{

"cells": [

{

"cell\_type": "code",

"execution\_count": 37,

"metadata": {

"scrolled": true

},

"outputs": [],

"source": [

"from keras.datasets import mnist\n",

"import pandas as pd\n",

"import numpy as np\n",

"from keras.models import Sequential\n",

"from keras.layers import Dense,Activation,Dropout,Conv2D,MaxPooling2D,Flatten\n",

"from keras.utils import np\_utils\n",

"from sklearn.model\_selection import train\_test\_split\n",

"(X\_train, y\_train), (X\_test, y\_test) = mnist.load\_data()\n"

]

},

{

"cell\_type": "code",

"execution\_count": 3,

"metadata": {

"scrolled": true

},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"[[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0\n",

" 0 0 0 0 0 0 0 0 0 0]\n",

" [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0\n",

" 0 0 0 0 0 0 0 0 0 0]\n",

" [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0\n",

" 0 0 0 0 0 0 0 0 0 0]\n",

" [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0\n",

" 0 0 0 0 0 0 0 0 0 0]\n",

" [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0\n",

" 0 0 0 0 0 0 0 0 0 0]\n",

" [ 0 0 0 0 0 0 0 0 0 0 0 0 3 18 18 18 126 136\n",

" 175 26 166 255 247 127 0 0 0 0]\n",

" [ 0 0 0 0 0 0 0 0 30 36 94 154 170 253 253 253 253 253\n",

" 225 172 253 242 195 64 0 0 0 0]\n",

" [ 0 0 0 0 0 0 0 49 238 253 253 253 253 253 253 253 253 251\n",

" 93 82 82 56 39 0 0 0 0 0]\n",

" [ 0 0 0 0 0 0 0 18 219 253 253 253 253 253 198 182 247 241\n",

" 0 0 0 0 0 0 0 0 0 0]\n",

" [ 0 0 0 0 0 0 0 0 80 156 107 253 253 205 11 0 43 154\n",

" 0 0 0 0 0 0 0 0 0 0]\n",

" [ 0 0 0 0 0 0 0 0 0 14 1 154 253 90 0 0 0 0\n",

" 0 0 0 0 0 0 0 0 0 0]\n",

" [ 0 0 0 0 0 0 0 0 0 0 0 139 253 190 2 0 0 0\n",

" 0 0 0 0 0 0 0 0 0 0]\n",

" [ 0 0 0 0 0 0 0 0 0 0 0 11 190 253 70 0 0 0\n",

" 0 0 0 0 0 0 0 0 0 0]\n",

" [ 0 0 0 0 0 0 0 0 0 0 0 0 35 241 225 160 108 1\n",

" 0 0 0 0 0 0 0 0 0 0]\n",

" [ 0 0 0 0 0 0 0 0 0 0 0 0 0 81 240 253 253 119\n",

" 25 0 0 0 0 0 0 0 0 0]\n",

" [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 45 186 253 253\n",

" 150 27 0 0 0 0 0 0 0 0]\n",

" [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 16 93 252\n",

" 253 187 0 0 0 0 0 0 0 0]\n",

" [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 249\n",

" 253 249 64 0 0 0 0 0 0 0]\n",

" [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 46 130 183 253\n",

" 253 207 2 0 0 0 0 0 0 0]\n",

" [ 0 0 0 0 0 0 0 0 0 0 0 0 39 148 229 253 253 253\n",

" 250 182 0 0 0 0 0 0 0 0]\n",

" [ 0 0 0 0 0 0 0 0 0 0 24 114 221 253 253 253 253 201\n",

" 78 0 0 0 0 0 0 0 0 0]\n",

" [ 0 0 0 0 0 0 0 0 23 66 213 253 253 253 253 198 81 2\n",

" 0 0 0 0 0 0 0 0 0 0]\n",

" [ 0 0 0 0 0 0 18 171 219 253 253 253 253 195 80 9 0 0\n",

" 0 0 0 0 0 0 0 0 0 0]\n",

" [ 0 0 0 0 55 172 226 253 253 253 253 244 133 11 0 0 0 0\n",

" 0 0 0 0 0 0 0 0 0 0]\n",

" [ 0 0 0 0 136 253 253 253 212 135 132 16 0 0 0 0 0 0\n",

" 0 0 0 0 0 0 0 0 0 0]\n",

" [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0\n",

" 0 0 0 0 0 0 0 0 0 0]\n",

" [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0\n",

" 0 0 0 0 0 0 0 0 0 0]\n",

" [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0\n",

" 0 0 0 0 0 0 0 0 0 0]]\n",

"5\n"

