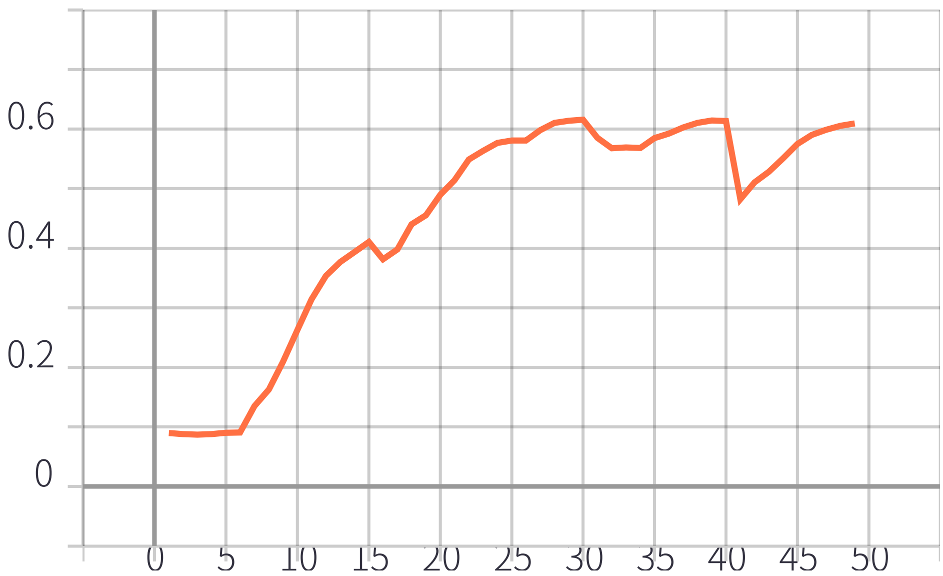
DLCV
HW2_REPORT

R07945010
許展銘
collaborator(r07942094 陳柏文)

- Baseline Model

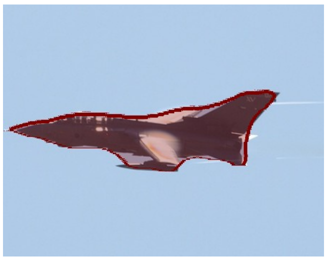
1. Pre-Process:
   ColorJitter() in torchvision.transforms to randomly change the brightness, contrast and saturation of an image.
   Normalize() in torchvision.transforms to do the normalization using the default value same as the imagenet pretrain.
2.
   - Training loss versus number of training iterations(about 8500 iter)

   - IoU score on validation set versus number of training iterations(50 epoch)

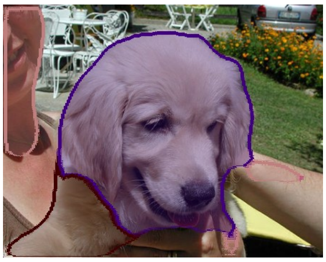
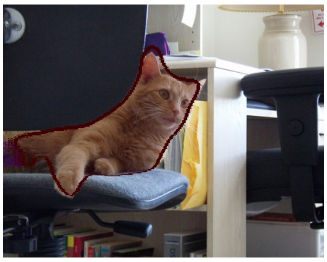


3. 左至右，上到下依序為 class0 ~ class8







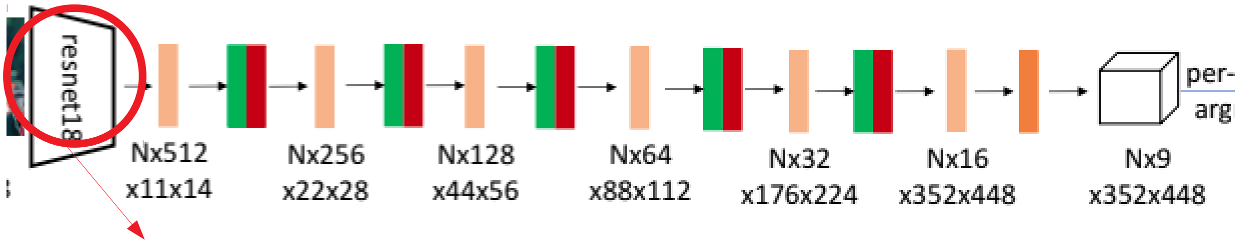




4. mIoU score and per-class IoU score in validation set

```
class #0 : 0.90016
class #1 : 0.73246
class #2 : 0.66146
class #3 : 0.68062
class #4 : 0.32371
class #5 : 0.48778
class #6 : 0.55084
class #7 : 0.69821
class #8 : 0.62506

mean_iou: 0.628923

Testing mIoU: 0.6289231044202348
```

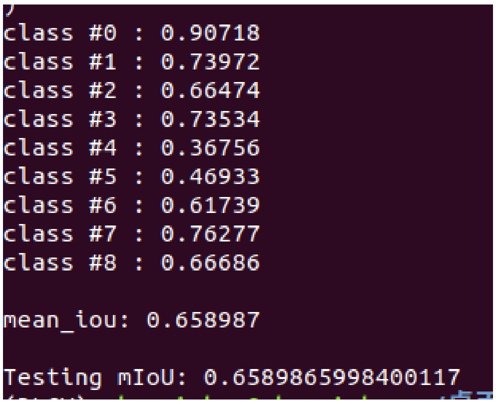
highest IoU score: class 0
lowest IoU score: class 4
reason:
Because there is always something different in TV, so it will increase the classification difficulty, and most of images may consist of many backgrounds, so it is easy for model to preclude the background from specific classes.

