Transfer Learning MNIST

- Train a simple convnet on the MNIST dataset the first 5 digits [0..4].
- Freeze convolutional layers and fine-tune dense layers for the classification of digits [5..9].

Import MNIST data and create 2 datasets with one dataset having digits from

Import the mnist dataset from keras datasets

```
#Importing important modules
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.callbacks import ModelCheckpoint, EarlyStopping
#Installing Tensorboard for Colab
!pip install tensorboardcolab
```

```
Requirement already satisfied: tensorboardcolab in /usr/local/lib/python3.6/dist-pack
```

```
from keras.datasets import mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

Creating two datasets one with digits below 5 and one with 5 and above

```
x_train_lt5 = x_train[y_train < 5]
y_train_lt5 = y_train[y_train < 5]
x_test_lt5 = x_test[y_test < 5]
y_test_lt5 = y_test[y_test < 5]
```

```
set(y_train_lt5)
```

```
{0, 1, 2, 3, 4}
```

```
x_train_gt5 = x_train[y_train >= 5]
y_train_gt5 = y_train[y_train >= 5] - 5  # make classes start at 0 for
x_test_gt5 = x_test[y_test >= 5]         # np_utils.to_categorical
y_test_gt5 = y_test[y_test >= 5] - 5
```

```
set(y_train_gt5)
```

```
{0, 1, 2, 3, 4}
```

```
set(y_train)
```

```
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
```

Check

Verify shapes of x_train, y_train, x_test and y_test for both the datasets with the below given shape

```
x_test.shape
```

```
(10000, 28, 28)
```

```
print(x_train_lt5.shape)
print(y_train_lt5.shape)
print(x_test_lt5.shape)
print(y_test_lt5.shape)
```

```
(30596, 28, 28)
(30596,)
(5139, 28, 28)
(5139,)
```

```
print(x_train_gt5.shape)
print(y_train_gt5.shape)
print(x_test_gt5.shape)
print(y_test_gt5.shape)
```

```
(29404, 28, 28)
(29404,)
(4861, 28, 28)
(4861,)
```

Let us take only the dataset (x_train, y_train, x_test, y_test) for Integers 0 to 4 ir

Reshape x_train and x_test to a 4 Dimensional array (channel = 1) to pass it int

```
x_train_lt5 = x_train_lt5.reshape(x_train_lt5.shape[0],28,28,1)
```

```
x_train_lt5[0].shape
```

```
(28, 28, 1)
```

```
x_test_lt5 = x_test_lt5.reshape(x_test_lt5.shape[0],28,28,1)
```

```
x_test_lt5[0].shape
```

```
(28, 28, 1)
```

Change into float32 datatype and Normalize x_train and x_test by dividing it by

```
x_train_lt5 = x_train_lt5.astype('float32')
x_test_lt5 = x_test_lt5.astype('float32')

#Normalizing the input
x_train_lt5 /= 255.0
x_test_lt5 /= 255.0
```

Check

Verify the shapes of the X_train and X_test with the shapes given below.

```
print('x_train shape:', x_train_lt5.shape)
print(x_train_lt5.shape[0], 'train samples')
print(x_test_lt5.shape[0], 'test samples')
```

```
x_train shape: (30596, 28, 28, 1)
30596 train samples
5139 test samples
```

```
print('X_train shape:', x_train_lt5.shape)
print('X_test shape:', x_test_lt5.shape)
```

```
X_train shape: (30596, 28, 28, 1)
X_test shape: (5139, 28, 28, 1)
```

```
batch_size = 128
num_classes = 5
epochs = 5
```

Use One-hot encoding to divide y_train and y_test into required no of output cla

```
# convert class vectors to binary class matrices
y_train_lt5 = keras.utils.to_categorical(y_train_lt5, num_classes)
y_test_lt5 = keras.utils.to_categorical(y_test_lt5, num_classes)
```

