README

- **Pytorch version**
  1.0.0
- **The training argument of the classifier**
  Batch size: 4
  Learning rate: 0.001
  Optimization method: SGD
  Epoch number: 6
- **Classification testing accuracy**

```
Accuracy of the network on the 10000 test images: 64 %
Accuracy of plane : 53 %
Accuracy of   car : 76 %
Accuracy of  bird : 52 %
Accuracy of   cat : 49 %
Accuracy of  deer : 64 %
Accuracy of   dog : 49 %
Accuracy of  frog : 75 %
Accuracy of horse : 62 %
Accuracy of  ship : 86 %
Accuracy of truck : 75 %
```

- **The tracking testing result**
  With classifier_papam.pth:

```
OTB2013 Best: result/OTB2013/DCFNet_test(0.5353)
```

  With standard tracking model stored in param.pth:

```
OTB2013 Best: result/OTB2013/DCFNet_test(0.6375)
```

- **Description of how the neural network is trained**
  Forward propagation
  During the forward propagation, we take a training image and pass it through the whole network. On our first training example, since all of the weights values in kernels were randomly initialized, the output will probably be something like [.1 .1 .1 .1 ... .1], basically an output doesn't give preference to any number in particular. The network, with its current weights, isn't able to look for those low-level features (e.g., curves, edges, corners, etc.) or thus isn't able to make any reasonable conclusion about what the classification might be. This goes to the loss function part of the backpropagation.

  Back propagation
  A loss function can be defined in many different ways. Let's say the output of error function is L. As we can imagine, the loss will be extremely high for the first couple of training images. In order to get the predicated label as close as possible to the training label, we want to minimize the amount of loss we have. Visualizing this as just as optimization problem in calculus, we want to find out which inputs (weights in our case) most directly contributed to the loss of the network. This is the mathematical equivalent of a dL/dW where W are the weights at a particular layer. The backprob is performed through the network to determine which weights contributed most to the

loss and find ways to adjust them so that the loss decreases. Once we compute this derivative, we then apply gradient descent to update the all the weights of the filters. The learning rate is a hyper-parameter which controls the step we take each time to update the weights. The bigger the steps are and the less time the model takes to converge on an optimal set of weights. However, a learning rate that is too high could result in jumps that are too large and not precise enough to reach the optimal point.

The process of forward propagation, back propagation and weights update is one training epoch. The program will repeat this process for a fixed number of iterations.

- **Description of how the tracker work**
  The DCF network is constituted with two components: feature extractor and correlation filter.
  Feature extractor contains some convolutional layers which encode the prior tracking knowledge in the off-line training process. Behind these convolutional layers is the correlation filter layer. Correlation filters are a class of classifiers, which are specifically optimized to produce sharp peaks in the correlation output, primarily to achieve accurate localization of targets in scenes. A correlation filter layer can efficiently complete online learning and tracking by defining the network output as the probability heatmap of object location.
  Specifically, a video is split into many single frames and each frame corresponds to an image. Given an image to be tracked, DCFNet first obtains the feature of target image using feature extractor and adds hanning window to the feature. Now we get the searching patch of image and manage to get the gaussian response by doing correlation operator between the searching patch and discriminant correlation filter. The peak value and its location of response are found in different scale. Once the best scale of peak is found, the coordination of the peak response can be determined and hence we get the shift value of target object. Then the image is equipped with new bounding box and the weights of correlation filter are updated.
  This process contains forward propagation and weights update (i.e. backprop), therefore the DCF is updated online.