0.25))

model_2.add(layers.Flatten())
model_2.add(layers.Dense(100, activation='relu'))
model_2.add(layers.Dense(10, activation='softmax'))
model_2.summary()

model_2.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model_2.fit(train_images, train_labels, epochs=5)
test_loss, test_acc = model_2.evaluate(test_images,  test_labels, verbose=2)
print("\nTest accuracy: %.1f%%" % (100.0 * test_acc))
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 24, 24, 10)        260

 max_pooling2d (MaxPooling2D  (None, 12, 12, 10)       0
 )

 conv2d_1 (Conv2D)           (None, 8, 8, 20)          5020

 max_pooling2d_1 (MaxPooling  (None, 4, 4, 20)         0
 2D)

 dropout (Dropout)           (None, 4, 4, 20)          0

 flatten (Flatten)           (None, 320)               0

 dense (Dense)               (None, 100)               32100

 dense_1 (Dense)             (None, 10)                1010

=================================================================
Total params: 38,390
Trainable params: 38,390
Non-trainable params: 0
_________________________________________________________________
```

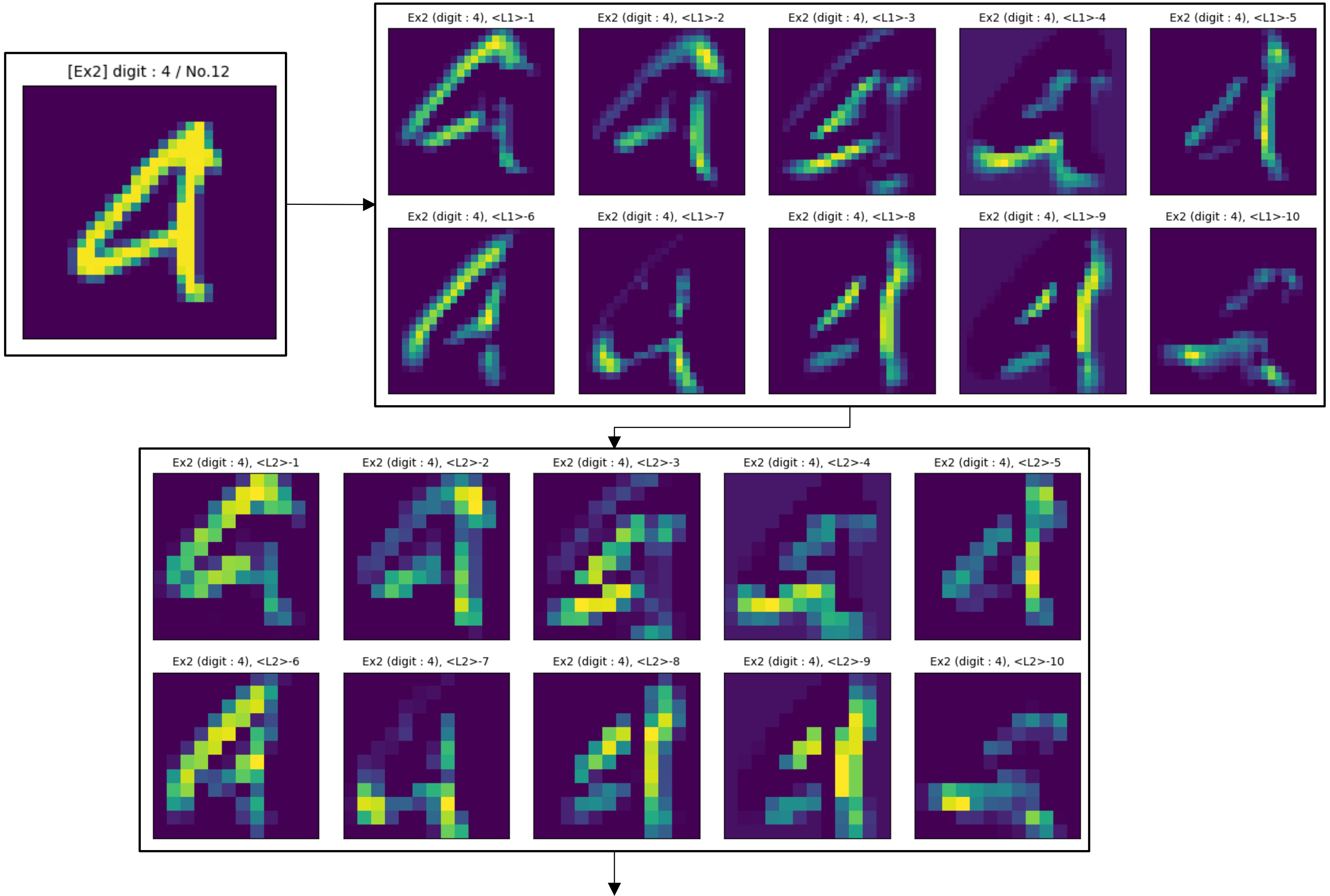
```
Epoch 1/5
985/985 [==============================] - 4s 3ms/step - loss: 0.3070 - accuracy: 0.9049
Epoch 2/5
985/985 [==============================] - 3s 3ms/step - loss: 0.1049 - accuracy: 0.9670
Epoch 3/5
985/985 [==============================] - 3s 3ms/step - loss: 0.0754 - accuracy: 0.9757
Epoch 4/5
985/985 [==============================] - 3s 3ms/step - loss: 0.0610 - accuracy: 0.9799
Epoch 5/5
985/985 [==============================] - 3s 3ms/step - loss: 0.0510 - accuracy: 0.9840
329/329 - 0s - loss: 0.0570 - accuracy: 0.9828 - 484ms/epoch - 1ms/step

