Final_Report

Li Yuan

July 24, 2021

1 Backgound and Introduction

We will initialize two main methods appearing in Neural Network.

1. The first one is a new method in Image Data Augmentation.
2. The second one is a new method in redesigning Committee Vote.

Currently, most image data augmentation techniques are only using one input to shift, rotate, add white noise. However, my new method will incorporate random multiple inputs to extract their common obvious features by committee votes based on each class. Our method performs well on small size data.

The present mature method of committee vote is to choose mode or median as the final decision, but those methods don't consider probability and weights. However, I will come up with a method which make use of probability distribution and appearing frequency as weights to redesign my new committee vote.

1.1 Introduce our Python3.8 environment to reproduce my results

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.pyplot import imshow
from keras.datasets import mnist

from keras.preprocessing.image import ImageDataGenerator
import os

import sklearn
from sklearn.utils import shuffle

from tensorflow import keras
from tensorflow.keras import layers

import copy
%config InlineBackend.figure_format = 'pdf'

import warnings
warnings.filterwarnings('ignore')
```

2 The current general image augmentation method review

2.1 Load Digit Written MNIST database as our new method experimental data

[2]:
```
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

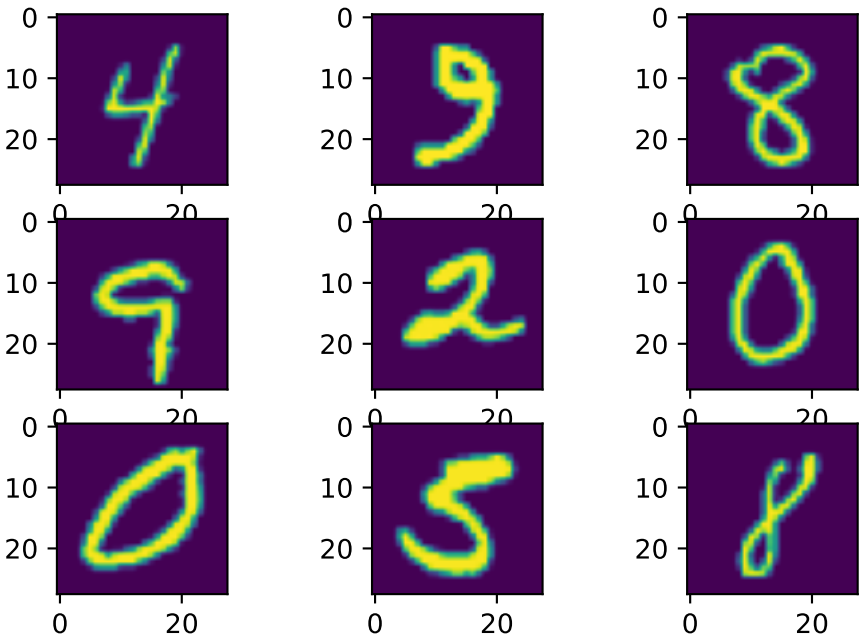
2.2 reshape to be [samples][width][height][channels] and change to float type

[3]:
```
X_train = X_train.reshape((X_train.shape[0], 28, 28, 1))
X_train = X_train.astype('float32')
# Set seed
np.random.seed(12345)
```