```
set(y_train_lt5)
```

```
{0, 1, 2, 3, 4}
```

```
# input image dimensions
img_rows, img_cols = 28, 28
```

```
#Keras expects data to be in the format (N_E,N_H,N_W,N_C)
#N_E = Number of Examples, N_H = height, N_W = Width, N_C = Number of Channels.
x_train_lt5 = x_train_lt5.reshape(x_train_lt5.shape[0], img_rows, img_cols, 1)
x_test_lt5 = x_test_lt5.reshape(x_test_lt5.shape[0], img_rows, img_cols, 1)
input_shape = (img_rows, img_cols, 1)
```

Build a sequential model with 2 Convolutional layers with 32 kernels of size (3, size (2,2) followed by a drop out layer to be trained for classification of digits 0

```
#Initialize the model
model = Sequential()

#Add a Convolutional Layer with 32 filters of size 3X3 and activation function as 'ReLU'
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))

#Add a Convolutional Layer with 64 filters of size 3X3 and activation function as 'ReLU'
model.add(Conv2D(64, (3, 3), activation='relu'))

#Add a MaxPooling Layer of size 2X2
model.add(MaxPooling2D(pool_size=(2, 2)))

#Apply Dropout with 0.25 probability
model.add(Dropout(0.25))
```

Post that flatten the data and add 2 Dense layers with 128 neurons and neuron 'relu' and 'softmax' respectively. Add dropout layer inbetween if necessary

```
#Flatten the layer
model.add(Flatten())

#Add Fully Connected Layer with 128 units and activation function as 'ReLU'
model.add(Dense(128, activation='relu'))
#Apply Dropout with 0.5 probability
model.add(Dropout(0.5))

#Add Fully Connected Layer with 10 units and activation function as 'softmax'
model.add(Dense(num_classes, activation='softmax'))
```

Print the training and test accuracy for 5 epochs

```
from keras.optimizers import Adam
from keras.losses import categorical_crossentropy

#To use adam optimizer for learning weights with learning rate = 0.001
```

```
optimizer = Adam(lr=0.001)
#Set the loss function and optimizer for the model training
model.compile(loss=categorical_crossentropy,
              optimizer=optimizer,
              metrics=['accuracy'])


#Import tensorboardcolab modules for creating a tensorboard call back which will passed in
from tensorboardcolab import TensorBoardColab, TensorBoardColabCallback

#Tensorboard callback is going to be added to model.fit function to draw graphs of loss va
tbc = TensorBoardColab()
```

```
Wait for 8 seconds...
TensorBoard link:
https://c5246f93.ngrok.io
```

```
#Adding Early stopping callback to the fit function is going to stop the training,
#if the val_loss is not going to change even '0.001' for more than 10 continous epochs

early_stopping = EarlyStopping(monitor='val_loss', min_delta=0.001, patience=10)

#Adding Model Checkpoint callback to the fit function is going to save the weights wheneve
#Hence saving the best weights occurred during training

model_checkpoint =  ModelCheckpoint('mnist_cnn_checkpoint_{epoch:02d}_loss{val_loss:.4f}.h
                                                          monitor='val_loss',
                                                          verbose=1,
                                                          save_best_only=True,
                                                          save_weights_only=True,
                                                          mode='auto',
                                                          period=1)


#Training on the dataset and adding the all the callbacks to the fit function.
#Once the training starts, results start appearing on Tensorboard after 1 epoch
model.fit(x_train_lt5, y_train_lt5,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test_lt5, y_test_lt5),
          callbacks=[TensorBoardColabCallback(tbc),early_stopping,model_checkpoint])
```

```
Train on 30596 samples, validate on 5139 samples
Epoch 1/5
30596/30596 [==============================] - 5s 153us/step - loss: 0.1168 - acc: 0.

Epoch 00001: val_loss improved from inf to 0.01987, saving model to mnist_cnn_checkpo
Epoch 2/5
30596/30596 [==============================] - 4s 142us/step - loss: 0.0318 - acc: 0.

Epoch 00002: val_loss improved from 0.01987 to 0.01163, saving model to mnist_cnn_che
Epoch 3/5
30596/30596 [==============================] - 4s 144us/step - loss: 0.0220 - acc: 0.