Test accuracy: 98.3%
```

Accuracy : 98.3%

Learning Time : 16.15sec

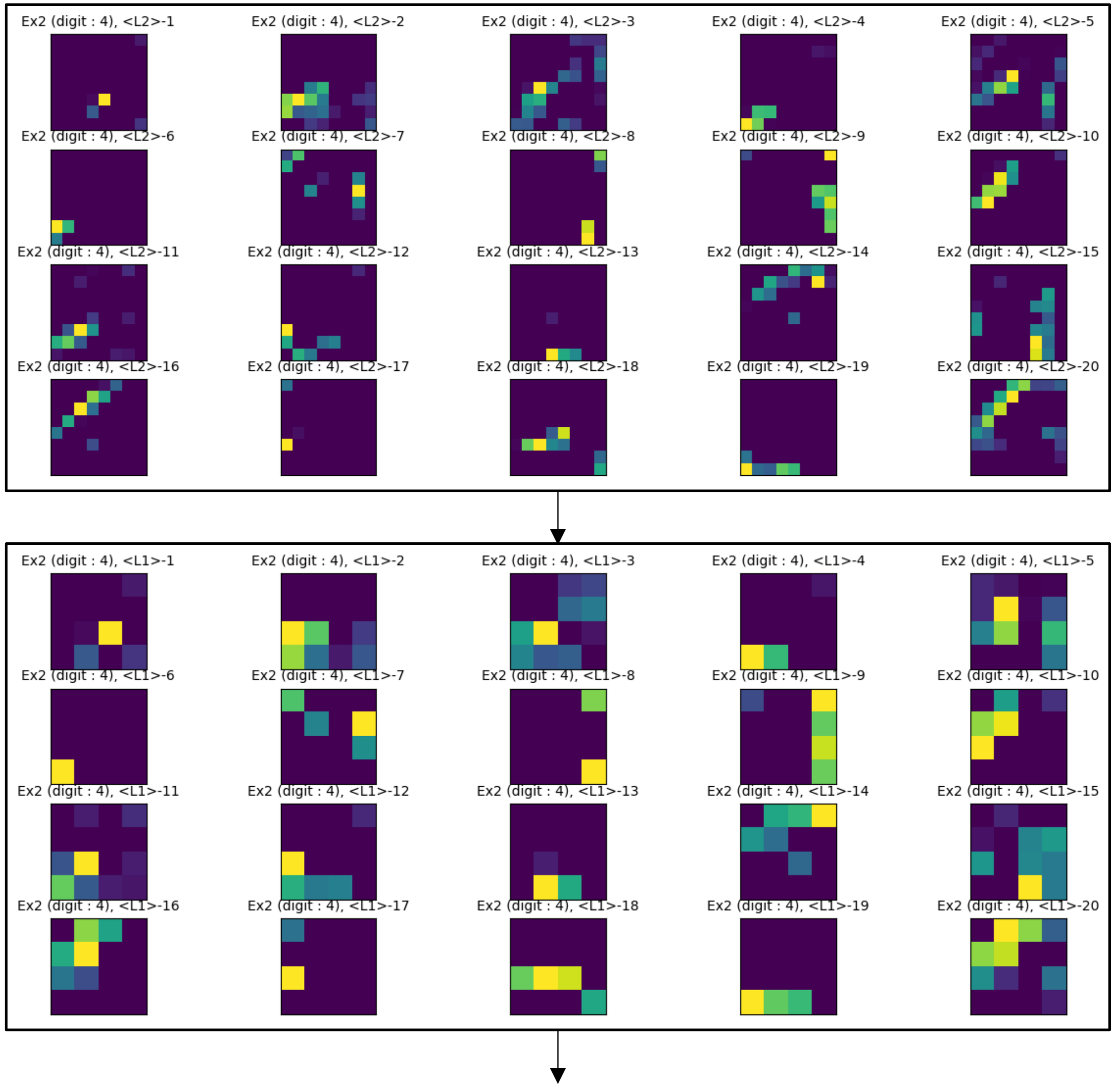
Hand-written Digit recognition

- Neural Network Model2 (Convolution Neural Network)



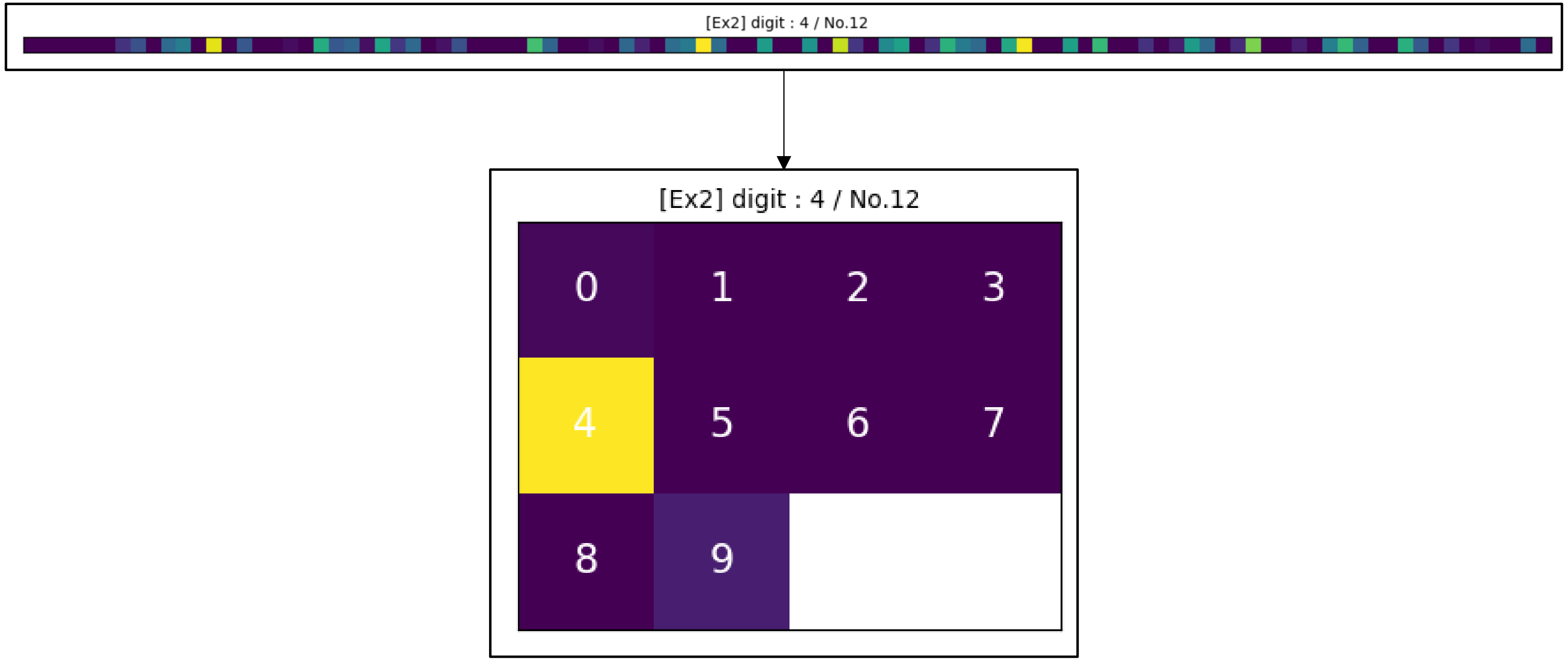
Hand-written Digit recognition

- Neural Network Model2 (Convolution Neural Network)



