{

"cells": [

{

"cell\_type": "code",

"execution\_count": 1,

"metadata": {

"id": "pZleFuZvLkrb"

},

"outputs": [

{

"name": "stderr",

"output\_type": "stream",

"text": [

"Using TensorFlow backend.\n",

"c:\\users\\danny\\appdata\\local\\programs\\python\\python36\\lib\\site-packages\\tensorflow\\python\\framework\\dtypes.py:516: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.\n",

" \_np\_qint8 = np.dtype([(\"qint8\", np.int8, 1)])\n",

"c:\\users\\danny\\appdata\\local\\programs\\python\\python36\\lib\\site-packages\\tensorflow\\python\\framework\\dtypes.py:517: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.\n",

" \_np\_quint8 = np.dtype([(\"quint8\", np.uint8, 1)])\n",

"c:\\users\\danny\\appdata\\local\\programs\\python\\python36\\lib\\site-packages\\tensorflow\\python\\framework\\dtypes.py:518: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.\n",

" \_np\_qint16 = np.dtype([(\"qint16\", np.int16, 1)])\n",

"c:\\users\\danny\\appdata\\local\\programs\\python\\python36\\lib\\site-packages\\tensorflow\\python\\framework\\dtypes.py:519: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.\n",

" \_np\_quint16 = np.dtype([(\"quint16\", np.uint16, 1)])\n",

"c:\\users\\danny\\appdata\\local\\programs\\python\\python36\\lib\\site-packages\\tensorflow\\python\\framework\\dtypes.py:520: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.\n",

" \_np\_qint32 = np.dtype([(\"qint32\", np.int32, 1)])\n",

"c:\\users\\danny\\appdata\\local\\programs\\python\\python36\\lib\\site-packages\\tensorflow\\python\\framework\\dtypes.py:525: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.\n",

" np\_resource = np.dtype([(\"resource\", np.ubyte, 1)])\n",

"c:\\users\\danny\\appdata\\local\\programs\\python\\python36\\lib\\site-packages\\tensorboard\\compat\\tensorflow\_stub\\dtypes.py:541: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.\n",

" \_np\_qint8 = np.dtype([(\"qint8\", np.int8, 1)])\n",

"c:\\users\\danny\\appdata\\local\\programs\\python\\python36\\lib\\site-packages\\tensorboard\\compat\\tensorflow\_stub\\dtypes.py:542: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.\n",

" \_np\_quint8 = np.dtype([(\"quint8\", np.uint8, 1)])\n",

"c:\\users\\danny\\appdata\\local\\programs\\python\\python36\\lib\\site-packages\\tensorboard\\compat\\tensorflow\_stub\\dtypes.py:543: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.\n",

" \_np\_qint16 = np.dtype([(\"qint16\", np.int16, 1)])\n",

"c:\\users\\danny\\appdata\\local\\programs\\python\\python36\\lib\\site-packages\\tensorboard\\compat\\tensorflow\_stub\\dtypes.py:544: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.\n",

" \_np\_quint16 = np.dtype([(\"quint16\", np.uint16, 1)])\n",

"c:\\users\\danny\\appdata\\local\\programs\\python\\python36\\lib\\site-packages\\tensorboard\\compat\\tensorflow\_stub\\dtypes.py:545: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.\n",

" \_np\_qint32 = np.dtype([(\"qint32\", np.int32, 1)])\n",

"c:\\users\\danny\\appdata\\local\\programs\\python\\python36\\lib\\site-packages\\tensorboard\\compat\\tensorflow\_stub\\dtypes.py:550: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.\n",

" np\_resource = np.dtype([(\"resource\", np.ubyte, 1)])\n"

]

}

],

"source": [

"import numpy as np\n",

"from keras.datasets import mnist\n",

"from keras.models import Model\n",

"from keras.layers import Dense, Input\n",

"from keras.datasets import mnist\n",

"import pandas as pd\n",

"from keras.models import Sequential\n",

"from keras.layers import Dense,Activation,Dropout,Conv2D,MaxPooling2D,Flatten,UpSampling2D\n",

