Mnist_with_keras

June 15, 2025

[2]:
```
from keras.datasets import mnist

(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/mnist.npz
11490434/11490434             1s
0us/step
```

[3]:
```
X_train.shape , y_train.shape , X_test.shape , y_test.shape
```

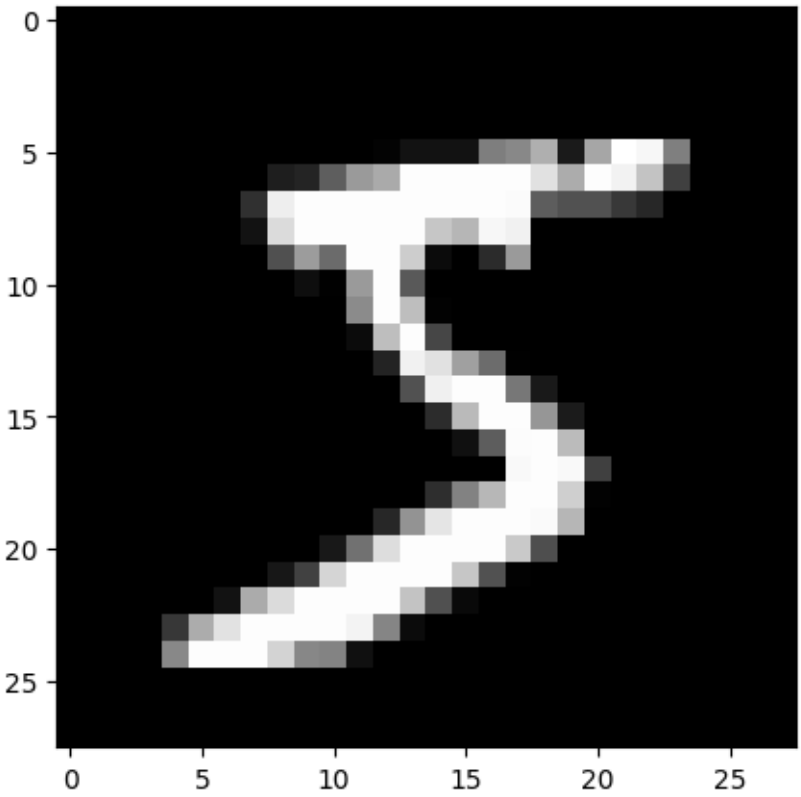
[3]:
```
((60000, 28, 28), (60000,), (10000, 28, 28), (10000,))
```

[4]:
```
X_train.min(), X_train.max(), X_test.min(), X_test.max()
```

[4]:
```
(np.uint8(0), np.uint8(255), np.uint8(0), np.uint8(255))
```

[5]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(5 ,5))
plt.imshow(X_train[0], cmap='gray')
```

[5]:
```
<matplotlib.image.AxesImage at 0x7c523f9cf210>
```



[6]:
```
X_train = X_train.reshape(-1 ,784) / 255
X_test = X_test.reshape(-1 , 784) / 255
```

[7]:
```
X_train.shape , y_train.shape , X_test.shape , y_test.shape
```

[7]: ((60000, 784), (60000,), (10000, 784), (10000,))

[8]:
```
X_train.min(), X_train.max(), X_test.min(), X_test.max()
```

[8]: (np.float64(0.0), np.float64(1.0), np.float64(0.0), np.float64(1.0))

[9]:
```
X_train[0]
```

[9]:
```
array([0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
```

```
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.01176471, 0.07058824, 0.07058824,
0.07058824, 0.49411765, 0.53333333, 0.68627451, 0.10196078,
0.65098039, 1.        , 0.96862745, 0.49803922, 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.11764706, 0.14117647, 0.36862745, 0.60392157,
0.66666667, 0.99215686, 0.99215686, 0.99215686, 0.99215686,
0.99215686, 0.88235294, 0.6745098 , 0.99215686, 0.94901961,
0.76470588, 0.25098039, 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.19215686, 0.93333333,
0.99215686, 0.99215686, 0.99215686, 0.99215686, 0.99215686,
0.99215686, 0.99215686, 0.99215686, 0.98431373, 0.36470588,
0.32156863, 0.32156863, 0.21960784, 0.15294118, 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.07058824, 0.85882353, 0.99215686, 0.99215686,
0.99215686, 0.99215686, 0.99215686, 0.77647059, 0.71372549,
0.96862745, 0.94509804, 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.31372549, 0.61176471, 0.41960784, 0.99215686, 0.99215686,
0.80392157, 0.04313725, 0.        , 0.16862745, 0.60392157,
0.        , 0.        , 0.        , 0.        , 0.        ,
```