Hand-written Digit recognition

- Neural Network Model2 (Convolution Neural Network)



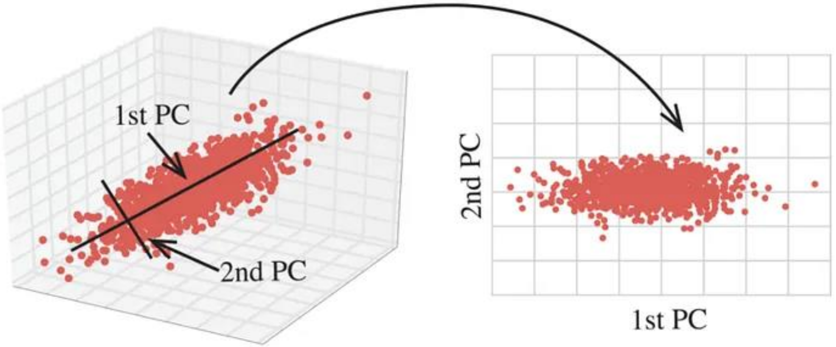
Hand-written Digit recognition

- Principal component analysis (PCA)

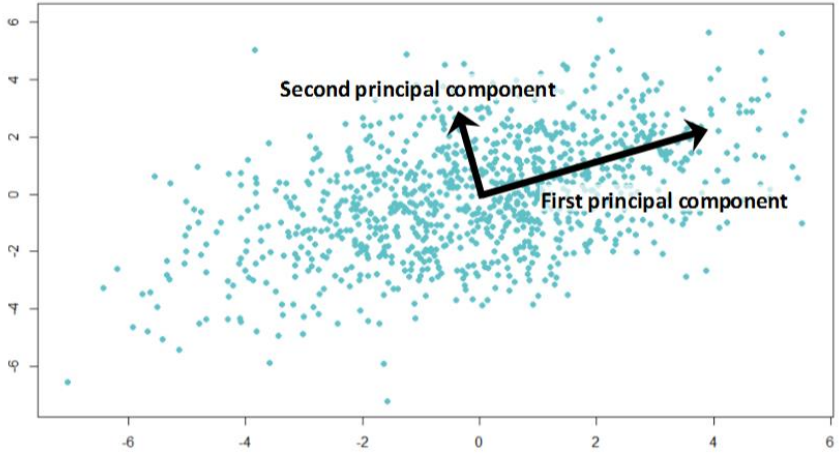
PCA simplifies the complexity in high-dimensional data