Epoch 00003: val_loss improved from 0.01163 to 0.01009, saving model to mnist_cnn_che
Epoch 4/5
30596/30596 [==============================] - 4s 140us/step - loss: 0.0161 - acc: 0.

Epoch 00004: val_loss improved from 0.01009 to 0.00533, saving model to mnist_cnn_che
Epoch 5/5
30596/30596 [==============================] - 4s 143us/step - loss: 0.0130 - acc: 0.

Epoch 00005: val_loss improved from 0.00533 to 0.00438, saving model to mnist_cnn_che
<keras.callbacks.History at 0x7f7030112828>
```

```
#Testing the model on test set
score = model.evaluate(x_test_lt5, y_test_lt5)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
5139/5139 [==============================] - 1s 109us/step
Test loss: 0.004378508583054539
Test accuracy: 0.9986378672893559
```

Use the model trained on 0 to 4 digit classification and train it on the dataset w

Transfer learning keeping only the dense layers to be trainable)

Make only the dense layers to be trainable and convolutional layers to be non-t

Check model summary to see model layer names

```
for layers in model.layers:
    print(layers.name)
    if('dense' not in layers.name):
        layers.trainable = False
        print(layers.name + 'is not trainable\n')
    if('dense' in layers.name):
        print(layers.name + ' is trainable\n')
```

```
conv2d_3
conv2d_3is not trainable

conv2d_4
conv2d_4is not trainable

max_pooling2d_2
max_pooling2d_2is not trainable

dropout_3
dropout_3is not trainable

flatten_2
flatten_2is not trainable

dense_3
dense_3 is trainable

dropout_4
dropout_4is not trainable

dense_4
dense_4 is trainable
```

Do the required preprocessing for `x_train_gt5` also same as `x_train_lt5` and `y_train_lt5`

1. Reshape
2. Change to float32 datatype
3. Normalize (dividing with 255)
4. y_train and y_test Convert into one-hot vectors

Reshape

```
x_train_gt5 = x_train_gt5.reshape(x_train_gt5.shape[0],28,28,1)
```

```
x_train_gt5.shape
```

```
(29404, 28, 28, 1)
```

```
set(y_train_gt5)
```

```
{0, 1, 2, 3, 4}
```

```
x_test_gt5 = x_test_gt5.reshape(x_test_gt5.shape[0],28,28,1)
```

Change to Float and Normalize

```
x_train_gt5 = x_train_gt5.astype('float32')
x_test_gt5 = x_test_gt5.astype('float32')

#Normalizing the input
x_train_gt5 /= 255.0
x_test_gt5 /= 255.0
```

OneHot

```
# convert class vectors to binary class matrices
y_train_gt5 = keras.utils.to_categorical(y_train_gt5, num_classes)
y_test_gt5 = keras.utils.to_categorical(y_test_gt5, num_classes)
```

Check

Verify the shapes with the given below.

```
print(x_train_gt5.shape)
print(y_train_gt5.shape)
print(x_test_gt5.shape)
print(y_test_gt5.shape)
```

```
(29404, 28, 28, 1)
(29404, 5)
(4861, 28, 28, 1)
(4861, 5)
```

Print the accuracy for classification of digits 5 to 9

```
#Training on the dataset and adding the all the callbacks to the fit function.
#Once the training starts, results start appearing on Tensorboard after 1 epoch
model.fit(x_train_gt5, y_train_gt5,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test_gt5, y_test_gt5),
          callbacks=[TensorBoardColabCallback(tbc),early_stopping,model_checkpoint])
```

```
Train on 29404 samples, validate on 4861 samples
Epoch 1/5
  128/29404 [..............................] - ETA: 5s - loss: 5.6763 - acc: 0.2812/u
  'Discrepancy between trainable weights and collected trainable'
29404/29404 [==============================] - 4s 147us/step - loss: 0.1961 - acc: 0.

