```
{
 "cells": [
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "### Import the necessary packages"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 17,
   "metadata": {},
   "outputs": [],
   "source": [
    "import numpy as np\n",
    "import pandas as pd\n",
    "import matplotlib.pyplot as plt\n",
    "from keras.utils import np_utils\n",
    "from tensorflow.keras.datasets import mnist\n",
    "from tensorflow.keras.models import Sequential\n",
    "from tensorflow.keras.layers import Conv2D, Dense, Flatten\n",
    "from tensorflow.keras.optimizers import Adam\n",
    "from tensorflow.keras.models import load_model\n",
    "from PIL import Image, ImageOps"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "### Load data"
   ]
  },
  {
```

```
{
 "cell_type": "code",
 "execution_count": 2,
 "metadata": {},
 "outputs": [],
 "source": [
  "(X_train, y_train), (X_test, y_test) = mnist.load_data()"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "### Data Analysis"
 ]
},
{
 "cell_type": "code",
 "execution_count": 3,
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "(60000, 28, 28)\n",
    "(10000, 28, 28)\n"
   ]
  }
 ],
 "source": [
  "print(X_train.shape)\n",
  "print(X_test.shape)"
 ]
```

},
{
"cell_type": "code",
"execution_count": 4,
"metadata": {},
"outputs": [
{
"data": {
"text/plain": [
"array([[0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"   0,   0],\n",
"   [0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"   0,   0],\n",
"   [0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"   0,   0],\n",
"   [0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"   0,   0],\n",
"   [0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"   0,   0],\n",
"   [0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   3,\n",
"   18, 18, 18, 126, 136, 175, 26, 166, 255, 247, 127,   0,   0,\n",
"   0,   0],\n",
"   [0,   0,   0,   0,   0,   0,   0,   0, 30, 36, 94, 154, 170,\n",
"   253, 253, 253, 253, 253, 225, 172, 253, 242, 195, 64,   0,   0,\n",
"   0,   0],\n",
"   [0,   0,   0,   0,   0,   0,   0, 49, 238, 253, 253, 253, 253,\n",
"   253, 253, 253, 253, 251, 93, 82, 82, 56, 39,   0,   0,   0,\n",
"   0,   0],\n",
"   [0,   0,   0,   0,   0,   0,   0, 18, 219, 253, 253, 253, 253,\n",

```
"      253, 198, 182, 247, 241,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"        0,   0],\n",
"     [  0,   0,   0,   0,   0,   0,   0,   0,  80, 156, 107, 253, 253,\n",
"      205,  11,   0,  43, 154,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"        0,   0],\n",
"     [  0,   0,   0,   0,   0,   0,   0,   0,   0,  14,   1, 154, 253,\n",
"       90,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"        0,   0],\n",
"     [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 139, 253,\n",
"      190,   2,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"        0,   0],\n",
"     [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  11, 190,\n",
"      253,  70,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"        0,   0],\n",
"     [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  35,\n",
"      241, 225, 160, 108,   1,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"        0,   0],\n",
"     [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"       81, 240, 253, 253, 119,  25,   0,   0,   0,   0,   0,   0,   0,\n",
"        0,   0],\n",
"     [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"        0,  45, 186, 253, 253, 150,  27,   0,   0,   0,   0,   0,   0,\n",
"        0,   0],\n",
"     [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"        0,   0,  16,  93, 252, 253, 187,   0,   0,   0,   0,   0,   0,\n",
"        0,   0],\n",
"     [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"        0,   0,   0,   0, 249, 253, 249,  64,   0,   0,   0,   0,   0,\n",
"        0,   0],\n",
"     [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"        0,  46, 130, 183, 253, 253, 207,   2,   0,   0,   0,   0,   0,\n",
"        0,   0],\n",
"     [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  39,\n",
"      148, 229, 253, 253, 253, 250, 182,   0,   0,   0,   0,   0,   0,\n",
```

"   0,   0],\n",
"   [0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 24, 114, 221,\n",
"   253, 253, 253, 253, 201, 78,   0,   0,   0,   0,   0,   0,   0,\n",
"   0,   0],\n",
"   [0,   0,   0,   0,   0,   0,   0,   0, 23, 66, 213, 253, 253,\n",
"   253, 253, 198, 81,   2,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"   0,   0],\n",
"   [0,   0,   0,   0,   0,   0, 18, 171, 219, 253, 253, 253, 253,\n",
"   195, 80,   9,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"   0,   0],\n",
"   [0,   0,   0,   0, 55, 172, 226, 253, 253, 253, 253, 244, 133,\n",
"   11,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"   0,   0],\n",
"   [0,   0,   0,   0, 136, 253, 253, 253, 212, 135, 132, 16,   0,\n",
"   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"   0,   0],\n",
"   [0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"   0,   0],\n",
"   [0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"   0,   0],\n",
"   [0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"   0,   0]], dtype=uint8)"
 ]
},
"execution_count": 4,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"X_train[0]"

```
 ]
},
{
 "cell_type": "code",
 "execution_count": 5,
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "5"
    ]
   },
   "execution_count": 5,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "y_train[0]"
 ]
},
{
 "cell_type": "code",
 "execution_count": 6,
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "<matplotlib.image.AxesImage at 0x1ed310b5930>"
    ]
   },
   "execution_count": 6,
```