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

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

"seed = 7\n",

"np.random.seed(seed)\n",

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

]

},

{

"cell\_type": "code",

"execution\_count": 2,

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/",

"height": 476

},

"id": "aF56bXUeW2-6",

"outputId": "dd1d19a9-b732-46de-c221-b520a406f45d"

},

"outputs": [],

"source": [

"a = Input(shape=(32,))\n",

"b = Dense(32, activation=\"relu\")(a)\n",

"model = Model(inputs=a, outputs=b)"

]

},

{

"cell\_type": "code",

"execution\_count": 3,

"metadata": {

"id": "37d6ZcZfQrl0"

},

"outputs": [],

"source": [

"\n",

"inputs = Input(shape=(8,))\n",

"hidden1 = Dense(10, activation=\"relu\")(inputs)\n",

"hidden2 = Dense(8, activation=\"relu\")(hidden1)\n",

"outputs = Dense(1, activation=\"sigmoid\")(hidden2)\n",

"model = Model(inputs=inputs, outputs=outputs)"

]

},

{

"cell\_type": "code",

"execution\_count": 4,

"metadata": {

"id": "TtHnUi-8MA6m"

},

"outputs": [],

"source": [

"X\_train = X\_train.reshape(X\_train.shape[0],28\*28).astype(\"float32\")\n",

"X\_test = X\_test.reshape(X\_test.shape[0], 28\*28).astype(\"float32\")\n",

"X\_train = X\_train / 255\n",

"X\_test = X\_test / 255\n",

"input\_img = Input(shape=(784,))\n",

"x = Dense(128, activation=\"relu\")(input\_img)\n",

"encoded = Dense(64, activation=\"relu\")(x)\n",

"x = Dense(128, activation=\"relu\")(encoded)\n",

"decoded = Dense(784, activation=\"sigmoid\")(x)\n",

"autoencoder = Model(input\_img, decoded)"

]

},

{

"cell\_type": "code",

"execution\_count": 5,

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "jEfcAqhnM0a-",

"outputId": "33ed4e4d-9aad-448f-f9a2-c1b4b9ea7250"

},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"WARNING:tensorflow:From c:\\users\\danny\\appdata\\local\\programs\\python\\python36\\lib\\site-packages\\tensorflow\\python\\ops\\nn\_impl.py:180: add\_dispatch\_support.<locals>.wrapper (from tensorflow.python.ops.array\_ops) is deprecated and will be removed in a future version.\n",

"Instructions for updating:\n",

"Use tf.where in 2.0, which has the same broadcast rule as np.where\n",

"WARNING:tensorflow:From c:\\users\\danny\\appdata\\local\\programs\\python\\python36\\lib\\site-packages\\keras\\backend\\tensorflow\_backend.py:422: The name tf.global\_variables is deprecated. Please use tf.compat.v1.global\_variables instead.\n",

"\n",

"Train on 60000 samples, validate on 10000 samples\n",

"Epoch 1/10\n",

" - 2s - loss: 0.2233 - accuracy: 0.7897 - val\_loss: 0.1406 - val\_accuracy: 0.8062\n",

"Epoch 2/10\n",

" - 2s - loss: 0.1251 - accuracy: 0.8101 - val\_loss: 0.1119 - val\_accuracy: 0.8111\n",

"Epoch 3/10\n",

" - 2s - loss: 0.1080 - accuracy: 0.8125 - val\_loss: 0.1017 - val\_accuracy: 0.8123\n",

"Epoch 4/10\n",

" - 2s - loss: 0.0998 - accuracy: 0.8135 - val\_loss: 0.0955 - val\_accuracy: 0.8128\n",

"Epoch 5/10\n",

" - 2s - loss: 0.0944 - accuracy: 0.8140 - val\_loss: 0.0912 - val\_accuracy: 0.8134\n",