while retaining trends and patterns.

more simply, It can be seen as dimensional reduction or projection.



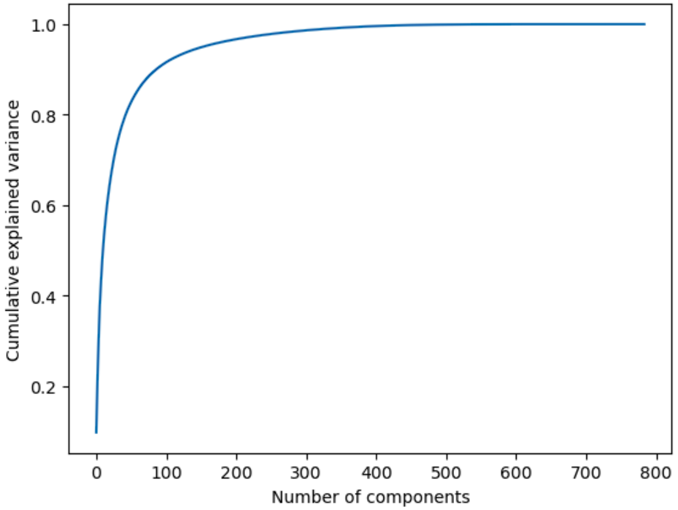
$\mathbb{R}^3 \rightarrow \mathbb{R}^2$



$\mathbb{R}^2$

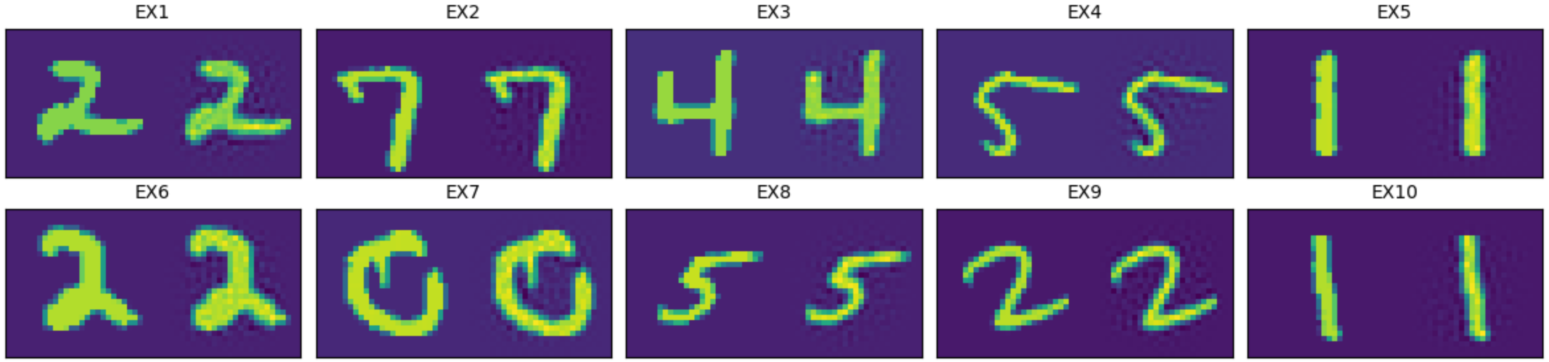
Hand-written Digit recognition

- Principal component analysis (PCA)