"metadata": {},
"output_type": "execute_result"
},
{
"data": {
"image/png":
"iVBORw0KGgoAAAANSUhEUgAAAPsAAAD4CAYAAAAq5pAIAAAAOXRFWHRTb2Z0d2FyZQBNYXRwbG90bGliIHZlc
nNpb24zLjUuMiwgaHR0cHM6Ly9tYXRwbG90bGliLm9yZy8qNh9FAAAACXBIWXMAAAsTAAALEwEAmpwYAAAQtkl
EQVR4nO3df0zUaX4H8DcCt1dkFVxw5g4sUE+6rLst1B3qBd1lU6uSXAKmXVe9RHprR7srNba0Jq9BJJ6RPesibP
jiYmc5x5HoE2p42KyIXaXG/eGXy7CcguUXzOgRMS9vcqPp3+YZc+VeQZnvvNDPu9XYiLz5sv3k9l9+x3mmZknCoACES1
7K8I9ABGFBstOJATLTiQEy04kBMtOJERMKE92Z3wKnsGJUJ6SSBRTWjIS1q5eNAuo7Dt27MCJEycQHR2Nd955B1VV
Vdrv9wxO4HBueSCnJCKNU7+o9Jr5/TB+xYoVOHXqFAoKCvDcc89h7969yMrK8vfHEVGQ+V323Nxc9PX1obh+/Mz
M7h48SIKCwuNnI2IDOR32VNSUjA0NVNSUjA0NLTw9fDwMFJSUh75PqvqVCqfTCafTidxN3A326
YjIC7/LPjIygnXr1i18nZqaipGREUOGiIiLj+V12p9OJDRs2ID09HbGxsdizZw8aGxuNnI2IDOT30tvtc3BxKSkpw+fJlRlEdH4+
zZs/j444+NnI2IDBTQOntTUxOampmMmoWIgogvlyUSgmUnEoJlxKCZScSgmUnEoJlJxKCZScSgmUnEoJlJxKCZScSgmUnEoJlJxKCZScSgmUnEoJlJxKCZScSgmUnEoJlJxKCZScSgm
UnEoJlJxKCZScSgmUnEoJlJxKCZScSgmUnEoJlJxKCZScSgmUnEoJlJxKCZScSgmUnEoJlJxKCZScSgmUnEoJlJxKCZScSgmUnEoJlJxKCZScSgmUnEoJlJxKCZScSgm
UnEiKgXVwp8kXF6P8TRycnBfX8PX+X7jWbi5vXHpu2flybx70Zpc3dP/ya1+yXL/5Ue+ytuc+0+R+/W6rNv/W3H2rzcAio
7P39/Ziensbc3BxmZ2dhsViMmouIDBbwlf2VV17B7du3jZiFiIKIv7MTCRFQ2ZVScDgcuH79OqxW6+6xHdZ5fwAJcjrFQlI9j
sTp5VSCnI6IABPQwfsuWLRgdHUVycjKuXLmCmzdvoqWWl5aHvsdlssNlsAIAeZ18gpyOiAAR0ZR8dHQUATExMoL6+Hr
m5uYYMRUTG87vscXFxixI+PX/j79u3b0dXdXZdhgRGQsvx/Gm0wm1NfXP/ghMTGora3F5cuXDRtsOYnO8qqDN1VOx2n
z05QRt/vlm72vCa1br14tb/lC/3hxOTb9+WptX/WinNm99odZr1j/zufbYSs+favNvtihtHon8Lnt/fz+ys7MNHIWIgolLb0
RCsOxEQrDsREKw7ERCsOxEQvAtrgaYy/8jbf7Dc6e0eWas97diLmczak6b/TI6b5Sr9nTI7PaY5+6pV+a
i7veqs0jEa/sREKw7ERCsOxEQrDsREKw7ERCsOxEQrDsREJwnd0AT/WMn4i+fW/P/5Amyf/xT9gGeowcd2duf
79z7xmU/P6dXLTyf/R5sH05L2B1Tde2YmEYNmJhGDZiYRg2YmEYNmJhGDZiYRg2YmE4Dq7wsDq7AWbH3Nr87apMXtfkPduo
/7jm6I16bt7/5tjbXeevWH2jzvm1/4uMpDEeevWH2jzvm1x2nzuzpg23/ftN71mA0e0hyID7fpvoMfCK
zueqs0jEa/sREKw7ERCsOxEQrDsREKw7ERCsOxEQrDsREJwnd0AT/WMavOPfrNOm2fGeowcx1ClY5u1+af39FfW
79z7xmU/P6dXLTyf/R5sH05L2B1Tde2YmEYNmJhGDZiYRg2YmEYNmJhGDZiYRg2YmE4Dq7AWbH3Nr87apXtfkPduo
/7jm6I16bt7/5tjbXeevWH2jzvm1x2nzuzpg23/ftN71mA0e0hyID7fpvoMfCKzuRECw7kRAsO5EQLDuREKw7ERCsOxE
QLDuREFxnD4E1P/5Amyf/xzPafO72pDbf+PzrXrMbL53VH1pYP5vIYP5vLLb7XZ4PB50dnYu
3JaYmAiHw4Hk2oGkTpY1JcNdS4QMhZmTcZSwsBoJjPbYCdlmYwSq/4vTKNigu7ZrObPk9n1HYq/P9eOO/dwz/ZqmFPAYBpBSmeV0k9HQ/NduRGq3VbMiYLqccdX2NSaqsm9Kr0vnH4D7tS5rEBIT+K3CHfKCqoMORQRQk/nQn+0ZH4RLMDUP4RP2BFVlFP...