Epoch 00001: val_loss did not improve from 0.00438
Epoch 2/5
29404/29404 [==============================] - 4s 141us/step - loss: 0.0550 - acc: 0.

Epoch 00002: val_loss did not improve from 0.00438
Epoch 3/5
29404/29404 [==============================] - 4s 142us/step - loss: 0.0395 - acc: 0.

Epoch 00003: val_loss did not improve from 0.00438
Epoch 4/5
29404/29404 [==============================] - 4s 143us/step - loss: 0.0296 - acc: 0.

Epoch 00004: val_loss did not improve from 0.00438
Epoch 5/5
29404/29404 [==============================] - 4s 144us/step - loss: 0.0245 - acc: 0.

Epoch 00005: val_loss did not improve from 0.00438
<keras.callbacks.History at 0x7f703d121f60>
```

```
#Testing the model on test set
score = model.evaluate(x_test_gt5, y_test_gt5)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
4861/4861 [==============================] - 1s 109us/step
Test loss: 0.020349998070301942
Test accuracy: 0.9942398683398478
```

Text classification using TF-IDF

Load the dataset from sklearn.datasets

```
from sklearn.datasets import fetch_20newsgroups

categories = ['alt.atheism', 'soc.religion.christian', 'comp.graphics', 'sci.med']
```

Training data

```
twenty_train = fetch_20newsgroups(subset='train', categories=categories, shuffle=True, ran
```

```
Downloading 20news dataset. This may take a few minutes.
```

Test data

```
twenty_test = fetch_20newsgroups(subset='test', categories=categories, shuffle=True, rando
```

a. You can access the values for the target variable using .target attribute

b. You can access the name of the class in the target variable with .target_nam

