# Final Project Individual Report

Joanne Chung
Group 7

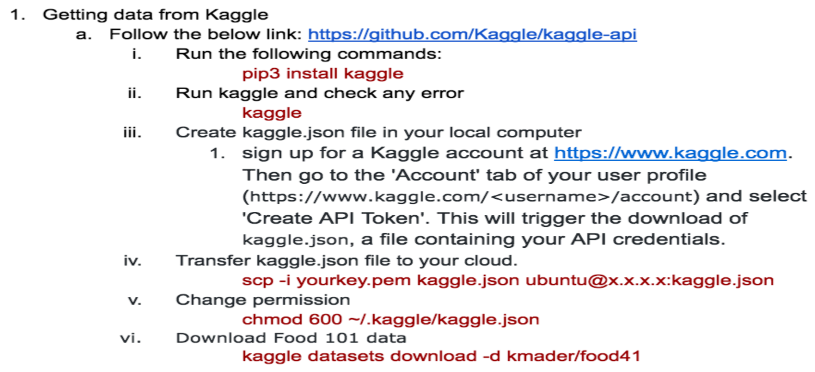
## 1. Introduction. An overview of the project and an outline of the shared work.

Businesses such as Yelp contain pictures of menu items relevant to particular restaurants uploaded by users to accompany their text reviews. The addition of these images to the restaurant review page adds value to the Yelp business model, giving other users a visual depiction of the menu items they are reviewing. My team make a project process(preprocess, design a model, train and test a model, update a model conclusion) and work together each step.

## 2. Description of your individual work. Provide some background information on the development of the algorithm and include necessary equations and figures.

First work is to read HDF5 file and create train_loader for PyTorch CNN model. A prototype code is only read HDF5 file and it can't make a mini-batch and tensor model After advising from you, I found a way to create train_loader in PyTorch. Second work is to find a best CNN Model. During the training, find a best model which is 6 convolutions and max-pooling layers with 0.5 Dropout. Last work is to make a plot for showing sample images. Reading images from a HDF5 files and show twenty images with labels.

The below figure is how go get data from Kaggle.



1. Getting data from Kaggle
   a. Follow the below link: https://github.com/Kaggle/kaggle-api
      i. Run the following commands:
         pip3 install kaggle
      ii. Run kaggle and check any error
         kaggle
      iii. Create kaggle.json file in your local computer
         1. sign up for a Kaggle account at https://www.kaggle.com. Then go to the 'Account' tab of your user profile (https://www.kaggle.com/<username>/account) and select 'Create API Token'. This will trigger the download of kaggle.json, a file containing your API credentials.
      iv. Transfer kaggle.json file to your cloud.
         scp -i yourkey.pem kaggle.json ubuntu@x.x.x.x:kaggle.json
      v. Change permission
         chmod 600 ~/.kaggle/kaggle.json
      vi. Download Food 101 data
         kaggle datasets download -d kmader/food41

## 3. Describe the portion of the work that you did on the project in detail. It can be figures, codes, explanation, pre-processing, training, etc.

First, reading HDF5 file and making train_loader for PyTorch CNN model is pre-processing part. Second, plotting sample images is data overview. Last, finding a best CNN model is related training.

```python
# ----------- 1. Read HDF5 files and make train_loader for PyTorch
class DatasetFromHdf5(torch.utils.data.Dataset):
    def __init__(self, file_path):
        super(DatasetFromHdf5, self).__init__()
        hf = h5py.File(file_path)
        self.data = hf.get('images')
        self.target = hf.get('labels')
        self.classes = hf.get('categories')

    def __getitem__(self, index):
        return torch.from_numpy(self.data[index, :, :, :].T).float(), self.target[index]

    def __len__(self):
        return self.data.shape[0]

# create datasets and data loaders
train_dataset = DatasetFromHdf5(h5train_file)
test_dataset = DatasetFromHdf5(h5test_file)
train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
test_loader = torch.utils.data.DataLoader(test_dataset, batch_size=batch_size, shuffle=False)
```

