a2-490576560-520653377

October 21, 2022

1 COMP5318 Assignment 2: Image Classification

1.0.1 Group number: 102, SID1: 490576560, SID2: 520653377

1.1 Setup and dependencies

All the required libraries/dependencies and the plotting environment are listed and set up here.

[ ]:
```
import time

import numpy as np
import tensorflow as tf
from tensorflow import keras
import matplotlib as mpl
import matplotlib.pyplot as plt
import pandas as pd

from sklearn.base import clone
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.model_selection import ParameterGrid
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import GridSearchCV
from scikeras.wrappers import KerasClassifier

from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

# Make the notebook's output stable across runs.
# Random seed is set to 0 consistently.
np.random.seed(0)
tf.random.set_seed(0)
initializer = tf.keras.initializers.GlorotUniform(seed=0)
keras.backend.clear_session()
```

1.2 1. Data loading, preprocessing, and exploration

The documentation for the data loading function can be accessed here.

1.2.1 1.1 Load data and declare variables

[ ]:
```
# Load the Fashion-MNIST dataset training and test sets as numpy arrays
(X_train_original, y_train_original), (X_test_original, y_test_original) =␣
 ↪keras.datasets.fashion_mnist.load_data()
assert X_train_original.shape == (60000, 28, 28)
assert X_test_original.shape == (10000, 28, 28)
assert y_train_original.shape == (60000,)
assert y_test_original.shape == (10000,)

# An ordered list of the class names
class_names = ["T-shirt/top",
               "Trouser",
               "Pullover",
               "Dress",
               "Coat",
               "Sandal",
               "Shirt",
               "Sneaker",
               "Bag",
               "Ankle boot"
              ]

# Declare size of the image
IMAGE_SIZE = X_train_original[0].shape
```

1.2.2 1.2 Data processing

[ ]:
```
# Normalise data
X_train_full = X_train_original.reshape(X_train_original.shape[0], -1) #␣
 ↪Flatten data from 3D to 2D
y_train_full = y_train_original.copy()
X_test = X_test_original.reshape(X_test_original.shape[0], -1) # Flatten data␣
 ↪from 3D to 2D
y_test = y_test_original.copy()

scaler = MinMaxScaler()
scaler.fit(X_train_full)
X_train_full = scaler.transform(X_train_full) # apply normalisation to the␣
 ↪training set
X_test = scaler.transform(X_test) # apply normalisation to the test set

X_train_full = X_train_full.reshape(X_train_original.shape[0], *IMAGE_SIZE) #␣
 ↪restore the dimention from 2D to 3D
```

```
X_test = X_test.reshape(X_test_original.shape[0], *IMAGE_SIZE) # restore the␣
 ↪dimention from 2D to 3D

# Create validation set from the training set
X_train, X_valid, y_train, y_valid = train_test_split(X_train_full,␣
 ↪y_train_full, train_size=0.9, stratify=y_train_full)
```

1.2.3 1.3 Data exploration

[ ]:
```
print(f"The original training set is {X_train_original.shape[0]} images with␣
 ↪{X_train_original[0].shape} pixels, \
without normalization ({X_train_original.dtype}).")
print(f"The original test set is {X_test_original.shape[0]} images with␣
 ↪{X_test_original[0].shape} pixels, \
without normalization ({X_test_original.dtype}).\n")

print(f"The size of training set is {X_train.shape[0]} ({X_train.dtype}), \
the size of validation set is {X_valid.shape[0]} ({X_valid.dtype}), and\
the size of test set is {X_test.shape[0]} ({X_test.dtype})")
```

```
The original training set is 60000 images with (28, 28) pixels, without
normalization (uint8).
The original test set is 10000 images with (28, 28) pixels, without
normalization (uint8).

The size of training set is 54000 (float64), the size of validation set is 6000
(float64), andthe size of test set is 10000 (float64)
```

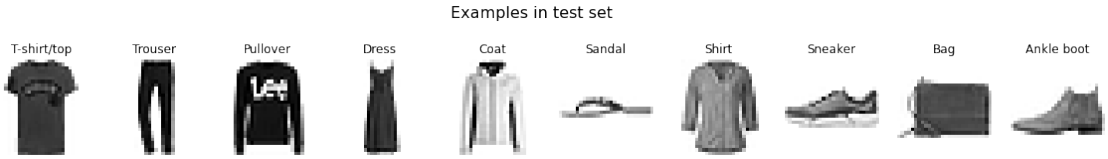
[ ]:
```
def show_distribution(y):
    """Simple way to show a label distribution."""
    result = []
    for i in range(len(class_names)):
        result.append((y == i).sum())
    return result

print(f"There are {len(set(y_train_original))} different classes: {np.
 ↪unique(y_train_original)}")
print(f"The label distribution of training set is {show_distribution(y_train)}")
print(f"The label distribution of validation set is␣
 ↪{show_distribution(y_valid)}")
print(f"The label distribution of test set is {show_distribution(y_test)}")
```

```
There are 10 different classes: [0 1 2 3 4 5 6 7 8 9]
The label distribution of training set is [5400, 5400, 5400, 5400, 5400, 5400,
5400, 5400, 5400, 5400]
The label distribution of validation set is [600, 600, 600, 600, 600, 600, 600,
600, 600, 600]
The label distribution of test set is [1000, 1000, 1000, 1000, 1000, 1000, 1000,
```

```
1000, 1000, 1000]
```

[ ]:
```python
def plot_examples(X, y, title=""):
    """Plot a grid of images from different classes."""
    # Size figure depending on the size of the grid
    plt.figure(figsize=(20, 2))
    plt.suptitle(title, fontsize=16,x=0.5,y=1.2,)

    index = []
    # search index
    for i in range(len(class_names)):
        for j in range(len(y)):
            if i == y[j]:
                index.append(j)
                break

    # Plot the image at appropriate place in grid
    for i in range(len(index)):
        plt.subplot(1, len(index), i + 1)
        plt.imshow(X[index[i]], cmap="binary")
        plt.title(class_names[y[index[i]]])
        plt.axis('off')

    plt.show()

print("Some examples:")

plot_examples(X_train, y_train, "Examples in training set")
plot_examples(X_valid, y_valid, "Examples in validation set")
plot_examples(X_test, y_test, "Examples in test set")
```

```
Some examples:
```



1.3 2. Algorithm design and setup

1.3.1 2.1 K-nearest neighbors

First, A group of simple algorithms from the first 6 weeks are compared. They are: K-nearest neighbors, Naive Bayes, Decision tree, and Random forest. We simply train the model with default/simple parameters on the full training set, and test their accuracy.

[ ]:
```
neigh = KNeighborsClassifier(n_neighbors=10) # k shoule be less than␣
↪sqrt(#training_examples), commercial packages typically use k=10

# Training and timer
time_stamp = time.time()
neigh.fit(X_train_full.reshape(X_train_full.shape[0], -1), y_train_full)
neigh_training_time = time.time() - time_stamp
```

[ ]:
```
nb = GaussianNB()

# Training and timer
time_stamp = time.time()
nb.fit(X_train_full.reshape(X_train_full.shape[0], -1), y_train_full)
nb_training_time = time.time() - time_stamp
```

[ ]:
```
# Running in arround 30s

tree = DecisionTreeClassifier(criterion='entropy', random_state=0) # without␣
↪setting max_depth results in overfitting.

# Training and timer
time_stamp = time.time()
tree.fit(X_train_full.reshape(X_train_full.shape[0], -1), y_train_full)
tree_training_time = time.time() - time_stamp
```

[ ]:
```
# Running in arround 90s

rnd = RandomForestClassifier(criterion='entropy', random_state=0) #␣
↪n_estimators=100 by default

# Training and timer
time_stamp = time.time()
```

```
rnd.fit(X_train_full.reshape(X_train_full.shape[0], -1), y_train_full)
rnd_training_time = time.time() - time_stamp
```

[ ]:
```
print(f"The accuracy of KNN is {neigh.score(X_test.reshape(X_test.shape[0],␣
↪-1), y_test):.4f}, training time is {neigh_training_time:.2f} s.")
print(f"The accuracy of NB is {nb.score(X_test.reshape(X_test.shape[0], -1),␣
↪y_test):.4f}, training time is {nb_training_time:.2f} s.")
print(f"The accuracy of DT is {tree.score(X_test.reshape(X_test.shape[0], -1),␣
↪y_test):.4f}, training time is {tree_training_time:.2f} s.")
print(f"The accuracy of RF is {rnd.score(X_test.reshape(X_test.shape[0], -1),␣
↪y_test):.4f}, training time is {rnd_training_time:.2f} s.")
```

```
The accuracy of KNN is 0.8519, training time is 0.05 s.
The accuracy of NB is 0.5838, training time is 0.54 s.
The accuracy of DT is 0.8001, training time is 28.93 s.
The accuracy of RF is 0.8760, training time is 78.00 s.
```

Althouth Random Forest performs best, the training time is relatively long. Noticing that KNN is simple but with a content accuracy among them, the training time is also tiny. Therefore, KNN is chosen.

1.3.2 2.2 Fully connected neural network

First, the numbers of layers need to be settled. Apart from the **input layer** and **output layer**, the numebr of hidden layer can be a variable. According to Cybenko(1998), any function (including discontinuous) can be approximated to arbitrary small error by a network with two hidden layers. To make the model small, we choose **two hidden layers**.

Number of neurons in the input layer: 784
For numerical attributes, basically 1 neuron per attribute, in this dataset, we have 28 * 28 = 784 atttributes each example. Thus, the number of neurons of input layer should be 784. Simply, we just use **keras.layers.Flatten(input_shape)**

Number of neurons in the output layer: 10
1 for each class. Therefore, the number of the output layers should be 10. The **softmax** function ($\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$) converts the raw outputs of this layer into a probability distribution over the classes.

Now we choose the hidden layers.
Sigmoid is the most widely used transfer function. We simply set most paras by default as well as the basic **SGD** learning algorithm. Since our labels are in index form rather than encoded as one-hot vectors, as we discussed earlier, we utilise the **sparse_categorical_crossentropy** loss. Then we observe the trend of the numbers of neurons with respect to MSE.

[ ]:
```
def test_build_mlp(num1=50, num2=50):
    """Build the MLP model with the specified number of neurons."""
    # Set Random seed to 0
    initializer = tf.keras.initializers.GlorotUniform(seed=0)
    # Define a test MLP model
    model = keras.models.Sequential([
```

```
        keras.layers.Flatten(input_shape=IMAGE_SIZE),
        keras.layers.Dense(num1, activation="sigmoid",␣
 ↪kernel_initializer=initializer),
        keras.layers.Dense(num2, activation="sigmoid",␣
 ↪kernel_initializer=initializer),
        keras.layers.Dense(len(class_names), activation="softmax",␣
 ↪kernel_initializer=initializer)
    ])

    opt = keras.optimizers.SGD() # default learning_rate=0.01
    model.compile(loss='sparse_categorical_crossentropy',
                  optimizer=opt,
                  metrics=['accuracy'])

    # model.summary()
    return model

def test_train_mlp(num1, num2, max_epochs=50, criterion=0.02):
    """Training the model.
    max_epochs: the maximum number of epochs to terminate.
    criterion: stop when the difference between the loss of the last 5 epoch is␣
 ↪less than."""
    # Train the classifier.
    mlp = test_build_mlp(num1, num2)
    loss_list = []
    for i in range(max_epochs):
        history = mlp.fit(X_train, y_train, validation_data=(X_valid, y_valid),␣
 ↪epochs=1)
        loss_list.append(history.history["loss"][0])

        # Stop condition
        if len(loss_list) > 5 and loss_list[-6] - loss_list[-1] < criterion:
            print(len(loss_list))
            break

    return loss_list[-1], len(loss_list)
```