2.3 Feature Standardization

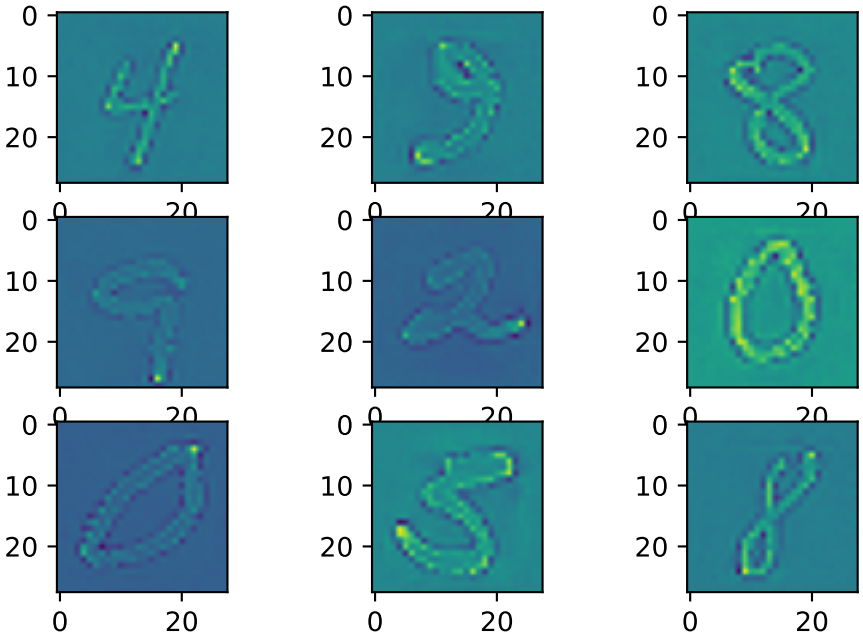
This method will centralize and standardize the data. The raw data sometimes will benefit from this manipulation because of some outliers and different scales.

[4]:
```
# define data preparation
datagen = ImageDataGenerator(featurewise_center=True,␣
 ↪featurewise_std_normalization=True)
# fit parameters from data
datagen.fit(X_train)
# configure batch size and retrieve one batch of images
for X_batch, y_batch in datagen.flow(X_train, y_train, batch_size=9, seed=123):
    # create a grid of 3x3 images
    for i in range(0, 9):
        plt.subplot(330 + 1 + i)
        plt.imshow(X_batch[i].reshape(28, 28))
    # show the plot
    plt.show()
    break
```



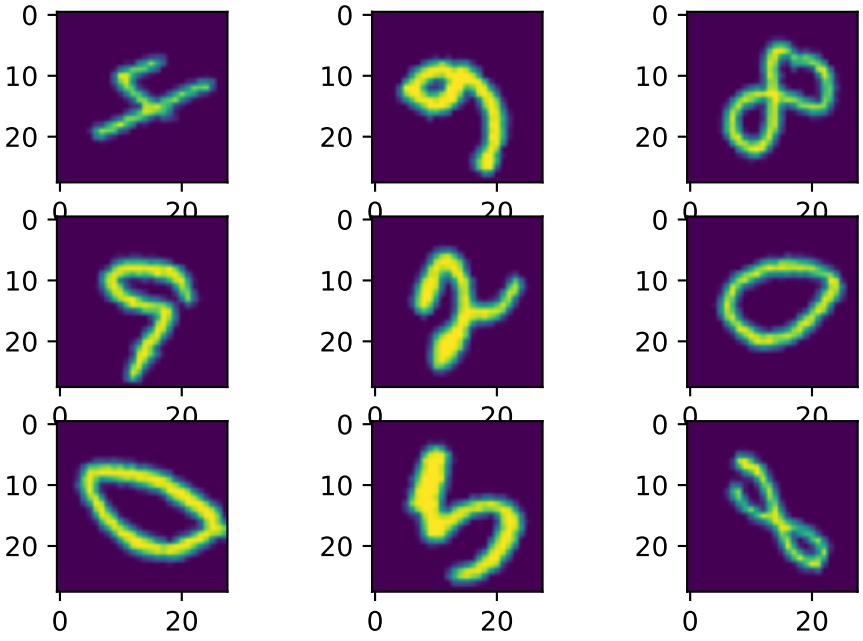
2.4 ZCA Whitening

[5]:
```
datagen = ImageDataGenerator(zca_whitening=True)
datagen.fit(X_train)
for X_batch, y_batch in datagen.flow(X_train, y_train, batch_size=9, seed=123):
    for i in range(0, 9):
        plt.subplot(330 + 1 + i)
        plt.imshow(X_batch[i].reshape(28, 28))
    plt.show()
    break
```



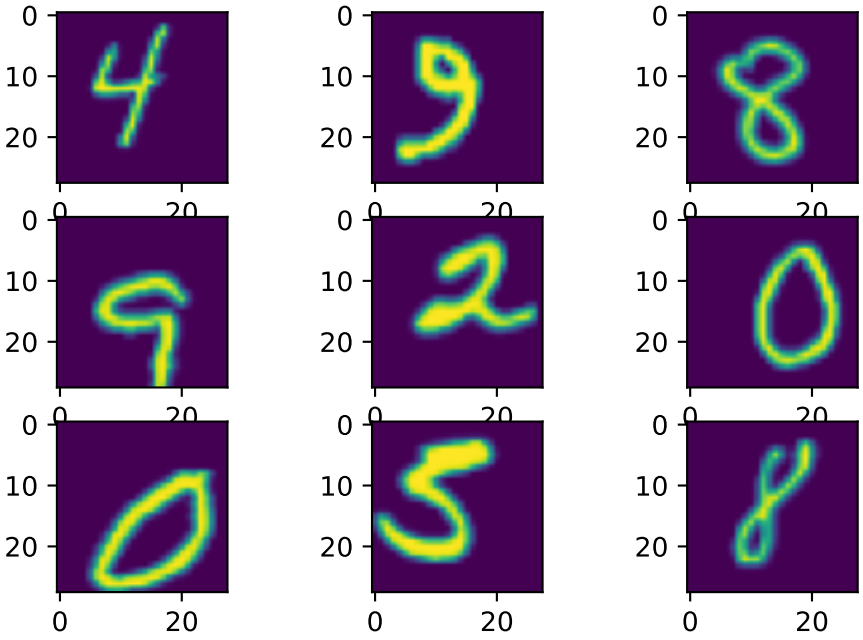
2.5 Random Rotations

[6]:
```
datagen = ImageDataGenerator(rotation_range=90)
```



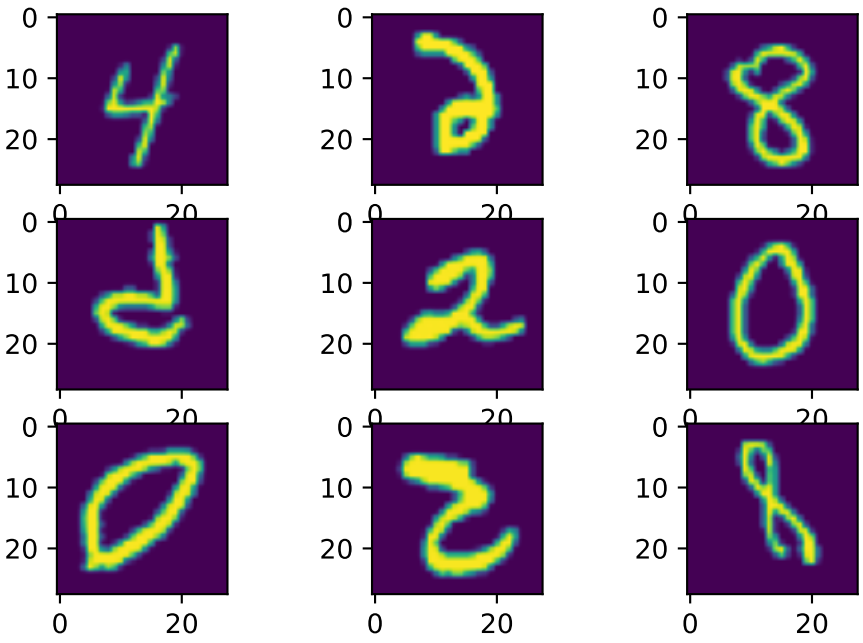
2.6 Random Shifts

[8]:
```
# define data preparation
shift = 0.2
datagen = ImageDataGenerator(width_shift_range=shift, height_shift_range=shift)
```



2.7 Random Flips

[12]:
```
datagen = ImageDataGenerator(horizontal_flip=True, vertical_flip=True)
```



[ ]: