Instructions

Step1:

对原数据进行缩放或者切割；（支持格式：jpeg、png）
Resize or crop raw data; (data format: jpeg,png)

Step2：

创建每个图像对应的 mask 图像；（支持格式：jpeg、png）（mask 图像必须为单通道图）

Create mask image for each data. (Mask image: jpeg, png; single channel)

步骤：(Steps :)

1、下载程序 JS Segment Annotator（见 readme 文件），解压缩（运行环境 linux）；

Download program - JS Segment Annotator & unzip it.

2、按照说明在/data 文件夹下，新建样本标注文件 xxx.json，修改 or 增加标注类别，分别按照样例添加图像和原始 mask 生成路径和名字；（样例文件 example.json）

3、修改/js 文件夹下 main.js 文件，使 dataURL 指向 你自己建立的 xxx.json 文件；

Follow the user guide of JS Segment Annotator, and then set your own labels and data.

4、运行主目录下文件 index.html 进入图像界面进行标注；（可以使用默认的 firefox 浏览器）

Make your own mask files.

5、保存每个图像生成的原始 mask 文件；

Save all mask files.

6、运行 segnet_labeling.m 程序（需要第三方函数 natsortfiles（见 readme 文件），建议加入为 Matlab 的库函数），将前面生成的原始 mask 图像（默认支持 png）转换为 segnet 的 mask 图像，并同时生成可视化图像（后缀为_colour）。在弹出对话框中分别输入：选择原始 mask 图像文件夹（图像不能放到桌面，必须在某个文件夹下）；生成的 segnet　mask 图像保存文件夹；

Run Matlab code - segnet_labeling.m to convert masks which you made above to the SegNet format. (Support png format default)

Note: 1) if you already have had another type mask file, you only need to create a Transforming Interface Program to make SegNet format mask file, and skip this step.
2) The mask image of SegNet is a label image whose pixel value means your original pixel class and position. So the mask image pixel value must be integer and maximum value must be your class number -1 （Because the label of Caffe is from 0 to max_class-1 ）. For example, if your project has 12 classes, the maximum value of your mask image pixel must not exceed 11. Otherwise, you will get an error like this:

Check failed: status == CUBLAS_STATUS_SUCCESS (11 vs. 0) CUBLAS_STATUS_MAPPING_ERROR

Step3：

建立 list 文件；（可以使用 create_list.sh）

Create list file.

格式：（数据与 mask 之间空一格）

```
/SegNet /train/0001TP_006690.png   /SegNet /trainannot/0001TP_006690.png
/SegNet /train/0001TP_006720.png   /SegNet /trainannot/0001TP_006720.png
```

Step4：

根据输入图像大小，修改网络 upsample 层的参数；

Change to your own parameter of upsample layer, according to input size.

Details:

The parameters of upsample layer: (code: see src\caffe\proto\caffe.proto)

```
message UpsampleParameter {
   // DEPRECATED. No need to specify upsampling scale factors when
   // exact output shape is given by upsample_h, upsample_w parameters.
   optional uint32 scale = 1 [default = 2];
   // DEPRECATED. No need to specify upsampling scale factors when
   // exact output shape is given by upsample_h, upsample_w parameters.
   optional uint32 scale_h = 2;
   // DEPRECATED. No need to specify upsampling scale factors when
   // exact output shape is given by upsample_h, upsample_w parameters.
   optional uint32 scale_w = 3;
   // DEPRECATED. Specify exact output height using upsample_h. This
   // parameter only works when scale is 2
   optional bool pad_out_h = 4 [default = false];
   // DEPRECATED. Specify exact output width using upsample_w. This
   // parameter only works when scale is 2
   optional bool pad_out_w = 5 [default = false];
   optional uint32 upsample_h = 6;
   optional uint32 upsample_w = 7;
}
```

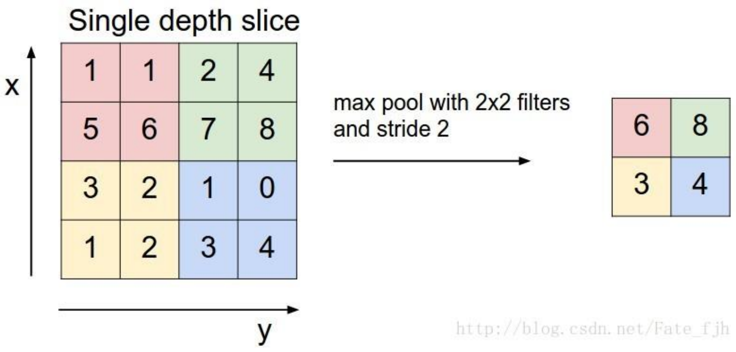
And the parameters: scale, scale_h, scale_w, pad_out_h, pad_out_w, upsample_h, upsample_w can be modified.

If you have exact output shape, you only need to modify “upsample_h” and “upsample_w”.

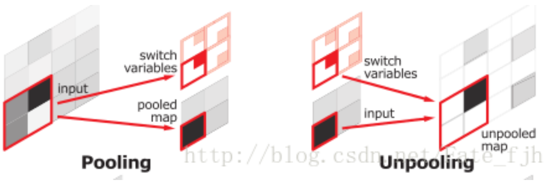
The author does not recommend to use others parameters; but I will give some explanations for how to use them.

1. Theoretical basis:

Pooling step



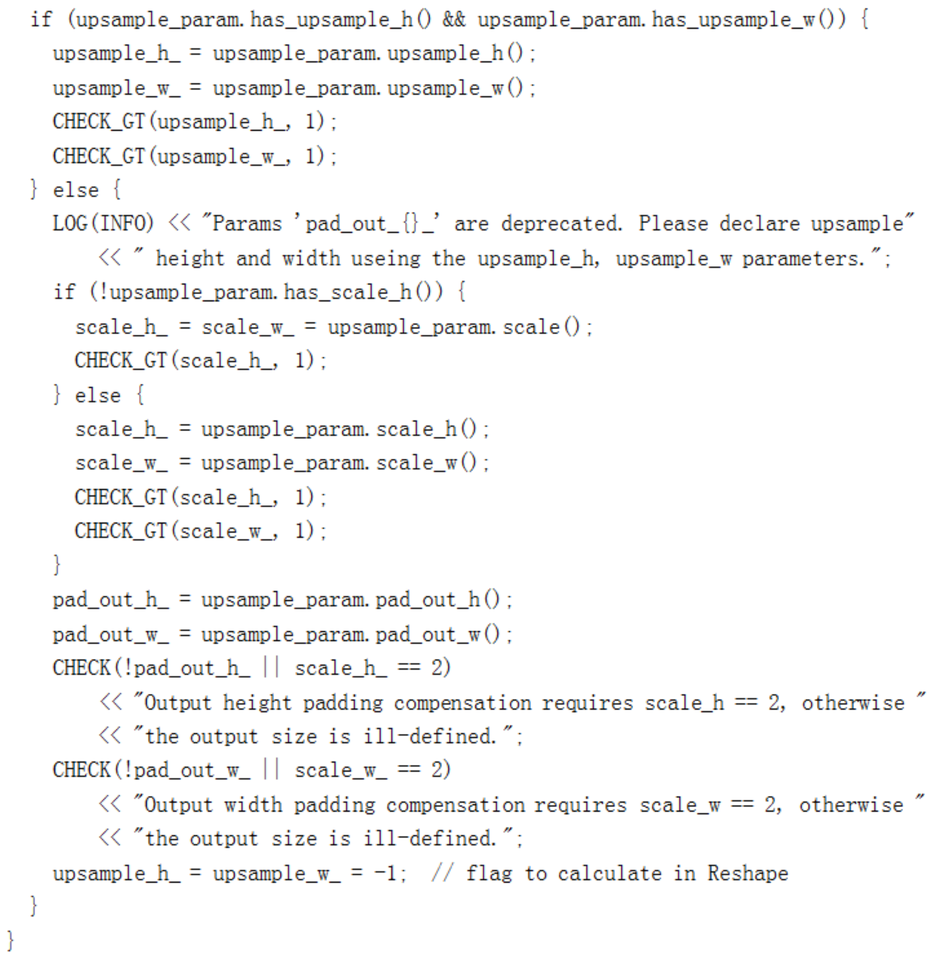
Upsampling



These two images explain: scale is 2 (2*2 pooling and upsampling), as 1/2 original width/height size (for pooling), as 2-fold pooling layer feature size (for upsampling).

2. Parameter pad_out: padding size for adjusting true size.

3. Parameter Priority: (code: see src\caffe\layers\upsample_layer.cpp)
I. Detect whether parameters upsample_h and upsample_w exist or not.
If exist, use them; if not, go to II.
II. Detect whether parameter scale_h exist or not.
If not, scale_h_ = scale_w_ = scale; if exist, go to III.
III. scale_h_ = scale_h, and scale_w_= scale_w.
Only scale_h_ = scale_w_ = 2, Parameters: pad_out_h，pad_out_w can be used.



```
  if (upsample_param.has_upsample_h() && upsample_param.has_upsample_w()) {
    upsample_h_ = upsample_param.upsample_h();
    upsample_w_ = upsample_param.upsample_w();
    CHECK_GT(upsample_h_, 1);
    CHECK_GT(upsample_w_, 1);
  } else {
    LOG(INFO) << "Params 'pad_out_{}_' are deprecated. Please declare upsample"
        << " height and width useing the upsample_h, upsample_w parameters.";
    if (!upsample_param.has_scale_h()) {
      scale_h_ = scale_w_ = upsample_param.scale();
      CHECK_GT(scale_h_, 1);
    } else {
      scale_h_ = upsample_param.scale_h();
      scale_w_ = upsample_param.scale_w();
      CHECK_GT(scale_h_, 1);
      CHECK_GT(scale_w_, 1);
    }
    pad_out_h_ = upsample_param.pad_out_h();
    pad_out_w_ = upsample_param.pad_out_w();
    CHECK(!pad_out_h_ || scale_h_ == 2)
        << "Output height padding compensation requires scale_h == 2, otherwise "
        << "the output size is ill-defined.";
    CHECK(!pad_out_w_ || scale_w_ == 2)
        << "Output width padding compensation requires scale_w == 2, otherwise "
        << "the output size is ill-defined.";
    upsample_h_ = upsample_w_ = -1;  // flag to calculate in Reshape
  }
}
```

4. Calculation formula

And when you use parameters of II/III, the output feature map size is:

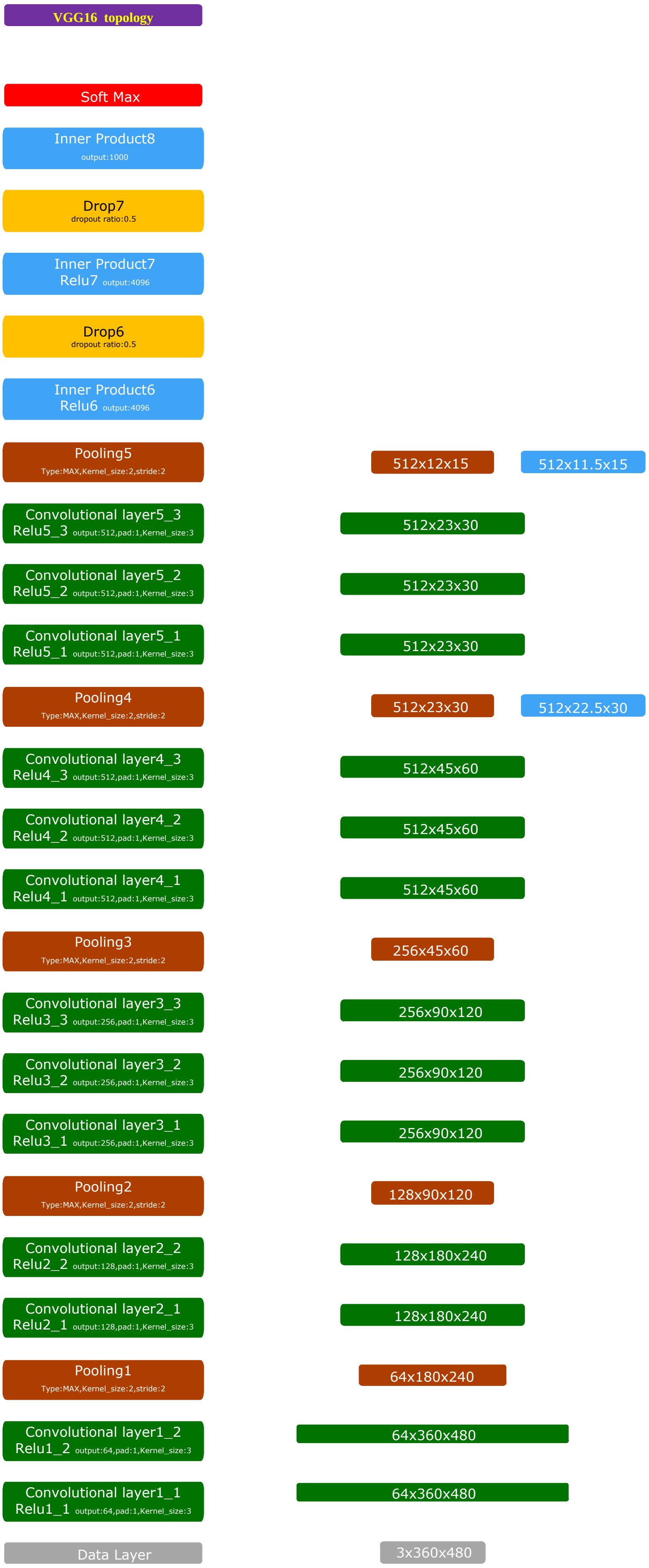
```
upsample_h_ = bottom[0]->height() * scale_h_ - int(pad_out_h_);
      // upsample_h_ = height * scale - pad_out
      // upsample_h_ = height * 2 - pad_out
upsample_w_ = bottom[0]->width() * scale_w_ - int(pad_out_w_);
```

```
void UpsampleLayer<Dtype>::Reshape(const vector<Blob<Dtype>*>& bottom,
      const vector<Blob<Dtype>*>& top) {
  CHECK_EQ(4, bottom[0]->num_axes()) << "Input must have 4 axes, "
      << "corresponding to (num, channels, height, width)";
  CHECK_EQ(4, bottom[1]->num_axes()) << "Input mask must have 4 axes, "
      << "corresponding to (num, channels, height, width)";
  CHECK_EQ(bottom[0]->num(), bottom[1]->num());
  CHECK_EQ(bottom[0]->channels(), bottom[1]->channels());
  CHECK_EQ(bottom[0]->height(), bottom[1]->height());
  CHECK_EQ(bottom[0]->width(), bottom[1]->width());

  if (upsample_h_ <= 0 || upsample_w_ <= 0) {
    upsample_h_ = bottom[0]->height() * scale_h_ - int(pad_out_h_);
    upsample_w_ = bottom[0]->width() * scale_w_ - int(pad_out_w_);
  }
  top[0]->Reshape(bottom[0]->num(), bottom[0]->channels(), upsample_h_,
      upsample_w_);
  channels_ = bottom[0]->channels();
  height_ = bottom[0]->height();
  width_ = bottom[0]->width();
}
```