- Improved model
  1. model architecture
     (Reference: resnet, URL:https://github.com/KaimingHe/deep-residual-networks)



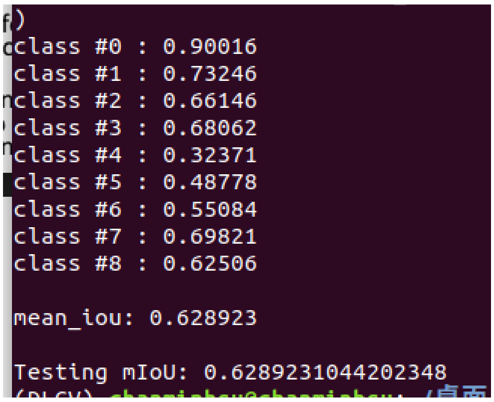
switch to resnet34

  2. First, from model architecture, I used the resnet34 with more convolution layers on the left to extract the feature, on the right side, using up-convolution layer to get the classification of each pixels. The bottleneck design can reduce the parameters efficiently. Additionally, the data augment performed in preprocess can also increase the training data set in order to improve the segmentation quality. Note that I added the dropout layer but comes out the worse mIoU, so I withdrew it in the model.
  3. mIoU score and per-class IoU score in validation set



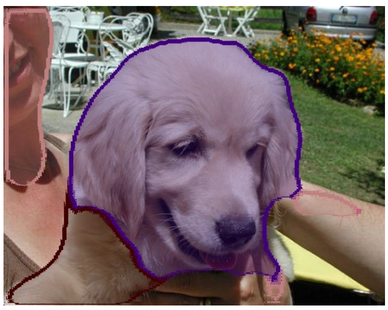
```
class #0 : 0.90718
class #1 : 0.73972
class #2 : 0.66474
class #3 : 0.73534
class #4 : 0.36756
class #5 : 0.46933
class #6 : 0.61739
class #7 : 0.76277
class #8 : 0.66686

mean_iou: 0.658987

Testing mIoU: 0.6589865998400117
```

Improved Model



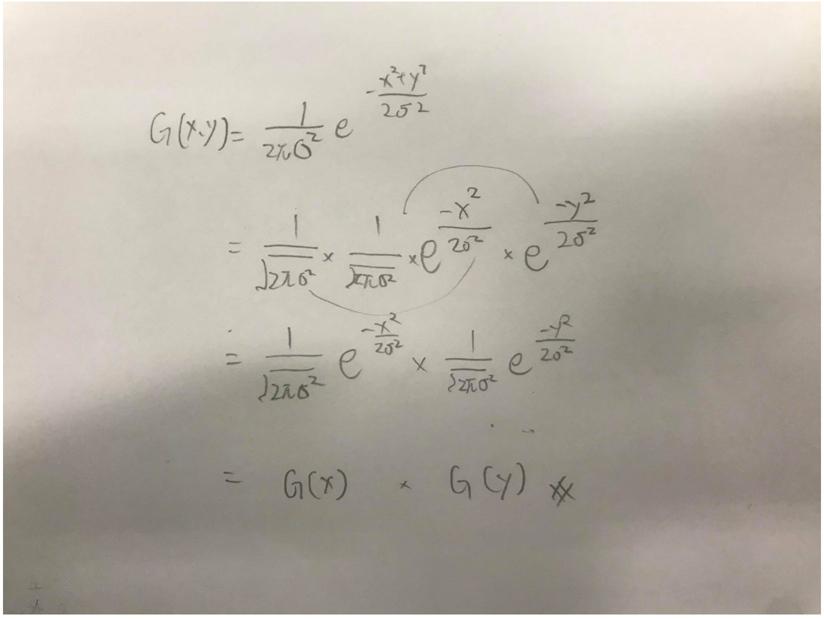
```
class #0 : 0.90016
class #1 : 0.73246
class #2 : 0.66146
class #3 : 0.68062
class #4 : 0.32371
class #5 : 0.48778
class #6 : 0.55084
class #7 : 0.69821
class #8 : 0.62506

mean_iou: 0.628923

Testing mIoU: 0.6289231044202348
```

Baseline Model



- Problem2

1.



$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$= \frac{1}{\sqrt{2\pi\sigma^2}} \times \frac{1}{\sqrt{2\pi\sigma^2}} \times e^{\frac{-x^2}{2\sigma^2}} \times e^{\frac{-y^2}{2\sigma^2}}$$

$$= \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-x^2}{2\sigma^2}} \times \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-y^2}{2\sigma^2}}$$

$$= G(x) \times G(y)$$

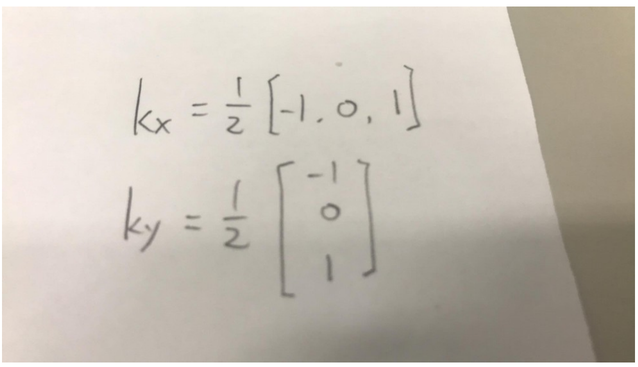
2.

lena.png

gaussain.png

We can see that the 2D Gaussian filter smooths the original lena.png.

3.



$$k_x = \frac{1}{2}[-1, 0, 1]$$

$$k_y = \frac{1}{2}\begin{bmatrix}-1\\0\\1\end{bmatrix}$$

Ix

Iy

Ix 是對橫軸做微分，故可以看出直線條。
Iy 是對縱軸做微分，故可以看出橫線條。

4.

lena.png 進行 gradient magnitude 的線條較清晰


Gaussian-filtered image 的影像則較為模糊的線條