```
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.05490196,
0.00392157, 0.60392157, 0.99215686, 0.35294118, 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.54509804,
0.99215686, 0.74509804, 0.00784314, 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.04313725, 0.74509804, 0.99215686,
0.2745098 , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.1372549 , 0.94509804, 0.88235294, 0.62745098,
0.42352941, 0.00392157, 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.31764706, 0.94117647, 0.99215686, 0.99215686, 0.46666667,
0.09803922, 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.17647059,
0.72941176, 0.99215686, 0.99215686, 0.58823529, 0.10588235,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.0627451 , 0.36470588,
0.98823529, 0.99215686, 0.73333333, 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.97647059, 0.99215686,
0.97647059, 0.25098039, 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
```

```
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.18039216, 0.50980392,
0.71764706, 0.99215686, 0.99215686, 0.81176471, 0.00784314,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.15294118,
0.58039216, 0.89803922, 0.99215686, 0.99215686, 0.99215686,
0.98039216, 0.71372549, 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.09411765, 0.44705882, 0.86666667, 0.99215686, 0.99215686,
0.99215686, 0.99215686, 0.78823529, 0.30588235, 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.09019608, 0.25882353, 0.83529412, 0.99215686,
0.99215686, 0.99215686, 0.99215686, 0.77647059, 0.31764706,
0.00784314, 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.07058824, 0.67058824, 0.85882353,
0.99215686, 0.99215686, 0.99215686, 0.99215686, 0.76470588,
0.31372549, 0.03529412, 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.21568627, 0.6745098 ,
0.88627451, 0.99215686, 0.99215686, 0.99215686, 0.99215686,
0.95686275, 0.52156863, 0.04313725, 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.53333333, 0.99215686, 0.99215686, 0.99215686,
0.83137255, 0.52941176, 0.51764706, 0.0627451 , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
```

```
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        ])
```

[10]:
```
y_train[0]
```

[10]:
```
np.uint8(5)
```

[11]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.utils import to_categorical
```

[12]:
```
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
```

[13]:
```
y_train.shape , y_test.shape
```

[13]:
```
((60000, 10), (10000, 10))
```

[14]:
```
y_train[0]
```

[14]:
```
array([0., 0., 0., 0., 0., 1., 0., 0., 0., 0.])
```

[15]:
```
model = Sequential([
    Dense(64, activation='relu', input_shape=(784,)),
    Dense(64, activation='relu'),
    Dense(10, activation='softmax')
])

model.compile(optimizer='Adam', loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])

model.summary()
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 64)	50,240
dense_1 (Dense)	(None, 64)	4,160
dense_2 (Dense)	(None, 10)	650

Total params: 55,050 (215.04 KB)

Trainable params: 55,050 (215.04 KB)

Non-trainable params: 0 (0.00 B)

1 Model Summary

First Hidden Layer * No. of inputs: 782 * N0. of neurons in hidden layer 1: 64 * Total No.of Parameters: 784* *italicized text*64(input-hidden1 weights) + 64(hidden1 biases)=4160

Second Hidden Layer * No. of inputs: 64 * No. of neurons in hidden layer 2: 64 * Total No. of Parameters: 64*64(hidden1-hidden2 weights) + 64(hidden2 biases)=4160

Output Layer * No. of inputs: 64 * No. of neurons in output layer: 10 * Total No. of Parameters: 64*10(hidden2-output weights) + 10(output biases)=650

Total No. of Parameters: 50240+4160+650= 55050

2 Alternative Model Building

1. Define then add layers with activation

```
model = Sequential()
model.add(Dense(64, activation='rule',input_shape=(784,)))
model.add(Dense(64, activation='rule'))
model.add(Dense(10,activation='softmax'))
```

2. Define then add layers without activation

```
from tensorflow.keras.layers import Activation
model = Sequential()
model.add(Danse(64, input_dim=784))
model.add(Activation('rule'))
model.add(Danse(64))
model.add(Activation('rule'))
```

```
model.add(Danse(10))
model.add(Activation('softmax'))
```

3. Functional API syntax