[ ]:
```
# Running in arround 960s

# some possible numbers to choose
hidden_layer_1 = [100, 200, 300, 400, 500, 600]

# results
loss_history_1 = []
epoch_history_1 = []

# for the first hidden layer
```

```
for i in hidden_layer_1:
    loss, epoch = test_train_mlp(i, 50)
    loss_history_1.append(loss)
    epoch_history_1.append(epoch)
```

```
1688/1688 [==============================] - 2s 1ms/step - loss: 2.0764 -
accuracy: 0.4214 - val_loss: 1.7342 - val_accuracy: 0.5058
1688/1688 [==============================] - 2s 1ms/step - loss: 1.4601 -
accuracy: 0.5927 - val_loss: 1.2573 - val_accuracy: 0.6330
1688/1688 [==============================] - 2s 995us/step - loss: 1.1359 -
accuracy: 0.6638 - val_loss: 1.0339 - val_accuracy: 0.6807
1688/1688 [==============================] - 2s 994us/step - loss: 0.9574 -
accuracy: 0.6956 - val_loss: 0.8923 - val_accuracy: 0.7097
1688/1688 [==============================] - 2s 1ms/step - loss: 0.8384 -
accuracy: 0.7185 - val_loss: 0.7962 - val_accuracy: 0.7290
1688/1688 [==============================] - 2s 1ms/step - loss: 0.7582 -
accuracy: 0.7356 - val_loss: 0.7316 - val_accuracy: 0.7415
1688/1688 [==============================] - 2s 1ms/step - loss: 0.7038 -
accuracy: 0.7480 - val_loss: 0.6869 - val_accuracy: 0.7547
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6652 -
accuracy: 0.7582 - val_loss: 0.6541 - val_accuracy: 0.7613
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6357 -
accuracy: 0.7669 - val_loss: 0.6280 - val_accuracy: 0.7683
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6115 -
accuracy: 0.7766 - val_loss: 0.6060 - val_accuracy: 0.7760
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5906 -
accuracy: 0.7858 - val_loss: 0.5866 - val_accuracy: 0.7842
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5720 -
accuracy: 0.7937 - val_loss: 0.5691 - val_accuracy: 0.7918
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5552 -
accuracy: 0.8006 - val_loss: 0.5531 - val_accuracy: 0.7987
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5400 -
accuracy: 0.8083 - val_loss: 0.5388 - val_accuracy: 0.8080
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5265 -
accuracy: 0.8136 - val_loss: 0.5260 - val_accuracy: 0.8110
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5145 -
accuracy: 0.8180 - val_loss: 0.5147 - val_accuracy: 0.8147
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5039 -
accuracy: 0.8226 - val_loss: 0.5046 - val_accuracy: 0.8168
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4945 -
accuracy: 0.8259 - val_loss: 0.4956 - val_accuracy: 0.8202
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4862 -
accuracy: 0.8286 - val_loss: 0.4876 - val_accuracy: 0.8240
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4786 -
accuracy: 0.8315 - val_loss: 0.4803 - val_accuracy: 0.8270
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4718 -
accuracy: 0.8339 - val_loss: 0.4737 - val_accuracy: 0.8288
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4656 -
```

```
accuracy: 0.8361 - val_loss: 0.4676 - val_accuracy: 0.8335
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4599 -
accuracy: 0.8381 - val_loss: 0.4620 - val_accuracy: 0.8353
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4546 -
accuracy: 0.8399 - val_loss: 0.4568 - val_accuracy: 0.8367
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4497 -
accuracy: 0.8415 - val_loss: 0.4520 - val_accuracy: 0.8390
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4450 -
accuracy: 0.8432 - val_loss: 0.4475 - val_accuracy: 0.8402
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4407 -
accuracy: 0.8445 - val_loss: 0.4433 - val_accuracy: 0.8422
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4367 -
accuracy: 0.8461 - val_loss: 0.4394 - val_accuracy: 0.8427
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4328 -
accuracy: 0.8472 - val_loss: 0.4357 - val_accuracy: 0.8448
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4292 -
accuracy: 0.8487 - val_loss: 0.4322 - val_accuracy: 0.8452
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4257 -
accuracy: 0.8496 - val_loss: 0.4290 - val_accuracy: 0.8463
31
1688/1688 [==============================] - 2s 1ms/step - loss: 1.9981 -
accuracy: 0.4755 - val_loss: 1.6055 - val_accuracy: 0.5913
1688/1688 [==============================] - 2s 1ms/step - loss: 1.3420 -
accuracy: 0.6401 - val_loss: 1.1452 - val_accuracy: 0.6605
1688/1688 [==============================] - 2s 1ms/step - loss: 1.0257 -
accuracy: 0.6861 - val_loss: 0.9321 - val_accuracy: 0.6963
1688/1688 [==============================] - 2s 1ms/step - loss: 0.8658 -
accuracy: 0.7105 - val_loss: 0.8156 - val_accuracy: 0.7187
1688/1688 [==============================] - 2s 1ms/step - loss: 0.7734 -
accuracy: 0.7275 - val_loss: 0.7435 - val_accuracy: 0.7348
1688/1688 [==============================] - 2s 1ms/step - loss: 0.7139 -
accuracy: 0.7414 - val_loss: 0.6949 - val_accuracy: 0.7455
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6727 -
accuracy: 0.7525 - val_loss: 0.6600 - val_accuracy: 0.7552
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6419 -
accuracy: 0.7624 - val_loss: 0.6329 - val_accuracy: 0.7663
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6170 -
accuracy: 0.7723 - val_loss: 0.6102 - val_accuracy: 0.7773
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5957 -
accuracy: 0.7833 - val_loss: 0.5903 - val_accuracy: 0.7863
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5767 -
accuracy: 0.7928 - val_loss: 0.5724 - val_accuracy: 0.7933
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5596 -
accuracy: 0.8006 - val_loss: 0.5563 - val_accuracy: 0.7975
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5443 -
accuracy: 0.8072 - val_loss: 0.5419 - val_accuracy: 0.8037
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5306 -
accuracy: 0.8125 - val_loss: 0.5289 - val_accuracy: 0.8103
```

```
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5184 -
accuracy: 0.8173 - val_loss: 0.5173 - val_accuracy: 0.8148
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5076 -
accuracy: 0.8215 - val_loss: 0.5070 - val_accuracy: 0.8198
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4980 -
accuracy: 0.8250 - val_loss: 0.4978 - val_accuracy: 0.8212
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4895 -
accuracy: 0.8277 - val_loss: 0.4896 - val_accuracy: 0.8237
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4818 -
accuracy: 0.8307 - val_loss: 0.4822 - val_accuracy: 0.8253
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4749 -
accuracy: 0.8332 - val_loss: 0.4754 - val_accuracy: 0.8283
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4686 -
accuracy: 0.8353 - val_loss: 0.4693 - val_accuracy: 0.8313
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4629 -
accuracy: 0.8379 - val_loss: 0.4636 - val_accuracy: 0.8330
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4576 -
accuracy: 0.8404 - val_loss: 0.4584 - val_accuracy: 0.8333
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4527 -
accuracy: 0.8416 - val_loss: 0.4536 - val_accuracy: 0.8358
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4482 -
accuracy: 0.8433 - val_loss: 0.4491 - val_accuracy: 0.8385
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4439 -
accuracy: 0.8447 - val_loss: 0.4450 - val_accuracy: 0.8402
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4400 -
accuracy: 0.8461 - val_loss: 0.4411 - val_accuracy: 0.8407
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4362 -
accuracy: 0.8470 - val_loss: 0.4375 - val_accuracy: 0.8413
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4327 -
accuracy: 0.8480 - val_loss: 0.4341 - val_accuracy: 0.8425
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4294 -
accuracy: 0.8491 - val_loss: 0.4309 - val_accuracy: 0.8438
30
1688/1688 [==============================] - 3s 1ms/step - loss: 1.9690 -
accuracy: 0.4764 - val_loss: 1.5522 - val_accuracy: 0.5837
1688/1688 [==============================] - 2s 1ms/step - loss: 1.2944 -
accuracy: 0.6431 - val_loss: 1.1066 - val_accuracy: 0.6697
1688/1688 [==============================] - 2s 1ms/step - loss: 0.9952 -
accuracy: 0.6926 - val_loss: 0.9086 - val_accuracy: 0.7040
1688/1688 [==============================] - 2s 1ms/step - loss: 0.8437 -
accuracy: 0.7182 - val_loss: 0.7964 - val_accuracy: 0.7295
1688/1688 [==============================] - 2s 1ms/step - loss: 0.7535 -
accuracy: 0.7358 - val_loss: 0.7265 - val_accuracy: 0.7422
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6964 -
accuracy: 0.7488 - val_loss: 0.6811 - val_accuracy: 0.7530
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6581 -
accuracy: 0.7598 - val_loss: 0.6488 - val_accuracy: 0.7628
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6296 -
```

```
accuracy: 0.7699 - val_loss: 0.6234 - val_accuracy: 0.7740
1688/1688 [==============================] - 3s 1ms/step - loss: 0.6064 -
accuracy: 0.7790 - val_loss: 0.6020 - val_accuracy: 0.7810
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5864 -
accuracy: 0.7870 - val_loss: 0.5831 - val_accuracy: 0.7903
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5688 -
accuracy: 0.7954 - val_loss: 0.5662 - val_accuracy: 0.7962
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5529 -
accuracy: 0.8028 - val_loss: 0.5511 - val_accuracy: 0.8022
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5388 -
accuracy: 0.8083 - val_loss: 0.5375 - val_accuracy: 0.8065
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5262 -
accuracy: 0.8138 - val_loss: 0.5253 - val_accuracy: 0.8108
1688/1688 [==============================] - 3s 1ms/step - loss: 0.5150 -
accuracy: 0.8180 - val_loss: 0.5146 - val_accuracy: 0.8150
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5051 -
accuracy: 0.8214 - val_loss: 0.5050 - val_accuracy: 0.8183
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4963 -
accuracy: 0.8248 - val_loss: 0.4964 - val_accuracy: 0.8205
1688/1688 [==============================] - 3s 1ms/step - loss: 0.4885 -
accuracy: 0.8270 - val_loss: 0.4888 - val_accuracy: 0.8248
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4814 -
accuracy: 0.8300 - val_loss: 0.4818 - val_accuracy: 0.8260
1688/1688 [==============================] - 3s 1ms/step - loss: 0.4750 -
accuracy: 0.8321 - val_loss: 0.4755 - val_accuracy: 0.8285
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4692 -
accuracy: 0.8346 - val_loss: 0.4697 - val_accuracy: 0.8308
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4638 -
accuracy: 0.8365 - val_loss: 0.4644 - val_accuracy: 0.8325
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4589 -
accuracy: 0.8381 - val_loss: 0.4595 - val_accuracy: 0.8343
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4543 -
accuracy: 0.8397 - val_loss: 0.4549 - val_accuracy: 0.8360
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4500 -
accuracy: 0.8411 - val_loss: 0.4507 - val_accuracy: 0.8367
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4460 -
accuracy: 0.8426 - val_loss: 0.4467 - val_accuracy: 0.8370
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4422 -
accuracy: 0.8440 - val_loss: 0.4430 - val_accuracy: 0.8383
1688/1688 [==============================] - 3s 1ms/step - loss: 0.4386 -
accuracy: 0.8450 - val_loss: 0.4395 - val_accuracy: 0.8393
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4352 -
accuracy: 0.8459 - val_loss: 0.4363 - val_accuracy: 0.8408
29
1688/1688 [==============================] - 3s 2ms/step - loss: 1.9480 -
accuracy: 0.4653 - val_loss: 1.5226 - val_accuracy: 0.5795
1688/1688 [==============================] - 3s 2ms/step - loss: 1.2741 -
accuracy: 0.6367 - val_loss: 1.0943 - val_accuracy: 0.6667
```

```
1688/1688 [==============================] - 3s 2ms/step - loss: 0.9860 -
accuracy: 0.6929 - val_loss: 0.9016 - val_accuracy: 0.7058
1688/1688 [==============================] - 3s 2ms/step - loss: 0.8379 -
accuracy: 0.7202 - val_loss: 0.7915 - val_accuracy: 0.7277
1688/1688 [==============================] - 3s 2ms/step - loss: 0.7494 -
accuracy: 0.7370 - val_loss: 0.7228 - val_accuracy: 0.7433
1688/1688 [==============================] - 3s 2ms/step - loss: 0.6930 -
accuracy: 0.7492 - val_loss: 0.6776 - val_accuracy: 0.7548
1688/1688 [==============================] - 3s 2ms/step - loss: 0.6547 -
accuracy: 0.7604 - val_loss: 0.6454 - val_accuracy: 0.7652
1688/1688 [==============================] - 3s 2ms/step - loss: 0.6261 -
accuracy: 0.7711 - val_loss: 0.6199 - val_accuracy: 0.7740
1688/1688 [==============================] - 3s 2ms/step - loss: 0.6028 -
accuracy: 0.7805 - val_loss: 0.5983 - val_accuracy: 0.7817
1688/1688 [==============================] - 3s 2ms/step - loss: 0.5826 -
accuracy: 0.7891 - val_loss: 0.5792 - val_accuracy: 0.7898
1688/1688 [==============================] - 3s 2ms/step - loss: 0.5648 -
accuracy: 0.7968 - val_loss: 0.5621 - val_accuracy: 0.7978
1688/1688 [==============================] - 3s 2ms/step - loss: 0.5488 -
accuracy: 0.8034 - val_loss: 0.5469 - val_accuracy: 0.8045
1688/1688 [==============================] - 3s 2ms/step - loss: 0.5347 -
accuracy: 0.8095 - val_loss: 0.5333 - val_accuracy: 0.8085
1688/1688 [==============================] - 3s 2ms/step - loss: 0.5222 -
accuracy: 0.8148 - val_loss: 0.5213 - val_accuracy: 0.8138
1688/1688 [==============================] - 3s 2ms/step - loss: 0.5112 -
accuracy: 0.8191 - val_loss: 0.5108 - val_accuracy: 0.8170
1688/1688 [==============================] - 3s 2ms/step - loss: 0.5016 -
accuracy: 0.8222 - val_loss: 0.5014 - val_accuracy: 0.8190
1688/1688 [==============================] - 3s 2ms/step - loss: 0.4931 -
accuracy: 0.8250 - val_loss: 0.4932 - val_accuracy: 0.8220
1688/1688 [==============================] - 3s 2ms/step - loss: 0.4855 -
accuracy: 0.8283 - val_loss: 0.4858 - val_accuracy: 0.8240
1688/1688 [==============================] - 3s 2ms/step - loss: 0.4787 -
accuracy: 0.8309 - val_loss: 0.4791 - val_accuracy: 0.8267
1688/1688 [==============================] - 3s 2ms/step - loss: 0.4725 -
accuracy: 0.8333 - val_loss: 0.4730 - val_accuracy: 0.8293
1688/1688 [==============================] - 3s 2ms/step - loss: 0.4669 -
accuracy: 0.8352 - val_loss: 0.4675 - val_accuracy: 0.8307
1688/1688 [==============================] - 3s 2ms/step - loss: 0.4618 -
accuracy: 0.8371 - val_loss: 0.4624 - val_accuracy: 0.8327
1688/1688 [==============================] - 3s 2ms/step - loss: 0.4571 -
accuracy: 0.8388 - val_loss: 0.4577 - val_accuracy: 0.8345
1688/1688 [==============================] - 3s 2ms/step - loss: 0.4527 -
accuracy: 0.8403 - val_loss: 0.4534 - val_accuracy: 0.8355
1688/1688 [==============================] - 3s 2ms/step - loss: 0.4486 -
accuracy: 0.8415 - val_loss: 0.4493 - val_accuracy: 0.8372
1688/1688 [==============================] - 3s 2ms/step - loss: 0.4447 -
accuracy: 0.8426 - val_loss: 0.4456 - val_accuracy: 0.8385
```

```
1688/1688 [==============================] - 3s 2ms/step - loss: 0.4411 -
accuracy: 0.8437 - val_loss: 0.4420 - val_accuracy: 0.8398
1688/1688 [==============================] - 3s 2ms/step - loss: 0.4377 -
accuracy: 0.8448 - val_loss: 0.4387 - val_accuracy: 0.8400
28
1688/1688 [==============================] - 4s 2ms/step - loss: 1.9466 -
accuracy: 0.4900 - val_loss: 1.5072 - val_accuracy: 0.6040
1688/1688 [==============================] - 3s 2ms/step - loss: 1.2489 -
accuracy: 0.6576 - val_loss: 1.0637 - val_accuracy: 0.6797
1688/1688 [==============================] - 3s 2ms/step - loss: 0.9579 -
accuracy: 0.7007 - val_loss: 0.8769 - val_accuracy: 0.7113
1688/1688 [==============================] - 4s 2ms/step - loss: 0.8172 -
accuracy: 0.7249 - val_loss: 0.7744 - val_accuracy: 0.7318
1688/1688 [==============================] - 4s 2ms/step - loss: 0.7355 -
accuracy: 0.7410 - val_loss: 0.7116 - val_accuracy: 0.7452
1688/1688 [==============================] - 3s 2ms/step - loss: 0.6839 -
accuracy: 0.7527 - val_loss: 0.6701 - val_accuracy: 0.7553
1688/1688 [==============================] - 3s 2ms/step - loss: 0.6483 -
accuracy: 0.7632 - val_loss: 0.6397 - val_accuracy: 0.7665
1688/1688 [==============================] - 3s 2ms/step - loss: 0.6211 -
accuracy: 0.7730 - val_loss: 0.6152 - val_accuracy: 0.7770
1688/1688 [==============================] - 3s 2ms/step - loss: 0.5985 -
accuracy: 0.7824 - val_loss: 0.5942 - val_accuracy: 0.7857
1688/1688 [==============================] - 3s 2ms/step - loss: 0.5788 -
accuracy: 0.7910 - val_loss: 0.5756 - val_accuracy: 0.7930
1688/1688 [==============================] - 3s 2ms/step - loss: 0.5614 -
accuracy: 0.7985 - val_loss: 0.5589 - val_accuracy: 0.7998
1688/1688 [==============================] - 3s 2ms/step - loss: 0.5458 -
accuracy: 0.8055 - val_loss: 0.5439 - val_accuracy: 0.8027
1688/1688 [==============================] - 3s 2ms/step - loss: 0.5320 -
accuracy: 0.8109 - val_loss: 0.5307 - val_accuracy: 0.8093
1688/1688 [==============================] - 3s 2ms/step - loss: 0.5199 -
accuracy: 0.8159 - val_loss: 0.5191 - val_accuracy: 0.8140
1688/1688 [==============================] - 3s 2ms/step - loss: 0.5093 -
accuracy: 0.8202 - val_loss: 0.5088 - val_accuracy: 0.8172
1688/1688 [==============================] - 3s 2ms/step - loss: 0.5000 -
accuracy: 0.8231 - val_loss: 0.4997 - val_accuracy: 0.8218
1688/1688 [==============================] - 3s 2ms/step - loss: 0.4918 -
accuracy: 0.8260 - val_loss: 0.4917 - val_accuracy: 0.8232
1688/1688 [==============================] - 3s 2ms/step - loss: 0.4844 -
accuracy: 0.8289 - val_loss: 0.4845 - val_accuracy: 0.8257
1688/1688 [==============================] - 3s 2ms/step - loss: 0.4779 -
accuracy: 0.8314 - val_loss: 0.4779 - val_accuracy: 0.8280
1688/1688 [==============================] - 4s 2ms/step - loss: 0.4719 -
accuracy: 0.8336 - val_loss: 0.4720 - val_accuracy: 0.8305
1688/1688 [==============================] - 4s 2ms/step - loss: 0.4664 -
accuracy: 0.8354 - val_loss: 0.4665 - val_accuracy: 0.8317
1688/1688 [==============================] - 3s 2ms/step - loss: 0.4614 -
```

```
accuracy: 0.8370 - val_loss: 0.4615 - val_accuracy: 0.8337
1688/1688 [==============================] - 3s 2ms/step - loss: 0.4567 -
accuracy: 0.8389 - val_loss: 0.4568 - val_accuracy: 0.8347
1688/1688 [==============================] - 3s 2ms/step - loss: 0.4524 -
accuracy: 0.8405 - val_loss: 0.4525 - val_accuracy: 0.8365
1688/1688 [==============================] - 3s 2ms/step - loss: 0.4483 -
accuracy: 0.8421 - val_loss: 0.4485 - val_accuracy: 0.8368
1688/1688 [==============================] - 3s 2ms/step - loss: 0.4445 -
accuracy: 0.8434 - val_loss: 0.4447 - val_accuracy: 0.8382
1688/1688 [==============================] - 3s 2ms/step - loss: 0.4410 -
accuracy: 0.8446 - val_loss: 0.4412 - val_accuracy: 0.8383
1688/1688 [==============================] - 3s 2ms/step - loss: 0.4376 -
accuracy: 0.8454 - val_loss: 0.4379 - val_accuracy: 0.8400
28
1688/1688 [==============================] - 5s 2ms/step - loss: 1.9301 -
accuracy: 0.4911 - val_loss: 1.4847 - val_accuracy: 0.6127
1688/1688 [==============================] - 4s 2ms/step - loss: 1.2330 -
accuracy: 0.6584 - val_loss: 1.0528 - val_accuracy: 0.6855
1688/1688 [==============================] - 4s 2ms/step - loss: 0.9478 -
accuracy: 0.7055 - val_loss: 0.8670 - val_accuracy: 0.7185
1688/1688 [==============================] - 4s 2ms/step - loss: 0.8065 -
accuracy: 0.7298 - val_loss: 0.7646 - val_accuracy: 0.7335
1688/1688 [==============================] - 4s 3ms/step - loss: 0.7259 -
accuracy: 0.7451 - val_loss: 0.7040 - val_accuracy: 0.7473
1688/1688 [==============================] - 4s 2ms/step - loss: 0.6765 -
accuracy: 0.7565 - val_loss: 0.6646 - val_accuracy: 0.7593
1688/1688 [==============================] - 4s 2ms/step - loss: 0.6425 -
accuracy: 0.7664 - val_loss: 0.6356 - val_accuracy: 0.7692
1688/1688 [==============================] - 4s 2ms/step - loss: 0.6164 -
accuracy: 0.7765 - val_loss: 0.6120 - val_accuracy: 0.7792
1688/1688 [==============================] - 4s 2ms/step - loss: 0.5947 -
accuracy: 0.7854 - val_loss: 0.5917 - val_accuracy: 0.7880
1688/1688 [==============================] - 4s 2ms/step - loss: 0.5758 -
accuracy: 0.7936 - val_loss: 0.5738 - val_accuracy: 0.7937
1688/1688 [==============================] - 4s 2ms/step - loss: 0.5591 -
accuracy: 0.8005 - val_loss: 0.5578 - val_accuracy: 0.8002
1688/1688 [==============================] - 4s 2ms/step - loss: 0.5442 -
accuracy: 0.8068 - val_loss: 0.5436 - val_accuracy: 0.8053
1688/1688 [==============================] - 4s 2ms/step - loss: 0.5310 -
accuracy: 0.8124 - val_loss: 0.5309 - val_accuracy: 0.8113
1688/1688 [==============================] - 4s 2ms/step - loss: 0.5193 -
accuracy: 0.8174 - val_loss: 0.5196 - val_accuracy: 0.8147
1688/1688 [==============================] - 4s 3ms/step - loss: 0.5089 -
accuracy: 0.8210 - val_loss: 0.5095 - val_accuracy: 0.8172
1688/1688 [==============================] - 4s 2ms/step - loss: 0.4997 -
accuracy: 0.8240 - val_loss: 0.5006 - val_accuracy: 0.8202
1688/1688 [==============================] - 4s 2ms/step - loss: 0.4915 -
accuracy: 0.8263 - val_loss: 0.4926 - val_accuracy: 0.8222
```

```
1688/1688 [==============================] - 4s 2ms/step - loss: 0.4841 -
accuracy: 0.8294 - val_loss: 0.4853 - val_accuracy: 0.8253
1688/1688 [==============================] - 4s 2ms/step - loss: 0.4775 -
accuracy: 0.8320 - val_loss: 0.4787 - val_accuracy: 0.8273
1688/1688 [==============================] - 4s 2ms/step - loss: 0.4714 -
accuracy: 0.8340 - val_loss: 0.4727 - val_accuracy: 0.8293
1688/1688 [==============================] - 4s 2ms/step - loss: 0.4659 -
accuracy: 0.8363 - val_loss: 0.4672 - val_accuracy: 0.8317
1688/1688 [==============================] - 4s 2ms/step - loss: 0.4608 -
accuracy: 0.8381 - val_loss: 0.4622 - val_accuracy: 0.8343
1688/1688 [==============================] - 4s 3ms/step - loss: 0.4561 -
accuracy: 0.8398 - val_loss: 0.4575 - val_accuracy: 0.8348
1688/1688 [==============================] - 4s 3ms/step - loss: 0.4517 -
accuracy: 0.8415 - val_loss: 0.4532 - val_accuracy: 0.8350
1688/1688 [==============================] - 4s 2ms/step - loss: 0.4477 -
accuracy: 0.8429 - val_loss: 0.4491 - val_accuracy: 0.8357
1688/1688 [==============================] - 4s 2ms/step - loss: 0.4439 -
accuracy: 0.8442 - val_loss: 0.4454 - val_accuracy: 0.8370
1688/1688 [==============================] - 4s 2ms/step - loss: 0.4403 -
accuracy: 0.8455 - val_loss: 0.4419 - val_accuracy: 0.8387
1688/1688 [==============================] - 4s 2ms/step - loss: 0.4369 -
accuracy: 0.8463 - val_loss: 0.4386 - val_accuracy: 0.8400
28
```

[ ]:
```
loss_history_1 = [round(i, 4) for i in loss_history_1]
print(f"The min loss is {min(loss_history_1)} when neurons is equal to␣
 ↪{hidden_layer_1[loss_history_1.index(min(loss_history_1))]}")

print(f"Loss history: {loss_history_1}")
print(f"Epoch history: {epoch_history_1}")
```

```
The min loss is 0.4257 when neurons is equal to 100
Loss history: [0.4257, 0.4294, 0.4352, 0.4377, 0.4376, 0.4369]
Epoch history: [31, 30, 29, 28, 28, 28]
```

It shows that when the number of neurons of the first layer is 100, the loss is minimal. Although the epochs is slightly larger, which may enhance the performance, the training time is much faster because of the lesser neurons. To keep the model small, we choose the 100 as the first number of neurons of the hidden layers.

[ ]:
```
# Running in arround 340s

# some possible numbers to choose
hidden_layer_2 = [20, 40, 60, 80]

# results
loss_history_2 = []
epoch_history_2 = []
```

```
# for the first hidden layer
for i in hidden_layer_2:
    loss, epoch = test_train_mlp(100, i)
    loss_history_2.append(loss)
    epoch_history_2.append(epoch)
```

```
1688/1688 [==============================] - 2s 1ms/step - loss: 2.1365 -
accuracy: 0.3363 - val_loss: 1.8808 - val_accuracy: 0.4397
1688/1688 [==============================] - 2s 1ms/step - loss: 1.6100 -
accuracy: 0.5462 - val_loss: 1.3973 - val_accuracy: 0.6027
1688/1688 [==============================] - 2s 1ms/step - loss: 1.2698 -
accuracy: 0.6288 - val_loss: 1.1570 - val_accuracy: 0.6437
1688/1688 [==============================] - 2s 1ms/step - loss: 1.0734 -
accuracy: 0.6632 - val_loss: 0.9944 - val_accuracy: 0.6720
1688/1688 [==============================] - 2s 1ms/step - loss: 0.9393 -
accuracy: 0.6847 - val_loss: 0.8852 - val_accuracy: 0.6940
1688/1688 [==============================] - 2s 1ms/step - loss: 0.8490 -
accuracy: 0.7011 - val_loss: 0.8106 - val_accuracy: 0.7128
1688/1688 [==============================] - 2s 1ms/step - loss: 0.7857 -
accuracy: 0.7149 - val_loss: 0.7569 - val_accuracy: 0.7232
1688/1688 [==============================] - 2s 1ms/step - loss: 0.7393 -
accuracy: 0.7264 - val_loss: 0.7167 - val_accuracy: 0.7350
1688/1688 [==============================] - 2s 1ms/step - loss: 0.7038 -
accuracy: 0.7383 - val_loss: 0.6854 - val_accuracy: 0.7442
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6753 -
accuracy: 0.7479 - val_loss: 0.6598 - val_accuracy: 0.7515
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6513 -
accuracy: 0.7579 - val_loss: 0.6377 - val_accuracy: 0.7603
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6300 -
accuracy: 0.7667 - val_loss: 0.6180 - val_accuracy: 0.7698
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6105 -
accuracy: 0.7761 - val_loss: 0.6001 - val_accuracy: 0.7800
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5926 -
accuracy: 0.7845 - val_loss: 0.5838 - val_accuracy: 0.7873
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5763 -
accuracy: 0.7928 - val_loss: 0.5691 - val_accuracy: 0.7943
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5615 -
accuracy: 0.7998 - val_loss: 0.5558 - val_accuracy: 0.8000
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5480 -
accuracy: 0.8062 - val_loss: 0.5437 - val_accuracy: 0.8055
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5358 -
accuracy: 0.8114 - val_loss: 0.5326 - val_accuracy: 0.8105
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5246 -
accuracy: 0.8160 - val_loss: 0.5224 - val_accuracy: 0.8142
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5144 -
accuracy: 0.8201 - val_loss: 0.5130 - val_accuracy: 0.8182
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5051 -
```

```
accuracy: 0.8235 - val_loss: 0.5045 - val_accuracy: 0.8212
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4966 -
accuracy: 0.8267 - val_loss: 0.4966 - val_accuracy: 0.8233
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4888 -
accuracy: 0.8295 - val_loss: 0.4894 - val_accuracy: 0.8258
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4818 -
accuracy: 0.8322 - val_loss: 0.4829 - val_accuracy: 0.8288
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4754 -
accuracy: 0.8341 - val_loss: 0.4769 - val_accuracy: 0.8307
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4695 -
accuracy: 0.8367 - val_loss: 0.4714 - val_accuracy: 0.8312
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4641 -
accuracy: 0.8385 - val_loss: 0.4664 - val_accuracy: 0.8318
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4591 -
accuracy: 0.8406 - val_loss: 0.4617 - val_accuracy: 0.8335
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4545 -
accuracy: 0.8422 - val_loss: 0.4573 - val_accuracy: 0.8357
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4502 -
accuracy: 0.8438 - val_loss: 0.4532 - val_accuracy: 0.8367
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4461 -
accuracy: 0.8452 - val_loss: 0.4493 - val_accuracy: 0.8382
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4423 -
accuracy: 0.8461 - val_loss: 0.4456 - val_accuracy: 0.8402
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4386 -
accuracy: 0.8471 - val_loss: 0.4422 - val_accuracy: 0.8412
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4352 -
accuracy: 0.8483 - val_loss: 0.4389 - val_accuracy: 0.8427
34
1688/1688 [==============================] - 2s 1ms/step - loss: 2.0688 -
accuracy: 0.4429 - val_loss: 1.7436 - val_accuracy: 0.5248
1688/1688 [==============================] - 2s 1ms/step - loss: 1.4827 -
accuracy: 0.5942 - val_loss: 1.2786 - val_accuracy: 0.6217
1688/1688 [==============================] - 2s 1ms/step - loss: 1.1467 -
accuracy: 0.6560 - val_loss: 1.0372 - val_accuracy: 0.6682
1688/1688 [==============================] - 2s 1ms/step - loss: 0.9572 -
accuracy: 0.6917 - val_loss: 0.8919 - val_accuracy: 0.7018
1688/1688 [==============================] - 2s 1ms/step - loss: 0.8381 -
accuracy: 0.7173 - val_loss: 0.7973 - val_accuracy: 0.7225
1688/1688 [==============================] - 2s 1ms/step - loss: 0.7588 -
accuracy: 0.7369 - val_loss: 0.7331 - val_accuracy: 0.7407
1688/1688 [==============================] - 2s 1ms/step - loss: 0.7039 -
accuracy: 0.7502 - val_loss: 0.6880 - val_accuracy: 0.7530
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6644 -
accuracy: 0.7601 - val_loss: 0.6547 - val_accuracy: 0.7613
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6344 -
accuracy: 0.7695 - val_loss: 0.6286 - val_accuracy: 0.7713
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6100 -
accuracy: 0.7794 - val_loss: 0.6067 - val_accuracy: 0.7792
```

```
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5893 -
accuracy: 0.7881 - val_loss: 0.5875 - val_accuracy: 0.7873
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5709 -
accuracy: 0.7961 - val_loss: 0.5704 - val_accuracy: 0.7938
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5546 -
accuracy: 0.8038 - val_loss: 0.5550 - val_accuracy: 0.8002
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5401 -
accuracy: 0.8096 - val_loss: 0.5412 - val_accuracy: 0.8078
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5271 -
accuracy: 0.8148 - val_loss: 0.5290 - val_accuracy: 0.8122
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5157 -
accuracy: 0.8193 - val_loss: 0.5181 - val_accuracy: 0.8152
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5056 -
accuracy: 0.8221 - val_loss: 0.5084 - val_accuracy: 0.8190
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4965 -
accuracy: 0.8249 - val_loss: 0.4997 - val_accuracy: 0.8212
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4884 -
accuracy: 0.8276 - val_loss: 0.4919 - val_accuracy: 0.8237
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4810 -
accuracy: 0.8300 - val_loss: 0.4847 - val_accuracy: 0.8255
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4743 -
accuracy: 0.8324 - val_loss: 0.4781 - val_accuracy: 0.8280
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4682 -
accuracy: 0.8351 - val_loss: 0.4720 - val_accuracy: 0.8302
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4625 -
accuracy: 0.8367 - val_loss: 0.4663 - val_accuracy: 0.8322
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4572 -
accuracy: 0.8389 - val_loss: 0.4610 - val_accuracy: 0.8337
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4522 -
accuracy: 0.8406 - val_loss: 0.4561 - val_accuracy: 0.8352
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4475 -
accuracy: 0.8423 - val_loss: 0.4515 - val_accuracy: 0.8365
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4431 -
accuracy: 0.8439 - val_loss: 0.4471 - val_accuracy: 0.8392
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4390 -
accuracy: 0.8458 - val_loss: 0.4430 - val_accuracy: 0.8408
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4350 -
accuracy: 0.8473 - val_loss: 0.4391 - val_accuracy: 0.8417
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4313 -
accuracy: 0.8484 - val_loss: 0.4355 - val_accuracy: 0.8428
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4277 -
accuracy: 0.8493 - val_loss: 0.4320 - val_accuracy: 0.8438
31
1688/1688 [==============================] - 2s 1ms/step - loss: 2.0653 -
accuracy: 0.4333 - val_loss: 1.7175 - val_accuracy: 0.5725
1688/1688 [==============================] - 2s 1ms/step - loss: 1.4502 -
accuracy: 0.6172 - val_loss: 1.2486 - val_accuracy: 0.6430
1688/1688 [==============================] - 2s 1ms/step - loss: 1.1186 -
```

```
accuracy: 0.6708 - val_loss: 1.0098 - val_accuracy: 0.6792
1688/1688 [==============================] - 2s 1ms/step - loss: 0.9312 -
accuracy: 0.6975 - val_loss: 0.8688 - val_accuracy: 0.7050
1688/1688 [==============================] - 2s 1ms/step - loss: 0.8186 -
accuracy: 0.7177 - val_loss: 0.7807 - val_accuracy: 0.7285
1688/1688 [==============================] - 2s 1ms/step - loss: 0.7449 -
accuracy: 0.7365 - val_loss: 0.7199 - val_accuracy: 0.7443
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6929 -
accuracy: 0.7490 - val_loss: 0.6760 - val_accuracy: 0.7550
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6547 -
accuracy: 0.7605 - val_loss: 0.6432 - val_accuracy: 0.7638
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6253 -
accuracy: 0.7723 - val_loss: 0.6170 - val_accuracy: 0.7745
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6011 -
accuracy: 0.7823 - val_loss: 0.5949 - val_accuracy: 0.7825
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5802 -
accuracy: 0.7918 - val_loss: 0.5755 - val_accuracy: 0.7927
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5617 -
accuracy: 0.7999 - val_loss: 0.5581 - val_accuracy: 0.8000
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5453 -
accuracy: 0.8067 - val_loss: 0.5425 - val_accuracy: 0.8060
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5308 -
accuracy: 0.8123 - val_loss: 0.5287 - val_accuracy: 0.8102
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5180 -
accuracy: 0.8170 - val_loss: 0.5165 - val_accuracy: 0.8147
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5068 -
accuracy: 0.8216 - val_loss: 0.5058 - val_accuracy: 0.8178
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4969 -
accuracy: 0.8253 - val_loss: 0.4963 - val_accuracy: 0.8207
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4882 -
accuracy: 0.8278 - val_loss: 0.4879 - val_accuracy: 0.8245
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4804 -
accuracy: 0.8301 - val_loss: 0.4802 - val_accuracy: 0.8263
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4733 -
accuracy: 0.8326 - val_loss: 0.4733 - val_accuracy: 0.8288
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4668 -
accuracy: 0.8351 - val_loss: 0.4669 - val_accuracy: 0.8313
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4609 -
accuracy: 0.8374 - val_loss: 0.4611 - val_accuracy: 0.8343
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4554 -
accuracy: 0.8396 - val_loss: 0.4557 - val_accuracy: 0.8372
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4503 -
accuracy: 0.8411 - val_loss: 0.4507 - val_accuracy: 0.8382
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4455 -
accuracy: 0.8426 - val_loss: 0.4460 - val_accuracy: 0.8402
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4410 -
accuracy: 0.8441 - val_loss: 0.4416 - val_accuracy: 0.8412
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4368 -
```

```
accuracy: 0.8457 - val_loss: 0.4375 - val_accuracy: 0.8427
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4328 -
accuracy: 0.8472 - val_loss: 0.4337 - val_accuracy: 0.8432
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4290 -
accuracy: 0.8485 - val_loss: 0.4301 - val_accuracy: 0.8437
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4254 -
accuracy: 0.8498 - val_loss: 0.4267 - val_accuracy: 0.8443
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4220 -
accuracy: 0.8510 - val_loss: 0.4235 - val_accuracy: 0.8455
31
1688/1688 [==============================] - 2s 1ms/step - loss: 2.0711 -
accuracy: 0.4232 - val_loss: 1.7194 - val_accuracy: 0.5698
1688/1688 [==============================] - 2s 1ms/step - loss: 1.4452 -
accuracy: 0.6097 - val_loss: 1.2367 - val_accuracy: 0.6330
1688/1688 [==============================] - 2s 1ms/step - loss: 1.1097 -
accuracy: 0.6600 - val_loss: 1.0070 - val_accuracy: 0.6763
1688/1688 [==============================] - 2s 1ms/step - loss: 0.9330 -
accuracy: 0.6980 - val_loss: 0.8732 - val_accuracy: 0.7097
1688/1688 [==============================] - 2s 1ms/step - loss: 0.8199 -
accuracy: 0.7241 - val_loss: 0.7812 - val_accuracy: 0.7327
1688/1688 [==============================] - 2s 1ms/step - loss: 0.7412 -
accuracy: 0.7427 - val_loss: 0.7169 - val_accuracy: 0.7463
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6868 -
accuracy: 0.7548 - val_loss: 0.6723 - val_accuracy: 0.7592
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6485 -
accuracy: 0.7645 - val_loss: 0.6399 - val_accuracy: 0.7665
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6200 -
accuracy: 0.7739 - val_loss: 0.6147 - val_accuracy: 0.7755
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5972 -
accuracy: 0.7819 - val_loss: 0.5938 - val_accuracy: 0.7827
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5779 -
accuracy: 0.7901 - val_loss: 0.5758 - val_accuracy: 0.7893
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5611 -
accuracy: 0.7970 - val_loss: 0.5599 - val_accuracy: 0.7968
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5463 -
accuracy: 0.8042 - val_loss: 0.5458 - val_accuracy: 0.8028
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5330 -
accuracy: 0.8087 - val_loss: 0.5331 - val_accuracy: 0.8083
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5211 -
accuracy: 0.8138 - val_loss: 0.5218 - val_accuracy: 0.8122
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5105 -
accuracy: 0.8182 - val_loss: 0.5116 - val_accuracy: 0.8153
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5009 -
accuracy: 0.8219 - val_loss: 0.5025 - val_accuracy: 0.8188
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4923 -
accuracy: 0.8252 - val_loss: 0.4942 - val_accuracy: 0.8210
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4845 -
accuracy: 0.8284 - val_loss: 0.4867 - val_accuracy: 0.8238
```

```
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4774 -
accuracy: 0.8313 - val_loss: 0.4799 - val_accuracy: 0.8267
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4710 -
accuracy: 0.8336 - val_loss: 0.4736 - val_accuracy: 0.8280
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4650 -
accuracy: 0.8363 - val_loss: 0.4677 - val_accuracy: 0.8307
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4595 -
accuracy: 0.8381 - val_loss: 0.4623 - val_accuracy: 0.8330
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4544 -
accuracy: 0.8404 - val_loss: 0.4573 - val_accuracy: 0.8340
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4496 -
accuracy: 0.8418 - val_loss: 0.4525 - val_accuracy: 0.8355
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4451 -
accuracy: 0.8432 - val_loss: 0.4481 - val_accuracy: 0.8362
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4409 -
accuracy: 0.8446 - val_loss: 0.4439 - val_accuracy: 0.8378
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4369 -
accuracy: 0.8457 - val_loss: 0.4400 - val_accuracy: 0.8390
1688/1688 [==============================] - 3s 2ms/step - loss: 0.4331 -
accuracy: 0.8465 - val_loss: 0.4363 - val_accuracy: 0.8400
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4295 -
accuracy: 0.8476 - val_loss: 0.4328 - val_accuracy: 0.8408
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4261 -
accuracy: 0.8487 - val_loss: 0.4294 - val_accuracy: 0.8420
31
```

[ ]:
```
loss_history_2 = [round(i, 4) for i in loss_history_2]
print(f"The min loss is {min(loss_history_2)} when neurons is equal to␣
  ↪{hidden_layer_2[loss_history_2.index(min(loss_history_2))]}")

print(f"Loss history: {loss_history_2}")
print(f"Epoch history: {epoch_history_2}")
```

```
The min loss is 0.422 when neurons is equal to 60
Loss history: [0.4352, 0.4277, 0.422, 0.4261]
Epoch history: [34, 31, 31, 31]
```

We select 60 as the neurons of the second hidden layer, as it achieves the best result with a less epochs, and the training time is similar.

1.3.3 2.3 Convolutional neural network

The architecture of CNN is settled in this section.

The input shape shoule be (28, 28, 1). This last dimension indicates the channel. The dataset consists of grey images, thus, the channel is 1. Although usually the CNN performs well, the theory of determining the architecture, such as the number of convolutional layers, the number of filters, is unclear. Sometimes it is determined by experiment and experience. Given that the condition is limited, we just choose the similar architecture from the tutorial. In other words, we

choose two conv and pool blocks with 32, 64 (empirically it should be exponentiation of 2) filters respectively, follower by a simple FC layer.

We also choose Max Pooling due to the distribution of pixels each image (features are presented as large value of pixels).

We have discussed the ReLU and Softmax previously.

The drouout layer is added to avoid overfitting.

The rest paras such as filter size, strides, and learning rate, will be tuned in section 3.3.

[ ]:
```
keras.Sequential([
    # Specify the input shape
    keras.Input(shape=(*IMAGE_SIZE, 1)),

    # Conv and pool block 1
    keras.layers.Conv2D(32, kernel_size=(3, 3), activation="relu", strides=(1,␣
 ↪1), kernel_initializer=initializer),
    keras.layers.MaxPooling2D(pool_size=(2, 2), padding='same'), # padding␣
 ↪evenly

    # Conv and pool block 2
    keras.layers.Conv2D(64, kernel_size=(3, 3), activation="relu", strides=(1,␣
 ↪1), kernel_initializer=initializer),
    keras.layers.MaxPooling2D(pool_size=(2, 2), padding='same'), # padding␣
 ↪evenly

    # Flatten and classify using dense output layer
    keras.layers.Flatten(),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(10, activation="softmax",␣
 ↪kernel_initializer=initializer),
])
```

[ ]:
```
<keras.engine.sequential.Sequential at 0x1d1b7770d60>
```

1.4 3. Hyperparameter tuning

[ ]:
```
# Helper method in this section.
def get_result(estimator, paras, X_train=X_train, y_train=y_train,␣
 ↪X_valid=X_valid, y_valid=y_valid, epochs=1):
    """get grid search result.
    estimator: model to be tuned.
    paras: an instance of ParameterGrid."""
    # Return a dict
    result = {
        "best_paras": None,
        "best_score": 0,
```

```
        "best_estimator": None,
        "results": [],
    }

    i = 0

    # Grid search for each combination.
    for para in paras:
        # Set para
        current_estimator = clone(estimator)
        current_estimator.set_params(**para)

        # Training and timer
        t1 = time.time()
        if epochs == 1:
            current_estimator.fit(X_train, y_train)
        else:
            current_estimator.fit(X_train, y_train, epochs=epochs)
        t2 = time.time()

        # Score on validation set
        score = current_estimator.score(X_valid, y_valid)
        t3 = time.time()

        # result for each combination
        temp = {}
        temp["paras"] = para
        temp["training_time"] = t2 - t1
        temp["validation_time"] = t3 - t2
        temp["score"] =score

        # Update the best result
        result["results"].append(temp)
        if score > result["best_score"]:
            result["best_paras"] = para
            result["best_score"] = score
            result["best_estimator"] = current_estimator

        i += 1
        print(f"{i} out of {len(list(paras))} finished: {para}")

    return result

def show_results(name, result, X_test=X_test, y_test=y_test):
    """Show the results."""
    print(f"Results for {name}:")
    print("Best parameters: {}".format(result["best_paras"]))
```

```
    print("Best validation score: {:.4f}".format(result["best_score"]))
    print("Test set score: {:.4f}".format(result["best_estimator"].
↪score(X_test, y_test)))

    # table of results
    df = pd.DataFrame(
        columns=["Score", "training_time", "validation_time"],
        index=[str(result["results"][i]["paras"]) for i in␣
↪range(len(result["results"]))]
        )

    for i in range(len(result["results"])):
        df.loc[str(result["results"][i]["paras"])] = [
            round(result["results"][i]["score"], 4),
            round(result["results"][i]["training_time"], 2),
            round(result["results"][i]["validation_time"], 2)
            ]

    df.to_csv(f'{name}_results.csv')
    display(df)
```

1.4.1 3.1 K-nearest neighbors

First, to determine a rough trend of the accuracy with different k, we calculate an accuracy every 10 with different k, until k = 245 (sqrt(#examples)), e.g., k = [1, 11, 21, ..., 241]

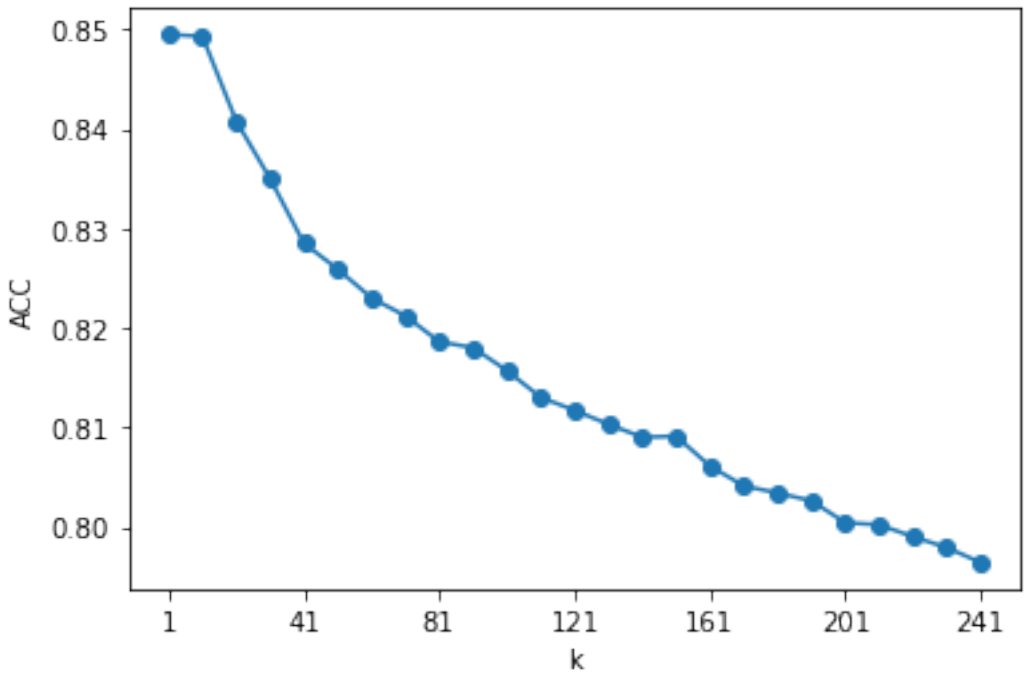
[ ]:
```
# Running in arroung 190s

k_value = [1 + i for i in range(245)]
k_acc = []

for i in k_value:
    if i % 10 != 1:
        continue
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train_full.reshape(X_train_full.shape[0], -1), y_train_full)
    k_acc.append(knn.score(X_test.reshape(X_test.shape[0], -1), y_test))
    print(f"k = {i}\t finished.")
```

```
k = 1    finished.
k = 11   finished.
k = 21   finished.
k = 31   finished.
k = 41   finished.
k = 51   finished.
k = 61   finished.
k = 71   finished.
k = 81   finished.
```

```
k = 91   finished.
k = 101  finished.
k = 111  finished.
k = 121  finished.
k = 131  finished.
k = 141  finished.
k = 151  finished.
k = 161  finished.
k = 171  finished.
k = 181  finished.
k = 191  finished.
k = 201  finished.
k = 211  finished.
k = 221  finished.
k = 231  finished.
k = 241  finished.
```

[ ]:
```
plt.figure(figsize=(2, 2))
fig, ax = plt.subplots()

ax.plot([i*10 + 1 for i in range(25)], k_acc, marker="o")
ax.set(xlabel="k", ylabel="ACC", xticks=range(1, 250, 40))
plt.show()
```

```
<Figure size 144x144 with 0 Axes>
```



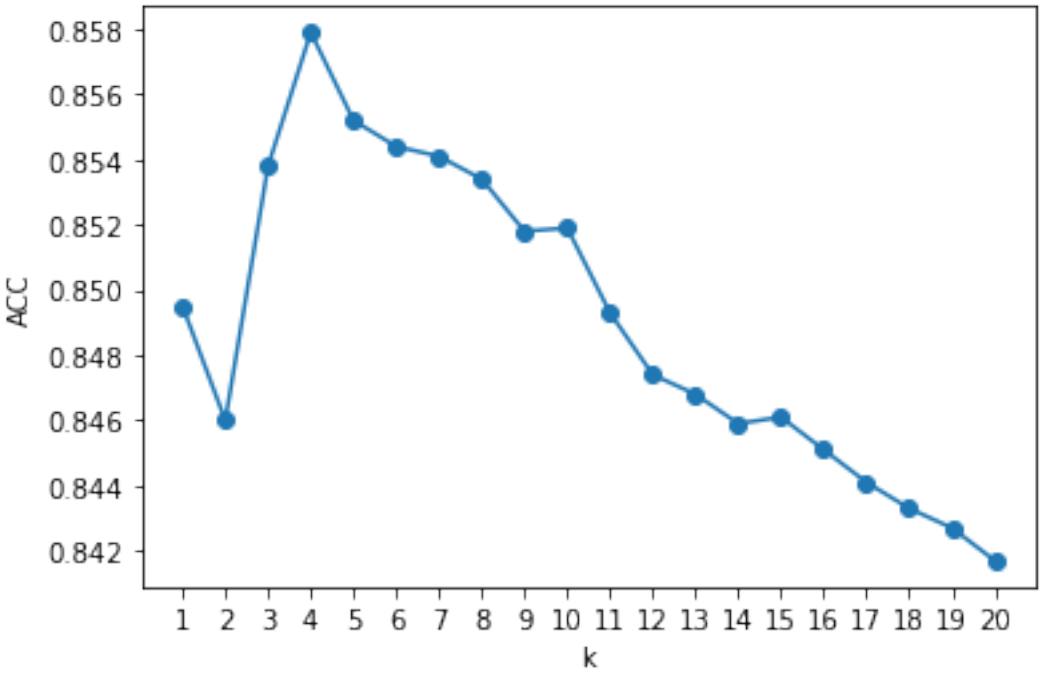
From the figure, the trend is roughly decending. Therefore, we can choose a range of k in [1, 21]

[ ]:
```
# Running in arroung 140s

k_value = [i for i in range(1,21)]
k_acc = []

for i in k_value:
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train_full.reshape(X_train_full.shape[0], -1), y_train_full)
    k_acc.append(knn.score(X_test.reshape(X_test.shape[0], -1), y_test))
    print(f"k = {i}\t finished.")
```

```
k = 1    finished.
k = 2    finished.
k = 3    finished.
k = 4    finished.
k = 5    finished.
k = 6    finished.
k = 7    finished.
k = 8    finished.
k = 9    finished.
k = 10   finished.
k = 11   finished.
k = 12   finished.
k = 13   finished.
k = 14   finished.
k = 15   finished.
k = 16   finished.
k = 17   finished.
k = 18   finished.
k = 19   finished.
k = 20   finished.
```

[ ]:
```
plt.figure(figsize=(2, 2))
fig, ax = plt.subplots()

ax.plot(k_value, k_acc, marker="o")
ax.set(xlabel="k", ylabel="ACC", xticks=k_value)
plt.show()
```

```
<Figure size 144x144 with 0 Axes>
```



From the figure above, we chose k = [3, 5, 9]. For p, we chose p [1, 2], it represents manhattan_distance and euclidean_distance respectively. We also consider the weight (uniform, distance) each point contribute to.

If we use CV with 10 folds, totally we need 3 * 2 * 2 * cv = 120 on 90% training set. Considering the running time, we use standard grid search instead.

[ ]:
```
# Setting parameters
param_grid = {
    'n_neighbors': [3, 5, 9],
    'p': [1, 2],
    "weights": ["uniform", "distance"],
    }

knn_paras = ParameterGrid(param_grid)

print(f"There are {len(list(knn_paras))} combinations.")
print("Parameter grid:\n{}".format(param_grid))

# ------------------------------------------------------------
# CV for tuning
# Setting the 10 fold stratified cross-validation

# cvKFold=StratifiedKFold(n_splits=10, shuffle=True, random_state=0)
```

```
# grid_search = GridSearchCV(KNeighborsClassifier(), param_grid, cv=cvKFold,␣
 ↪return_train_score=True, verbose=3)
# grid_search.fit(X_train_full.reshape(60000, -1), y_train_full)
```

```
There are 12 combinations.
Parameter grid:
{'n_neighbors': [3, 5, 9], 'p': [1, 2], 'weights': ['uniform', 'distance']}
```

[ ]:
```
# Running in around 410s

# Tuning KNN paras, 12 combination in total.
knn_result = get_result(
    KNeighborsClassifier(),
    knn_paras,
    X_train=X_train.reshape(X_train.shape[0], -1),
    X_valid=X_valid.reshape(X_valid.shape[0], -1)
    )
```

```
1 out of 12 finished: {'n_neighbors': 3, 'p': 1, 'weights': 'uniform'}
2 out of 12 finished: {'n_neighbors': 3, 'p': 1, 'weights': 'distance'}
3 out of 12 finished: {'n_neighbors': 3, 'p': 2, 'weights': 'uniform'}
4 out of 12 finished: {'n_neighbors': 3, 'p': 2, 'weights': 'distance'}
5 out of 12 finished: {'n_neighbors': 5, 'p': 1, 'weights': 'uniform'}
6 out of 12 finished: {'n_neighbors': 5, 'p': 1, 'weights': 'distance'}
7 out of 12 finished: {'n_neighbors': 5, 'p': 2, 'weights': 'uniform'}
8 out of 12 finished: {'n_neighbors': 5, 'p': 2, 'weights': 'distance'}
9 out of 12 finished: {'n_neighbors': 9, 'p': 1, 'weights': 'uniform'}
10 out of 12 finished: {'n_neighbors': 9, 'p': 1, 'weights': 'distance'}
11 out of 12 finished: {'n_neighbors': 9, 'p': 2, 'weights': 'uniform'}
12 out of 12 finished: {'n_neighbors': 9, 'p': 2, 'weights': 'distance'}
```

[ ]:
```
# Running in around 110s

show_results("KNN", knn_result, X_test=X_test.reshape(X_test.shape[0], -1))
```

```
Results for KNN:
Best parameters: {'n_neighbors': 3, 'p': 1, 'weights': 'distance'}
Best validation score: 0.8653
Test set score: 0.8567
```

	Score	training_time	\
{'n_neighbors': 3, 'p': 1, 'weights': 'uniform'}	0.8622	0.05	
{'n_neighbors': 3, 'p': 1, 'weights': 'distance'}	0.8653	0.05	
{'n_neighbors': 3, 'p': 2, 'weights': 'uniform'}	0.8583	0.05	
{'n_neighbors': 3, 'p': 2, 'weights': 'distance'}	0.8602	0.04	
{'n_neighbors': 5, 'p': 1, 'weights': 'uniform'}	0.8603	0.05	
{'n_neighbors': 5, 'p': 1, 'weights': 'distance'}	0.8622	0.04	
{'n_neighbors': 5, 'p': 2, 'weights': 'uniform'}	0.855	0.05	
{'n_neighbors': 5, 'p': 2, 'weights': 'distance'}	0.8577	0.05	

```
{'n_neighbors': 9, 'p': 1, 'weights': 'uniform'}   0.8628          0.05
{'n_neighbors': 9, 'p': 1, 'weights': 'distance'}  0.8645          0.04
{'n_neighbors': 9, 'p': 2, 'weights': 'uniform'}   0.8512          0.05
{'n_neighbors': 9, 'p': 2, 'weights': 'distance'}  0.8533          0.05
```

```
                                                   validation_time
{'n_neighbors': 3, 'p': 1, 'weights': 'uniform'}             40.38
{'n_neighbors': 3, 'p': 1, 'weights': 'distance'}            40.33
{'n_neighbors': 3, 'p': 2, 'weights': 'uniform'}              2.81
{'n_neighbors': 3, 'p': 2, 'weights': 'distance'}             2.68
{'n_neighbors': 5, 'p': 1, 'weights': 'uniform'}             40.39
{'n_neighbors': 5, 'p': 1, 'weights': 'distance'}            40.23
{'n_neighbors': 5, 'p': 2, 'weights': 'uniform'}              2.92
{'n_neighbors': 5, 'p': 2, 'weights': 'distance'}             2.68
{'n_neighbors': 9, 'p': 1, 'weights': 'uniform'}             40.42
{'n_neighbors': 9, 'p': 1, 'weights': 'distance'}            40.57
{'n_neighbors': 9, 'p': 2, 'weights': 'uniform'}              3.06
{'n_neighbors': 9, 'p': 2, 'weights': 'distance'}             2.89
```

1.4.2 3.2 Fully connected neural network

From section 2.2, we settled the numbers of neurons (100, 20) in hidden layers. Although the number of hidden layers as well as number of neurons are also hyperparameter, to avoid a great running time due to a number of combination of paras, we design the structure first and tune the other paras in this section.

[ ]:
```
def build_mlp(activation_function="relu"):
    """Build a Keras MLP for 10 class classification with desired parameters."""

    model = keras.models.Sequential()

    # Add the input layer
    model.add(keras.layers.Flatten(input_shape=IMAGE_SIZE))

    # Add the hidden layers with activation function
    model.add(keras.layers.Dense(100, activation=activation_function,␣
 ↪kernel_initializer=initializer))
    model.add(keras.layers.Dense(60, activation=activation_function,␣
 ↪kernel_initializer=initializer))

    # Add the output layer for 10 class classification
    model.add(keras.layers.Dense(10, activation="softmax",␣
 ↪kernel_initializer=initializer))

    return model

# Create a KerasClassifier object which works with sklearn grid searches
```

```
# We need to pass default values of arguments in build_mlp if we wish to tune␣
  ↪them
keras_classifier = KerasClassifier(build_mlp,
                                   activation_function="relu",
                                   loss="sparse_categorical_crossentropy",
                                   optimizer="sgd",
                                   optimizer__lr=0.01,
                                   metrics=["accuracy"]
                                  )

param_grid = {
    "optimizer__lr": [0.1, 0.01, 0.001],
    "activation_function": ["relu", "sigmoid", "tanh", None],
    "optimizer": ["sgd", "Adam"],
}

mlp_paras = ParameterGrid(param_grid)

print(f"There are {len(list(mlp_paras))} combinations.")
print("Parameter grid:\n{}".format(param_grid))
```

```
There are 24 combinations.
Parameter grid:
{'optimizer__lr': [0.1, 0.01, 0.001], 'activation_function': ['relu', 'sigmoid',
'tanh', None], 'optimizer': ['sgd', 'Adam']}
```

[ ]:
```
# Running in arround 1800s

mlp_result = get_result(keras_classifier, mlp_paras, epochs=30)
```

```
Epoch 1/30

c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\gradient_descent.py:111: UserWarning: The
`lr` argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)

1688/1688 [==============================] - 2s 974us/step - loss: 0.5523 -
accuracy: 0.8009
Epoch 2/30
1688/1688 [==============================] - 2s 965us/step - loss: 0.4040 -
accuracy: 0.8533
Epoch 3/30
1688/1688 [==============================] - 2s 968us/step - loss: 0.3632 -
accuracy: 0.8651
Epoch 4/30
1688/1688 [==============================] - 2s 963us/step - loss: 0.3385 -
accuracy: 0.8740
Epoch 5/30
```

```
1688/1688 [==============================] - 2s 967us/step - loss: 0.3194 -
accuracy: 0.8820
Epoch 6/30
1688/1688 [==============================] - 2s 955us/step - loss: 0.3053 -
accuracy: 0.8870
Epoch 7/30
1688/1688 [==============================] - 2s 953us/step - loss: 0.2929 -
accuracy: 0.8906
Epoch 8/30
1688/1688 [==============================] - 2s 975us/step - loss: 0.2823 -
accuracy: 0.8937
Epoch 9/30
1688/1688 [==============================] - 2s 954us/step - loss: 0.2732 -
accuracy: 0.8967
Epoch 10/30
1688/1688 [==============================] - 2s 949us/step - loss: 0.2654 -
accuracy: 0.8996
Epoch 11/30
1688/1688 [==============================] - 2s 947us/step - loss: 0.2582 -
accuracy: 0.9033
Epoch 12/30
1688/1688 [==============================] - 2s 957us/step - loss: 0.2512 -
accuracy: 0.9045
Epoch 13/30
1688/1688 [==============================] - 2s 953us/step - loss: 0.2447 -
accuracy: 0.9069
Epoch 14/30
1688/1688 [==============================] - 2s 963us/step - loss: 0.2378 -
accuracy: 0.9090
Epoch 15/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2328 -
accuracy: 0.9118
Epoch 16/30
1688/1688 [==============================] - 2s 997us/step - loss: 0.2268 -
accuracy: 0.9141
Epoch 17/30
1688/1688 [==============================] - 2s 967us/step - loss: 0.2222 -
accuracy: 0.9161
Epoch 18/30
1688/1688 [==============================] - 2s 943us/step - loss: 0.2172 -
accuracy: 0.9175
Epoch 19/30
1688/1688 [==============================] - 2s 942us/step - loss: 0.2127 -
accuracy: 0.9184
Epoch 20/30
1688/1688 [==============================] - 2s 944us/step - loss: 0.2062 -
accuracy: 0.9213
Epoch 21/30
```

```
1688/1688 [==============================] - 2s 944us/step - loss: 0.2040 -
accuracy: 0.9217
Epoch 22/30
1688/1688 [==============================] - 2s 944us/step - loss: 0.1996 -
accuracy: 0.9235
Epoch 23/30
1688/1688 [==============================] - 2s 970us/step - loss: 0.1948 -
accuracy: 0.9253
Epoch 24/30
1688/1688 [==============================] - 2s 958us/step - loss: 0.1900 -
accuracy: 0.9268
Epoch 25/30
1688/1688 [==============================] - 2s 974us/step - loss: 0.1877 -
accuracy: 0.9287
Epoch 26/30
1688/1688 [==============================] - 2s 946us/step - loss: 0.1853 -
accuracy: 0.9280
Epoch 27/30
1688/1688 [==============================] - 2s 946us/step - loss: 0.1824 -
accuracy: 0.9291
Epoch 28/30
1688/1688 [==============================] - 2s 943us/step - loss: 0.1764 -
accuracy: 0.9323
Epoch 29/30
1688/1688 [==============================] - 2s 940us/step - loss: 0.1765 -
accuracy: 0.9331
Epoch 30/30
1688/1688 [==============================] - 2s 942us/step - loss: 0.1718 -
accuracy: 0.9343
188/188 [==============================] - 0s 698us/step
1 out of 24 finished: {'activation_function': 'relu', 'optimizer': 'sgd',
'optimizer__lr': 0.1}
Epoch 1/30

c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\gradient_descent.py:111: UserWarning: The
`lr` argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)

1688/1688 [==============================] - 2s 960us/step - loss: 0.7777 -
accuracy: 0.7462
Epoch 2/30
1688/1688 [==============================] - 2s 983us/step - loss: 0.5046 -
accuracy: 0.8268
Epoch 3/30
1688/1688 [==============================] - 2s 961us/step - loss: 0.4568 -
accuracy: 0.8404
Epoch 4/30
1688/1688 [==============================] - 2s 956us/step - loss: 0.4297 -
```

```
accuracy: 0.8486
Epoch 5/30
1688/1688 [==============================] - 2s 959us/step - loss: 0.4100 -
accuracy: 0.8567
Epoch 6/30
1688/1688 [==============================] - 2s 962us/step - loss: 0.3938 -
accuracy: 0.8626
Epoch 7/30
1688/1688 [==============================] - 2s 957us/step - loss: 0.3807 -
accuracy: 0.8661
Epoch 8/30
1688/1688 [==============================] - 2s 958us/step - loss: 0.3701 -
accuracy: 0.8691
Epoch 9/30
1688/1688 [==============================] - 2s 972us/step - loss: 0.3599 -
accuracy: 0.8726
Epoch 10/30
1688/1688 [==============================] - 2s 957us/step - loss: 0.3518 -
accuracy: 0.8746
Epoch 11/30
1688/1688 [==============================] - 2s 958us/step - loss: 0.3432 -
accuracy: 0.8784
Epoch 12/30
1688/1688 [==============================] - 2s 968us/step - loss: 0.3363 -
accuracy: 0.8798
Epoch 13/30
1688/1688 [==============================] - 2s 961us/step - loss: 0.3298 -
accuracy: 0.8813
Epoch 14/30
1688/1688 [==============================] - 2s 966us/step - loss: 0.3232 -
accuracy: 0.8843
Epoch 15/30
1688/1688 [==============================] - 2s 978us/step - loss: 0.3170 -
accuracy: 0.8862
Epoch 16/30
1688/1688 [==============================] - 2s 953us/step - loss: 0.3113 -
accuracy: 0.8880
Epoch 17/30
1688/1688 [==============================] - 2s 960us/step - loss: 0.3054 -
accuracy: 0.8906
Epoch 18/30
1688/1688 [==============================] - 2s 965us/step - loss: 0.3015 -
accuracy: 0.8906
Epoch 19/30
1688/1688 [==============================] - 2s 960us/step - loss: 0.2961 -
accuracy: 0.8935
Epoch 20/30
1688/1688 [==============================] - 2s 960us/step - loss: 0.2902 -
```

```
accuracy: 0.8953
Epoch 21/30
1688/1688 [==============================] - 2s 966us/step - loss: 0.2866 -
accuracy: 0.8962
Epoch 22/30
1688/1688 [==============================] - 2s 960us/step - loss: 0.2831 -
accuracy: 0.8973
Epoch 23/30
1688/1688 [==============================] - 2s 959us/step - loss: 0.2786 -
accuracy: 0.8990
Epoch 24/30
1688/1688 [==============================] - 2s 957us/step - loss: 0.2753 -
accuracy: 0.9011
Epoch 25/30
1688/1688 [==============================] - 2s 959us/step - loss: 0.2708 -
accuracy: 0.9031
Epoch 26/30
1688/1688 [==============================] - 2s 961us/step - loss: 0.2676 -
accuracy: 0.9030
Epoch 27/30
1688/1688 [==============================] - 2s 954us/step - loss: 0.2641 -
accuracy: 0.9035
Epoch 28/30
1688/1688 [==============================] - 2s 951us/step - loss: 0.2602 -
accuracy: 0.9058
Epoch 29/30
1688/1688 [==============================] - 2s 957us/step - loss: 0.2569 -
accuracy: 0.9074
Epoch 30/30
1688/1688 [==============================] - 2s 962us/step - loss: 0.2533 -
accuracy: 0.9084
188/188 [==============================] - 0s 696us/step
2 out of 24 finished: {'activation_function': 'relu', 'optimizer': 'sgd',
'optimizer__lr': 0.01}
Epoch 1/30

c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\gradient_descent.py:111: UserWarning: The
`lr` argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)

1688/1688 [==============================] - 2s 953us/step - loss: 1.6533 -
accuracy: 0.5011
Epoch 2/30
1688/1688 [==============================] - 2s 957us/step - loss: 0.9808 -
accuracy: 0.6989
Epoch 3/30
1688/1688 [==============================] - 2s 963us/step - loss: 0.7788 -
accuracy: 0.7530
```

```
Epoch 4/30
1688/1688 [==============================] - 2s 970us/step - loss: 0.6913 -
accuracy: 0.7784
Epoch 5/30
1688/1688 [==============================] - 2s 961us/step - loss: 0.6389 -
accuracy: 0.7931
Epoch 6/30
1688/1688 [==============================] - 2s 951us/step - loss: 0.6022 -
accuracy: 0.8038
Epoch 7/30
1688/1688 [==============================] - 2s 950us/step - loss: 0.5752 -
accuracy: 0.8100
Epoch 8/30
1688/1688 [==============================] - 2s 985us/step - loss: 0.5540 -
accuracy: 0.8154
Epoch 9/30
1688/1688 [==============================] - 2s 961us/step - loss: 0.5372 -
accuracy: 0.8201
Epoch 10/30
1688/1688 [==============================] - 2s 952us/step - loss: 0.5237 -
accuracy: 0.8233
Epoch 11/30
1688/1688 [==============================] - 2s 951us/step - loss: 0.5119 -
accuracy: 0.8260
Epoch 12/30
1688/1688 [==============================] - 2s 951us/step - loss: 0.5018 -
accuracy: 0.8291
Epoch 13/30
1688/1688 [==============================] - 2s 955us/step - loss: 0.4934 -
accuracy: 0.8316
Epoch 14/30
1688/1688 [==============================] - 2s 952us/step - loss: 0.4858 -
accuracy: 0.8334
Epoch 15/30
1688/1688 [==============================] - 2s 951us/step - loss: 0.4789 -
accuracy: 0.8355
Epoch 16/30
1688/1688 [==============================] - 2s 951us/step - loss: 0.4726 -
accuracy: 0.8372
Epoch 17/30
1688/1688 [==============================] - 2s 954us/step - loss: 0.4668 -
accuracy: 0.8389
Epoch 18/30
1688/1688 [==============================] - 2s 951us/step - loss: 0.4617 -
accuracy: 0.8407
Epoch 19/30
1688/1688 [==============================] - 2s 962us/step - loss: 0.4567 -
accuracy: 0.8422
```

```
Epoch 20/30
1688/1688 [==============================] - 2s 958us/step - loss: 0.4521 -
accuracy: 0.8430
Epoch 21/30
1688/1688 [==============================] - 2s 970us/step - loss: 0.4480 -
accuracy: 0.8443
Epoch 22/30
1688/1688 [==============================] - 2s 959us/step - loss: 0.4440 -
accuracy: 0.8464
Epoch 23/30
1688/1688 [==============================] - 2s 954us/step - loss: 0.4401 -
accuracy: 0.8477
Epoch 24/30
1688/1688 [==============================] - 2s 955us/step - loss: 0.4368 -
accuracy: 0.8489
Epoch 25/30
1688/1688 [==============================] - 2s 958us/step - loss: 0.4330 -
accuracy: 0.8501
Epoch 26/30
1688/1688 [==============================] - 2s 955us/step - loss: 0.4299 -
accuracy: 0.8511
Epoch 27/30
1688/1688 [==============================] - 2s 964us/step - loss: 0.4265 -
accuracy: 0.8523
Epoch 28/30
1688/1688 [==============================] - 2s 966us/step - loss: 0.4236 -
accuracy: 0.8534
Epoch 29/30
1688/1688 [==============================] - 2s 962us/step - loss: 0.4208 -
accuracy: 0.8543
Epoch 30/30
1688/1688 [==============================] - 2s 967us/step - loss: 0.4179 -
accuracy: 0.8547
188/188 [==============================] - 0s 697us/step
3 out of 24 finished: {'activation_function': 'relu', 'optimizer': 'sgd',
'optimizer__lr': 0.001}
Epoch 1/30

c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\adam.py:114: UserWarning: The `lr`
argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)

1688/1688 [==============================] - 2s 1ms/step - loss: 1.9412 -
accuracy: 0.2261
Epoch 2/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.9550 -
accuracy: 0.1905
Epoch 3/30
```

```
1688/1688 [==============================] - 2s 1ms/step - loss: 1.9386 -
accuracy: 0.1932
Epoch 4/30
1688/1688 [==============================] - 2s 1ms/step - loss: 2.0199 -
accuracy: 0.1769
Epoch 5/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.8893 -
accuracy: 0.1932
Epoch 6/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.8357 -
accuracy: 0.1922
Epoch 7/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.8877 -
accuracy: 0.1916
Epoch 8/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.8569 -
accuracy: 0.1944
Epoch 9/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.7971 -
accuracy: 0.1955
Epoch 10/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.7549 -
accuracy: 0.1944
Epoch 11/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.7524 -
accuracy: 0.1949
Epoch 12/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.7534 -
accuracy: 0.1946
Epoch 13/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.7534 -
accuracy: 0.1991
Epoch 14/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.7526 -
accuracy: 0.1978
Epoch 15/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.7529 -
accuracy: 0.1973
Epoch 16/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.7523 -
accuracy: 0.1963
Epoch 17/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.7540 -
accuracy: 0.2010
Epoch 18/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.7521 -
accuracy: 0.1990
Epoch 19/30
```

```
1688/1688 [==============================] - 2s 1ms/step - loss: 1.7552 -
accuracy: 0.1960
Epoch 20/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.7526 -
accuracy: 0.2008
Epoch 21/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.7532 -
accuracy: 0.1969
Epoch 22/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.7537 -
accuracy: 0.1965
Epoch 23/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.7529 -
accuracy: 0.1979
Epoch 24/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.7510 -
accuracy: 0.2008
Epoch 25/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.7542 -
accuracy: 0.1971
Epoch 26/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.7529 -
accuracy: 0.1967
Epoch 27/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.7525 -
accuracy: 0.1975
Epoch 28/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.7518 -
accuracy: 0.2006
Epoch 29/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.7531 -
accuracy: 0.1986
Epoch 30/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.7521 -
accuracy: 0.1974
188/188 [==============================] - 0s 711us/step
4 out of 24 finished: {'activation_function': 'relu', 'optimizer': 'Adam',
'optimizer__lr': 0.1}
Epoch 1/30

c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\adam.py:114: UserWarning: The `lr`
argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)

1688/1688 [==============================] - 2s 1ms/step - loss: 0.5480 -
accuracy: 0.8045
Epoch 2/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4461 -
```

```
accuracy: 0.8405
Epoch 3/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4232 -
accuracy: 0.8490
Epoch 4/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4166 -
accuracy: 0.8520
Epoch 5/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4067 -
accuracy: 0.8541
Epoch 6/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3957 -
accuracy: 0.8585
Epoch 7/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3893 -
accuracy: 0.8604
Epoch 8/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3806 -
accuracy: 0.8635
Epoch 9/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3764 -
accuracy: 0.8653
Epoch 10/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3885 -
accuracy: 0.8606
Epoch 11/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3822 -
accuracy: 0.8637
Epoch 12/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3769 -
accuracy: 0.8666
Epoch 13/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3724 -
accuracy: 0.8665
Epoch 14/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3640 -
accuracy: 0.8696
Epoch 15/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3579 -
accuracy: 0.8716
Epoch 16/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3598 -
accuracy: 0.8717
Epoch 17/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3612 -
accuracy: 0.8706
Epoch 18/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3477 -
```

```
accuracy: 0.8741
Epoch 19/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3455 -
accuracy: 0.8755
Epoch 20/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3510 -
accuracy: 0.8743
Epoch 21/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3454 -
accuracy: 0.8756
Epoch 22/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3569 -
accuracy: 0.8719
Epoch 23/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3540 -
accuracy: 0.8714
Epoch 24/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3470 -
accuracy: 0.8765
Epoch 25/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3413 -
accuracy: 0.8769
Epoch 26/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3398 -
accuracy: 0.8784
Epoch 27/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3349 -
accuracy: 0.8797
Epoch 28/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3364 -
accuracy: 0.8791
Epoch 29/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3312 -
accuracy: 0.8805
Epoch 30/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3364 -
accuracy: 0.8785
188/188 [==============================] - 0s 763us/step
5 out of 24 finished: {'activation_function': 'relu', 'optimizer': 'Adam',
'optimizer__lr': 0.01}
Epoch 1/30

c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\adam.py:114: UserWarning: The `lr`
argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)

1688/1688 [==============================] - 2s 1ms/step - loss: 0.5115 -
accuracy: 0.8196
```

```
Epoch 2/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3767 -
accuracy: 0.8640
Epoch 3/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3391 -
accuracy: 0.8759
Epoch 4/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3171 -
accuracy: 0.8829
Epoch 5/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2985 -
accuracy: 0.8889
Epoch 6/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2863 -
accuracy: 0.8931
Epoch 7/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2728 -
accuracy: 0.8998
Epoch 8/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2595 -
accuracy: 0.9026
Epoch 9/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2523 -
accuracy: 0.9046
Epoch 10/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2444 -
accuracy: 0.9075
Epoch 11/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2326 -
accuracy: 0.9126
Epoch 12/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2266 -
accuracy: 0.9146
Epoch 13/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2185 -
accuracy: 0.9170
Epoch 14/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2130 -
accuracy: 0.9197
Epoch 15/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2058 -
accuracy: 0.9209
Epoch 16/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1997 -
accuracy: 0.9235
Epoch 17/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1960 -
accuracy: 0.9259
```

```
Epoch 18/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1888 -
accuracy: 0.9283
Epoch 19/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1849 -
accuracy: 0.9289
Epoch 20/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1831 -
accuracy: 0.9306
Epoch 21/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1796 -
accuracy: 0.9313
Epoch 22/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1696 -
accuracy: 0.9346
Epoch 23/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1685 -
accuracy: 0.9353
Epoch 24/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1644 -
accuracy: 0.9372
Epoch 25/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1600 -
accuracy: 0.9397
Epoch 26/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1554 -
accuracy: 0.9404
Epoch 27/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1527 -
accuracy: 0.9417
Epoch 28/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1472 -
accuracy: 0.9431
Epoch 29/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1449 -
accuracy: 0.9436
Epoch 30/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1413 -
accuracy: 0.9453
188/188 [==============================] - 0s 715us/step
6 out of 24 finished: {'activation_function': 'relu', 'optimizer': 'Adam',
'optimizer__lr': 0.001}
Epoch 1/30

c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\gradient_descent.py:111: UserWarning: The
`lr` argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)
```

```
1688/1688 [==============================] - 2s 1ms/step - loss: 0.9854 -
accuracy: 0.6590
Epoch 2/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5370 -
accuracy: 0.8100
Epoch 3/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4667 -
accuracy: 0.8337
Epoch 4/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4331 -
accuracy: 0.8441
Epoch 5/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4111 -
accuracy: 0.8536
Epoch 6/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3940 -
accuracy: 0.8597
Epoch 7/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3807 -
accuracy: 0.8630
Epoch 8/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3702 -
accuracy: 0.8664
Epoch 9/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3603 -
accuracy: 0.8699
Epoch 10/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3519 -
accuracy: 0.8719
Epoch 11/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3434 -
accuracy: 0.8769
Epoch 12/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3369 -
accuracy: 0.8778
Epoch 13/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3300 -
accuracy: 0.8795
Epoch 14/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3236 -
accuracy: 0.8824
Epoch 15/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3174 -
accuracy: 0.8844
Epoch 16/30
1688/1688 [==============================] - 2s 1000us/step - loss: 0.3119 -
accuracy: 0.8860
Epoch 17/30
```

```
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3055 -
accuracy: 0.8898
Epoch 18/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3014 -
accuracy: 0.8887
Epoch 19/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2965 -
accuracy: 0.8922
Epoch 20/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2909 -
accuracy: 0.8929
Epoch 21/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2869 -
accuracy: 0.8934
Epoch 22/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2832 -
accuracy: 0.8965
Epoch 23/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2790 -
accuracy: 0.8980
Epoch 24/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2752 -
accuracy: 0.8989
Epoch 25/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2707 -
accuracy: 0.9003
Epoch 26/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2678 -
accuracy: 0.9012
Epoch 27/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2637 -
accuracy: 0.9027
Epoch 28/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2594 -
accuracy: 0.9035
Epoch 29/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2578 -
accuracy: 0.9043
Epoch 30/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2529 -
accuracy: 0.9071
188/188 [==============================] - 0s 734us/step
7 out of 24 finished: {'activation_function': 'sigmoid', 'optimizer': 'sgd',
'optimizer__lr': 0.1}
Epoch 1/30

c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\gradient_descent.py:111: UserWarning: The
```

```
`lr` argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)
1688/1688 [==============================] - 2s 1ms/step - loss: 2.0653 -
accuracy: 0.4333
Epoch 2/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.4501 -
accuracy: 0.6122
Epoch 3/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.1183 -
accuracy: 0.6688
Epoch 4/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.9311 -
accuracy: 0.6953
Epoch 5/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.8186 -
accuracy: 0.7176
Epoch 6/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.7447 -
accuracy: 0.7360
Epoch 7/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6929 -
accuracy: 0.7507
Epoch 8/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6548 -
accuracy: 0.7615
Epoch 9/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6253 -
accuracy: 0.7721
Epoch 10/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6012 -
accuracy: 0.7828
Epoch 11/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5803 -
accuracy: 0.7923
Epoch 12/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5617 -
accuracy: 0.8006
Epoch 13/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5454 -
accuracy: 0.8066
Epoch 14/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5309 -
accuracy: 0.8126
Epoch 15/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5182 -
accuracy: 0.8170
Epoch 16/30
```

```
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5069 -
accuracy: 0.8212
Epoch 17/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4970 -
accuracy: 0.8254
Epoch 18/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4882 -
accuracy: 0.8286
Epoch 19/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4804 -
accuracy: 0.8312
Epoch 20/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4734 -
accuracy: 0.8324
Epoch 21/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4669 -
accuracy: 0.8351
Epoch 22/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4610 -
accuracy: 0.8372
Epoch 23/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4554 -
accuracy: 0.8387
Epoch 24/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4504 -
accuracy: 0.8408
Epoch 25/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4454 -
accuracy: 0.8418
Epoch 26/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4410 -
accuracy: 0.8448
Epoch 27/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4367 -
accuracy: 0.8464
Epoch 28/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4327 -
accuracy: 0.8467
Epoch 29/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4290 -
accuracy: 0.8483
Epoch 30/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4255 -
accuracy: 0.8492
188/188 [==============================] - 0s 739us/step
8 out of 24 finished: {'activation_function': 'sigmoid', 'optimizer': 'sgd',
'optimizer__lr': 0.01}
Epoch 1/30
```

```
c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\gradient_descent.py:111: UserWarning: The
`lr` argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)
1688/1688 [==============================] - 2s 994us/step - loss: 2.3311 -
accuracy: 0.1289
Epoch 2/30
1688/1688 [==============================] - 2s 1ms/step - loss: 2.2469 -
accuracy: 0.3896
Epoch 3/30
1688/1688 [==============================] - 2s 1ms/step - loss: 2.2102 -
accuracy: 0.4634
Epoch 4/30
1688/1688 [==============================] - 2s 994us/step - loss: 2.1686 -
accuracy: 0.5458
Epoch 5/30
1688/1688 [==============================] - 2s 990us/step - loss: 2.1194 -
accuracy: 0.5681
Epoch 6/30
1688/1688 [==============================] - 2s 993us/step - loss: 2.0606 -
accuracy: 0.6006
Epoch 7/30
1688/1688 [==============================] - 2s 997us/step - loss: 1.9917 -
accuracy: 0.6125
Epoch 8/30
1688/1688 [==============================] - 2s 994us/step - loss: 1.9150 -
accuracy: 0.6008
Epoch 9/30
1688/1688 [==============================] - 2s 992us/step - loss: 1.8350 -
accuracy: 0.6037
Epoch 10/30
1688/1688 [==============================] - 2s 996us/step - loss: 1.7565 -
accuracy: 0.5934
Epoch 11/30
1688/1688 [==============================] - 2s 994us/step - loss: 1.6831 -
accuracy: 0.6012
Epoch 12/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.6165 -
accuracy: 0.5999
Epoch 13/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.5566 -
accuracy: 0.6091
Epoch 14/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.5027 -
accuracy: 0.6119
Epoch 15/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.4539 -
```

```
accuracy: 0.6214
Epoch 16/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.4096 -
accuracy: 0.6267
Epoch 17/30
1688/1688 [==============================] - 2s 989us/step - loss: 1.3689 -
accuracy: 0.6340
Epoch 18/30
1688/1688 [==============================] - 2s 997us/step - loss: 1.3313 -
accuracy: 0.6452
Epoch 19/30
1688/1688 [==============================] - 2s 998us/step - loss: 1.2962 -
accuracy: 0.6489
Epoch 20/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.2632 -
accuracy: 0.6524
Epoch 21/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.2323 -
accuracy: 0.6591
Epoch 22/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.2030 -
accuracy: 0.6629
Epoch 23/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.1752 -
accuracy: 0.6652
Epoch 24/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.1488 -
accuracy: 0.6693
Epoch 25/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.1238 -
accuracy: 0.6722
Epoch 26/30
1688/1688 [==============================] - 2s 999us/step - loss: 1.1000 -
accuracy: 0.6747
Epoch 27/30
1688/1688 [==============================] - 2s 998us/step - loss: 1.0774 -
accuracy: 0.6773
Epoch 28/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.0560 -
accuracy: 0.6810
Epoch 29/30
1688/1688 [==============================] - 2s 997us/step - loss: 1.0356 -
accuracy: 0.6818
Epoch 30/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.0163 -
accuracy: 0.6852
188/188 [==============================] - 0s 744us/step
9 out of 24 finished: {'activation_function': 'sigmoid', 'optimizer': 'sgd',
```

```
'optimizer__lr': 0.001}
Epoch 1/30

c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\adam.py:114: UserWarning: The `lr`
argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)

1688/1688 [==============================] - 2s 1ms/step - loss: 1.6611 -
accuracy: 0.2850
Epoch 2/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.6164 -
accuracy: 0.2919
Epoch 3/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.5782 -
accuracy: 0.2944
Epoch 4/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.5460 -
accuracy: 0.3080
Epoch 5/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.5494 -
accuracy: 0.3295
Epoch 6/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.6137 -
accuracy: 0.3158
Epoch 7/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.5360 -
accuracy: 0.3406
Epoch 8/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.7608 -
accuracy: 0.2815
Epoch 9/30
1688/1688 [==============================] - 2s 1ms/step - loss: 2.1013 -
accuracy: 0.1883
Epoch 10/30
1688/1688 [==============================] - 2s 1ms/step - loss: 2.0346 -
accuracy: 0.1886
Epoch 11/30
1688/1688 [==============================] - 2s 1ms/step - loss: 2.0469 -
accuracy: 0.1911
Epoch 12/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.8669 -
accuracy: 0.2439
Epoch 13/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.6129 -
accuracy: 0.3397
Epoch 14/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.6031 -
accuracy: 0.3514
```

```
Epoch 15/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.5742 -
accuracy: 0.3729
Epoch 16/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.5823 -
accuracy: 0.3680
Epoch 17/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.5034 -
accuracy: 0.4039
Epoch 18/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.4689 -
accuracy: 0.4148
Epoch 19/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.5391 -
accuracy: 0.3947
Epoch 20/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.4537 -
accuracy: 0.4192
Epoch 21/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.5505 -
accuracy: 0.3939
Epoch 22/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.4749 -
accuracy: 0.4164
Epoch 23/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.4844 -
accuracy: 0.4178
Epoch 24/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.4685 -
accuracy: 0.4247
Epoch 25/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.4752 -
accuracy: 0.4231
Epoch 26/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.5007 -
accuracy: 0.4201
Epoch 27/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.4804 -
accuracy: 0.4323
Epoch 28/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.4532 -
accuracy: 0.4409
Epoch 29/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.4527 -
accuracy: 0.4308
Epoch 30/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.4818 -
accuracy: 0.4280
```

```
188/188 [==============================] - 0s 758us/step
10 out of 24 finished: {'activation_function': 'sigmoid', 'optimizer': 'Adam',
'optimizer__lr': 0.1}
Epoch 1/30

c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\adam.py:114: UserWarning: The `lr`
argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)

1688/1688 [==============================] - 2s 1ms/step - loss: 0.5692 -
accuracy: 0.7917
Epoch 2/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4766 -
accuracy: 0.8277
Epoch 3/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4439 -
accuracy: 0.8389
Epoch 4/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4365 -
accuracy: 0.8410
Epoch 5/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4318 -
accuracy: 0.8442
Epoch 6/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4265 -
accuracy: 0.8456
Epoch 7/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4108 -
accuracy: 0.8529
Epoch 8/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4088 -
accuracy: 0.8534
Epoch 9/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4006 -
accuracy: 0.8553
Epoch 10/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3939 -
accuracy: 0.8552
Epoch 11/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3994 -
accuracy: 0.8549
Epoch 12/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3880 -
accuracy: 0.8599
Epoch 13/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3838 -
accuracy: 0.8606
Epoch 14/30
```

```
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3784 -
accuracy: 0.8630
Epoch 15/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3842 -
accuracy: 0.8603
Epoch 16/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3757 -
accuracy: 0.8646
Epoch 17/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3822 -
accuracy: 0.8637
Epoch 18/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3779 -
accuracy: 0.8635
Epoch 19/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3759 -
accuracy: 0.8659
Epoch 20/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3760 -
accuracy: 0.8647
Epoch 21/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3698 -
accuracy: 0.8671
Epoch 22/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3671 -
accuracy: 0.8664
Epoch 23/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3708 -
accuracy: 0.8669
Epoch 24/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3694 -
accuracy: 0.8668
Epoch 25/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3734 -
accuracy: 0.8648
Epoch 26/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3627 -
accuracy: 0.8679
Epoch 27/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3743 -
accuracy: 0.8650
Epoch 28/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3674 -
accuracy: 0.8667
Epoch 29/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3644 -
accuracy: 0.8694
Epoch 30/30
```

```
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3713 -
accuracy: 0.8659
188/188 [==============================] - 0s 741us/step
11 out of 24 finished: {'activation_function': 'sigmoid', 'optimizer': 'Adam',
'optimizer__lr': 0.01}
Epoch 1/30

c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\adam.py:114: UserWarning: The `lr`
argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)

1688/1688 [==============================] - 2s 1ms/step - loss: 0.6677 -
accuracy: 0.7802
Epoch 2/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4039 -
accuracy: 0.8548
Epoch 3/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3615 -
accuracy: 0.8694
Epoch 4/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3375 -
accuracy: 0.8782
Epoch 5/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3197 -
accuracy: 0.8845
Epoch 6/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3047 -
accuracy: 0.8879
Epoch 7/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2923 -
accuracy: 0.8935
Epoch 8/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2793 -
accuracy: 0.8969
Epoch 9/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2704 -
accuracy: 0.9001
Epoch 10/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2610 -
accuracy: 0.9039
Epoch 11/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2520 -
accuracy: 0.9064
Epoch 12/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2450 -
accuracy: 0.9090
Epoch 13/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2369 -
```

```
accuracy: 0.9118
Epoch 14/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2307 -
accuracy: 0.9137
Epoch 15/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2231 -
accuracy: 0.9170
Epoch 16/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2175 -
accuracy: 0.9193
Epoch 17/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2115 -
accuracy: 0.9206
Epoch 18/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2037 -
accuracy: 0.9248
Epoch 19/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1993 -
accuracy: 0.9261
Epoch 20/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1959 -
accuracy: 0.9275
Epoch 21/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1897 -
accuracy: 0.9289
Epoch 22/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1858 -
accuracy: 0.9311
Epoch 23/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1797 -
accuracy: 0.9334
Epoch 24/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1761 -
accuracy: 0.9343
Epoch 25/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1724 -
accuracy: 0.9366
Epoch 26/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1672 -
accuracy: 0.9383
Epoch 27/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1628 -
accuracy: 0.9395
Epoch 28/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1588 -
accuracy: 0.9416
Epoch 29/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1547 -
```

```
accuracy: 0.9429
Epoch 30/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1531 -
accuracy: 0.9430
188/188 [==============================] - 0s 750us/step
12 out of 24 finished: {'activation_function': 'sigmoid', 'optimizer': 'Adam',
'optimizer__lr': 0.001}
Epoch 1/30

c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\gradient_descent.py:111: UserWarning: The
`lr` argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)

1688/1688 [==============================] - 2s 976us/step - loss: 0.5242 -
accuracy: 0.8104
Epoch 2/30
1688/1688 [==============================] - 2s 983us/step - loss: 0.4022 -
accuracy: 0.8542
Epoch 3/30
1688/1688 [==============================] - 2s 976us/step - loss: 0.3634 -
accuracy: 0.8670
Epoch 4/30
1688/1688 [==============================] - 2s 984us/step - loss: 0.3389 -
accuracy: 0.8759
Epoch 5/30
1688/1688 [==============================] - 2s 981us/step - loss: 0.3202 -
accuracy: 0.8826
Epoch 6/30
1688/1688 [==============================] - 2s 985us/step - loss: 0.3045 -
accuracy: 0.8871
Epoch 7/30
1688/1688 [==============================] - 2s 989us/step - loss: 0.2912 -
accuracy: 0.8921
Epoch 8/30
1688/1688 [==============================] - 2s 972us/step - loss: 0.2820 -
accuracy: 0.8958
Epoch 9/30
1688/1688 [==============================] - 2s 964us/step - loss: 0.2721 -
accuracy: 0.8979
Epoch 10/30
1688/1688 [==============================] - 2s 979us/step - loss: 0.2639 -
accuracy: 0.9009
Epoch 11/30
1688/1688 [==============================] - 2s 975us/step - loss: 0.2566 -
accuracy: 0.9040
Epoch 12/30
1688/1688 [==============================] - 2s 980us/step - loss: 0.2492 -
accuracy: 0.9077
```

```
Epoch 13/30
1688/1688 [==============================] - 2s 972us/step - loss: 0.2417 -
accuracy: 0.9093
Epoch 14/30
1688/1688 [==============================] - 2s 976us/step - loss: 0.2352 -
accuracy: 0.9120
Epoch 15/30
1688/1688 [==============================] - 2s 983us/step - loss: 0.2286 -
accuracy: 0.9143
Epoch 16/30
1688/1688 [==============================] - 2s 972us/step - loss: 0.2236 -
accuracy: 0.9167
Epoch 17/30
1688/1688 [==============================] - 2s 987us/step - loss: 0.2169 -
accuracy: 0.9195
Epoch 18/30
1688/1688 [==============================] - 2s 978us/step - loss: 0.2124 -
accuracy: 0.9198
Epoch 19/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2067 -
accuracy: 0.9219
Epoch 20/30
1688/1688 [==============================] - 2s 981us/step - loss: 0.2011 -
accuracy: 0.9235
Epoch 21/30
1688/1688 [==============================] - 2s 975us/step - loss: 0.1967 -
accuracy: 0.9265
Epoch 22/30
1688/1688 [==============================] - 2s 974us/step - loss: 0.1943 -
accuracy: 0.9265
Epoch 23/30
1688/1688 [==============================] - 2s 974us/step - loss: 0.1868 -
accuracy: 0.9306
Epoch 24/30
1688/1688 [==============================] - 2s 973us/step - loss: 0.1837 -
accuracy: 0.9321
Epoch 25/30
1688/1688 [==============================] - 2s 986us/step - loss: 0.1792 -
accuracy: 0.9334
Epoch 26/30
1688/1688 [==============================] - 2s 976us/step - loss: 0.1773 -
accuracy: 0.9332
Epoch 27/30
1688/1688 [==============================] - 2s 977us/step - loss: 0.1731 -
accuracy: 0.9351
Epoch 28/30
1688/1688 [==============================] - 2s 973us/step - loss: 0.1688 -
accuracy: 0.9366
```

```
Epoch 29/30
1688/1688 [==============================] - 2s 987us/step - loss: 0.1660 -
accuracy: 0.9361
Epoch 30/30
1688/1688 [==============================] - 2s 980us/step - loss: 0.1613 -
accuracy: 0.9403
188/188 [==============================] - 0s 720us/step
13 out of 24 finished: {'activation_function': 'tanh', 'optimizer': 'sgd',
'optimizer__lr': 0.1}
Epoch 1/30

c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\gradient_descent.py:111: UserWarning: The
`lr` argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)

1688/1688 [==============================] - 2s 983us/step - loss: 0.7262 -
accuracy: 0.7684
Epoch 2/30
1688/1688 [==============================] - 2s 985us/step - loss: 0.4912 -
accuracy: 0.8301
Epoch 3/30
1688/1688 [==============================] - 2s 982us/step - loss: 0.4465 -
accuracy: 0.8421
Epoch 4/30
1688/1688 [==============================] - 2s 986us/step - loss: 0.4212 -
accuracy: 0.8510
Epoch 5/30
1688/1688 [==============================] - 2s 986us/step - loss: 0.4032 -
accuracy: 0.8583
Epoch 6/30
1688/1688 [==============================] - 2s 985us/step - loss: 0.3888 -
accuracy: 0.8631
Epoch 7/30
1688/1688 [==============================] - 2s 985us/step - loss: 0.3774 -
accuracy: 0.8667
Epoch 8/30
1688/1688 [==============================] - 2s 981us/step - loss: 0.3676 -
accuracy: 0.8690
Epoch 9/30
1688/1688 [==============================] - 2s 985us/step - loss: 0.3593 -
accuracy: 0.8717
Epoch 10/30
1688/1688 [==============================] - 2s 984us/step - loss: 0.3524 -
accuracy: 0.8732
Epoch 11/30
1688/1688 [==============================] - 2s 983us/step - loss: 0.3446 -
accuracy: 0.8777
Epoch 12/30
```

```
1688/1688 [==============================] - 2s 981us/step - loss: 0.3389 -
accuracy: 0.8787
Epoch 13/30
1688/1688 [==============================] - 2s 984us/step - loss: 0.3332 -
accuracy: 0.8797
Epoch 14/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3275 -
accuracy: 0.8830
Epoch 15/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3224 -
accuracy: 0.8849
Epoch 16/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3176 -
accuracy: 0.8848
Epoch 17/30
1688/1688 [==============================] - 2s 983us/step - loss: 0.3125 -
accuracy: 0.8881
Epoch 18/30
1688/1688 [==============================] - 2s 984us/step - loss: 0.3087 -
accuracy: 0.8883
Epoch 19/30
1688/1688 [==============================] - 2s 984us/step - loss: 0.3045 -
accuracy: 0.8906
Epoch 20/30
1688/1688 [==============================] - 2s 982us/step - loss: 0.3000 -
accuracy: 0.8918
Epoch 21/30
1688/1688 [==============================] - 2s 981us/step - loss: 0.2965 -
accuracy: 0.8929
Epoch 22/30
1688/1688 [==============================] - 2s 981us/step - loss: 0.2929 -
accuracy: 0.8944
Epoch 23/30
1688/1688 [==============================] - 2s 984us/step - loss: 0.2895 -
accuracy: 0.8952
Epoch 24/30
1688/1688 [==============================] - 2s 982us/step - loss: 0.2862 -
accuracy: 0.8958
Epoch 25/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2825 -
accuracy: 0.8982
Epoch 26/30
1688/1688 [==============================] - 2s 981us/step - loss: 0.2797 -
accuracy: 0.8985
Epoch 27/30
1688/1688 [==============================] - 2s 987us/step - loss: 0.2761 -
accuracy: 0.8996
Epoch 28/30
```

```
1688/1688 [==============================] - 2s 988us/step - loss: 0.2733 -
accuracy: 0.9002
Epoch 29/30
1688/1688 [==============================] - 2s 983us/step - loss: 0.2705 -
accuracy: 0.9011
Epoch 30/30
1688/1688 [==============================] - 2s 981us/step - loss: 0.2675 -
accuracy: 0.9025
188/188 [==============================] - 0s 723us/step
14 out of 24 finished: {'activation_function': 'tanh', 'optimizer': 'sgd',
'optimizer__lr': 0.01}
Epoch 1/30

c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\gradient_descent.py:111: UserWarning: The
`lr` argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)

1688/1688 [==============================] - 2s 973us/step - loss: 1.3950 -
accuracy: 0.6037
Epoch 2/30
1688/1688 [==============================] - 2s 974us/step - loss: 0.8948 -
accuracy: 0.7302
Epoch 3/30
1688/1688 [==============================] - 2s 970us/step - loss: 0.7577 -
accuracy: 0.7646
Epoch 4/30
1688/1688 [==============================] - 2s 976us/step - loss: 0.6848 -
accuracy: 0.7839
Epoch 5/30
1688/1688 [==============================] - 2s 990us/step - loss: 0.6372 -
accuracy: 0.7955
Epoch 6/30
1688/1688 [==============================] - 2s 975us/step - loss: 0.6028 -
accuracy: 0.8040
Epoch 7/30
1688/1688 [==============================] - 2s 981us/step - loss: 0.5769 -
accuracy: 0.8100
Epoch 8/30
1688/1688 [==============================] - 2s 977us/step - loss: 0.5562 -
accuracy: 0.8153
Epoch 9/30
1688/1688 [==============================] - 2s 976us/step - loss: 0.5394 -
accuracy: 0.8190
Epoch 10/30
1688/1688 [==============================] - 2s 985us/step - loss: 0.5257 -
accuracy: 0.8216
Epoch 11/30
1688/1688 [==============================] - 2s 980us/step - loss: 0.5140 -
```

```
accuracy: 0.8249
Epoch 12/30
1688/1688 [==============================] - 2s 974us/step - loss: 0.5038 -
accuracy: 0.8273
Epoch 13/30
1688/1688 [==============================] - 2s 977us/step - loss: 0.4952 -
accuracy: 0.8290
Epoch 14/30
1688/1688 [==============================] - 2s 976us/step - loss: 0.4876 -
accuracy: 0.8322
Epoch 15/30
1688/1688 [==============================] - 2s 978us/step - loss: 0.4807 -
accuracy: 0.8336
Epoch 16/30
1688/1688 [==============================] - 2s 978us/step - loss: 0.4744 -
accuracy: 0.8355
Epoch 17/30
1688/1688 [==============================] - 2s 983us/step - loss: 0.4687 -
accuracy: 0.8368
Epoch 18/30
1688/1688 [==============================] - 2s 984us/step - loss: 0.4635 -
accuracy: 0.8387
Epoch 19/30
1688/1688 [==============================] - 2s 982us/step - loss: 0.4587 -
accuracy: 0.8406
Epoch 20/30
1688/1688 [==============================] - 2s 989us/step - loss: 0.4543 -
accuracy: 0.8410
Epoch 21/30
1688/1688 [==============================] - 2s 984us/step - loss: 0.4500 -
accuracy: 0.8426
Epoch 22/30
1688/1688 [==============================] - 2s 989us/step - loss: 0.4461 -
accuracy: 0.8431
Epoch 23/30
1688/1688 [==============================] - 2s 998us/step - loss: 0.4424 -
accuracy: 0.8457
Epoch 24/30
1688/1688 [==============================] - 2s 993us/step - loss: 0.4390 -
accuracy: 0.8461
Epoch 25/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4354 -
accuracy: 0.8473
Epoch 26/30
1688/1688 [==============================] - 2s 994us/step - loss: 0.4323 -
accuracy: 0.8484
Epoch 27/30
1688/1688 [==============================] - 2s 993us/step - loss: 0.4292 -
```

```
accuracy: 0.8498
Epoch 28/30
1688/1688 [==============================] - 2s 987us/step - loss: 0.4263 -
accuracy: 0.8500
Epoch 29/30
1688/1688 [==============================] - 2s 989us/step - loss: 0.4235 -
accuracy: 0.8516
Epoch 30/30
1688/1688 [==============================] - 2s 977us/step - loss: 0.4210 -
accuracy: 0.8514
188/188 [==============================] - 0s 709us/step
15 out of 24 finished: {'activation_function': 'tanh', 'optimizer': 'sgd',
'optimizer__lr': 0.001}
Epoch 1/30

c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\adam.py:114: UserWarning: The `lr`
argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)

1688/1688 [==============================] - 2s 1ms/step - loss: 2.8471 -
accuracy: 0.1004
Epoch 2/30
1688/1688 [==============================] - 2s 1ms/step - loss: 2.8309 -
accuracy: 0.1002
Epoch 3/30
1688/1688 [==============================] - 2s 1ms/step - loss: 2.8583 -
accuracy: 0.1000
Epoch 4/30
1688/1688 [==============================] - 2s 1ms/step - loss: 2.8601 -
accuracy: 0.0985
Epoch 5/30
1688/1688 [==============================] - 2s 1ms/step - loss: 2.8354 -
accuracy: 0.1016
Epoch 6/30
1688/1688 [==============================] - 2s 1ms/step - loss: 2.8326 -
accuracy: 0.1012
Epoch 7/30
1688/1688 [==============================] - 2s 1ms/step - loss: 2.8048 -
accuracy: 0.1018
Epoch 8/30
1688/1688 [==============================] - 2s 1ms/step - loss: 2.8371 -
accuracy: 0.0993
Epoch 9/30
1688/1688 [==============================] - 2s 1ms/step - loss: 2.8450 -
accuracy: 0.1003
Epoch 10/30
1688/1688 [==============================] - 2s 1ms/step - loss: 2.8541 -
accuracy: 0.0982
```

```
Epoch 11/30
1688/1688 [==============================] - 2s 1ms/step - loss: 2.8407 -
accuracy: 0.1018
Epoch 12/30
1688/1688 [==============================] - 2s 1ms/step - loss: 2.8154 -
accuracy: 0.1008
Epoch 13/30
1688/1688 [==============================] - 2s 1ms/step - loss: 2.8465 -
accuracy: 0.1002
Epoch 14/30
1688/1688 [==============================] - 2s 1ms/step - loss: 2.8406 -
accuracy: 0.1006
Epoch 15/30
1688/1688 [==============================] - 2s 1ms/step - loss: 2.8792 -
accuracy: 0.0994
Epoch 16/30
1688/1688 [==============================] - 2s 1ms/step - loss: 2.8425 -
accuracy: 0.1004
Epoch 17/30
1688/1688 [==============================] - 2s 1ms/step - loss: 2.8374 -
accuracy: 0.0991
Epoch 18/30
1688/1688 [==============================] - 2s 1ms/step - loss: 2.8663 -
accuracy: 0.0981
Epoch 19/30
1688/1688 [==============================] - 2s 1ms/step - loss: 2.8442 -
accuracy: 0.0996
Epoch 20/30
1688/1688 [==============================] - 2s 1ms/step - loss: 2.8399 -
accuracy: 0.0998
Epoch 21/30
1688/1688 [==============================] - 2s 1ms/step - loss: 2.8872 -
accuracy: 0.0993
Epoch 22/30
1688/1688 [==============================] - 2s 1ms/step - loss: 2.8069 -
accuracy: 0.1004
Epoch 23/30
1688/1688 [==============================] - 2s 1ms/step - loss: 2.8597 -
accuracy: 0.0996
Epoch 24/30
1688/1688 [==============================] - 2s 1ms/step - loss: 2.8370 -
accuracy: 0.0980
Epoch 25/30
1688/1688 [==============================] - 2s 1ms/step - loss: 2.8713 -
accuracy: 0.1023
Epoch 26/30
1688/1688 [==============================] - 2s 1ms/step - loss: 2.8610 -
accuracy: 0.1001
```

```
Epoch 27/30
1688/1688 [==============================] - 2s 1ms/step - loss: 2.8411 -
accuracy: 0.0986
Epoch 28/30
1688/1688 [==============================] - 2s 1ms/step - loss: 2.8385 -
accuracy: 0.0982
Epoch 29/30
1688/1688 [==============================] - 2s 1ms/step - loss: 2.8479 -
accuracy: 0.0997
Epoch 30/30
1688/1688 [==============================] - 2s 1ms/step - loss: 2.8448 -
accuracy: 0.1005
188/188 [==============================] - 0s 782us/step
16 out of 24 finished: {'activation_function': 'tanh', 'optimizer': 'Adam',
'optimizer__lr': 0.1}
Epoch 1/30

c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\adam.py:114: UserWarning: The `lr`
argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)

1688/1688 [==============================] - 2s 1ms/step - loss: 0.6807 -
accuracy: 0.7537
Epoch 2/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6639 -
accuracy: 0.7633
Epoch 3/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6622 -
accuracy: 0.7596
Epoch 4/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6841 -
accuracy: 0.7497
Epoch 5/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6846 -
accuracy: 0.7524
Epoch 6/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6729 -
accuracy: 0.7712
Epoch 7/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.7334 -
accuracy: 0.7183
Epoch 8/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6781 -
accuracy: 0.7455
Epoch 9/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6698 -
accuracy: 0.7530
Epoch 10/30
```

```
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6718 -
accuracy: 0.7595
Epoch 11/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6580 -
accuracy: 0.7683
Epoch 12/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6510 -
accuracy: 0.7666
Epoch 13/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6532 -
accuracy: 0.7651
Epoch 14/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6654 -
accuracy: 0.7623
Epoch 15/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6647 -
accuracy: 0.7625
Epoch 16/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6374 -
accuracy: 0.7735
Epoch 17/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6382 -
accuracy: 0.7731
Epoch 18/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6738 -
accuracy: 0.7502
Epoch 19/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6679 -
accuracy: 0.7573
Epoch 20/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6393 -
accuracy: 0.7760
Epoch 21/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6284 -
accuracy: 0.7768
Epoch 22/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6436 -
accuracy: 0.7764
Epoch 23/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6380 -
accuracy: 0.7764
Epoch 24/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6272 -
accuracy: 0.7832
Epoch 25/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6444 -
accuracy: 0.7638
Epoch 26/30
```

```
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6753 -
accuracy: 0.7539
Epoch 27/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6846 -
accuracy: 0.7473
Epoch 28/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6539 -
accuracy: 0.7651
Epoch 29/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6943 -
accuracy: 0.7530
Epoch 30/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6696 -
accuracy: 0.7642
188/188 [==============================] - 0s 768us/step
17 out of 24 finished: {'activation_function': 'tanh', 'optimizer': 'Adam',
'optimizer__lr': 0.01}
Epoch 1/30

c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\adam.py:114: UserWarning: The `lr`
argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)

1688/1688 [==============================] - 2s 1ms/step - loss: 0.4843 -
accuracy: 0.8262
Epoch 2/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3712 -
accuracy: 0.8659
Epoch 3/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3352 -
accuracy: 0.8773
Epoch 4/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3142 -
accuracy: 0.8840
Epoch 5/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2957 -
accuracy: 0.8907
Epoch 6/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2834 -
accuracy: 0.8943
Epoch 7/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2723 -
accuracy: 0.8994
Epoch 8/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2596 -
accuracy: 0.9034
Epoch 9/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2502 -
```

```
accuracy: 0.9055
Epoch 10/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2447 -
accuracy: 0.9082
Epoch 11/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2366 -
accuracy: 0.9115
Epoch 12/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2310 -
accuracy: 0.9133
Epoch 13/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2226 -
accuracy: 0.9159
Epoch 14/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2189 -
accuracy: 0.9175
Epoch 15/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2099 -
accuracy: 0.9211
Epoch 16/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2066 -
accuracy: 0.9221
Epoch 17/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2027 -
accuracy: 0.9238
Epoch 18/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1975 -
accuracy: 0.9244
Epoch 19/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1907 -
accuracy: 0.9288
Epoch 20/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1877 -
accuracy: 0.9291
Epoch 21/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1851 -
accuracy: 0.9309
Epoch 22/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1799 -
accuracy: 0.9330
Epoch 23/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1776 -
accuracy: 0.9334
Epoch 24/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1743 -
accuracy: 0.9334
Epoch 25/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1701 -
```

```
accuracy: 0.9351
Epoch 26/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1649 -
accuracy: 0.9381
Epoch 27/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1664 -
accuracy: 0.9367
Epoch 28/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1604 -
accuracy: 0.9391
Epoch 29/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1589 -
accuracy: 0.9396
Epoch 30/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1561 -
accuracy: 0.9415
188/188 [==============================] - 0s 759us/step
18 out of 24 finished: {'activation_function': 'tanh', 'optimizer': 'Adam',
'optimizer__lr': 0.001}
Epoch 1/30

c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\gradient_descent.py:111: UserWarning: The
`lr` argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)

1688/1688 [==============================] - 2s 1ms/step - loss: 0.6285 -
accuracy: 0.7890
Epoch 2/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4883 -
accuracy: 0.8287
Epoch 3/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4665 -
accuracy: 0.8383
Epoch 4/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4565 -
accuracy: 0.8393
Epoch 5/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4475 -
accuracy: 0.8433
Epoch 6/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4412 -
accuracy: 0.8469
Epoch 7/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4356 -
accuracy: 0.8478
Epoch 8/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4341 -
accuracy: 0.8479
```

```
Epoch 9/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4287 -
accuracy: 0.8503
Epoch 10/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4274 -
accuracy: 0.8506
Epoch 11/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4256 -
accuracy: 0.8507
Epoch 12/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4238 -
accuracy: 0.8525
Epoch 13/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4217 -
accuracy: 0.8522
Epoch 14/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4209 -
accuracy: 0.8518
Epoch 15/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4187 -
accuracy: 0.8534
Epoch 16/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4173 -
accuracy: 0.8540
Epoch 17/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4144 -
accuracy: 0.8540
Epoch 18/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4137 -
accuracy: 0.8529
Epoch 19/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4134 -
accuracy: 0.8550
Epoch 20/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4103 -
accuracy: 0.8560
Epoch 21/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4115 -
accuracy: 0.8558
Epoch 22/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4092 -
accuracy: 0.8565
Epoch 23/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4088 -
accuracy: 0.8556
Epoch 24/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4081 -
accuracy: 0.8572
```

```
Epoch 25/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4066 -
accuracy: 0.8577
Epoch 26/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4067 -
accuracy: 0.8567
Epoch 27/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4074 -
accuracy: 0.8554
Epoch 28/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4051 -
accuracy: 0.8567
Epoch 29/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4051 -
accuracy: 0.8571
Epoch 30/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4054 -
accuracy: 0.8575
188/188 [==============================] - 0s 749us/step
19 out of 24 finished: {'activation_function': None, 'optimizer': 'sgd',
'optimizer__lr': 0.1}
Epoch 1/30

c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\gradient_descent.py:111: UserWarning: The
`lr` argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)

1688/1688 [==============================] - 2s 1ms/step - loss: 0.6610 -
accuracy: 0.7794
Epoch 2/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4938 -
accuracy: 0.8293
Epoch 3/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4627 -
accuracy: 0.8385
Epoch 4/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4478 -
accuracy: 0.8437
Epoch 5/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4373 -
accuracy: 0.8485
Epoch 6/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4296 -
accuracy: 0.8525
Epoch 7/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4238 -
accuracy: 0.8530
Epoch 8/30
```

```
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4195 -
accuracy: 0.8547
Epoch 9/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4151 -
accuracy: 0.8555
Epoch 10/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4121 -
accuracy: 0.8566
Epoch 11/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4094 -
accuracy: 0.8581
Epoch 12/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4065 -
accuracy: 0.8595
Epoch 13/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4046 -
accuracy: 0.8589
Epoch 14/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4019 -
accuracy: 0.8602
Epoch 15/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4000 -
accuracy: 0.8606
Epoch 16/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3985 -
accuracy: 0.8611
Epoch 17/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3958 -
accuracy: 0.8621
Epoch 18/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3953 -
accuracy: 0.8607
Epoch 19/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3929 -
accuracy: 0.8628
Epoch 20/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3908 -
accuracy: 0.8627
Epoch 21/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3906 -
accuracy: 0.8626
Epoch 22/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3896 -
accuracy: 0.8632
Epoch 23/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3883 -
accuracy: 0.8642
Epoch 24/30
```

```
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3872 -
accuracy: 0.8651
Epoch 25/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3858 -
accuracy: 0.8656
Epoch 26/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3855 -
accuracy: 0.8650
Epoch 27/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3843 -
accuracy: 0.8648
Epoch 28/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3828 -
accuracy: 0.8651
Epoch 29/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3825 -
accuracy: 0.8657
Epoch 30/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3815 -
accuracy: 0.8658
188/188 [==============================] - 0s 763us/step
20 out of 24 finished: {'activation_function': None, 'optimizer': 'sgd',
'optimizer__lr': 0.01}
Epoch 1/30

c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\gradient_descent.py:111: UserWarning: The
`lr` argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)

1688/1688 [==============================] - 2s 1ms/step - loss: 1.1842 -
accuracy: 0.6423
Epoch 2/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.7551 -
accuracy: 0.7579
Epoch 3/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6598 -
accuracy: 0.7864
Epoch 4/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6095 -
accuracy: 0.7996
Epoch 5/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5773 -
accuracy: 0.8082
Epoch 6/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5540 -
accuracy: 0.8153
Epoch 7/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5369 -
```

```
accuracy: 0.8185
Epoch 8/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5231 -
accuracy: 0.8222
Epoch 9/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5120 -
accuracy: 0.8264
Epoch 10/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5030 -
accuracy: 0.8291
Epoch 11/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4950 -
accuracy: 0.8313
Epoch 12/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4879 -
accuracy: 0.8331
Epoch 13/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4821 -
accuracy: 0.8339
Epoch 14/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4768 -
accuracy: 0.8369
Epoch 15/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4721 -
accuracy: 0.8376
Epoch 16/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4676 -
accuracy: 0.8384
Epoch 17/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4636 -
accuracy: 0.8404
Epoch 18/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4600 -
accuracy: 0.8411
Epoch 19/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4566 -
accuracy: 0.8429
Epoch 20/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4535 -
accuracy: 0.8436
Epoch 21/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4506 -
accuracy: 0.8446
Epoch 22/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4480 -
accuracy: 0.8453
Epoch 23/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4455 -
```

```
accuracy: 0.8465
Epoch 24/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4434 -
accuracy: 0.8470
Epoch 25/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4408 -
accuracy: 0.8482
Epoch 26/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4389 -
accuracy: 0.8489
Epoch 27/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4368 -
accuracy: 0.8492
Epoch 28/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4350 -
accuracy: 0.8497
Epoch 29/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4333 -
accuracy: 0.8507
Epoch 30/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4316 -
accuracy: 0.8507
188/188 [==============================] - 0s 759us/step
21 out of 24 finished: {'activation_function': None, 'optimizer': 'sgd',
'optimizer__lr': 0.001}
Epoch 1/30

c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\adam.py:114: UserWarning: The `lr`
argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)

1688/1688 [==============================] - 2s 1ms/step - loss: 11.5627 -
accuracy: 0.7412
Epoch 2/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1019.8622 -
accuracy: 0.7256
Epoch 3/30
1688/1688 [==============================] - 2s 1ms/step - loss: 32.1146 -
accuracy: 0.7461
Epoch 4/30
1688/1688 [==============================] - 2s 1ms/step - loss: 15.4382 -
accuracy: 0.7470
Epoch 5/30
1688/1688 [==============================] - 2s 1ms/step - loss: 914.6314 -
accuracy: 0.7195
Epoch 6/30
1688/1688 [==============================] - 2s 1ms/step - loss: 52.7696 -
accuracy: 0.7643
```

```
Epoch 7/30
1688/1688 [==============================] - 2s 1ms/step - loss: 28.1203 -
accuracy: 0.7514
Epoch 8/30
1688/1688 [==============================] - 2s 1ms/step - loss: 833.8857 -
accuracy: 0.7334
Epoch 9/30
1688/1688 [==============================] - 2s 1ms/step - loss: 52.6891 -
accuracy: 0.7636
Epoch 10/30
1688/1688 [==============================] - 2s 1ms/step - loss: 38.0421 -
accuracy: 0.7512
Epoch 11/30
1688/1688 [==============================] - 2s 1ms/step - loss: 934.8671 -
accuracy: 0.7513
Epoch 12/30
1688/1688 [==============================] - 2s 1ms/step - loss: 41.6756 -
accuracy: 0.7646
Epoch 13/30
1688/1688 [==============================] - 2s 1ms/step - loss: 716.1636 -
accuracy: 0.7335
Epoch 14/30
1688/1688 [==============================] - 2s 1ms/step - loss: 213.9136 -
accuracy: 0.7755
Epoch 15/30
1688/1688 [==============================] - 2s 1ms/step - loss: 56.5122 -
accuracy: 0.7596
Epoch 16/30
1688/1688 [==============================] - 2s 1ms/step - loss: 735.7065 -
accuracy: 0.7509
Epoch 17/30
1688/1688 [==============================] - 2s 1ms/step - loss: 60.1095 -
accuracy: 0.7732
Epoch 18/30
1688/1688 [==============================] - 2s 1ms/step - loss: 610.9733 -
accuracy: 0.7379
Epoch 19/30
1688/1688 [==============================] - 2s 1ms/step - loss: 318.1247 -
accuracy: 0.7743
Epoch 20/30
1688/1688 [==============================] - 2s 1ms/step - loss: 59.1774 -
accuracy: 0.7636
Epoch 21/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1068.1047 -
accuracy: 0.7574
Epoch 22/30
1688/1688 [==============================] - 2s 1ms/step - loss: 63.0517 -
accuracy: 0.7805
```

```
Epoch 23/30
1688/1688 [==============================] - 2s 1ms/step - loss: 109.9158 -
accuracy: 0.7563
Epoch 24/30
1688/1688 [==============================] - 2s 1ms/step - loss: 353.7023 -
accuracy: 0.7634
Epoch 25/30
1688/1688 [==============================] - 2s 1ms/step - loss: 576.0347 -
accuracy: 0.7564
Epoch 26/30
1688/1688 [==============================] - 2s 1ms/step - loss: 65.1387 -
accuracy: 0.7748
Epoch 27/30
1688/1688 [==============================] - 2s 1ms/step - loss: 641.3946 -
accuracy: 0.7534
Epoch 28/30
1688/1688 [==============================] - 2s 1ms/step - loss: 77.1088 -
accuracy: 0.7787
Epoch 29/30
1688/1688 [==============================] - 2s 1ms/step - loss: 856.2628 -
accuracy: 0.7558
Epoch 30/30
1688/1688 [==============================] - 2s 1ms/step - loss: 136.4784 -
accuracy: 0.7841
188/188 [==============================] - 0s 756us/step
22 out of 24 finished: {'activation_function': None, 'optimizer': 'Adam',
'optimizer__lr': 0.1}
Epoch 1/30

c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\adam.py:114: UserWarning: The `lr`
argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)

1688/1688 [==============================] - 2s 1ms/step - loss: 0.7356 -
accuracy: 0.7692
Epoch 2/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.8600 -
accuracy: 0.7878
Epoch 3/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5032 -
accuracy: 0.8278
Epoch 4/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5822 -
accuracy: 0.8106
Epoch 5/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6317 -
accuracy: 0.8097
Epoch 6/30
```

```
1688/1688 [==============================] - 2s 1ms/step - loss: 1.5931 -
accuracy: 0.8140
Epoch 7/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4842 -
accuracy: 0.8362
Epoch 8/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5471 -
accuracy: 0.8221
Epoch 9/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5862 -
accuracy: 0.8167
Epoch 10/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.0495 -
accuracy: 0.8127
Epoch 11/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4656 -
accuracy: 0.8416
Epoch 12/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5994 -
accuracy: 0.8154
Epoch 13/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5602 -
accuracy: 0.8216
Epoch 14/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6120 -
accuracy: 0.8150
Epoch 15/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.8399 -
accuracy: 0.8131
Epoch 16/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4671 -
accuracy: 0.8408
Epoch 17/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.9075 -
accuracy: 0.8171
Epoch 18/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4823 -
accuracy: 0.8362
Epoch 19/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6057 -
accuracy: 0.8153
Epoch 20/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5552 -
accuracy: 0.8232
Epoch 21/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6068 -
accuracy: 0.8156
Epoch 22/30
```

```
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6159 -
accuracy: 0.8217
Epoch 23/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.8498 -
accuracy: 0.8244
Epoch 24/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6292 -
accuracy: 0.8258
Epoch 25/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4953 -
accuracy: 0.8360
Epoch 26/30
1688/1688 [==============================] - 2s 1ms/step - loss: 1.0496 -
accuracy: 0.8182
Epoch 27/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4680 -
accuracy: 0.8424
Epoch 28/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.7327 -
accuracy: 0.8166
Epoch 29/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5536 -
accuracy: 0.8310
Epoch 30/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6967 -
accuracy: 0.8202
188/188 [==============================] - 0s 757us/step
23 out of 24 finished: {'activation_function': None, 'optimizer': 'Adam',
'optimizer__lr': 0.01}
Epoch 1/30

c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\adam.py:114: UserWarning: The `lr`
argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)

1688/1688 [==============================] - 2s 1ms/step - loss: 0.5494 -
accuracy: 0.8081
Epoch 2/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4708 -
accuracy: 0.8358
Epoch 3/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4517 -
accuracy: 0.8435
Epoch 4/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4407 -
accuracy: 0.8465
Epoch 5/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4332 -
```

```
accuracy: 0.8486
Epoch 6/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4258 -
accuracy: 0.8503
Epoch 7/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4213 -
accuracy: 0.8515
Epoch 8/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4145 -
accuracy: 0.8552
Epoch 9/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4121 -
accuracy: 0.8542
Epoch 10/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4123 -
accuracy: 0.8549
Epoch 11/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4079 -
accuracy: 0.8572
Epoch 12/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4043 -
accuracy: 0.8584
Epoch 13/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4021 -
accuracy: 0.8588
Epoch 14/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3993 -
accuracy: 0.8600
Epoch 15/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3970 -
accuracy: 0.8600
Epoch 16/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3963 -
accuracy: 0.8599
Epoch 17/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3944 -
accuracy: 0.8606
Epoch 18/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3921 -
accuracy: 0.8602
Epoch 19/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3914 -
accuracy: 0.8627
Epoch 20/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3906 -
accuracy: 0.8622
Epoch 21/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3899 -
```

```
accuracy: 0.8620
Epoch 22/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3882 -
accuracy: 0.8628
Epoch 23/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3872 -
accuracy: 0.8637
Epoch 24/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3878 -
accuracy: 0.8625
Epoch 25/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3855 -
accuracy: 0.8639
Epoch 26/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3839 -
accuracy: 0.8640
Epoch 27/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3828 -
accuracy: 0.8639
Epoch 28/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3820 -
accuracy: 0.8646
Epoch 29/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3806 -
accuracy: 0.8659
Epoch 30/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3806 -
accuracy: 0.8650
188/188 [==============================] - 0s 751us/step
24 out of 24 finished: {'activation_function': None, 'optimizer': 'Adam',
'optimizer__lr': 0.001}
```

[ ]:
```
show_results("MLP", mlp_result)
```

```
Results for MLP:
Best parameters: {'activation_function': 'sigmoid', 'optimizer': 'Adam',
'optimizer__lr': 0.001}
Best validation score: 0.8940
313/313 [==============================] - 0s 791us/step
Test set score: 0.8886

                                                    Score training_time  \
{'activation_function': 'relu', 'optimizer': 's…   0.872         49.03
{'activation_function': 'relu', 'optimizer': 's…  0.8833         49.13
{'activation_function': 'relu', 'optimizer': 's…  0.8488         48.98
{'activation_function': 'relu', 'optimizer': 'A…   0.198         52.16
{'activation_function': 'relu', 'optimizer': 'A…  0.8577         52.98
{'activation_function': 'relu', 'optimizer': 'A…  0.8927         52.72
{'activation_function': 'sigmoid', 'optimizer':…  0.8867         51.64
```

{'activation_function': 'sigmoid', 'optimizer':…	0.8455	51.55
{'activation_function': 'sigmoid', 'optimizer':…	0.6782	51.52
{'activation_function': 'sigmoid', 'optimizer':…	0.4063	54.35
{'activation_function': 'sigmoid', 'optimizer':…	0.8545	53.68
{'activation_function': 'sigmoid', 'optimizer':…	0.894	53.34
{'activation_function': 'tanh', 'optimizer': 's…	0.8873	50.07
{'activation_function': 'tanh', 'optimizer': 's…	0.8873	50.41
{'activation_function': 'tanh', 'optimizer': 's…	0.8467	50.21
{'activation_function': 'tanh', 'optimizer': 'A…	0.1	57.98
{'activation_function': 'tanh', 'optimizer': 'A…	0.7078	57.59
{'activation_function': 'tanh', 'optimizer': 'A…	0.8857	56.22
{'activation_function': None, 'optimizer': 'sgd…	0.8398	52.35
{'activation_function': None, 'optimizer': 'sgd…	0.8512	57.59
{'activation_function': None, 'optimizer': 'sgd…	0.8485	52.61
{'activation_function': None, 'optimizer': 'Ada…	0.7832	55.43
{'activation_function': None, 'optimizer': 'Ada…	0.7968	55.06
{'activation_function': None, 'optimizer': 'Ada…	0.8475	54.47

	validation_time
{'activation_function': 'relu', 'optimizer': 's…	0.31
{'activation_function': 'relu', 'optimizer': 's…	0.25
{'activation_function': 'relu', 'optimizer': 's…	0.25
{'activation_function': 'relu', 'optimizer': 'A…	0.25
{'activation_function': 'relu', 'optimizer': 'A…	0.26
{'activation_function': 'relu', 'optimizer': 'A…	0.25
{'activation_function': 'sigmoid', 'optimizer':…	0.25
{'activation_function': 'sigmoid', 'optimizer':…	0.26
{'activation_function': 'sigmoid', 'optimizer':…	0.26
{'activation_function': 'sigmoid', 'optimizer':…	0.26
{'activation_function': 'sigmoid', 'optimizer':…	0.25
{'activation_function': 'sigmoid', 'optimizer':…	0.26
{'activation_function': 'tanh', 'optimizer': 's…	0.25
{'activation_function': 'tanh', 'optimizer': 's…	0.25
{'activation_function': 'tanh', 'optimizer': 's…	0.25
{'activation_function': 'tanh', 'optimizer': 'A…	0.27
{'activation_function': 'tanh', 'optimizer': 'A…	0.26
{'activation_function': 'tanh', 'optimizer': 'A…	0.27
{'activation_function': None, 'optimizer': 'sgd…	0.26
{'activation_function': None, 'optimizer': 'sgd…	0.26
{'activation_function': None, 'optimizer': 'sgd…	0.26
{'activation_function': None, 'optimizer': 'Ada…	0.26
{'activation_function': None, 'optimizer': 'Ada…	0.26
{'activation_function': None, 'optimizer': 'Ada…	0.27

From the table above, both {'activation_function': 'relu', 'optimizer': 'Adam', 'optimizer___lr': 0.001} and {'activation_function': 'sigmoid', 'optimizer': 'Adam', 'optimizer___lr': 0.001} are good. Considering the limitations of Sigmoid activation function, we choose **ReLU** in this study.

Then we observe the trend of epochs with these paras.

[ ]:
```
# build a model with paras we just chose
mlp_model = build_mlp(activation_function="relu")

mlp_model.summary()

mlp_model.compile(loss='sparse_categorical_crossentropy',
              optimizer=keras.optimizers.Adam(learning_rate=0.001),
              metrics=['accuracy'])
```

```
Model: "sequential_35"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_35 (Flatten)        (None, 784)               0

 dense_103 (Dense)           (None, 100)               78500

 dense_104 (Dense)           (None, 60)                6060

 dense_105 (Dense)           (None, 10)                610

=================================================================
Total params: 85,170
Trainable params: 85,170
Non-trainable params: 0
_________________________________________________________________
```

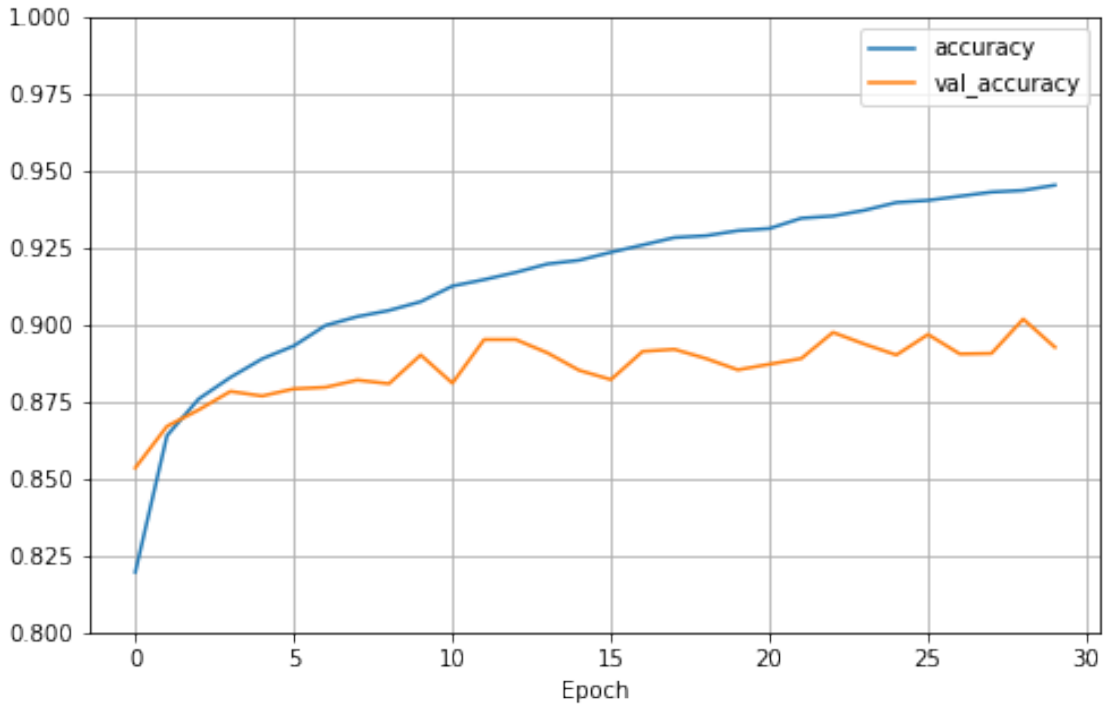
[ ]:
```
mlp_history = mlp_model.fit(X_train, y_train, epochs=30,␣
 ↪validation_data=(X_valid, y_valid))
```

```
Epoch 1/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5115 -
accuracy: 0.8196 - val_loss: 0.3997 - val_accuracy: 0.8535
Epoch 2/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3767 -
accuracy: 0.8640 - val_loss: 0.3651 - val_accuracy: 0.8670
Epoch 3/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3391 -
accuracy: 0.8759 - val_loss: 0.3397 - val_accuracy: 0.8723
Epoch 4/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3171 -
accuracy: 0.8829 - val_loss: 0.3278 - val_accuracy: 0.8783
Epoch 5/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2985 -
accuracy: 0.8889 - val_loss: 0.3359 - val_accuracy: 0.8768
Epoch 6/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2863 -
accuracy: 0.8931 - val_loss: 0.3239 - val_accuracy: 0.8792
```

```
Epoch 7/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2728 -
accuracy: 0.8998 - val_loss: 0.3243 - val_accuracy: 0.8797
Epoch 8/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2595 -
accuracy: 0.9026 - val_loss: 0.3194 - val_accuracy: 0.8820
Epoch 9/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2523 -
accuracy: 0.9046 - val_loss: 0.3274 - val_accuracy: 0.8808
Epoch 10/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2444 -
accuracy: 0.9075 - val_loss: 0.3019 - val_accuracy: 0.8902
Epoch 11/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2326 -
accuracy: 0.9126 - val_loss: 0.3307 - val_accuracy: 0.8810
Epoch 12/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2266 -
accuracy: 0.9146 - val_loss: 0.3046 - val_accuracy: 0.8952
Epoch 13/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2185 -
accuracy: 0.9170 - val_loss: 0.3076 - val_accuracy: 0.8952
Epoch 14/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2130 -
accuracy: 0.9197 - val_loss: 0.3079 - val_accuracy: 0.8908
Epoch 15/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2058 -
accuracy: 0.9209 - val_loss: 0.3383 - val_accuracy: 0.8852
Epoch 16/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1997 -
accuracy: 0.9235 - val_loss: 0.3601 - val_accuracy: 0.8822
Epoch 17/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1960 -
accuracy: 0.9259 - val_loss: 0.3288 - val_accuracy: 0.8913
Epoch 18/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1888 -
accuracy: 0.9283 - val_loss: 0.3342 - val_accuracy: 0.8920
Epoch 19/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1849 -
accuracy: 0.9289 - val_loss: 0.3312 - val_accuracy: 0.8890
Epoch 20/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1831 -
accuracy: 0.9306 - val_loss: 0.3298 - val_accuracy: 0.8853
Epoch 21/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1796 -
accuracy: 0.9313 - val_loss: 0.3571 - val_accuracy: 0.8872
Epoch 22/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1696 -
accuracy: 0.9346 - val_loss: 0.3485 - val_accuracy: 0.8890
```

```
Epoch 23/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1685 -
accuracy: 0.9353 - val_loss: 0.3449 - val_accuracy: 0.8975
Epoch 24/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1644 -
accuracy: 0.9372 - val_loss: 0.3374 - val_accuracy: 0.8937
Epoch 25/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1600 -
accuracy: 0.9397 - val_loss: 0.3626 - val_accuracy: 0.8902
Epoch 26/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1554 -
accuracy: 0.9404 - val_loss: 0.3714 - val_accuracy: 0.8968
Epoch 27/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1527 -
accuracy: 0.9417 - val_loss: 0.3586 - val_accuracy: 0.8905
Epoch 28/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1472 -
accuracy: 0.9431 - val_loss: 0.4041 - val_accuracy: 0.8907
Epoch 29/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1449 -
accuracy: 0.9436 - val_loss: 0.3459 - val_accuracy: 0.9018
Epoch 30/30
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1413 -
accuracy: 0.9453 - val_loss: 0.3922 - val_accuracy: 0.8927
```

[ ]:
```
# Convert the history dictionary to a Pandas dataframe and extract the␣
↪accuracies
accuracies = pd.DataFrame(mlp_history.history)[['accuracy', 'val_accuracy']]

# Plot the accuracies
accuracies.plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0.8, 1)
plt.xlabel('Epoch')
plt.show()
```



The accuracy of validation set fluctuated since epoch = 10, thus, we set the epochs = 10 to keep it small.

1.4.3 3.3 Convolutional neural network

In section 2.2, we fixed a set ot paras as well as the architecture of CNN.

[ ]:
```
def build_cnn(kernel_size=(3, 3), strides=(1, 1), activation_function="relu"):
    """Build a Keras CNN for 10 class classification with desired parameters."""

    model = keras.Sequential([
        # Specify the input shape
        keras.Input(shape=(*IMAGE_SIZE, 1)),

        # Conv and pool block 1
        keras.layers.Conv2D(
            32,
            kernel_size=kernel_size,
            activation=activation_function,
            strides=strides,
            kernel_initializer=initializer
            ),
        keras.layers.MaxPooling2D(pool_size=(2, 2), padding='same'), # padding␣
↪evenly
```

```
        # Conv and pool block 2
        keras.layers.Conv2D(
            64,
            kernel_size=kernel_size,
            activation=activation_function,
            strides=strides,
            kernel_initializer=initializer
            ),
        keras.layers.MaxPooling2D(pool_size=(2, 2), padding='same'), # padding␣
 ↪evenly

        # Flatten and classify using dense output layer
        keras.layers.Flatten(),
        keras.layers.Dropout(0.5),
        keras.layers.Dense(len(class_names), activation="softmax",␣
 ↪kernel_initializer=initializer),
    ])

    return model

# Create a KerasClassifier object which works with sklearn grid searches
# We need to pass default values of arguments in build_cnn if we wish to tune␣
 ↪them
keras_classifier = KerasClassifier(build_cnn,
                                   kernel_size=(3, 3),
                                   strides=(1, 1),
                                   activation_function="relu",
                                   loss="sparse_categorical_crossentropy",
                                   optimizer="adam",
                                   optimizer__lr=0.01,
                                   metrics=["accuracy"]
                                  )

# For an odd-sized filter, all the previous layer pixels would be symmetrical␣
 ↪around the output pixel.
param_grid = {
    "optimizer__lr": [0.01, 0.005, 0.001],
    "kernel_size": [(3, 3), (5, 5)],
    "strides": [(1, 1), (2, 2)]
}

cnn_paras = ParameterGrid(param_grid)

print(f"There are {len(list(cnn_paras))} combinations.")
print("Parameter grid:\n{}".format(param_grid))
```

```
There are 12 combinations.
Parameter grid:
{'optimizer__lr': [0.01, 0.005, 0.001], 'kernel_size': [(3, 3), (5, 5)],
'strides': [(1, 1), (2, 2)]}
```

[ ]:
```
# Running in arround 1100s

cnn_result = get_result(
    keras_classifier,
    cnn_paras,
    X_train=np.expand_dims(X_train, -1),
    X_valid=np.expand_dims(X_valid, -1),
    epochs=10
    )
```

```
Epoch 1/10

c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\adam.py:114: UserWarning: The `lr`
argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)

1688/1688 [==============================] - 14s 8ms/step - loss: 0.5044 -
accuracy: 0.8175
Epoch 2/10
1688/1688 [==============================] - 13s 8ms/step - loss: 0.4250 -
accuracy: 0.8472
Epoch 3/10
1688/1688 [==============================] - 13s 8ms/step - loss: 0.4157 -
accuracy: 0.8500
Epoch 4/10
1688/1688 [==============================] - 13s 8ms/step - loss: 0.4070 -
accuracy: 0.8526
Epoch 5/10
1688/1688 [==============================] - 13s 8ms/step - loss: 0.4085 -
accuracy: 0.8510
Epoch 6/10
1688/1688 [==============================] - 13s 8ms/step - loss: 0.4048 -
accuracy: 0.8523
Epoch 7/10
1688/1688 [==============================] - 13s 8ms/step - loss: 0.4004 -
accuracy: 0.8531
Epoch 8/10
1688/1688 [==============================] - 13s 8ms/step - loss: 0.3999 -
accuracy: 0.8557
Epoch 9/10
1688/1688 [==============================] - 13s 8ms/step - loss: 0.3996 -
accuracy: 0.8537
Epoch 10/10
```

```
1688/1688 [==============================] - 13s 8ms/step - loss: 0.4034 -
accuracy: 0.8536
188/188 [==============================] - 0s 2ms/step
1 out of 12 finished: {'kernel_size': (3, 3), 'optimizer__lr': 0.01, 'strides':
(1, 1)}
Epoch 1/10

c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\adam.py:114: UserWarning: The `lr`
argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)

1688/1688 [==============================] - 5s 3ms/step - loss: 0.6453 -
accuracy: 0.7617
Epoch 2/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.5320 -
accuracy: 0.8065
Epoch 3/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.5166 -
accuracy: 0.8136
Epoch 4/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.5010 -
accuracy: 0.8199
Epoch 5/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.4991 -
accuracy: 0.8179
Epoch 6/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.4936 -
accuracy: 0.8190
Epoch 7/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.4884 -
accuracy: 0.8219
Epoch 8/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.4878 -
accuracy: 0.8227
Epoch 9/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.4846 -
accuracy: 0.8223
Epoch 10/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.4837 -
accuracy: 0.8220
188/188 [==============================] - 0s 998us/step
2 out of 12 finished: {'kernel_size': (3, 3), 'optimizer__lr': 0.01, 'strides':
(2, 2)}
Epoch 1/10

c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\adam.py:114: UserWarning: The `lr`
argument is deprecated, use `learning_rate` instead.
```

```
  super().__init__(name, **kwargs)

1688/1688 [==============================] - 13s 8ms/step - loss: 0.4701 -
accuracy: 0.8295
Epoch 2/10
1688/1688 [==============================] - 13s 8ms/step - loss: 0.3734 -
accuracy: 0.8655
Epoch 3/10
1688/1688 [==============================] - 13s 8ms/step - loss: 0.3519 -
accuracy: 0.8721
Epoch 4/10
1688/1688 [==============================] - 13s 8ms/step - loss: 0.3374 -
accuracy: 0.8782
Epoch 5/10
1688/1688 [==============================] - 13s 8ms/step - loss: 0.3284 -
accuracy: 0.8799
Epoch 6/10
1688/1688 [==============================] - 13s 8ms/step - loss: 0.3258 -
accuracy: 0.8808
Epoch 7/10
1688/1688 [==============================] - 13s 8ms/step - loss: 0.3199 -
accuracy: 0.8826
Epoch 8/10
1688/1688 [==============================] - 13s 8ms/step - loss: 0.3155 -
accuracy: 0.8835
Epoch 9/10
1688/1688 [==============================] - 13s 8ms/step - loss: 0.3061 -
accuracy: 0.8873
Epoch 10/10
1688/1688 [==============================] - 13s 8ms/step - loss: 0.3098 -
accuracy: 0.8860
188/188 [==============================] - 0s 2ms/step
3 out of 12 finished: {'kernel_size': (3, 3), 'optimizer__lr': 0.005, 'strides':
(1, 1)}
Epoch 1/10

c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\adam.py:114: UserWarning: The `lr`
argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)

1688/1688 [==============================] - 5s 3ms/step - loss: 0.6295 -
accuracy: 0.7703
Epoch 2/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.4755 -
accuracy: 0.8267
Epoch 3/10
1688/1688 [==============================] - 4s 2ms/step - loss: 0.4458 -
accuracy: 0.8371
```

```
Epoch 4/10
1688/1688 [==============================] - 4s 2ms/step - loss: 0.4258 -
accuracy: 0.8459
Epoch 5/10
1688/1688 [==============================] - 4s 2ms/step - loss: 0.4152 -
accuracy: 0.8508
Epoch 6/10
1688/1688 [==============================] - 4s 2ms/step - loss: 0.4043 -
accuracy: 0.8530
Epoch 7/10
1688/1688 [==============================] - 4s 2ms/step - loss: 0.3981 -
accuracy: 0.8532
Epoch 8/10
1688/1688 [==============================] - 4s 2ms/step - loss: 0.3947 -
accuracy: 0.8547
Epoch 9/10
1688/1688 [==============================] - 4s 2ms/step - loss: 0.3920 -
accuracy: 0.8544
Epoch 10/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.3892 -
accuracy: 0.8576
188/188 [==============================] - 0s 1ms/step
4 out of 12 finished: {'kernel_size': (3, 3), 'optimizer__lr': 0.005, 'strides':
(2, 2)}
Epoch 1/10

c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\adam.py:114: UserWarning: The `lr`
argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)

1688/1688 [==============================] - 13s 8ms/step - loss: 0.5201 -
accuracy: 0.8115
Epoch 2/10
1688/1688 [==============================] - 13s 8ms/step - loss: 0.3668 -
accuracy: 0.8700
Epoch 3/10
1688/1688 [==============================] - 13s 8ms/step - loss: 0.3272 -
accuracy: 0.8831
Epoch 4/10
1688/1688 [==============================] - 14s 8ms/step - loss: 0.3026 -
accuracy: 0.8908
Epoch 5/10
1688/1688 [==============================] - 14s 8ms/step - loss: 0.2882 -
accuracy: 0.8964
Epoch 6/10
1688/1688 [==============================] - 15s 9ms/step - loss: 0.2735 -
accuracy: 0.9010
Epoch 7/10
```

```
1688/1688 [==============================] - 14s 8ms/step - loss: 0.2611 -
accuracy: 0.9049
Epoch 8/10
1688/1688 [==============================] - 14s 8ms/step - loss: 0.2527 -
accuracy: 0.9085
Epoch 9/10
1688/1688 [==============================] - 14s 8ms/step - loss: 0.2463 -
accuracy: 0.9091
Epoch 10/10
1688/1688 [==============================] - 14s 8ms/step - loss: 0.2414 -
accuracy: 0.9126
188/188 [==============================] - 1s 2ms/step
5 out of 12 finished: {'kernel_size': (3, 3), 'optimizer__lr': 0.001, 'strides':
(1, 1)}
Epoch 1/10

c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\adam.py:114: UserWarning: The `lr`
argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)

1688/1688 [==============================] - 5s 3ms/step - loss: 0.7486 -
accuracy: 0.7271
Epoch 2/10
1688/1688 [==============================] - 5s 3ms/step - loss: 0.5156 -
accuracy: 0.8138
Epoch 3/10
1688/1688 [==============================] - 5s 3ms/step - loss: 0.4618 -
accuracy: 0.8321
Epoch 4/10
1688/1688 [==============================] - 5s 3ms/step - loss: 0.4325 -
accuracy: 0.8451
Epoch 5/10
1688/1688 [==============================] - 5s 3ms/step - loss: 0.4119 -
accuracy: 0.8521
Epoch 6/10
1688/1688 [==============================] - 5s 3ms/step - loss: 0.3971 -
accuracy: 0.8567
Epoch 7/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.3840 -
accuracy: 0.8605
Epoch 8/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.3750 -
accuracy: 0.8638
Epoch 9/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.3660 -
accuracy: 0.8673
Epoch 10/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.3599 -
```

```
accuracy: 0.8704
188/188 [==============================] - 0s 992us/step
6 out of 12 finished: {'kernel_size': (3, 3), 'optimizer__lr': 0.001, 'strides':
(2, 2)}
Epoch 1/10

c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\adam.py:114: UserWarning: The `lr`
argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)

1688/1688 [==============================] - 14s 8ms/step - loss: 0.5870 -
accuracy: 0.7874
Epoch 2/10
1688/1688 [==============================] - 13s 8ms/step - loss: 0.4884 -
accuracy: 0.8250
Epoch 3/10
1688/1688 [==============================] - 13s 8ms/step - loss: 0.4843 -
accuracy: 0.8254
Epoch 4/10
1688/1688 [==============================] - 13s 8ms/step - loss: 0.4742 -
accuracy: 0.8286
Epoch 5/10
1688/1688 [==============================] - 13s 8ms/step - loss: 0.4735 -
accuracy: 0.8298
Epoch 6/10
1688/1688 [==============================] - 13s 8ms/step - loss: 0.4656 -
accuracy: 0.8309
Epoch 7/10
1688/1688 [==============================] - 13s 8ms/step - loss: 0.4777 -
accuracy: 0.8279
Epoch 8/10
1688/1688 [==============================] - 13s 8ms/step - loss: 0.4648 -
accuracy: 0.8329
Epoch 9/10
1688/1688 [==============================] - 15s 9ms/step - loss: 0.4661 -
accuracy: 0.8282
Epoch 10/10
1688/1688 [==============================] - 14s 8ms/step - loss: 0.4647 -
accuracy: 0.8328
188/188 [==============================] - 1s 3ms/step
7 out of 12 finished: {'kernel_size': (5, 5), 'optimizer__lr': 0.01, 'strides':
(1, 1)}
Epoch 1/10

c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\adam.py:114: UserWarning: The `lr`
argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)
```

```
1688/1688 [==============================] - 5s 3ms/step - loss: 0.6395 -
accuracy: 0.7672
Epoch 2/10
1688/1688 [==============================] - 5s 3ms/step - loss: 0.5430 -
accuracy: 0.8005
Epoch 3/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.5268 -
accuracy: 0.8028
Epoch 4/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.5178 -
accuracy: 0.8073
Epoch 5/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.5118 -
accuracy: 0.8103
Epoch 6/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.5085 -
accuracy: 0.8110
Epoch 7/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.5047 -
accuracy: 0.8121
Epoch 8/10
1688/1688 [==============================] - 4s 2ms/step - loss: 0.5016 -
accuracy: 0.8131
Epoch 9/10
1688/1688 [==============================] - 5s 3ms/step - loss: 0.4968 -
accuracy: 0.8129
Epoch 10/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.4983 -
accuracy: 0.8140
188/188 [==============================] - 0s 1ms/step
8 out of 12 finished: {'kernel_size': (5, 5), 'optimizer__lr': 0.01, 'strides':
(2, 2)}
Epoch 1/10

c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\adam.py:114: UserWarning: The `lr`
argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)

1688/1688 [==============================] - 15s 9ms/step - loss: 0.5323 -
accuracy: 0.8076
Epoch 2/10
1688/1688 [==============================] - 14s 8ms/step - loss: 0.4223 -
accuracy: 0.8460
Epoch 3/10
1688/1688 [==============================] - 15s 9ms/step - loss: 0.3962 -
accuracy: 0.8539
Epoch 4/10
1688/1688 [==============================] - 14s 8ms/step - loss: 0.3853 -
```

```
accuracy: 0.8608
Epoch 5/10
1688/1688 [==============================] - 14s 8ms/step - loss: 0.3756 -
accuracy: 0.8621
Epoch 6/10
1688/1688 [==============================] - 14s 8ms/step - loss: 0.3715 -
accuracy: 0.8636
Epoch 7/10
1688/1688 [==============================] - 14s 8ms/step - loss: 0.3687 -
accuracy: 0.8651
Epoch 8/10
1688/1688 [==============================] - 14s 8ms/step - loss: 0.3588 -
accuracy: 0.8683
Epoch 9/10
1688/1688 [==============================] - 14s 8ms/step - loss: 0.3582 -
accuracy: 0.8688
Epoch 10/10
1688/1688 [==============================] - 14s 8ms/step - loss: 0.3554 -
accuracy: 0.8701
188/188 [==============================] - 1s 3ms/step
9 out of 12 finished: {'kernel_size': (5, 5), 'optimizer__lr': 0.005, 'strides':
(1, 1)}
Epoch 1/10

c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\adam.py:114: UserWarning: The `lr`
argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)

1688/1688 [==============================] - 5s 3ms/step - loss: 0.6417 -
accuracy: 0.7657
Epoch 2/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.4959 -
accuracy: 0.8187
Epoch 3/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.4638 -
accuracy: 0.8282
Epoch 4/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.4484 -
accuracy: 0.8337
Epoch 5/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.4397 -
accuracy: 0.8359
Epoch 6/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.4279 -
accuracy: 0.8392
Epoch 7/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.4168 -
accuracy: 0.8424
```

```
Epoch 8/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.4161 -
accuracy: 0.8434
Epoch 9/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.4096 -
accuracy: 0.8454
Epoch 10/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.4080 -
accuracy: 0.8471
188/188 [==============================] - 0s 1ms/step
10 out of 12 finished: {'kernel_size': (5, 5), 'optimizer__lr': 0.005,
'strides': (2, 2)}
Epoch 1/10

c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\adam.py:114: UserWarning: The `lr`
argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)

1688/1688 [==============================] - 15s 9ms/step - loss: 0.5577 -
accuracy: 0.7958
Epoch 2/10
1688/1688 [==============================] - 14s 8ms/step - loss: 0.3883 -
accuracy: 0.8614
Epoch 3/10
1688/1688 [==============================] - 14s 8ms/step - loss: 0.3465 -
accuracy: 0.8748
Epoch 4/10
1688/1688 [==============================] - 14s 8ms/step - loss: 0.3230 -
accuracy: 0.8843
Epoch 5/10
1688/1688 [==============================] - 14s 8ms/step - loss: 0.3048 -
accuracy: 0.8896
Epoch 6/10
1688/1688 [==============================] - 14s 8ms/step - loss: 0.2918 -
accuracy: 0.8934
Epoch 7/10
1688/1688 [==============================] - 14s 8ms/step - loss: 0.2797 -
accuracy: 0.8995
Epoch 8/10
1688/1688 [==============================] - 14s 8ms/step - loss: 0.2705 -
accuracy: 0.9016
Epoch 9/10
1688/1688 [==============================] - 14s 8ms/step - loss: 0.2617 -
accuracy: 0.9042
Epoch 10/10
1688/1688 [==============================] - 14s 8ms/step - loss: 0.2557 -
accuracy: 0.9061
188/188 [==============================] - 1s 3ms/step
```

```
11 out of 12 finished: {'kernel_size': (5, 5), 'optimizer__lr': 0.001,
'strides': (1, 1)}
Epoch 1/10

c:\Program Files\Anaconda3-2021.11x64\lib\site-
packages\keras\optimizers\optimizer_v2\adam.py:114: UserWarning: The `lr`
argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)

1688/1688 [==============================] - 5s 3ms/step - loss: 0.7298 -
accuracy: 0.7438
Epoch 2/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.5274 -
accuracy: 0.8151
Epoch 3/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.4729 -
accuracy: 0.8302
Epoch 4/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.4409 -
accuracy: 0.8399
Epoch 5/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.4174 -
accuracy: 0.8481
Epoch 6/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.4015 -
accuracy: 0.8536
Epoch 7/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.3896 -
accuracy: 0.8577
Epoch 8/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.3779 -
accuracy: 0.8606
Epoch 9/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.3678 -
accuracy: 0.8639
Epoch 10/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.3632 -
accuracy: 0.8647
188/188 [==============================] - 0s 997us/step
12 out of 12 finished: {'kernel_size': (5, 5), 'optimizer__lr': 0.001,
'strides': (2, 2)}
```

[ ]:
```
show_results("CNN", cnn_result, X_test=np.expand_dims(X_test, -1))
```

```
Results for CNN:
Best parameters: {'kernel_size': (3, 3), 'optimizer__lr': 0.001, 'strides': (1,
1)}
Best validation score: 0.9175
313/313 [==============================] - 1s 2ms/step
```

```
Test set score: 0.9125
```

	Score	training_time	validation_time
{'kernel_size': (3, 3), 'optimizer__lr': 0.01, …	0.8717	130.88	0.54
{'kernel_size': (3, 3), 'optimizer__lr': 0.01, …	0.8602	43.34	0.31
{'kernel_size': (3, 3), 'optimizer__lr': 0.005,…	0.8988	130.16	0.54
{'kernel_size': (3, 3), 'optimizer__lr': 0.005,…	0.889	42.69	0.31
{'kernel_size': (3, 3), 'optimizer__lr': 0.001,…	0.9175	138.12	0.6
{'kernel_size': (3, 3), 'optimizer__lr': 0.001,…	0.889	45.52	0.31
{'kernel_size': (5, 5), 'optimizer__lr': 0.01, …	0.8502	136.82	0.61
{'kernel_size': (5, 5), 'optimizer__lr': 0.01, …	0.8483	44.92	0.32
{'kernel_size': (5, 5), 'optimizer__lr': 0.005,…	0.8833	141.77	0.61
{'kernel_size': (5, 5), 'optimizer__lr': 0.005,…	0.8688	43.64	0.32
{'kernel_size': (5, 5), 'optimizer__lr': 0.001,…	0.9137	140.6	0.61
{'kernel_size': (5, 5), 'optimizer__lr': 0.001,…	0.8857	43.86	0.31

From the table above, the best paras in our setting is: 'kernel_size': (3, 3), 'optimizer___lr': 0.001, 'strides': (1, 1)

Then we observe the trend of epochs with these paras.

[ ]:
```
# build a model with paras we just chose
cnn_model = build_cnn(
    kernel_size=(3, 3),
    strides=(1, 1),
    activation_function="relu"
    )

cnn_model.summary()

cnn_model.compile(loss='sparse_categorical_crossentropy',
              optimizer=keras.optimizers.Adam(learning_rate=0.001),
              metrics=['accuracy'])
```

```
Model: "sequential_48"
_________________________________________________________________
```

```
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_26 (Conv2D)          (None, 26, 26, 32)        320

 max_pooling2d_26 (MaxPoolin  (None, 13, 13, 32)       0
 g2D)

 conv2d_27 (Conv2D)          (None, 11, 11, 64)        18496

 max_pooling2d_27 (MaxPoolin  (None, 6, 6, 64)         0
 g2D)

 flatten_48 (Flatten)        (None, 2304)              0

 dropout_13 (Dropout)        (None, 2304)              0

 dense_118 (Dense)           (None, 10)                23050

=================================================================
Total params: 41,866
Trainable params: 41,866
Non-trainable params: 0
_________________________________________________________________
```

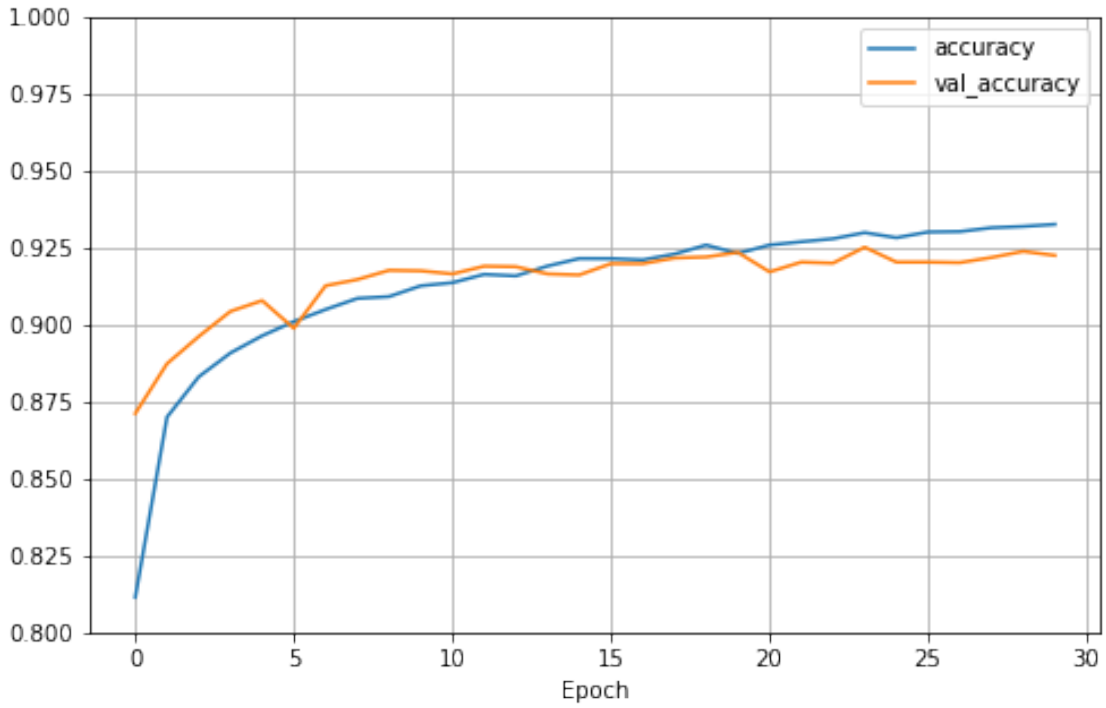
[ ]:
```
cnn_history = cnn_model.fit(
    np.expand_dims(X_train, -1),
    y_train,
    epochs=30,
    validation_data=(np.expand_dims(X_valid, -1), y_valid)
    )
```

```
Epoch 1/30
1688/1688 [==============================] - 14s 8ms/step - loss: 0.5201 -
accuracy: 0.8115 - val_loss: 0.3583 - val_accuracy: 0.8712
Epoch 2/30
1688/1688 [==============================] - 14s 8ms/step - loss: 0.3668 -
accuracy: 0.8700 - val_loss: 0.3091 - val_accuracy: 0.8873
Epoch 3/30
1688/1688 [==============================] - 14s 8ms/step - loss: 0.3272 -
accuracy: 0.8831 - val_loss: 0.2851 - val_accuracy: 0.8962
Epoch 4/30
1688/1688 [==============================] - 14s 8ms/step - loss: 0.3026 -
accuracy: 0.8908 - val_loss: 0.2628 - val_accuracy: 0.9043
Epoch 5/30
1688/1688 [==============================] - 14s 8ms/step - loss: 0.2882 -
accuracy: 0.8964 - val_loss: 0.2507 - val_accuracy: 0.9078
Epoch 6/30
1688/1688 [==============================] - 14s 8ms/step - loss: 0.2735 -
```

```
accuracy: 0.9010 - val_loss: 0.2696 - val_accuracy: 0.8988
Epoch 7/30
1688/1688 [==============================] - 14s 8ms/step - loss: 0.2611 -
accuracy: 0.9049 - val_loss: 0.2415 - val_accuracy: 0.9127
Epoch 8/30
1688/1688 [==============================] - 14s 8ms/step - loss: 0.2527 -
accuracy: 0.9085 - val_loss: 0.2334 - val_accuracy: 0.9147
Epoch 9/30
1688/1688 [==============================] - 14s 8ms/step - loss: 0.2463 -
accuracy: 0.9091 - val_loss: 0.2297 - val_accuracy: 0.9177
Epoch 10/30
1688/1688 [==============================] - 14s 8ms/step - loss: 0.2414 -
accuracy: 0.9126 - val_loss: 0.2257 - val_accuracy: 0.9175
Epoch 11/30
1688/1688 [==============================] - 14s 8ms/step - loss: 0.2351 -
accuracy: 0.9136 - val_loss: 0.2237 - val_accuracy: 0.9165
Epoch 12/30
1688/1688 [==============================] - 14s 8ms/step - loss: 0.2296 -
accuracy: 0.9164 - val_loss: 0.2211 - val_accuracy: 0.9190
Epoch 13/30
1688/1688 [==============================] - 14s 8ms/step - loss: 0.2274 -
accuracy: 0.9159 - val_loss: 0.2206 - val_accuracy: 0.9188
Epoch 14/30
1688/1688 [==============================] - 14s 8ms/step - loss: 0.2206 -
accuracy: 0.9191 - val_loss: 0.2210 - val_accuracy: 0.9165
Epoch 15/30
1688/1688 [==============================] - 13s 8ms/step - loss: 0.2181 -
accuracy: 0.9215 - val_loss: 0.2265 - val_accuracy: 0.9162
Epoch 16/30
1688/1688 [==============================] - 14s 8ms/step - loss: 0.2152 -
accuracy: 0.9215 - val_loss: 0.2166 - val_accuracy: 0.9198
Epoch 17/30
1688/1688 [==============================] - 14s 8ms/step - loss: 0.2128 -
accuracy: 0.9211 - val_loss: 0.2193 - val_accuracy: 0.9198
Epoch 18/30
1688/1688 [==============================] - 14s 8ms/step - loss: 0.2079 -
accuracy: 0.9229 - val_loss: 0.2151 - val_accuracy: 0.9217
Epoch 19/30
1688/1688 [==============================] - 14s 8ms/step - loss: 0.2018 -
accuracy: 0.9259 - val_loss: 0.2156 - val_accuracy: 0.9220
Epoch 20/30
1688/1688 [==============================] - 13s 8ms/step - loss: 0.2064 -
accuracy: 0.9232 - val_loss: 0.2105 - val_accuracy: 0.9235
Epoch 21/30
1688/1688 [==============================] - 14s 8ms/step - loss: 0.1998 -
accuracy: 0.9259 - val_loss: 0.2279 - val_accuracy: 0.9172
Epoch 22/30
1688/1688 [==============================] - 14s 8ms/step - loss: 0.2003 -
```

```
accuracy: 0.9269 - val_loss: 0.2213 - val_accuracy: 0.9203
Epoch 23/30
1688/1688 [==============================] - 14s 8ms/step - loss: 0.1965 -
accuracy: 0.9279 - val_loss: 0.2173 - val_accuracy: 0.9200
Epoch 24/30
1688/1688 [==============================] - 14s 8ms/step - loss: 0.1913 -
accuracy: 0.9299 - val_loss: 0.2124 - val_accuracy: 0.9252
Epoch 25/30
1688/1688 [==============================] - 14s 8ms/step - loss: 0.1930 -
accuracy: 0.9283 - val_loss: 0.2219 - val_accuracy: 0.9203
Epoch 26/30
1688/1688 [==============================] - 14s 8ms/step - loss: 0.1909 -
accuracy: 0.9301 - val_loss: 0.2231 - val_accuracy: 0.9203
Epoch 27/30
1688/1688 [==============================] - 13s 8ms/step - loss: 0.1885 -
accuracy: 0.9303 - val_loss: 0.2157 - val_accuracy: 0.9202
Epoch 28/30
1688/1688 [==============================] - 14s 8ms/step - loss: 0.1845 -
accuracy: 0.9315 - val_loss: 0.2181 - val_accuracy: 0.9218
Epoch 29/30
1688/1688 [==============================] - 14s 8ms/step - loss: 0.1838 -
accuracy: 0.9319 - val_loss: 0.2164 - val_accuracy: 0.9238
Epoch 30/30
1688/1688 [==============================] - 14s 8ms/step - loss: 0.1820 -
accuracy: 0.9326 - val_loss: 0.2155 - val_accuracy: 0.9225
```

[ ]:
```
# Convert the history dictionary to a Pandas dataframe and extract the␣
↪accuracies
accuracies = pd.DataFrame(cnn_history.history)[['accuracy', 'val_accuracy']]

# Plot the accuracies
accuracies.plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0.8, 1)
plt.xlabel('Epoch')
plt.show()
```



The accuracy of validation set fluctuated since epoch = 12, thus, we set the epochs = 12.

1.5 4. Final models

1.5.1 4.1 K-nearest neighbors

Best paras for K-nearest neighbors is:
'n_neighbors'= 3, 'p'= 1, 'weights'= 'distance'.

[ ]:
```
# Set the best paras
knn_best_paras = dict({'n_neighbors': 3, 'p': 1, 'weights': 'distance'})

knn = KNeighborsClassifier(**knn_best_paras)
knn_runtime = time.time()
knn.fit(X_train_full.reshape(X_train_full.shape[0], -1), y_train_full)
knn_runtime = time.time() - knn_runtime
```

[ ]:
```
# Running in around 120s

# Performance on test set.
print(f"KNN training time: {knn_runtime:.2f} s")
print(f"KNN score on the test set: {knn.score(X_test.reshape(X_test.shape[0],␣
  ↪-1), y_test):.4f}")
```

```
KNN training time: 0.07 s
KNN score on the test set: 0.8597
```

1.5.2 4.2 Fully connected neural network

We settled the size of layers (784, 100, 60, 10) in section 2.2; and settled the the best paras of our experimental settings:
'activation_function': 'relu', 'optimizer': 'Adam', 'optimizer___lr': 0.001
epochs: 10

[ ]:
```
# Running in around 25s

keras.backend.clear_session()

# Set Random seed to 0
initializer = tf.keras.initializers.GlorotUniform(seed=0)

# Build the final model
mlp = keras.models.Sequential([
    keras.layers.Flatten(input_shape=IMAGE_SIZE),
    keras.layers.Dense(100, activation="relu", kernel_initializer=initializer),
    keras.layers.Dense(60, activation="relu", kernel_initializer=initializer),
    keras.layers.Dense(10, activation="softmax", kernel_initializer=initializer)
])

mlp.summary()

# Complie the model
mlp.compile(loss='sparse_categorical_crossentropy',
              optimizer=keras.optimizers.Adam(learning_rate=0.001),
              metrics=['accuracy'])

# Train the model
mlp_runtime = time.time()
mlp.fit(X_train, y_train, epochs=10, validation_data=(X_valid, y_valid),␣
  ↪verbose=1)
mlp_runtime = time.time() - mlp_runtime
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 784)               0

 dense (Dense)               (None, 100)               78500

 dense_1 (Dense)             (None, 60)                6060

 dense_2 (Dense)             (None, 10)                610

=================================================================
```

```
Total params: 85,170
Trainable params: 85,170
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
1688/1688 [==============================] - 3s 2ms/step - loss: 0.5115 -
accuracy: 0.8196 - val_loss: 0.3997 - val_accuracy: 0.8535
Epoch 2/10
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3767 -
accuracy: 0.8640 - val_loss: 0.3651 - val_accuracy: 0.8670
Epoch 3/10
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3391 -
accuracy: 0.8759 - val_loss: 0.3397 - val_accuracy: 0.8723
Epoch 4/10
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3171 -
accuracy: 0.8829 - val_loss: 0.3278 - val_accuracy: 0.8783
Epoch 5/10
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2985 -
accuracy: 0.8889 - val_loss: 0.3359 - val_accuracy: 0.8768
Epoch 6/10
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2863 -
accuracy: 0.8931 - val_loss: 0.3239 - val_accuracy: 0.8792
Epoch 7/10
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2728 -
accuracy: 0.8998 - val_loss: 0.3243 - val_accuracy: 0.8797
Epoch 8/10
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2595 -
accuracy: 0.9026 - val_loss: 0.3194 - val_accuracy: 0.8820
Epoch 9/10
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2523 -
accuracy: 0.9046 - val_loss: 0.3274 - val_accuracy: 0.8808
Epoch 10/10
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2444 -
accuracy: 0.9075 - val_loss: 0.3019 - val_accuracy: 0.8902
```

[ ]:
```python
# Performance on test set.
loss, accuracy = mlp.evaluate(X_test, y_test)
print(f"MLP training time: {mlp_runtime:.2f} s")
print(f"MLP score on the test set: {accuracy:.4f}")
```

```
313/313 [==============================] - 0s 971us/step - loss: 0.3335 -
accuracy: 0.8864
MLP training time: 24.62 s
MLP score on the test set: 0.8864
```

1.5.3 4.3 Convolutional neural network

We designed a proper architecture of CNN in section 2.3; and settled the the best paras of our experimental settings:
'kernel_size': (3, 3), 'optimizer___lr': 0.001, 'strides': (1, 1)
epochs: 12

[ ]:
```
# Running in around 200s

keras.backend.clear_session()

# Set Random seed to 0
initializer = tf.keras.initializers.GlorotUniform(seed=0)

# Build the final model
cnn = keras.Sequential([
        # Specify the input shape
        keras.Input(shape=(*IMAGE_SIZE, 1)),

        # Conv and pool block 1
        keras.layers.Conv2D(
            32,
            kernel_size=(3, 3),
            activation="relu",
            strides=(1, 1),
            kernel_initializer=initializer
            ),
        keras.layers.MaxPooling2D(pool_size=(2, 2), padding='same'), # padding␣
↪evenly

        # Conv and pool block 2
        keras.layers.Conv2D(
            64,
            kernel_size=(3, 3),
            activation="relu",
            strides=(1, 1),
            kernel_initializer=initializer
            ),
        keras.layers.MaxPooling2D(pool_size=(2, 2), padding='same'), # padding␣
↪evenly

        # Flatten and classify using dense output layer
        keras.layers.Flatten(),
        keras.layers.Dropout(0.5),
        keras.layers.Dense(len(class_names), activation="softmax",␣
↪kernel_initializer=initializer),
    ])
```

```
cnn.summary()

# Complie the model
cnn.compile(loss='sparse_categorical_crossentropy',
              optimizer=keras.optimizers.Adam(learning_rate=0.001),
              metrics=['accuracy'])

# Train the model
cnn_runtime = time.time()
cnn.fit(np.expand_dims(X_train, -1), y_train, epochs=12, validation_data=(np.
 ↪expand_dims(X_valid, -1), y_valid), verbose=1)
cnn_runtime = time.time() - cnn_runtime
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 11, 11, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 flatten (Flatten)           (None, 2304)              0

 dropout (Dropout)           (None, 2304)              0

 dense (Dense)               (None, 10)                23050

=================================================================
Total params: 41,866
Trainable params: 41,866
Non-trainable params: 0
_________________________________________________________________
Epoch 1/12
1688/1688 [==============================] - 16s 10ms/step - loss: 0.5201 -
accuracy: 0.8115 - val_loss: 0.3583 - val_accuracy: 0.8712
Epoch 2/12
1688/1688 [==============================] - 16s 9ms/step - loss: 0.3668 -
accuracy: 0.8700 - val_loss: 0.3091 - val_accuracy: 0.8873
Epoch 3/12
1688/1688 [==============================] - 16s 9ms/step - loss: 0.3272 -
accuracy: 0.8831 - val_loss: 0.2851 - val_accuracy: 0.8962
Epoch 4/12
```

```
1688/1688 [==============================] - 16s 9ms/step - loss: 0.3026 -
accuracy: 0.8908 - val_loss: 0.2628 - val_accuracy: 0.9043
Epoch 5/12
1688/1688 [==============================] - 16s 9ms/step - loss: 0.2882 -
accuracy: 0.8964 - val_loss: 0.2507 - val_accuracy: 0.9078
Epoch 6/12
1688/1688 [==============================] - 16s 9ms/step - loss: 0.2735 -
accuracy: 0.9010 - val_loss: 0.2696 - val_accuracy: 0.8988
Epoch 7/12
1688/1688 [==============================] - 16s 9ms/step - loss: 0.2611 -
accuracy: 0.9049 - val_loss: 0.2415 - val_accuracy: 0.9127
Epoch 8/12
1688/1688 [==============================] - 16s 9ms/step - loss: 0.2527 -
accuracy: 0.9085 - val_loss: 0.2334 - val_accuracy: 0.9147
Epoch 9/12
1688/1688 [==============================] - 16s 9ms/step - loss: 0.2463 -
accuracy: 0.9091 - val_loss: 0.2297 - val_accuracy: 0.9177
Epoch 10/12
1688/1688 [==============================] - 16s 9ms/step - loss: 0.2414 -
accuracy: 0.9126 - val_loss: 0.2257 - val_accuracy: 0.9175
Epoch 11/12
1688/1688 [==============================] - 16s 9ms/step - loss: 0.2351 -
accuracy: 0.9136 - val_loss: 0.2237 - val_accuracy: 0.9165
Epoch 12/12
1688/1688 [==============================] - 17s 10ms/step - loss: 0.2296 -
accuracy: 0.9164 - val_loss: 0.2211 - val_accuracy: 0.9190
```

[ ]:
```python
# Performance on test set.
loss, accuracy = cnn.evaluate(np.expand_dims(X_test, -1), y_test)
print(f"CNN training time: {cnn_runtime:.2f} s")
print(f"CNN score on the test set: {accuracy:.4f}")
```

```
313/313 [==============================] - 1s 3ms/step - loss: 0.2414 -
accuracy: 0.9126
CNN training time: 191.29 s
CNN score on the test set: 0.9126
```

Finalized at 10/12 20:24