Technische Universität Hamburg
Institut für Mathematik
Lehrstuhl Numerische Mathematik

Jens-Peter M. Zemke
Jonas Grams

Mathematics of Neural Networks
winter semester 2021/2022
exercise sheet 6

Exercise 1: (4 points) Derive an FFT for $n = 3^k$.
Hint: Similar to the case $n = 2 \cdot m$ in the lecture notes on page 112[1] consider $n = 3 \cdot m$ and decompose the DFT of size n into three DFT's of size m.

Exercise 2: (4 points)

a) Count operations in the following two algorithms to compute the full convolution $\mathbf{y} = \mathbf{x} * \mathbf{f}$ of $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{f} \in \mathbb{R}^k$:
   (i) directly compute the convolution,
   (ii) enlarge the dimension of $\mathbf{x}$ and $\mathbf{f}$ to the smallest $2^\ell \geqslant n+k-1$ by appending trailing zeros and use the radix-2 FFT/IFFT (w/o the permutation).

b) Write a Python script that plots for given $n \in \mathbb{N}$ the operation count of both approaches. For which k is FFT better?

Hint: You can use the following Python code for part (ii) of a).

```
import numpy as np
from scipy.linalg import dft

def fft2(x):
    n = len(x)
    if n == 1:
        y = x
    else:
        y = np.zeros(n, dtype='complex')
        m = n//2
        omega = np.exp(-2*np.pi*1j/n)
        d = omega**np.arange(m)
        z_top = fft2(x[0:n:2])
        z_bot = d * fft2(x[1:n:2])
        y[0:m] = z_top + z_bot
        y[m:n] = z_top - z_bot
    return y
```

Exercise 3: (4 points) In exercise 2 on the previous exercise sheet you implemented evaluation and backpropagation in a convolutional layer using the function `convolve2D()` of the SciPy package. Now we want to utilize the FFT approach and the `im2col` approach to calculate the convolution.

a) Add a method `evaluate_fft(self, a)` that implements the evaluation of a 2D convolutional layer with FFT and IFFT. Use the functions `rfft2` and `irfft2` from `scipy.fft`[2]

[1] Version from 01.11.21
[2] More information can be found at https://docs.scipy.org/doc/scipy/reference/tutorial/fft.html

b) Add a method `evaluate_im2col(self, a)` that implements the evaluation of a 2D convolutionl layer with im2col. Use the functions `im2col` from the script `utils.py` to map the input to the corresponding im2col matrix. Use the attribute `cache` of `Conv2DLayer` to save the im2col matrix and the reshaped filterbank. Don't forget to reshape the result before applying the activation function.

c) Test your implementation with the code provided in `layers.py`

Exercise 4: (4 points) Compare our implementation of a neural network with TensorFlow.

a) Develop a convolutional neural network for the Fashion MNIST dataset. Try to achieve at least 80 percent accuracy.

b) Implement the network with TensorFlow and our library.

c) Compare the time needed for **one** epoch of training in both cases.Use `im2col` for evaluation in our implementation. The time for the TensorFlow network is printed while the network is trained.

You may use the following skeleton provided in `skeleton.py`.

```
import numpy             as np
import matplotlib.pyplot as plt
import tensorflow.keras  as tfk

from random import randrange
from time   import time      # For time measuring

from networks    import SequentialNet
from layers      import *
from optimizers  import *
from activations import *

DATA = np.load('fashion_mnist.npz')
x_train, y_train = DATA['x_train'].reshape(60000,28,28), DATA['y_train']
x_test,  y_test  = DATA['x_test'].reshape(10000,28,28), DATA['y_test']
x_train, x_test  = x_train / 255.0, x_test / 255.0

x    = x_train[:,np.newaxis,:,:]
x_TF = x_train[:,:,:,np.newaxis]

bs, ep, eta = 128, 10, .001
"""
Categories for Fashion MNIST. Category i is ct[i].
"""
ct = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress',
      'Coat', 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle␣Boot']

"""
TODO Set up the network with our library
"""
net = SequentialNet((1,28,28))

"""
Add the layers to your network. As first hidden layer
you can use for example:

net.add_conv2D((32,3,3),
               afun=ReLU(),
               optim=Adam(),
```

```
                 initializer=HeUniform(),
                 eval_method='im2col')
"""

"""
The Last layer should be a SoftMax Layer with 10 neurons
"""
net.add_dense(10, afun=SoftMax(),
                optim=Adam(),
                initializer=HeUniform())
"""
TODO Set up the network with TensorFlow
"""
input_shape = (28,28,1)
net_TF = tfk.Sequential()

net_TF.add(tfk.Input(shape=input_shape))
"""
Add the same layers as above to the network.
If you used a convolutional layer with 32 3x3 filters
as first layer, you can add it with
net_TF.add(tfk.layers.Conv2D(32, (3,3),
                             activation='relu',
                             kernel_initializer='he_uniform'))
"""

net_TF.add(tfk.Dense(10, activation='softmax',
                     kernel_initializer='he_uniform'))

opt = tfk.optimizers.Adam(eta)
net_TF.compile(optimizer=opt,
               loss='categorical_crossentropy',
               metrics=['accuracy'])


start = time()
net.train(x, y_train, batch_size=bs, epochs=1)
t_train = time() - start

"""
TODO Train the TensorFlow network. With metrics=['accuracy'] you
     get the time needed for training one epoch.
"""

y_test = np.argmax(y_test, 1).T



y_tilde_TF = net_TF.predict(x_test.reshape(10000, 28, 28, 1))
guess_TF   = np.argmax(y_tilde_TF, 1).T
print('Accuracy with TensorFlow =', np.sum(guess_TF == y_test)/100)
```