```
from tensorflow.kares.models import Model
from tensorflow.kares.models import input
inputs = input(shape=(784,))
x = Dense(64, activation='relu')(inputs)
x = Dense(64, activation='relu')(x)
predictions = Dense(10 , activations='softmax')(x)
model = Model(inputs=inputs,outputs=predictions)
```

[17]:
```
# Train the model
model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.3)
print('Training Completed \n')

# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test)
print(f'Test Loss: {loss:.4f}')
print(f'Test Accuracy: {accuracy * 100:.2f}%')
```

```
Epoch 1/10
1313/1313             6s 4ms/step -
accuracy: 0.9936 - loss: 0.0194 - val_accuracy: 0.9776 - val_loss: 0.0918
Epoch 2/10
1313/1313             11s 4ms/step -
accuracy: 0.9943 - loss: 0.0173 - val_accuracy: 0.9744 - val_loss: 0.1065
Epoch 3/10
1313/1313             11s 5ms/step -
accuracy: 0.9958 - loss: 0.0144 - val_accuracy: 0.9779 - val_loss: 0.0940
Epoch 4/10
1313/1313             10s 5ms/step -
accuracy: 0.9960 - loss: 0.0138 - val_accuracy: 0.9754 - val_loss: 0.1053
Epoch 5/10
1313/1313             9s 4ms/step -
accuracy: 0.9947 - loss: 0.0148 - val_accuracy: 0.9720 - val_loss: 0.1307
Epoch 6/10
1313/1313             6s 5ms/step -
accuracy: 0.9937 - loss: 0.0170 - val_accuracy: 0.9752 - val_loss: 0.1172
Epoch 7/10
1313/1313             5s 4ms/step -
accuracy: 0.9971 - loss: 0.0092 - val_accuracy: 0.9746 - val_loss: 0.1245
Epoch 8/10
1313/1313             10s 4ms/step -
accuracy: 0.9972 - loss: 0.0091 - val_accuracy: 0.9738 - val_loss: 0.1309
Epoch 9/10
1313/1313             5s 4ms/step -
accuracy: 0.9953 - loss: 0.0133 - val_accuracy: 0.9742 - val_loss: 0.1289
```

```
Epoch 10/10
1313/1313              9s 4ms/step -
accuracy: 0.9972 - loss: 0.0090 - val_accuracy: 0.9755 - val_loss: 0.1324
Training Completed

313/313             1s 2ms/step -
accuracy: 0.9691 - loss: 0.1569
Test Loss: 0.1310
Test Accuracy: 97.24%
```

[18]:
```
# Batch vs Epoch
batch_size = 32
training_samples = 60000
# Calculate the number of batches
batches_per_epoch = training_samples / batch_size
batches_per_epoch
```

[18]: 1875.0

3 Some Karas Options

[16]:
```
# Activation function
from tensorflow.keras.activation import relu, sigmoid, linear, softmax, tanh

# Optimizer
from tensorflow.keras.optimizers import SGD, RMSprop, Adam, Adadelta, Adagrad,␣
 ↪Adamax, Nadam, Ftrl

# loss function
from tensorflow.keras.losses import mean_squared_error, mean_absolute_error,␣
 ↪binary_crossentropy, categorical_crossentropy,␣
 ↪sparse_categorical_crossentropy, kullback_leibler_divergence, poisson,␣
 ↪cosine_similarity

# metrics
from tensorflow.keras.metrics import accuracy, binary_accuracy,␣
 ↪categorical_accuracy, sparse_categorical_accuracy
```

[21]:
```
def build_model(hidden_activation="relu",␣
 ↪output_activation='softmax',optimizer='Adam',␣
 ↪loss='categorical_crossentropy'):
  model = Sequential()
  model.add(Dense(64, input_shape=(784,), activation=hidden_activation))
  model.add(Dense(64, activation=hidden_activation))
  model.add(Dense(10, activation=output_activation))
  model.compile(optimizer=optimizer, loss=loss, metrics=['accuracy'])
  return model
```

[22]:
```
my_model = build_model()
my_model.summary()
```

```
Model: "sequential_2"
```

Layer (type)	Output Shape	Param #
dense_6 (Dense)	(None, 64)	50,240
dense_7 (Dense)	(None, 64)	4,160
dense_8 (Dense)	(None, 10)	650

```
Total params: 55,050 (215.04 KB)

Trainable params: 55,050 (215.04 KB)

Non-trainable params: 0 (0.00 B)
```

[29]:
```
history = my_model.fit(X_train, y_train, epochs=10, batch_size=32,␣
↪validation_split=0.3)
```