"Epoch 6/10\n",

" - 2s - loss: 0.0906 - accuracy: 0.8143 - val\_loss: 0.0880 - val\_accuracy: 0.8136\n",

"Epoch 7/10\n",

" - 2s - loss: 0.0882 - accuracy: 0.8145 - val\_loss: 0.0862 - val\_accuracy: 0.8137\n",

"Epoch 8/10\n",

" - 2s - loss: 0.0864 - accuracy: 0.8146 - val\_loss: 0.0847 - val\_accuracy: 0.8137\n",

"Epoch 9/10\n",

" - 2s - loss: 0.0851 - accuracy: 0.8147 - val\_loss: 0.0835 - val\_accuracy: 0.8138\n",

"Epoch 10/10\n",

" - 2s - loss: 0.0840 - accuracy: 0.8148 - val\_loss: 0.0825 - val\_accuracy: 0.8139\n"

]

}

],

"source": [

"encoder = Model(input\_img, encoded)\n",

"decoder\_input = Input(shape=(64,))\n",

"decoder\_layer = autoencoder.layers[-2](decoder\_input)\n",

"decoder\_layer = autoencoder.layers[-1](decoder\_layer)\n",

"decoder = Model(decoder\_input, decoder\_layer)\n",

"autoencoder.compile(loss=\"binary\_crossentropy\",\n",

"optimizer=\"adam\", metrics=[\"accuracy\"])\n",

"history=autoencoder.fit(X\_train, X\_train, validation\_data=(X\_test,\n",

"X\_test), epochs=10, batch\_size=256, shuffle=True,\n",

"verbose=2)"

]

},

{

"cell\_type": "code",

"execution\_count": 6,

"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": 7,

"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": 34,

"metadata": {},

"outputs": [],

"source": [

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

]

},

{

"cell\_type": "code",

"execution\_count": 9,

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"WARNING:tensorflow:From c:\\users\\danny\\appdata\\local\\programs\\python\\python36\\lib\\site-packages\\keras\\backend\\tensorflow\_backend.py:4070: The name tf.nn.max\_pool is deprecated. Please use tf.nn.max\_pool2d instead.\n",

"\n"

]

}

],

"source": [

"input\_img = Input(shape=(28,28,1))\n",

"x = Conv2D(16, (3,3), activation=\"relu\", padding=\"same\")(input\_img)\n",

"x = MaxPooling2D((2,2), padding=\"same\")(x)\n",

"x = Conv2D(8, (3,3), activation=\"relu\", padding=\"same\")(x)\n",

"x = MaxPooling2D((2,2), padding=\"same\")(x)\n",

"x = Conv2D(8, (3,3), activation=\"relu\", padding=\"same\")(x)\n",

"encoded = MaxPooling2D((2,2), padding=\"same\")(x)\n",

"x = Conv2D(8, (3,3), activation=\"relu\", padding=\"same\")(encoded)\n",

"x = UpSampling2D((2,2))(x)\n",

"x = Conv2D(8, (3,3), activation=\"relu\", padding=\"same\")(x)\n",

"x = UpSampling2D((2,2))(x)\n",

"x = Conv2D(16, (3,3), activation=\"relu\")(x)\n",

"x = UpSampling2D((2,2))(x)\n",

"decoded = Conv2D(1, (3, 3), activation=\"sigmoid\", padding=\"same\")(x) \n",

"autoencoder = Model(input\_img,decoded)"

]

},

{

"cell\_type": "raw",

"metadata": {

"id": "60PzpfaIOCD\_"

},

"source": [

"X\_retrain = X\_train.reshape(X\_train.shape[0],28\*28).astype(\"float32\")\n",

"X\_retest = X\_test.reshape(X\_test.shape[0], 28\*28).astype(\"float32\")\n",

"X\_retrain = X\_retrain / 255\n",

"X\_retest = X\_retest / 255\n",

"encoded\_imgs = encoder.predict(X\_retest)\n",

"decoded\_imgs = decoder.predict(encoded\_imgs)"