]

}

],

"source": [

"print(X\_train[0])\n",

"print(Y\_train[0])\n"

]

},

{

"cell\_type": "code",

"execution\_count": 4,

"metadata": {

"scrolled": true

},

"outputs": [

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 432x288 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"import matplotlib.pyplot as plt\n",

"\n",

"plt.imshow(X\_train[0], cmap=\"gray\")\n",

"plt.title(\"Label: \" + str(Y\_train[0]))\n",

"plt.axis(\"off\")\n",

"plt.show()\n"

]

},

{

"cell\_type": "code",

"execution\_count": 5,

"metadata": {},

"outputs": [

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 432x288 with 9 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"sub\_plot= 330\n",

"for i in range(0, 9):\n",

" ax = plt.subplot(sub\_plot+i+1)\n",

" ax.imshow(X\_train[i], cmap=\"gray\")\n",

" ax.set\_title(\"Label: \" + str(Y\_train[i]))\n",

" ax.axis(\"off\")\n",

"\n",

"plt.subplots\_adjust(hspace = .5)\n",

"plt.show()\n"

]

},

{

"cell\_type": "code",

"execution\_count": 45,

"metadata": {},

"outputs": [],

"source": [

"X\_train = X\_train.reshape(X\_train.shape[0],28\*28) / 255\n",

"X\_test = X\_test.reshape(X\_test.shape[0],28\*28) / 255\n",

"y\_train\_onehot = np\_utils.to\_categorical(y\_train, num\_classes=10)\n",

"y\_test\_onehot = np\_utils.to\_categorical(y\_test, num\_classes=10)"

]

},

{

"cell\_type": "code",

"execution\_count": 48,

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"(60000, 784)"

]

},

"execution\_count": 48,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"X\_train.shape"

]

},

{

"cell\_type": "code",

"execution\_count": 10,

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"Model: \"sequential\_3\"\n",

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n",

"Layer (type) Output Shape Param # \n",

"=================================================================\n",

"dense\_5 (Dense) (None, 256) 200960 \n",

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n",

"dropout\_2 (Dropout) (None, 256) 0 \n",

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n",

"dense\_6 (Dense) (None, 256) 65792 \n",

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n",

"dense\_7 (Dense) (None, 10) 2570 \n",

"=================================================================\n",

"Total params: 269,322\n",

"Trainable params: 269,322\n",

"Non-trainable params: 0\n",

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n"

]

}

],

"source": [

"model = Sequential()\n",

"model.add(Dense(256, input\_dim=28\*28, activation=\"relu\"))\n",

"model.add(Dropout(0.5))\n",

"model.add(Dense(256, activation=\"relu\"))\n",

"model.add(Dense(10, activation=\"softmax\"))\n",

"model.summary() #顯示摘要資訊"

]

},

{

"cell\_type": "code",

"execution\_count": 22,

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"Epoch 1/10\n",

"60000/60000 [==============================] - 37s 612us/step - loss: 0.1412 - accuracy: 0.9670\n",

"Epoch 2/10\n",

"60000/60000 [==============================] - 36s 602us/step - loss: 0.1323 - accuracy: 0.9678\n",

"Epoch 3/10\n",

"60000/60000 [==============================] - 36s 598us/step - loss: 0.1328 - accuracy: 0.9680\n",

"Epoch 4/10\n",

"60000/60000 [==============================] - 36s 597us/step - loss: 0.1271 - accuracy: 0.9694\n",

"Epoch 5/10\n",

"60000/60000 [==============================] - 35s 589us/step - loss: 0.1250 - accuracy: 0.9703\n",

"Epoch 6/10\n",

"60000/60000 [==============================] - 36s 606us/step - loss: 0.1234 - accuracy: 0.9700\n",