5. Example

The original model of SegNet used CamVid Database.
（http://mi.eng.cam.ac.uk/research/projects/VideoRec/CamVid/ ）
And the size of CamVid Database's images is 360*480 (height*width).
Meanwhile the models in SegNet-Tutorial-master/Example_Models & SegNet-Tutorial-master/Models all used VGG16 structure.
(https://github.com/alexgkendall/SegNet-Tutorial)
So, the feature maps of each layer show as below:



VGG16 topology

Soft Max

Inner Product8
output:1000

Drop7
dropout ratio:0.5

Inner Product7
Relu7 output:4096

Drop6
dropout ratio:0.5

Inner Product6
Relu6 output:4096

Pooling5
Type:MAX,Kernel_size:2,stride:2
512x12x15
512x11.5x15

Convolutional layer5_3
Relu5_3 output:512,pad:1,Kernel_size:3
512x23x30

Convolutional layer5_2
Relu5_2 output:512,pad:1,Kernel_size:3
512x23x30

Convolutional layer5_1
Relu5_1 output:512,pad:1,Kernel_size:3
512x23x30

Pooling4
Type:MAX,Kernel_size:2,stride:2
512x23x30
512x22.5x30

Convolutional layer4_3
Relu4_3 output:512,pad:1,Kernel_size:3
512x45x60

Convolutional layer4_2
Relu4_2 output:512,pad:1,Kernel_size:3
512x45x60

Convolutional layer4_1
Relu4_1 output:512,pad:1,Kernel_size:3
512x45x60

Pooling3
Type:MAX,Kernel_size:2,stride:2
256x45x60

Convolutional layer3_3
Relu3_3 output:256,pad:1,Kernel_size:3
256x90x120

Convolutional layer3_2
Relu3_2 output:256,pad:1,Kernel_size:3
256x90x120

Convolutional layer3_1
Relu3_1 output:256,pad:1,Kernel_size:3
256x90x120

Pooling2
Type:MAX,Kernel_size:2,stride:2
128x90x120

Convolutional layer2_2
Relu2_2 output:128,pad:1,Kernel_size:3
128x180x240

Convolutional layer2_1
Relu2_1 output:128,pad:1,Kernel_size:3
128x180x240

Pooling1
Type:MAX,Kernel_size:2,stride:2
64x180x240

Convolutional layer1_2
Relu1_2 output:64,pad:1,Kernel_size:3
64x360x480

Convolutional layer1_1
Relu1_1 output:64,pad:1,Kernel_size:3
64x360x480

Data Layer
3x360x480

According to Caffe pooling & convolution layer formula, the size of feature map is integer. So the first unsampling layer parameters are upsample_w = 30, upsample_h = 23 (scale = 2 is optional parameter, you can omit it.); and the second unsampling layer parameters are upsample_w = 60, upsample_h = 45. But the rest layers' parameter is only scale = 2 (Because the size of feature map can be divided with no remainder).

Step5：

计算并修改 class_weighting 参数（在 loss 层）；
Compute the parameters (class_weighting) of loss layer.

可以使用 class_weighting_compute.m；使用方法：(Method:)
Matlab 运行，在弹出对话框中分别输入：类别总数；选择 mask 图片文件夹；结果保存的 txt 文件夹路径；txt 文件保存的名字。（注：mask 文件夹中图片不可太多，可能会溢出，可以分别做成若干个 txt 文件，再把对应类权重求均值，且 mask 图像格式默认为 png，如果为 jpeg 需要进行修改）
Run code class_weighting_compute.m, and you will get a txt file. Then set the value to loss layer.

Step6：

训练+测试。
Train & test.

注：由于 SegNet 结构限制，类别暂时不支持超过 256 类（0-255，8bit），程序在 Matlab2010a、Matlab2014a 和 Matlab2016a 测试通过。