Using different models in Images Classification

Table of Contents:

0. Title and author
1. Summary of research questions
2. Datasets
3. Collaboration

Title and author:

1. **Title**:

   Using different models in Images Classification
2. **Author**:

- Name: Thai Quoc Hoang
- Date of birth: January 09, 2000
- Email: quocthai9120@gmail.com (mailto:quocthai9120@gmail.com) / qthai912@uw.edu (mailto:qthai912@uw.edu)

Summary of research questions:

Part 0: Using k-Nearest-Neighbors approach:

1. Do the results predicted by the model saturated with the other labels?

- We want to know whether our model can really classify the object or just able to get right from a slightly different score for each label.
- Approach:
  - Implement kNN model to classify images in both MNIST Fashion dataset and CIFAR_10 dataset.
    - Define a function to calculate the distances between the training images and the testing images using numpy.
    - Define a function to get k elements that nearest the testing images with distance is defined by the distance function.
    - Define a function to get the best result from k nearest elements using a function to compare k-nearest neighbors. Currently, the function to calculate the score of each label $a[i]$ that appears in k-nearest neighbors is: $\frac{1}{\frac{\sum_{j=1}^{n} label[i][j]}{n}} * n$, where n is the total times $a[i]$ appears in k-nearest neighbors. In words, the score of each label is mean of all distances between the image with the current label and the testing image multiply number of images for each label that appears in k-nearest neighbors).
    - Returns the scores of all labels for each testing image.

        - Visualize the images in the validation and test set with real label and predicted labels using matplotlib to verify that no error occurs.
        - Use the returned scores for all labels for each testing image to see how different the score for each label is.

2. How much different when using k-Nearest-Neighbors to classify images in Fashion MNIST dataset comparing to when using that to classify images in CIFAR_10 datasset?

    - Fashion MNIST dataset and CIFAR_10 dataset are much different datasets. Analyzing how each model works in each dataset can help us generalize the choice of model to choose when face a images classification task.
    - Approach:
        - Implement kNN model to classify images in MNIST Fashion dataset and CIFAR_10 dataset.
        - Compare the accuracy of the model in the two datasets.
        - Visualize images in each dataset and compare to the other dataset to understand why the results are different in the two datasets.

Part 1: Using Convolutional Neural Network approach:

In this part, we will focus on CIFAR_10 dataset. The Fashion MNIST is used only with the purpose of comparing to kNN model.

1. Do the results predicted by the model saturated with the other labels?

    - We want to know whether our model can really classify the object or just able to get right from a slightly different score for each label.
    - Approach:
        - Using Sequential from Keras to implement a Convolutional Neural Network model. To be fair, in this research we will use the same configuration of model for both dataset and analyze the different.
            - CIFAR_10 Current model summary:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 32)        2432      
_________________________________________________________________
activation_1 (Activation)    (None, 28, 28, 32)        0         
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 32)        0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 10, 10, 64)        51264     
_________________________________________________________________
activation_2 (Activation)    (None, 10, 10, 64)        0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 8, 8, 128)         73856     
_________________________________________________________________
activation_3 (Activation)    (None, 8, 8, 128)         0         
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 128)         0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 2048)              0         
_________________________________________________________________
dense_1 (Dense)              (None, 256)               524544    
_________________________________________________________________
activation_4 (Activation)    (None, 256)               0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 10)                2570      
_________________________________________________________________
activation_5 (Activation)    (None, 10)                0         
=================================================================
Total params: 654,666
Trainable params: 654,666
Non-trainable params: 0
```

- FASHION MNIST Current model summary:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_1 (Conv2D)            (None, 24, 24, 32)        832       
_________________________________________________________________
activation_1 (Activation)    (None, 24, 24, 32)        0         
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 32)        0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 8, 8, 64)          51264     
_________________________________________________________________
activation_2 (Activation)    (None, 8, 8, 64)          0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 6, 6, 128)         73856     
_________________________________________________________________
activation_3 (Activation)    (None, 6, 6, 128)         0         
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 3, 3, 128)         0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 1152)              0         
_________________________________________________________________
dense_1 (Dense)              (None, 256)               295168    
_________________________________________________________________
activation_4 (Activation)    (None, 256)               0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 10)                2570      
_________________________________________________________________
activation_5 (Activation)    (None, 10)                0         
=================================================================
Total params: 423,690
Trainable params: 423,690
Non-trainable params: 0
_________________________________________________________________
```