```
Epoch 1/10
1313/1313            7s 5ms/step -
accuracy: 0.9915 - loss: 0.0254 - val_accuracy: 0.9713 - val_loss: 0.1153
Epoch 2/10
1313/1313            9s 7ms/step -
accuracy: 0.9940 - loss: 0.0185 - val_accuracy: 0.9704 - val_loss: 0.1322
Epoch 3/10
1313/1313            5s 4ms/step -
accuracy: 0.9948 - loss: 0.0166 - val_accuracy: 0.9707 - val_loss: 0.1333
Epoch 4/10
1313/1313            6s 5ms/step -
accuracy: 0.9949 - loss: 0.0154 - val_accuracy: 0.9696 - val_loss: 0.1427
Epoch 5/10
1313/1313            5s 4ms/step -
accuracy: 0.9949 - loss: 0.0150 - val_accuracy: 0.9721 - val_loss: 0.1402
Epoch 6/10
1313/1313            6s 5ms/step -
accuracy: 0.9964 - loss: 0.0106 - val_accuracy: 0.9673 - val_loss: 0.1652
Epoch 7/10
1313/1313            9s 4ms/step -
accuracy: 0.9952 - loss: 0.0145 - val_accuracy: 0.9701 - val_loss: 0.1519
```

```
Epoch 8/10
1313/1313             6s 5ms/step -
accuracy: 0.9968 - loss: 0.0096 - val_accuracy: 0.9694 - val_loss: 0.1652
Epoch 9/10
1313/1313             11s 5ms/step -
accuracy: 0.9975 - loss: 0.0078 - val_accuracy: 0.9688 - val_loss: 0.1706
Epoch 10/10
1313/1313             10s 5ms/step -
accuracy: 0.9938 - loss: 0.0179 - val_accuracy: 0.9682 - val_loss: 0.1824
```

[25]:
```
batch_size = 32