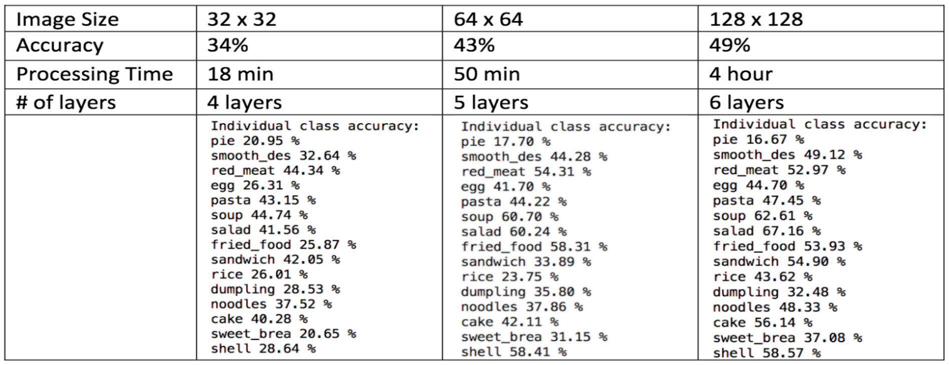
```python
# ----------- 2. Show sample images
# read in image data
image_data = h5py.File("food_train.h5", "r")
print('training database file:', image_data.filename)

# get labels and images for training data
labels = image_data.get('labels')
images = image_data.get('images')
target_classes = list(target_class.decode() for target_class in image_data.get("categories"))
print('')
print('image labels:', target_classes)
print('input shape:', images.shape)
print('labels shape:', labels.shape)

# show first 20 images
samplesize = 20
idx = slice(0, samplesize)
sample_labels = list(target_classes[x] for x in labels[idx])
sample_images = images[idx]
fig, m_x = plt.subplots(4, 5, figsize = (12, 12))
for ax, i in zip(m_x.flatten(), range(samplesize)):
    ax.imshow(sample_images[i])
    ax.set_title(sample_labels[i])
    ax.axis('off')
    # print(sample_labels[i])
    # print(sample_images[i])
    # print(labelnames[np.argmax(sample_labels[i])])
plt.show()
```

```python
# ----------- 3. Find a best CNN model
# define the network model
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.layer1 = nn.Sequential(                        # input size = 128 x 128
            nn.Conv2d(3, 64, kernel_size=10, padding=4),
            nn.BatchNorm2d(64),
            nn.ReLU(),
            nn.MaxPool2d(2, stride=2, padding=1))   # output size = 64 x 64
        self.layer2 = nn.Sequential(
            nn.Conv2d(64, 128, kernel_size=10, padding=4),
            nn.BatchNorm2d(128),
            nn.ReLU(),
            nn.MaxPool2d(2, stride=2, padding=1))   # output size = 32 x 32
        self.layer3 = nn.Sequential(
            nn.Conv2d(128, 256, kernel_size=10, padding=4),
            nn.BatchNorm2d(256),
            nn.ReLU(),
            nn.MaxPool2d(2, stride=2, padding=1))   # output size = 16 x 16
        self.layer4 = nn.Sequential(
            nn.Conv2d(256, 384, kernel_size=10, padding=4),
            nn.BatchNorm2d(384),
            nn.ReLU(),
            nn.MaxPool2d(2, stride=2, padding=1))   # output size = 8 x 8
        self.layer5 = nn.Sequential(
            nn.Conv2d(384, 256, kernel_size=4, padding=2),
            nn.BatchNorm2d(256),
            nn.ReLU(),
            nn.MaxPool2d(2, stride=2, padding=1))   # output size = 5 x 5
        self.layer6 = nn.Sequential(
            nn.Conv2d(256, 128, kernel_size=2),
            nn.BatchNorm2d(128),
            nn.ReLU(),
            nn.MaxPool2d(2, stride=2))              # output size = 2 x 2
        self.dropout = nn.Dropout(0.5)
        self.fc = nn.Linear(2 * 2 * 128, 12)
```

**4. Results. Describe the results of your experiments, using figures and tables wherever possible. Include all results (including all figures and tables) in the main body of the report, not in appendices. Provide an explanation of each figure and table that you include. Your discussions in this section will be the most important part of the report.**

At first, just reading HDF5 file with reading labels and images is not efficiently. Because we can't use mini-batch. So, I tried to create train_loader of CNN in PyTorch. Using train_loader in PyTorch, we are able to use Tensor and mini-batch. Second, I tested different size of images and compared accuracy based on similar CNN model. The below figure shows 128x128 pixels is the best size to use this project.

