**Machine Learning**

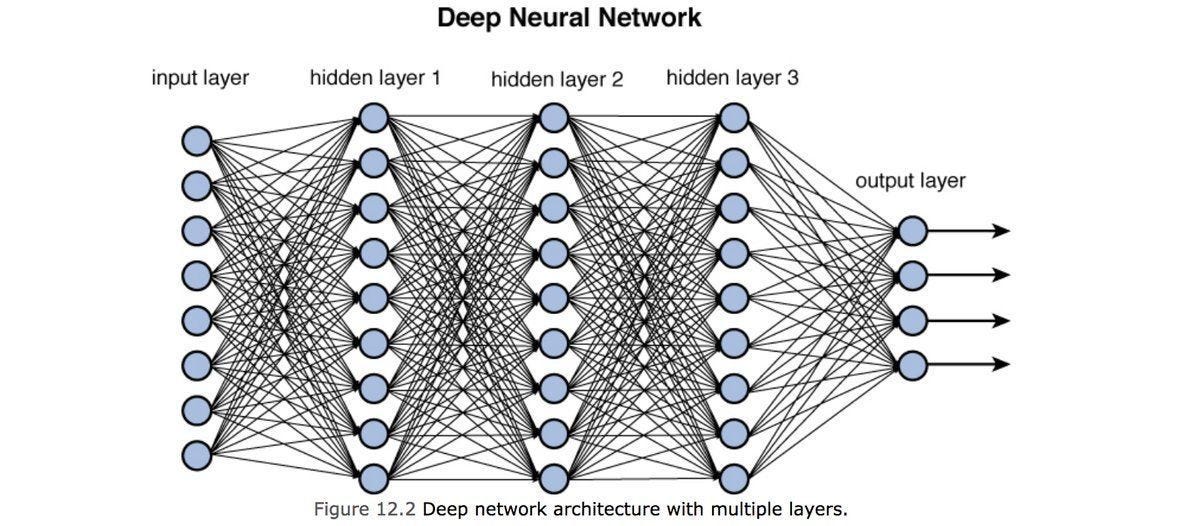
**Deep Neural Network**



|  |  |
| --- | --- |
| **이름(학번)** | **이 재 원 (32193430)** |
| **담당교수** | **조 재 형 교수님** |
| **제출일** | **2024. 12. 02** |

**What is Deep Neural Network?**

A **Deep Neural Network (DNN)** is a type of neural network with multiple layers, enabling it to learn hierarchical representations of data. This architecture excels in tackling complex tasks such as image and speech recognition, natural language processing (NLP), and beyond. However, despite its power, a DNN requires meticulous design and training to address challenges like overfitting and computational inefficiency effectively.



**Figure 1 – DNN architecture with multiple layers**

**Download the Data**

from sklearn.datasets import fetch\_openml

from sklearn.model\_selection import train\_test\_split

import numpy as np

X, y = fetch\_openml('mnist\_784', version=1, return\_X\_y=True)

y = y.astype(int)

X = ((X / 255.) - .5) \* 2

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

    X, y, test\_size=10000, random\_state=123, stratify=y)

**Crate a Test Set**

from sklearn.datasets import fetch\_openml

from sklearn.model\_selection import train\_test\_split

import numpy as np

X, y = fetch\_openml('mnist\_784', version=1, return\_X\_y=True)

y = y.astype(int)

X = ((X / 255.) - .5) \* 2

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

    X, y, test\_size=10000, random\_state=123, stratify=y)

import numpy as np

np.savez\_compressed('mnist\_scaled.npz',

                    X\_train=X\_train,

                    y\_train=y\_train,

                    X\_test=X\_test,

                    y\_test=y\_test)

**Take a Quick Look at the Data Structure**

import matplotlib.pyplot as plt

fig, ax = plt.subplots(nrows=2, ncols=5, sharex=True, sharey=True)

ax = ax.flatten()

for i in range(10):

    img = X\_train[y\_train == i][i].reshape(28, 28)