```
twenty_train.target
```

```
array([1, 1, 3, ..., 2, 2, 2])
```

```
twenty_train
```

```
 'From: dfuller@portal.hq.videocart.com (Dave Fuller)\nSubject: Re: thoughts on chri
 'From: halsall@murray.fordham.edu (Paul Halsall)\nSubject: Bible Unsuitable for New
 'From: twain@carson.u.washington.edu (Barbara Hlavin)\nSubject: Re: Is MSG sensitiv
 'From: Desiree_Bradley@mindlink.bc.ca (Desiree Bradley)\nSubject: Doing the work of
 'From: db7n+@andrew.cmu.edu (D. Andrew Byler)\nSubject: Re: What WAS the immaculate
 'From: Donald Mackie <Donald_Mackie@med.umich.edu>\nSubject: Re: Seeking advice/exp
 "From: weilej@cary115.its.rpi.edu (Jason Lee Weiler)\nSubject: Re: need a viewer fo
 "From: ednobles@sacam.OREN.ORTN.EDU (Edward d Nobles)\nSubject: POV .TGA's and Spee
 "Subject: .GL and .FLI specs\nFrom: arthur@qedbbs.com (Arthur Choung)\nOrganization
 'From: jbrown@batman.bmd.trw.com\nSubject: Re: Death Penalty / Gulf War (long)\nLin
 "From: bobbe@vice.ICO.TEK.COM (Robert Beauchaine)\nSubject: Re: thoughts on christi
 "From: rcomg@melomys.co.rmit.oz.AU (Mark Gregory)\nSubject: AVI file format?\nSumma
 'From: jimh@carson.u.washington.edu (James Hogan)\nSubject: Re: Yet more Rushdie [R
 'From: alastair@farli.otago.ac.nz (Alastair Thomson)\nSubject: Does \'Just/justifia
 "From: will@futon.webo.dg.com (Will Taber)\nSubject: [soc.religion.christian] Re: T
 'From: mathew <mathew@mantis.co.uk>\nSubject: Alt.Atheism FAQ: Introduction to Athe
 'From: timmbake@mcl.ucsb.edu (Bake Timmons)\nSubject: Re: Amusing atheists and agno
 'From: kmr4@po.CWRU.edu (Keith M. Ryan)\nSubject: Re: The Inimitable Rushdie\nOrgan
 "From: tonyo@pendragon.CNA.TEK.COM (Tony Ozrelic)\nSubject: Need info on cc:Mail fi
 "From: scornd7@technet.sg (Tang Chang Thai)\nSubject: Re: InterViews graphics packa
 'From: mmatusev@radford.vak12ed.edu (Melissa N. Matusevich)\nSubject: Re: Emphysema
 'From: livesey@solntze.wpd.sgi.com (Jon Livesey)\nSubject: Re: <Political Atheists?
 'From: rjs2@po.cwru.edu (Richard J. Szanto)\nSubject: Re: When are two people marri
 "From: rws2v@uvacs.cs.Virginia.EDU (Richard Stoakley)\nSubject: Need a good concave
 'From: Geoffrey_Hansen@mindlink.bc.ca (Geoffrey Hansen)\nSubject: Re: VESA on the S
 'From: erh0362@tesla.njit.edu\nSubject: Mormon beliefs about bastards\nOrganization
 'From: bil@okcforum.osrhe.edu (Bill Conner)\nSubject: Re: some thoughts.\nNntp-Post
 "From: mary@uicsl.csl.uiuc.edu (Mary E. Allison)\nSubject: Re: Is MSG sensitivity s
 'From: sieferme@stein.u.washington.edu (Eric Sieferman)\nSubject: Re: some thoughts
 'From: Lars.Jorgensen@p7.syntax.bbs.bad.se (Lars Jorgensen)\nSubject: Externel proc
 'From: maridai@comm.mot.com (Marida Ignacio)\nSubject: Re: "Accepting Jesus in your
 'From: h8902939@hkuxa.hku.hk (Abel)\nSubject: Developable Surface\nNntp-Posting-Hos
 'From: wsun@jeeves.ucsd.edu (Fiberman)\nSubject: Re: Is MSG sensitivity superstitio
 'From: mcovingt@aisun3.ai.uga.edu (Michael Covington)\nSubject: Re: When are two pe
 'From: Patrick C Leger <pl1u+@andrew.cmu.edu>\nSubject: Re: thoughts on christians\
 "From: diablo.UUCP!cboesel (Charles Boesel)\nSubject: Re: Postscript drawing prog\n
 "From: I3150101@dbstu1.rz.tu-bs.de (Benedikt Rosenau)\nSubject: Re: The Inimitable
 "From: nichael@bbn.com (Nichael Cramer)\nSubject: Re: Dead Sea Scrolls\nReply-To: n
 'From: atterlep@vela.acs.oakland.edu (Cardinal Ximenez)\nSubject: Re: A question th
 'From: cobb@alexia.lis.uiuc.edu (Mike Cobb)\nSubject: Re: After 2000 years, can we
 'From: parkin@Eng.Sun.COM (Michael Parkin)\nSubject: Re: Being right about messiahs
 'From: kxgst1+@pitt.edu (Kenneth Gilbert)\nSubject: Re: Any info. on Vasomotor Rhin
 "From: labson@borneo.corp.sgi.com (Joel Labson)\nSubject: Maybe?????\nOrganization:
 'From: dyer@spdcc.com (Steve Dyer)\nSubject: Re: Thrush ((was: Good Grief! (was Re:
 ...],
'filenames': array(['/root/scikit_learn_data/20news_home/20news-bydate-train/comp.gr
       '/root/scikit_learn_data/20news_home/20news-bydate-train/comp.graphics/38479'
       '/root/scikit_learn_data/20news_home/20news-bydate-train/soc.religion.christi
       ...,
       '/root/scikit_learn_data/20news_home/20news-bydate-train/sci.med/58112',
       '/root/scikit_learn_data/20news_home/20news-bydate-train/sci.med/58578',
       '/root/scikit_learn_data/20news_home/20news-bydate-train/sci.med/58895'],
      dtype='<U86'),
'target': array([1, 1, 3, ..., 2, 2, 2]),
'target_names': ['alt.atheism',
 'comp.graphics',
 'sci.med',
 'soc.religion.christian']}
```