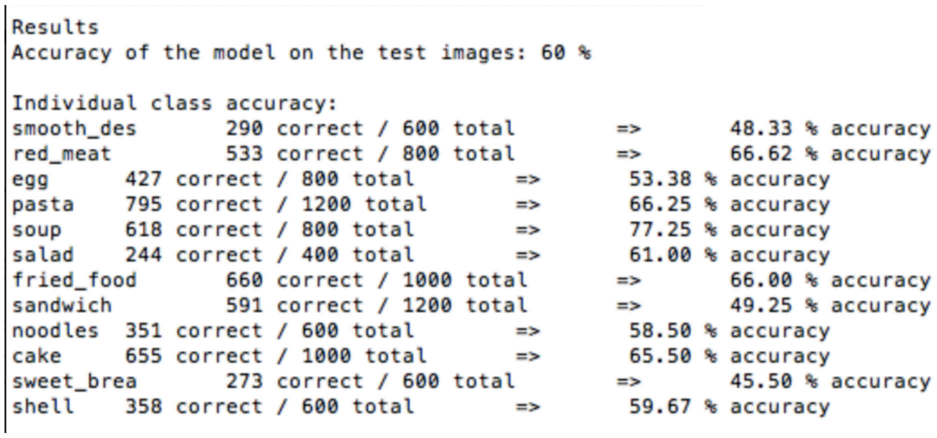
PyTorch CNN model – Food Image 101



| Image Size | 32 x 32 | 64 x 64 | 128 x 128 |
|---|---|---|---|
| Accuracy | 34% | 43% | 49% |
| Processing Time | 18 min | 50 min | 4 hour |
| # of layers | 4 layers | 5 layers | 6 layers |
|  | Individual class accuracy:<br>pie 20.95 %<br>smooth_des 32.64 %<br>red_meat 44.34 %<br>egg 26.31 %<br>pasta 43.15 %<br>soup 44.74 %<br>salad 41.56 %<br>fried_food 25.87 %<br>sandwich 42.05 %<br>rice 26.01 %<br>dumpling 28.53 %<br>noodles 37.52 %<br>cake 40.28 %<br>sweet_brea 20.65 %<br>shell 28.64 % | Individual class accuracy:<br>pie 17.70 %<br>smooth_des 44.28 %<br>red_meat 54.31 %<br>egg 41.70 %<br>pasta 44.22 %<br>soup 60.70 %<br>salad 60.24 %<br>fried_food 58.31 %<br>sandwich 33.89 %<br>rice 23.75 %<br>dumpling 35.80 %<br>noodles 37.86 %<br>cake 42.11 %<br>sweet_brea 31.15 %<br>shell 58.41 % | Individual class accuracy:<br>pie 16.67 %<br>smooth_des 49.12 %<br>red_meat 52.97 %<br>egg 44.70 %<br>pasta 47.45 %<br>soup 62.61 %<br>salad 67.16 %<br>fried_food 53.93 %<br>sandwich 54.90 %<br>rice 43.62 %<br>dumpling 32.48 %<br>noodles 48.33 %<br>cake 56.14 %<br>sweet_brea 37.08 %<br>shell 58.57 % |

**5. Summary and conclusions. Summarize the results you obtained, explain what you have learned, and suggest improvements that could be made in the future.**

From this project, I have learned how data preprocessing is essential and important. We spent 60% of time for preprocessing data. HDF5 file and OpenCV are helpful to manage data. Also, I have learned CNN is a perfect tool for image classification. During project, my team used Pytorch. PyTorch makes us to develop a model with small line of codes and should be careful to calculate an image size.



```
Results
Accuracy of the model on the test images: 60 %

Individual class accuracy:
smooth_des      290 correct / 600 total        =>       48.33 % accuracy
red_meat        533 correct / 800 total        =>       66.62 % accuracy
egg     427 correct / 800 total        =>       53.38 % accuracy
pasta   795 correct / 1200 total       =>       66.25 % accuracy
soup    618 correct / 800 total        =>       77.25 % accuracy
salad   244 correct / 400 total        =>       61.00 % accuracy
fried_food      660 correct / 1000 total       =>       66.00 % accuracy
sandwich        591 correct / 1200 total       =>       49.25 % accuracy
noodles 351 correct / 600 total        =>       58.50 % accuracy
cake    655 correct / 1000 total       =>       65.50 % accuracy
sweet_brea      273 correct / 600 total        =>       45.50 % accuracy
shell   358 correct / 600 total        =>       59.67 % accuracy
```

**6. Calculate the percentage of the code that you found or copied from the internet. For example, if you used 50 lines of code from the internet and then you modified 10 of lines and added another 15 lines of your own code, the percentage will be**

First code for HDF5 file: 50 %
# 13 - 4
# 13 + 5

Second code for plotting images:  28%
# 8 - 3
# 8 + 10

Third code for CNN model: 0 %

**7. References.**

Reading HDF5 file
https://github.com/twtygqyy/pytorch-vdsr/blob/master/dataset.py