B9bfP9vJrX5sW/9l9ds++985tdMRvHMfe41e6mxVHvss/90U5uvuaNfJ5/XphRJeGUnEoJlJxKCZScSgmUnEoJlJxKCZSc
SgmUnEkLMOvtAkf7ftd4X3g3auU/dWa/NT7y/XZtHzS3+6b5fePatfq/ZBk+r9tg5bUrLCa/sREKw7ERCsOxEQrDsREKw
7ERCsOxEQrDsREL4XGdPTU3F+fPnYTKZoJTCmTNncPLkSVRUVMBqtWJiYgLAg22cm5qagj6wvzLf0O8F/p03NoVokk
dlQj+bL1wrp6XwWfbZ2VmUlpbC5XIhPj4eH330Ea5cuQIAqK6uxvHjx4M+JBEFzmfZ3W433G43AODevXvo7u5GSkpK
0AcjImM91u/saWlpyMnJQWvrg5dglpSUoL29HXa7HQkJCYseY7Va4XQ64XQ6sTp5VcADE5F/llz2lStXoq6uDkePHsX0
9DROnz6N9evXIzs7G2NjY14fzttsNlgsFlgsFkxN3DVscCJ6PEsqe0xMDOrq6nDhwgXU19cDAMbHxzE/Pw+lFGw2G3Jzc
4M6KBEFZkllt9vt6O7uRnV19cJtZvOXu5ru2rULXV363TyJKLx8PkGXl5eH/fv3o6OjAy6XC8CDZba9e/ciOzsbSikMDAzg
0KFDQR+WiPzns+zXrl1DVNSj76eO5DV1InoUX0FHJATLTiQEy04kBMtOJATLTiQEy04kBMtOJATLTiQEy04kBMtOJATL
TiQEy04kBMtOJATLTiREFAAVqpONj49jcHBw4eukpCTcunUrVKd/LJE6W6TOBXA2fxk5W1paQ6w3
buJ3W2SJ2Ls0X+bHwYTyQEy04kRFjLfubMmXCeXitSZ4vUuQDO5q9QzRbSJ+iIKHz4MJ5ICJadSIiwlH3Hjh24efMmPv
nkE5SVlYVjBK/6+/sXPiPf6XSGdRa73Q6Px4POzs6F2xITE+FwONDb2wuHw+F1j71wzFZRUYHh4WG4XC64UFBSE
ZbbU1FRcvXoVN27cQFdXF44cOQIg/Pedt7lCeb+FdE1xxYoVqq+vT2VkZKjY2FjV1tamsrKywr7W+cWf/v5+9cwzz4R9
DgBq69atKicnR3V2di7cVlVVpcrKyhQAVVZWpiorKyNmtoqKClVaWhr2+81utXMXHx6uenh6VlZUV9vvO21
yhut9CfmXPzc1FX18f+vv7MTMzg4sXL6KwsDDUYzwRWlpaMDk5+dBthYWFqKmpAQDU1NSgqKgoDJMtPlukcLvdC7
sX/fY24+G+77zNFSohL3tKSgqGhoYWvh4eHo6o67srpdDQ0IDr16/j4MGD4R7nEcnJyXC73QAe/PCYzeYwT/SwkpIStLe3
w7du4evU1FSMjIyEegyvRkdHAQATExOr6+PuK2oPR7Pwg66ZrMZ4+PjYZ7oS5G0jfdi24xHwn0Xzu3PQ152p9OJDRs2I
D09HbGxsdizZw8aGxtDPcai4uLiEB8fv/D37du3o6urK8xTPayDmdUaGbjJvxQIIw6hIcwTfSmStvFebJvxSLjvwr39ecifLS0oK
FA9PT2qr69PHTt2LOzP3n7xJyMjQ7W1tam2tjbV1dUV9tlqa2vV6Oioun//vhoaGlLFxcXq6aefVlevv/vhoaGlKvv/66WrNmjXrvvfdUb2+vunLlikp
MTIyY2c6fP686OjpUe3u7amhoUGazOSyz5eXlKaWUam9vVy6XS7lcLLVQUBD2+87bXKG63/hyWSIh+AQdkRAsO5EQLDuRECw7kRAsO5EQLDuRECw7kRD/D2hlIVvotzDnAAAAAElFTkSuQmCC",