"Epoch 7/10\n",

"60000/60000 [==============================] - 36s 592us/step - loss: 0.1246 - accuracy: 0.9692\n",

"Epoch 8/10\n",

"60000/60000 [==============================] - 38s 636us/step - loss: 0.1238 - accuracy: 0.9691\n",

"Epoch 9/10\n",

"60000/60000 [==============================] - 35s 578us/step - loss: 0.1195 - accuracy: 0.9710\n",

"Epoch 10/10\n",

"60000/60000 [==============================] - 36s 602us/step - loss: 0.1197 - accuracy: 0.9713\n"

]

}

],

"source": [

"model.compile(loss=\"categorical\_crossentropy\", optimizer=\"adam\", metrics=[\"accuracy\"])\n",

"history=model.fit(X\_train,y\_train\_onehot, epochs=10, batch\_size=5)"

]

},

{

"cell\_type": "code",

"execution\_count": 24,

"metadata": {},

"outputs": [

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 432x288 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"import matplotlib.pyplot as plt\n",

"loss = history.history[\"loss\"]\n",

"epochs = range(1, len(loss)+1)\n",

"val\_loss = history.history[\"loss\"]\n",

"plt.plot(epochs, loss, \"bo\", label=\"Training Loss\")\n",

"plt.plot(epochs, val\_loss, \"r\", label=\"Validation Loss\")\n",

"plt.title(\"Training and Validation Loss\")\n",

"plt.xlabel(\"Epochs\")\n",

"plt.ylabel(\"Loss\")\n",

"plt.legend()\n",

"plt.show()"

]

},

{

"cell\_type": "code",

"execution\_count": 27,

"metadata": {

"scrolled": true

},

"outputs": [

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 432x288 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"acc = history.history[\"accuracy\"]\n",

"epochs = range(1, len(acc)+1)\n",

"val\_acc = history.history[\"accuracy\"]\n",

"plt.plot(epochs, acc, \"b-\", label=\"Training Acc\")\n",

"plt.plot(epochs, val\_acc, \"r--\", label=\"Validation Acc\")\n",

"plt.title(\"Training and Validation Accuracy\")\n",

"plt.xlabel(\"Epochs\")\n",

"plt.ylabel(\"Accuracy\")\n",

"plt.legend()\n",

"plt.show()"

]

},

{

"cell\_type": "code",

"execution\_count": 49,

"metadata": {},

"outputs": [],

"source": [

"(X\_train, y\_train), (X\_test, y\_test) = mnist.load\_data()\n",

"X\_train = X\_train.reshape(X\_train.shape[0],28,28,1)\n",

"X\_test = X\_test.reshape(X\_test.shape[0],28,28,1)\n",

"y\_train\_onehot = np\_utils.to\_categorical(y\_train, num\_classes=10)\n",

"y\_test\_onehot = np\_utils.to\_categorical(y\_test, num\_classes=10)"

]

},

{

"cell\_type": "code",

"execution\_count": 50,

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"(60000, 28, 28, 1)"

]

},

"execution\_count": 50,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"X\_train.shape"

]

},

{

"cell\_type": "code",

"execution\_count": 51,

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"Model: \"sequential\_11\"\n",

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n",

"Layer (type) Output Shape Param # \n",

"=================================================================\n",

"conv2d\_10 (Conv2D) (None, 28, 28, 16) 416 \n",

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n",

"max\_pooling2d\_7 (MaxPooling2 (None, 14, 14, 16) 0 \n",

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n",

"conv2d\_11 (Conv2D) (None, 14, 14, 32) 12832 \n",

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n",

"max\_pooling2d\_8 (MaxPooling2 (None, 7, 7, 32) 0 \n",

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n",

"dropout\_8 (Dropout) (None, 7, 7, 32) 0 \n",

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n",

"flatten\_3 (Flatten) (None, 1568) 0 \n",

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n",

"dense\_12 (Dense) (None, 128) 200832 \n",

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n",

"dropout\_9 (Dropout) (None, 128) 0 \n",

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n",

"dense\_13 (Dense) (None, 10) 1290 \n",

"=================================================================\n",

"Total params: 215,370\n",

"Trainable params: 215,370\n",

"Non-trainable params: 0\n",

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n"

]