```
[9]: %%time
NCOMPONENTS = 500

pca = sklearn.decomposition.PCA(n_components=NCOMPONENTS)
X_pca_train = pca.fit_transform(X_sc_train)
X_pca_test = pca.transform(X_sc_test)
pca_std = np.std(X_pca_train)
```

We choose Number of Principal component : 500
784(28 x 28) component → 500 key component



Hand-written Digit recognition

- Principal component analysis (PCA) + Neural Network ; Result

```
[129]: model = models.Sequential()
units = 256

model.add(layers.Dense(units, input_dim=NCOMPONENTS, activation='relu'))
model.add(layers.GaussianNoise(pca_std))
model.add(layers.Dense(units, activation='relu'))
model.add(layers.GaussianNoise(pca_std))
model.add(layers.Dropout(0.1))
model.add(layers.Dense(10, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='rmsprop', metrics=['categorical_accuracy'])

model.fit(X_pca_train, Y_train, epochs=20, batch_size=128, validation_split=0.15, verbose=2)

test_loss, test_acc = model.evaluate(X_pca_test, Y_test, verbose=2)
print("\nTest accuracy: %.1f%%" % (100.0 * test_acc))
```

Input layer 500

Hidden layer 256

Hidden layer 256

Output layer 10

```
Epoch 1/20
210/210 - 1s - loss: 0.9572 - categorical_accuracy: 0.7337 - val_loss: 0.3183 - val_categorical_accuracy: 0.9175 - 1s/epoch - 5ms/step
Epoch 2/20
210/210 - 1s - loss: 0.3735 - categorical_accuracy: 0.8925 - val_loss: 0.2361 - val_categorical_accuracy: 0.9352 - 516ms/epoch - 2ms/step
Epoch 3/20
210/210 - 1s - loss: 0.2546 - categorical_accuracy: 0.9251 - val_loss: 0.1998 - val_categorical_accuracy: 0.9481 - 539ms/epoch - 3ms/step
Epoch 4/20
210/210 - 1s - loss: 0.1875 - categorical_accuracy: 0.9421 - val_loss: 0.2190 - val_categorical_accuracy: 0.9420 - 528ms/epoch - 3ms/step
Epoch 5/20
210/210 - 1s - loss: 0.1493 - categorical_accuracy: 0.9525 - val_loss: 0.1884 - val_categorical_accuracy: 0.9530 - 552ms/epoch - 3ms/step