"text/plain": [

"<Figure size 432x288 with 1 Axes>"

]

},

"metadata": {},

"output_type": "display_data"

}

],

"source": [

"plt.imshow(X_train[0])"

]

},

{

"cell_type": "markdown",

"metadata": {},

"source": [

"### Data Pre-Processing"

]

},

{

```
 "cell_type": "code",
 "execution_count": 7,
 "metadata": {},
 "outputs": [],
 "source": [
  "X_train = X_train.reshape(60000, 28, 28, 1).astype('float32')\n",
  "X_test = X_test.reshape(10000, 28, 28, 1).astype('float32')"
 ]
},
{
 "cell_type": "code",
 "execution_count": 8,
 "metadata": {},
 "outputs": [],
 "source": [
  "number_of_classes = 10\n",
  "Y_train = np_utils.to_categorical(y_train, number_of_classes)\n",
  "Y_test = np_utils.to_categorical(y_test, number_of_classes)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 9,
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "array([0., 0., 0., 0., 0., 1., 0., 0., 0., 0.], dtype=float32)"
    ]
   },
   "execution_count": 9,
   "metadata": {},
   "output_type": "execute_result"
```

```
 }
],
"source": [
 "Y_train[0]"
]
},
{
"cell_type": "markdown",
"metadata": {},
"source": [
 "### Create model"
]
},
{
"cell_type": "code",
"execution_count": 10,
"metadata": {},
"outputs": [],
"source": [
 "model = Sequential()\n",
 "model.add(Conv2D(64, (3, 3), input_shape=(28, 28, 1), activation=\"relu\"))\n",
 "model.add(Conv2D(32, (3, 3), activation=\"relu\"))\n",
 "model.add(Flatten())\n",
 "model.add(Dense(number_of_classes, activation=\"softmax\"))"
]
},
{
"cell_type": "code",
"execution_count": 11,
"metadata": {},
"outputs": [],
"source": [
 "model.compile(loss='categorical_crossentropy', optimizer=\"Adam\", metrics=[\"accuracy\"])"
]
```

```
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "### Train the model"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 12,
  "metadata": {},
  "outputs": [
   {
    "name": "stdout",
    "output_type": "stream",
    "text": [
     "Epoch 1/5\n",
     "1875/1875 [==============================] - 16s 5ms/step - loss: 0.2158 - accuracy: 0.9518 - val_loss: 0.0964 - val_accuracy: 0.9707\n",
     "Epoch 2/5\n",
     "1875/1875 [==============================] - 9s 5ms/step - loss: 0.0682 - accuracy: 0.9794 - val_loss: 0.0674 - val_accuracy: 0.9805\n",
     "Epoch 3/5\n",
     "1875/1875 [==============================] - 9s 5ms/step - loss: 0.0478 - accuracy: 0.9844 - val_loss: 0.0852 - val_accuracy: 0.9759\n",
     "Epoch 4/5\n",
     "1875/1875 [==============================] - 9s 5ms/step - loss: 0.0336 - accuracy: 0.9893 - val_loss: 0.1202 - val_accuracy: 0.9719\n",
     "Epoch 5/5\n",
     "1875/1875 [==============================] - 9s 5ms/step - loss: 0.0270 - accuracy: 0.9914 - val_loss: 0.1036 - val_accuracy: 0.9777\n"
    ]
   },
   {
    "data": {
```

```
   "text/plain": [
    "<keras.callbacks.History at 0x1ed3324f7f0>"
   ]
  },
  "execution_count": 12,
  "metadata": {},
  "output_type": "execute_result"
 }
],
"source": [
 "model.fit(X_train, Y_train, batch_size=32, epochs=5, validation_data=(X_test,Y_test))"
]
},
{
"cell_type": "markdown",
"metadata": {},
"source": [
 "### Test the model"
]
},
{
"cell_type": "code",
"execution_count": 13,
"metadata": {},
"outputs": [
 {
  "name": "stdout",
  "output_type": "stream",
  "text": [
   "Metrics (Test Loss & Test Accuracy): \n",