- Visualize the images in the validation and test set with real label and predicted labels using matplotlib to verify that no error occurs.
- Compare the accuracy for each dataset while using CNN then compare with the result while using kNN.

2. What does the model learn in each layer?

- Understanding what happen in each layer can help us know how to tuning hyperparameters and help us avoid time consuming because of training bad model. In this part we only use the CIFAR_10 dataset.
- Approach:
  - Visualize the filters in each convolutional layers of the model by getting the weights of each layer using Keras.
  - Use serval images as training data and visualize the activations of these images through different layers in the model. Use Keras.Model(inputs=model.input, outputs=[layer.output for layer in model.layers]) to get all activations for each layer.
  - Use the convention that the brighter each pixel, the closer that pixel to 1 and analyze the properties the filters learn in different Convolutional layers.

Datasets:

In this project, I will use both Fashion MNIST dataset and CIFAR_10 dataset to analyze the choice of models in different datasets. The Fashion MNIST and CIFAR_10 datasets are very different (in Fashion MNIST dataset, all objects have the same scale and there are no background to mislead from training; in CIFAR_10 dataset, the objects in the dataset do not necessarily have the same scale and there are background to mislead from training).

Fashion MNIST:

Source and Information:

Fashion-MNIST dataset is a dataset of Zalando's article images. The datset contains 60000 training samples and 10000 testing samples and each sample is a 28x28 grayscale image, associated with a label from 10 classes in the dataset.

10 classes of Fashion MNIST dataset are:

0. T-shirt/top
1. Trouser
2. Pullover
3. Dress
4. Coat
5. Sandal
6. Shirt
7. Sneaker
8. Bag
9. Ankle boot

Dataset layout:

The dataset used in this project is the csv version of the original fashion MNIST dataset, which is downloaded from Kaggle in this link: https://www.kaggle.com/zalando-research/fashionmnist (https://www.kaggle.com/zalando-research/fashionmnist).

The dataset can be download through the link (Kaggle) provided above or can be directly download here: https://www.kaggle.com/zalando-research/fashionmnist/downloads/fashionmnist.zip/4 (https://www.kaggle.com/zalando-research/fashionmnist/downloads/fashionmnist.zip/4).

CIFAR_10 dataset:

Source and Information:

The CIFAR-10 dataset contains 60000 32x32 colour images in 10 classes (6000 images/class) and is divided into 2 parts:

- 50000 images for training (5000 images for each class).
- 10000 images for testing (1000 images for each class).

The classes are completely mutually exclusive and there is no overlap between classes.

10 classes of CIFAR-10 dataset are:

0. airplane
1. automobile
2. bird
3. cat
4. deer
5. dog
6. frog
7. horse
8. ship
9. truck

The dataset were collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton and can be accessed through the link provided here : https://www.cs.toronto.edu/~kriz/cifar.html (https://www.cs.toronto.edu/~kriz/cifar.html) or directly download here: https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz (https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz).

Dataset layout:

The CIFAR-10 datset used in this project is the Python Version.

The download compressed file contains 7 files: data_batch_1, data_batch_2, data_batch_4, data_batch_5,test_batch, and batches.meta, which are Python "pickled" objects.

The source of the dataset (https://www.cs.toronto.edu/~kriz/cifar.html (https://www.cs.toronto.edu/~kriz/cifar.html)) provides a function called 'unpickle(file)' to open such files and returns the data as dictionaries.

Collaboration:

I have done this project alone.