training_samples = 0.8 * 60000
iterations_per_epoch = training_samples / batch_size
iterations_per_epoch
```

[25]:
```
1500.0
```

[30]:
```
history.params
```

[30]:
```
{'verbose': 'auto', 'epochs': 10, 'steps': 1313}
```

[31]:
```
history.history.keys()
```

[31]:
```
dict_keys(['accuracy', 'loss', 'val_accuracy', 'val_loss'])
```

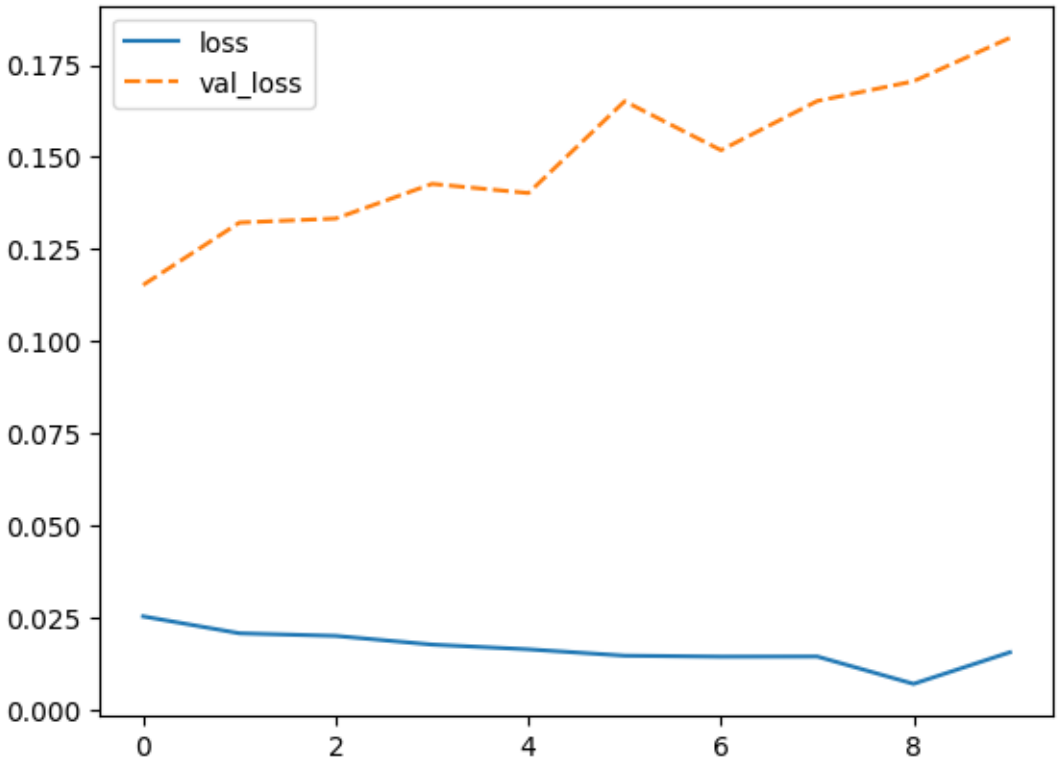
[32]:
```
import pandas as pd
history_df = pd.DataFrame(history.history)
history_df
```

[32]:

	accuracy	loss	val_accuracy	val_loss
0	0.991595	0.025370	0.971333	0.115310
1	0.992929	0.020782	0.970444	0.132237
2	0.993381	0.020074	0.970722	0.133330
3	0.994190	0.017717	0.969611	0.142737
4	0.994476	0.016430	0.972056	0.140246
5	0.995333	0.014707	0.967333	0.165164
6	0.994976	0.014446	0.970111	0.151878
7	0.995095	0.014481	0.969389	0.165226
8	0.997762	0.007059	0.968833	0.170586
9	0.994524	0.015622	0.968167	0.182351

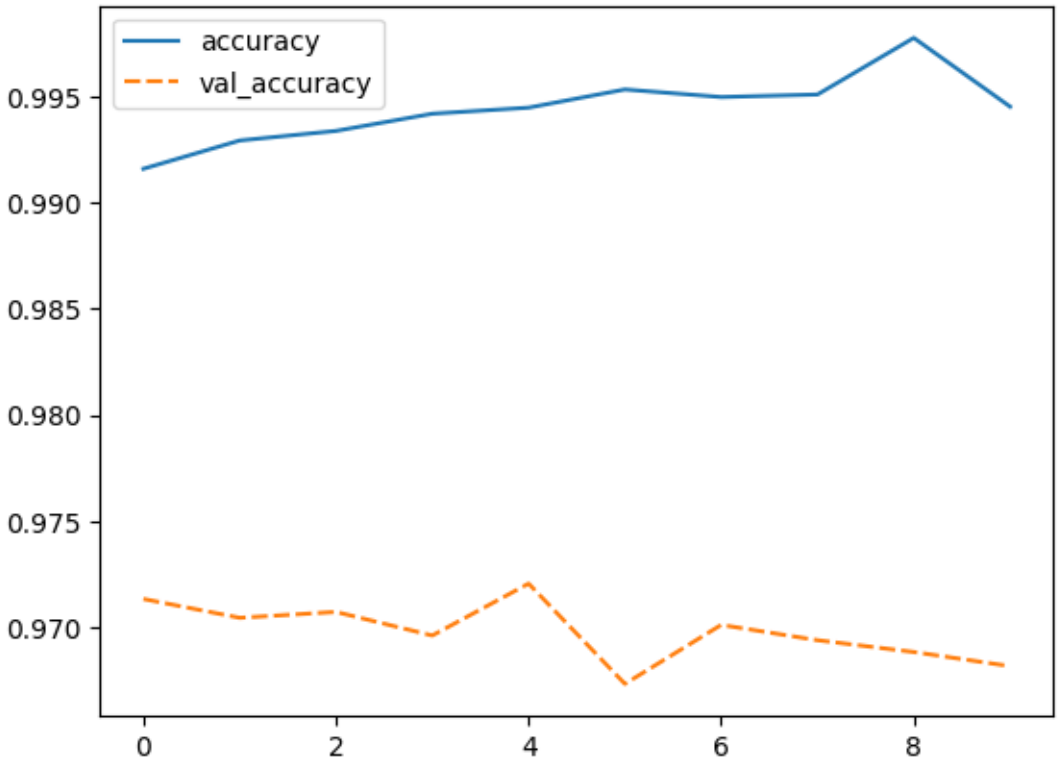
[33]:
```
# plot the loss
import seaborn as sns
sns.set_style()
sns.lineplot(data=history_df[['loss','val_loss']])
```

[33]:
```
<Axes: >
```



[34]:
```
# plot the accuracy
sns.lineplot(data=history_df[['accuracy','val_accuracy']])
```

[34]: <Axes: >


accuracy
val_accuracy
0.995
0.990
0.985
0.980
0.975
0.970
0
2
4
6
8

[ ]: