1-CIFAR-100 Classifier

Abstract:

- CIFAR-100 are labeled subsets of the 80 million tiny images dataset. They were collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton.
- This dataset is just like the CIFAR-10, except it has 100 classes containing 600 images each. There are 500 training images and 100 testing images per class.

API used:

- In this classifier problem I used Python, Numpy and PyTorch API. I choosed PyTorch because I used it recently in the udacity course.

Classifier:

- At the beginning I started with making the model required in the challenge exactly.
    1. I added two fully connected layers at the end of the model
    2. The learning rate is 0.01
    3. I started the training with 50 epochs
    4. I retrain the model with 100 epochs

Result:

- In the first 50 epochs the less validation loss was: 0.5599
  and as we noticed the model is starting to overfit.

```
Epoch: 44       Training Loss: 2.439327         Validation Loss: 0.559917
Validation loss decreased (0.569416 --> 0.559917).  Saving model ...
Epoch: 45       Training Loss: 2.432260         Validation Loss: 0.570649
Epoch: 46       Training Loss: 2.419281         Validation Loss: 0.570470
Epoch: 47       Training Loss: 2.414710         Validation Loss: 0.571579
Epoch: 48       Training Loss: 2.418477         Validation Loss: 0.562668
Epoch: 49       Training Loss: 2.416605         Validation Loss: 0.569048
Epoch: 50       Training Loss: 2.409982         Validation Loss: 0.565980
```

- The test loss was: 2.77 and the overall test accuracy was: 30%

- After finishing all 100 epochs the validation loss didn’t changed so much, less validation loss = 0.5122
- Test loss = 2.549 and overall test accuracy = 34%

```
Epoch: 94       Training Loss: 2.288477         Validation Loss: 0.512217
Validation loss decreased (0.523409 --> 0.512217).  Saving model ...
Epoch: 95       Training Loss: 2.290610         Validation Loss: 0.522618
Epoch: 96       Training Loss: 2.291896         Validation Loss: 0.529860
Epoch: 97       Training Loss: 2.285128         Validation Loss: 0.530260
Epoch: 98       Training Loss: 2.285722         Validation Loss: 0.520538
Epoch: 99       Training Loss: 2.283131         Validation Loss: 0.515918
Epoch: 100      Training Loss: 2.288603         Validation Loss: 0.528941
```

- You can see the jupeter notebook of the detailed code in the “required model” folder. Also you can see the trained model and test scripts.

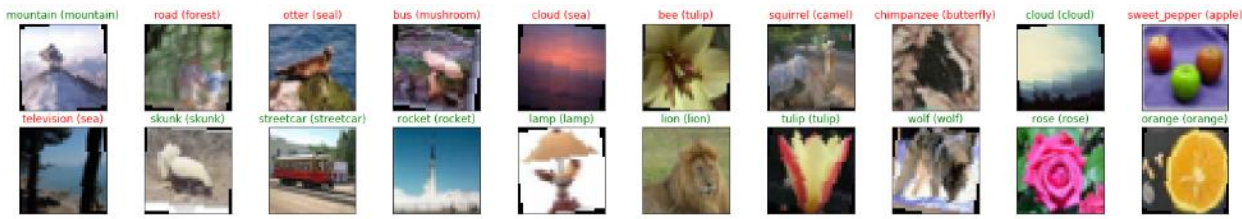
Enhancement classifier:

- I added Relu function to the conv layer and removed the BatchNorm layers, also put the dropout layers only after the flatten layer and the fully connected layer

Result:

- The validation loss decreased much better after 50 epoch validation loss was : 0.428. and after the 100 epochs the validation loss was about: 0.38
- Test loss = 1.879 and overall test accuracy = 50%

```
Epoch: 48       Training Loss: 1.746202         Validation Loss: 0.434916
Epoch: 49       Training Loss: 1.728068         Validation Loss: 0.431057
Validation loss decreased (0.431290 --> 0.431057).  Saving model ...
Epoch: 50       Training Loss: 1.718122         Validation Loss: 0.428084
Validation loss decreased (0.431057 --> 0.428084).  Saving model ...
```



- You can see the jupeter notebook of the detailed code in the “simple model” folder. Also you can see the trained model and test scripts.

Run method:

- Github link: https://github.com/Ahmed3sam/computer_vision_challenge
- You will found jupyter notebooks of all codes
- You might run the test script (with any console or with jupyter) to only see the test accuracy and predictions without retrain

Notes:

- The training time excceds 10 hours for the script.(using CPU)
- You should change the path of the dataset before start running the codes

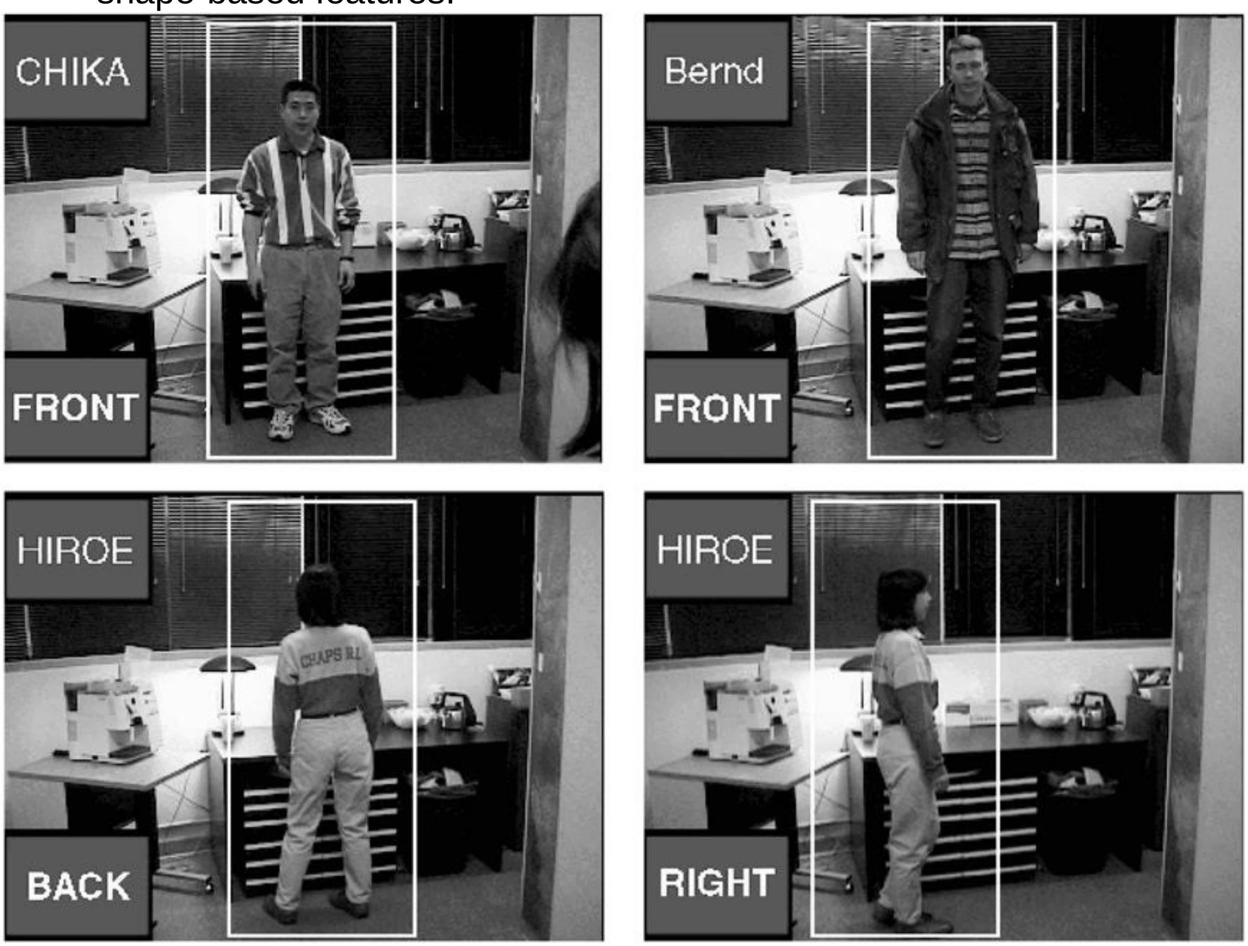
2- Problem:

Motivation:

- The problem: A client want to be able to track multiple people inside a store to identify patterns followed by their customers while shopping.
- Potentials: one static camera streaming live from inside the store.
- Explain

Solutions:

- In opposite of amazon go, the client here offer limited resources to track the customers in the store, so all what we have here to make process with the live stream of the camera to can track them
    1. We can make a Full-body person recognition system performed by multi-class SVMs that were trained on color images of people. The images were represented by different sets of color and shape-based features.



2. Make face recognition systems or object recognition with the convolutional neural networks

3. Add heat map system to the camera as we can see the heat maps of the customers

Different Technologies:

- Of course like we saw in the amazon go video, there are alot of technologies that we can use to track people in the store:
    1. Smartphones:
       We can use the smartphones in tracking people with many ways like:
        - Generating QR code for each customer or by email
        - Wifi signal tracking , every phone has unique mac adress
    2. Path tracker RFID tags at the buttom of every grocery cart in a supermarket
    3. Using loyality programs to track what customers buy
    4. Adobe Labs: The project is built on the Adobe Cloud Platform and pulls in data from Internet of Things sensors, beacons and if the brand has one, an app. Additionally, depending on the retailer, data is also taken from point-of-sale systems and online data.

References:

- https://www.cio.com/article/2383681/5-ways-to-track-in-store-customer-behavior.html
- http://knowledge.wharton.upenn.edu/article/tag-team-tracking-the-patterns-of-supermarket-shoppers/
- http://www.vcatechnology.com/footfall-heatmap-analysis
- https://www.adweek.com/digital/adobes-newest-labs-project-can-track-in-store-customers-in-real-time/