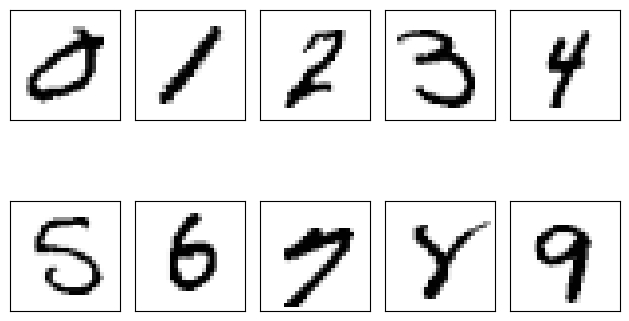
    ax[i].imshow(img, cmap='Greys')

ax[0].set\_xticks([])

ax[0].set\_yticks([])

plt.tight\_layout()

plt.show()

****

**Figure 2 – A few digits from the MNIST dataset**

**Model Training**

import tensorflow as tf

# Define the model

model = tf.keras.Sequential()

# Input layer

model.add(tf.keras.layers.Dense(

    units=64,  # Increased units

    activation=tf.keras.activations.relu,

    kernel\_initializer=tf.keras.initializers.GlorotNormal(),

    bias\_initializer=tf.keras.initializers.Constant(2.0),

    input\_shape=(784,)  # Input shape specified only in the first layer

))

# Additional hidden layers

model.add(tf.keras.layers.Dense(units=128, activation='relu'))  # Increased capacity

model.add(tf.keras.layers.Dense(units=64, activation='relu'))   # Add another hidden layer

# Output layer

model.add(tf.keras.layers.Dense(

    units=10,  # Output units should match the number of classes

    activation='softmax'  # Use softmax for multi-class classification

))

# Compile the model

model.compile(optimizer='adam',

              loss='sparse\_categorical\_crossentropy',

              metrics=['accuracy'])

# Train the model

model.fit(

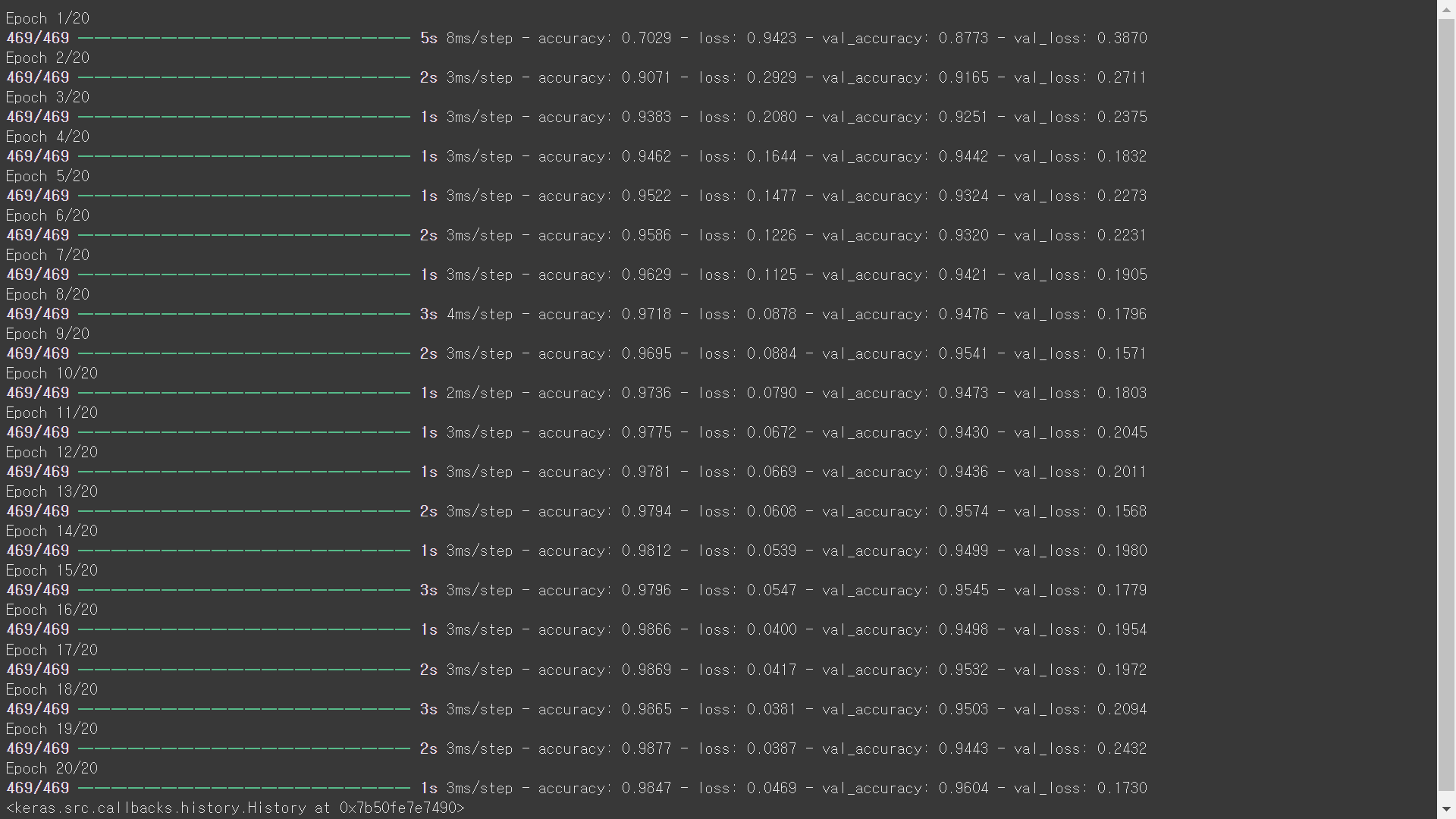
    x=X\_train[:15000],  # Training samples

    y=y\_train[:15000],  # Training labels

    validation\_data=(X\_train[15000:25000], y\_train[15000:25000]),  # Validation samples

    epochs=20  # Increase epochs for deeper networks

)



**Figure 3 – A DNN on a subset of training data**

**Model Summary**



**Figure 4 – Model Summary**

**Performance Measures**

import numpy as np

y\_test\_pred = model.predict(X\_test)

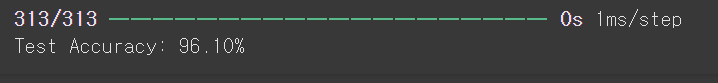
# Get the predicted class labels by finding the index of the maximum value along the second axis (axis=1)

y\_test\_pred = np.argmax(y\_test\_pred, axis=1)

acc = (np.sum(y\_test == y\_test\_pred)

       .astype(np.float32) / X\_test.shape[0])

print('Test Accuracy: %.2f%%' % (acc \* 100))



**Figure 5 – Test Accuracy**

**Misclassified Data**

miscl\_img = X\_test[y\_test != y\_test\_pred][:25]

correct\_lab = y\_test[y\_test != y\_test\_pred][:25]

miscl\_lab = y\_test\_pred[y\_test != y\_test\_pred][:25]

fig, ax = plt.subplots(nrows=5, ncols=5, sharex=True, sharey=True)

ax = ax.flatten()

for i in range(25):

    img = miscl\_img[i].reshape(28, 28)

    ax[i].imshow(img, cmap='Greys', interpolation='nearest')

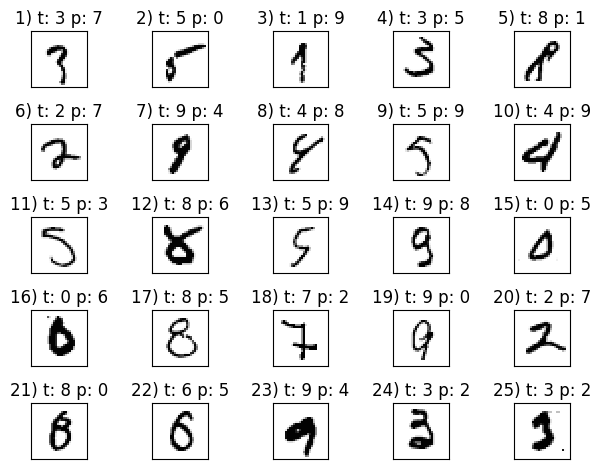
    ax[i].set\_title('%d) t: %d p: %d' % (i+1, correct\_lab[i], miscl\_lab[i]))

ax[0].set\_xticks([])

ax[0].set\_yticks([])

plt.tight\_layout()

plt.show()

****

**Figure 6 – Misclassified Data**

**Reference**

**Figure1**

[**https://miro.medium.com/v2/resize:fit:1199/1\*N8UXaiUKWurFLdmEhEHiWg.jpeg**](https://miro.medium.com/v2/resize:fit:1199/1*N8UXaiUKWurFLdmEhEHiWg.jpeg)