Epoch 15/20
210/210 - 1s - loss: 0.0326 - categorical_accuracy: 0.9897 - val_loss: 0.2253 - val_categorical_accuracy: 0.9630 - 660ms/epoch - 3ms/step
Epoch 16/20
210/210 - 1s - loss: 0.0291 - categorical_accuracy: 0.9903 - val_loss: 0.2378 - val_categorical_accuracy: 0.9638 - 665ms/epoch - 3ms/step
Epoch 17/20
210/210 - 1s - loss: 0.0277 - categorical_accuracy: 0.9911 - val_loss: 0.2299 - val_categorical_accuracy: 0.9683 - 551ms/epoch - 3ms/step
Epoch 18/20
210/210 - 1s - loss: 0.0231 - categorical_accuracy: 0.9919 - val_loss: 0.2378 - val_categorical_accuracy: 0.9655 - 652ms/epoch - 3ms/step
Epoch 19/20
210/210 - 1s - loss: 0.0239 - categorical_accuracy: 0.9925 - val_loss: 0.2489 - val_categorical_accuracy: 0.9663 - 651ms/epoch - 3ms/step
Epoch 20/20
210/210 - 1s - loss: 0.0220 - categorical_accuracy: 0.9923 - val_loss: 0.2712 - val_categorical_accuracy: 0.9649 - 683ms/epoch - 3ms/step
329/329 - 0s - loss: 0.2862 - categorical_accuracy: 0.9654 - 225ms/epoch - 685us/step

Test accuracy: 96.5%
```

Accuracy : 96.5%

Learning Time : 13.64sec

Summary

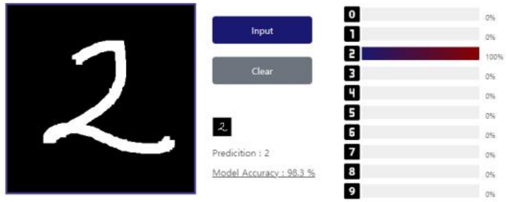
- Choosing Best Model

We choose Convolution Neural Network model with the low time cost and highest accuracy.

	Minimum Distance Classifier	Neural Network	Convolution Neural Network	Principal Component Analysis
Learning Time	23min 58sec	20.73sec	16.15sec	13.64sec
Accuracy	94.94%	97.2%	98.3%	96.5%

Digit recognition web

- https://pdg2619.github.io/digit_recog_web/



Reference

[1] Andrew Ng, Machine Learning, Coursera,

https://www.coursera.org/learn/machine-learning-course/home/info

[2] 공대선배 스트롱, 인공지능 분야 개념부터 진로까지. AI, 머신러닝, 딥러닝. + 근황토크 | 공신위 진로#2,

https://www.youtube.com/watch?v=7tf3lKh1pxU&t=693s

[3] Rocket Source, The 4 Machine Learning Models Imperative for Business Transformation,

https://www.rocketsource.com/blog/machine-learning-models/

[4] Data Flair, Clustering in Machine Learning – Algorithms that Every Data Scientist Uses,

https://data-flair.training/blogs/clustering-in-machine-learning/

[5] ML-Guy, An Executive Introduction to AI/ML/DL,

https://guyernest.medium.com/an-executive-introduction-to-ai-ml-dl-7f925e32d4bc

[6] Kaggle, Digit Recognizer, https://www.kaggle.com/competitions/digit-recognizer/data

[7] NehaCkumari, MNIST-Handwritten-Digit-Recognition-using-CNN,

https://github.com/NehaCkumari/MNIST-Handwritten-Digit-Recognition-using-CNN

[8] 네이쳐k, Training, Validation and Test sets 차이 및 정확한 용도 (훈련, 검정, 테스트 데이터 차이),

https://modern-manual.tistory.com/19

[9] OLGA CHERNYTSKA, Minimum Distance Classifier,

https://www.kaggle.com/code/olhacher/minimum-distance-classifier

[10] H2O.ai, Weights and Biases,

https://h2o.ai/wiki/weights-and-biases

Reference

[11] Emine Kesici, Principal Component Analysis in Practice,

https://medium.com/@emykes/principal-component-analysis-in-practice-4363f3b46104

[12] Jake Lever, Martin Krzywinski & Naomi Altman, Principal component analysis,

https://www.nature.com/articles/nmeth.4346

[13] HAN_PY, Convolutional Neural Network(CNN) _기초 개념,

https://han-py.tistory.com/230

[14]Irene Ban, [Deep learning/Machine learning] Convenient understanding of convolutional neural networks (CNN),

https://halfundecided.medium.com/%EB%94%A5%EB%9F%AC%EB%8B%9D-%EB%A8%B8%EC%8B%A0%EB%9F%AC%EB%8B%9D-cnn-convolutional-neural-networks-%EC%89%BD%EA%B2%8C-%EC%9D%B4%ED%95%B4%ED%95%98%EA%B8%B0-836869f88375

Thank you for your attention.

Fin.