]

},

{

"cell\_type": "code",

"execution\_count": 33,

"metadata": {},

"outputs": [

{

"ename": "NameError",

"evalue": "name 'y\_train' is not defined",

"output\_type": "error",

"traceback": [

"\u001b[1;31m---------------------------------------------------------------------------\u001b[0m",

"\u001b[1;31mNameError\u001b[0m Traceback (most recent call last)",

"\u001b[1;32m<ipython-input-33-0680b93dc039>\u001b[0m in \u001b[0;36m<module>\u001b[1;34m\u001b[0m\n\u001b[0;32m 2\u001b[0m \u001b[1;33m\u001b[0m\u001b[0m\n\u001b[0;32m 3\u001b[0m \u001b[0mplt\u001b[0m\u001b[1;33m.\u001b[0m\u001b[0mimshow\u001b[0m\u001b[1;33m(\u001b[0m\u001b[0mdecoded\_imgs\u001b[0m\u001b[1;33m[\u001b[0m\u001b[1;36m0\u001b[0m\u001b[1;33m]\u001b[0m\u001b[1;33m.\u001b[0m\u001b[0mreshape\u001b[0m\u001b[1;33m(\u001b[0m\u001b[1;36m28\u001b[0m\u001b[1;33m,\u001b[0m\u001b[1;36m28\u001b[0m\u001b[1;33m)\u001b[0m\u001b[1;33m,\u001b[0m \u001b[0mcmap\u001b[0m\u001b[1;33m=\u001b[0m\u001b[1;34m\"gray\"\u001b[0m\u001b[1;33m)\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\u001b[1;32m----> 4\u001b[1;33m \u001b[0mplt\u001b[0m\u001b[1;33m.\u001b[0m\u001b[0mtitle\u001b[0m\u001b[1;33m(\u001b[0m\u001b[1;34m\"Label: \"\u001b[0m \u001b[1;33m+\u001b[0m \u001b[0mstr\u001b[0m\u001b[1;33m(\u001b[0m\u001b[0my\_train\u001b[0m\u001b[1;33m[\u001b[0m\u001b[1;36m0\u001b[0m\u001b[1;33m]\u001b[0m\u001b[1;33m)\u001b[0m\u001b[1;33m)\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\u001b[0m\u001b[0;32m 5\u001b[0m \u001b[0mplt\u001b[0m\u001b[1;33m.\u001b[0m\u001b[0maxis\u001b[0m\u001b[1;33m(\u001b[0m\u001b[1;34m\"off\"\u001b[0m\u001b[1;33m)\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\u001b[0;32m 6\u001b[0m \u001b[0mplt\u001b[0m\u001b[1;33m.\u001b[0m\u001b[0mshow\u001b[0m\u001b[1;33m(\u001b[0m\u001b[1;33m)\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n",

"\u001b[1;31mNameError\u001b[0m: name 'y\_train' is not defined"

]

},

{

"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(decoded\_imgs[0].reshape(28,28), cmap=\"gray\")\n",

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

"plt.show()"

]

},

{

"cell\_type": "code",

"execution\_count": 69,

"metadata": {

"scrolled": true

},

"outputs": [

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 432x288 with 24 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"for i in range(1,9):\n",

" ax = plt.subplot(3,9,i)\n",

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

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

" ax = plt.subplot(3,9,i+9)\n",

" ax.imshow(encoded\_imgs[i].reshape(8,8), cmap=\"gray\")\n",

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

" ax = plt.subplot(3,9,i+18)\n",

" ax.imshow(decoded\_imgs[i].reshape(28,28), cmap=\"gray\")\n",

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

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

"plt.show()"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": []

}

],

"metadata": {

"colab": {

"name": "Untitled0.ipynb",

"provenance": []

},

"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": 1

}