}

],

"source": [

"model = Sequential()\n",

"model.add(Conv2D(16, kernel\_size=(5, 5), padding=\"same\", input\_shape=(28, 28, 1), activation=\"relu\"))\n",

"model.add(MaxPooling2D(pool\_size=(2, 2)))\n",

"model.add(Conv2D(32, kernel\_size=(5, 5), padding=\"same\", activation=\"relu\"))\n",

"model.add(MaxPooling2D(pool\_size=(2, 2)))\n",

"model.add(Dropout(0.5))\n",

"model.add(Flatten())\n",

"model.add(Dense(128, activation=\"relu\"))\n",

"model.add(Dropout(0.5))\n",

"model.add(Dense(10, activation=\"softmax\"))\n",

"model.summary() #顯示摘要資訊"

]

},

{

"cell\_type": "code",

"execution\_count": 52,

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"Epoch 1/10\n",

"60000/60000 [==============================] - 86s 1ms/step - loss: 0.6161 - accuracy: 0.8523\n",

"Epoch 2/10\n",

"60000/60000 [==============================] - 87s 1ms/step - loss: 0.3004 - accuracy: 0.9224\n",

"Epoch 3/10\n",

"60000/60000 [==============================] - 81s 1ms/step - loss: 0.2721 - accuracy: 0.9306\n",

"Epoch 4/10\n",

"60000/60000 [==============================] - 81s 1ms/step - loss: 0.2613 - accuracy: 0.9337\n",

"Epoch 5/10\n",

"60000/60000 [==============================] - 86s 1ms/step - loss: 0.2602 - accuracy: 0.9347\n",

"Epoch 6/10\n",

"60000/60000 [==============================] - 86s 1ms/step - loss: 0.2598 - accuracy: 0.9359\n",

"Epoch 7/10\n",

"60000/60000 [==============================] - 87s 1ms/step - loss: 0.2659 - accuracy: 0.9383\n",

"Epoch 8/10\n",

"60000/60000 [==============================] - 84s 1ms/step - loss: 0.2787 - accuracy: 0.9338\n",

"Epoch 9/10\n",

"60000/60000 [==============================] - 82s 1ms/step - loss: 0.2691 - accuracy: 0.9352\n",

"Epoch 10/10\n",

"60000/60000 [==============================] - 82s 1ms/step - loss: 0.2785 - accuracy: 0.9366\n"

]

}

],

"source": [

"model.compile(loss=\"categorical\_crossentropy\", optimizer=\"adam\", metrics=[\"accuracy\"])\n",

"history=model.fit(X\_train,y\_train\_onehot, epochs=10, batch\_size=5)"

]

},

{

"cell\_type": "code",

"execution\_count": 53,

"metadata": {},

"outputs": [

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 432x288 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"import matplotlib.pyplot as plt\n",

"loss = history.history[\"loss\"]\n",

"epochs = range(1, len(loss)+1)\n",

"val\_loss = history.history[\"loss\"]\n",

"plt.plot(epochs, loss, \"bo\", label=\"Training Loss\")\n",

"plt.plot(epochs, val\_loss, \"r\", label=\"Validation Loss\")\n",

"plt.title(\"Training and Validation Loss\")\n",

"plt.xlabel(\"Epochs\")\n",

"plt.ylabel(\"Loss\")\n",

"plt.legend()\n",

"plt.show()"

]

},

{

"cell\_type": "code",

"execution\_count": 54,

"metadata": {},

"outputs": [

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 432x288 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"acc = history.history[\"accuracy\"]\n",

"epochs = range(1, len(acc)+1)\n",

"val\_acc = history.history[\"accuracy\"]\n",

"plt.plot(epochs, acc, \"b-\", label=\"Training Acc\")\n",

"plt.plot(epochs, val\_acc, \"r--\", label=\"Validation Acc\")\n",

"plt.title(\"Training and Validation Accuracy\")\n",

"plt.xlabel(\"Epochs\")\n",

"plt.ylabel(\"Accuracy\")\n",

"plt.legend()\n",

"plt.show()"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": []

}

],

"metadata": {

"kernelspec": {

"display\_name": "Python 3",

"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": "3.6.8"

}

},

"nbformat": 4,

"nbformat\_minor": 4

}