   "[0.1035672277212143, 0.9776999950408936]\n"
  ]
 }
],
```

```
 "source": [
  "metrics = model.evaluate(X_test, Y_test, verbose=0)\n",
  "print(\"Metrics (Test Loss & Test Accuracy): \")\n",
  "print(metrics)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 14,
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "1/1 [==============================] - 0s 177ms/step\n",
    "[[6.43197941e-15 8.71634543e-21 7.98728167e-11 7.08215517e-12\n",
    "  2.27718335e-18 1.36703092e-15 2.37176042e-22 1.00000000e+00\n",
    "  4.51405352e-13 4.25453591e-13]\n",
    " [4.56659687e-15 1.54588287e-10 1.00000000e+00 1.20107971e-13\n",
    "  1.86926159e-19 3.90255250e-20 1.16102319e-11 4.27834925e-23\n",
    "  7.33884963e-17 1.86307852e-23]\n",
    " [1.37352282e-10 9.99961138e-01 3.40877750e-06 1.50240779e-12\n",
    "  1.99599867e-07 1.10004057e-05 6.72304851e-11 7.78906983e-09\n",
    "  2.42337919e-05 3.74607870e-13]\n",
    " [1.00000000e+00 5.39840355e-16 1.03082355e-10 4.23198737e-17\n",
    "  8.17481194e-10 2.49619574e-12 1.66041558e-09 5.06253395e-17\n",
    "  3.02219919e-13 5.55243709e-08]]\n"
   ]
  }
 ],
 "source": [
  "prediction = model.predict(X_test[:4])\n",
  "print(prediction)"
```

```
 ]
},
{
 "cell_type": "code",
 "execution_count": 15,
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "[7 2 1 0]\n",
    "[[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]\n",
    " [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]\n",
    " [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]\n",
    " [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]\n"
   ]
  }
 ],
 "source": [
  "print(numpy.argmax(prediction, axis=1))\n",
  "print(Y_test[:4])"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "### Save the model"
 ]
},
{
 "cell_type": "code",
 "execution_count": 16,
```

```
 "metadata": {},
 "outputs": [],
 "source": [
  "model.save(\"model.h5\")"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "### Test the saved model"
 ]
},
{
 "cell_type": "code",
 "execution_count": 22,
 "metadata": {},
 "outputs": [],
 "source": [
  "model=load_model(\"model.h5\")"
 ]
},
{
 "cell_type": "code",
 "execution_count": 23,
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "1/1 [==============================] - 0s 435ms/step\n",
    "0    8\n",
    "Name: Label, dtype: int64\n"
```

```
   ]
   }
  ],
  "source": [
   "img = Image.open(\"sample.png\").convert(\"L\")\n",
   "img = img.resize((28, 28))\n",
   "img2arr = np.array(img)\n",
   "img2arr = img2arr.reshape(1, 28, 28, 1)\n",
   "results  = model.predict(img2arr)\n",
   "results = np.argmax(results,axis = 1)\n",
   "results = pd.Series(results,name=\"Label\")\n",
   "print(results)"
  ]
 }
],
"metadata": {
 "kernelspec": {
  "display_name": "Python 3.10.8 ('venv': venv)",
  "language": "python",
  "name": "python3"
 },
 "language_info": {
  "codemirror_mode": {
   "name": "ipython",
   "version": 3
  },
  "file_extension": ".py",
  "mimetype": "text/x-python",
  "name": "python",
  "nbconvert_exporter": "python",
  "pygments_lexer": "ipython3",
  "version": "3.10.8"
 },
 "orig_nbformat": 4,
```

```
  "vscode": {
   "interpreter": {
    "hash": "72cf82f53b15019b5b640600623df8bcf4d62c2c60fee1ea51c8c07b395bb5c2"
   }
  }
 },
 "nbformat": 4,
 "nbformat_minor": 2
}
```