2019

The works-end summary

General business template

For consulting, management, planning, reporting

If you want to see farther, you have to stand on your shoulders.
你想看的更远就要站在 巨人视野之上 （理想的判断和学习）

A man walks fast, a group walks far.
一个人走的很快一群人走的很远 （格局观）

The proposition

What we hope to achieve in the short and long run

Add The Title

目前AIoT产品的设计个人理解需要以下几个因素同时存在才能落地

1、一个懂模型思维的科技型带队

2、正式的真实场景数据

3、可以接收的误差

4、产品的安全问题可控

5、一个学习型创新团队

Part One

From Technical Support Service to Technical Leadership Service
从技术支持服务到技术领导服务

From Perceptual Cognition to Rational Realization
从感性认知到理性实现（技术不是万能的不能为某些人的无知买单）

技术产品研究中心主要内容

1、论文复现
2、场景数据理解和验证
3、服务部署与模型安全

The Objectives

What we hope to achieve in the short and long run

深度学习算法架构师

(考虑重点：模型的上下限、传统方法结合、各自优势、模型安全和系统安全稳定性)
深度学习模型代码维护成本非常高，一旦引入确保安全是第一，这些模型具备学习能力并且算法的写作人本人的代码风格需要非常专业的人才能在很长的时间去理解。

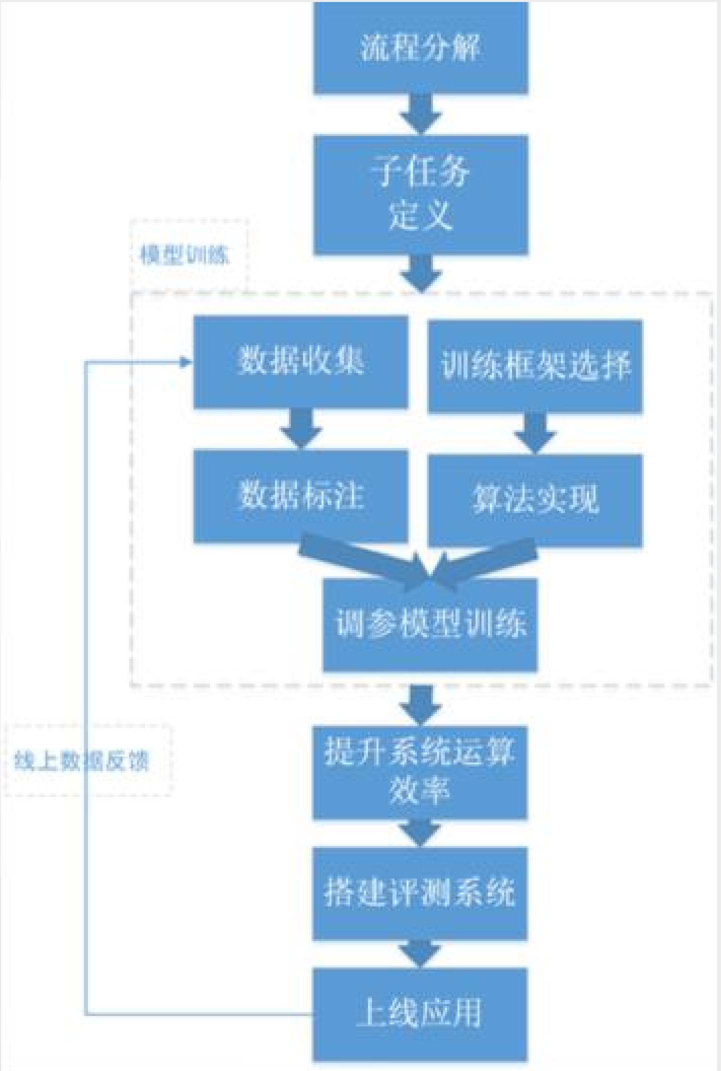
接下来我们将从以下两个产品落地的角度分析整个流程深度学习工程化项目的相关问题

设计一款高速的驾驶证识别系统

Process Flow

What we hope to achieve in the short and long run

技巧：使用预训练模型为自己数据打标签



The Objectives问题分析

50%论文阅读	20%工程调试问题分析	30%系统安全和稳定结构设计

时间和资源分配（问题找对了可以在整个过程节省50%的工程和后期维护成本）

ADD THE TITLE

- ✓ 驾驶证的识别是一个自然场景文本检测不是OCR
- ✓ OCR和STR的区别在于复杂的图像前后景问题和STR是文字定位和文字识别两个阶段而OCR只是单一的识别

ADD THE TITLE

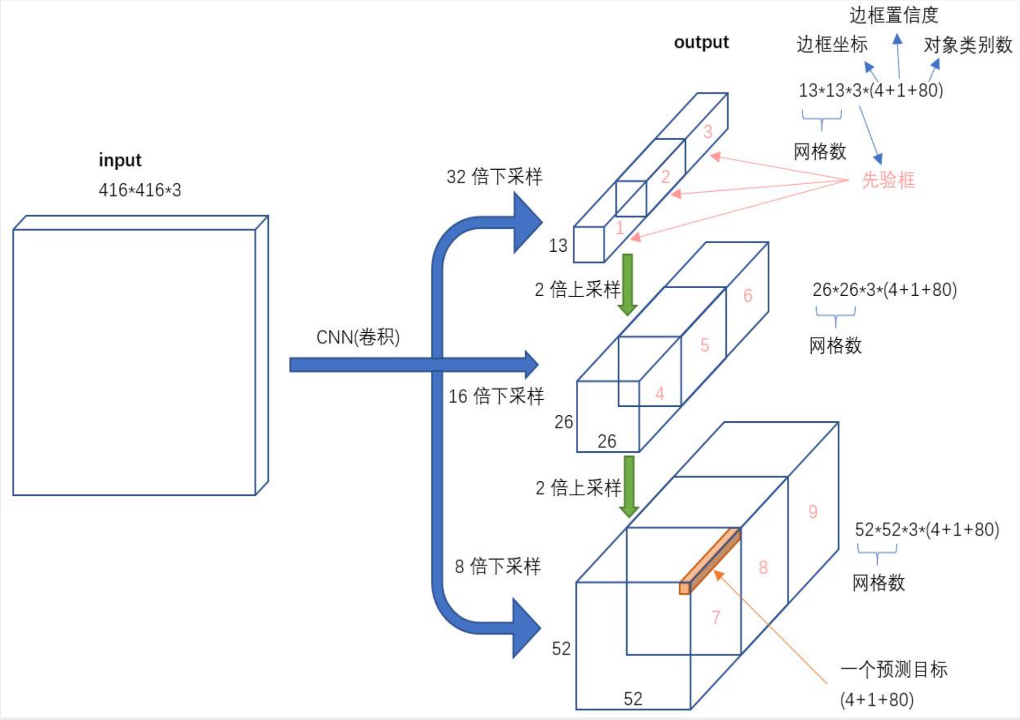
- ✓ 针对文本识别目前识别主流方法三个：CRNN、Attention、FAN
- ✓ 针对文字区域定位主流方法：CTPN、EAST、FOTS、PSENet、SOTA、MOSTR、DDR

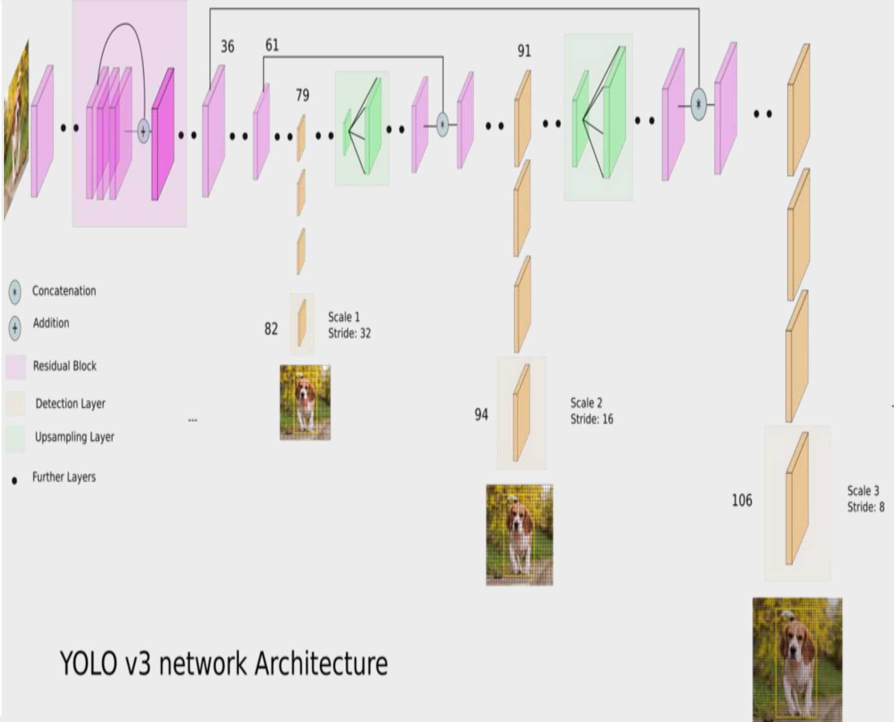
ADD THE TITLE

- ✓ 线下对预训练模型进行迁移学习测试
- ✓ 业务场景的定位准确性存在的问题
- ✓ 确定识别的准确率存在的问题
- ✓ 防止系统被模拟证件数据攻击和算法调用
- ✓ 预留系统自我防护机制

model farmwork

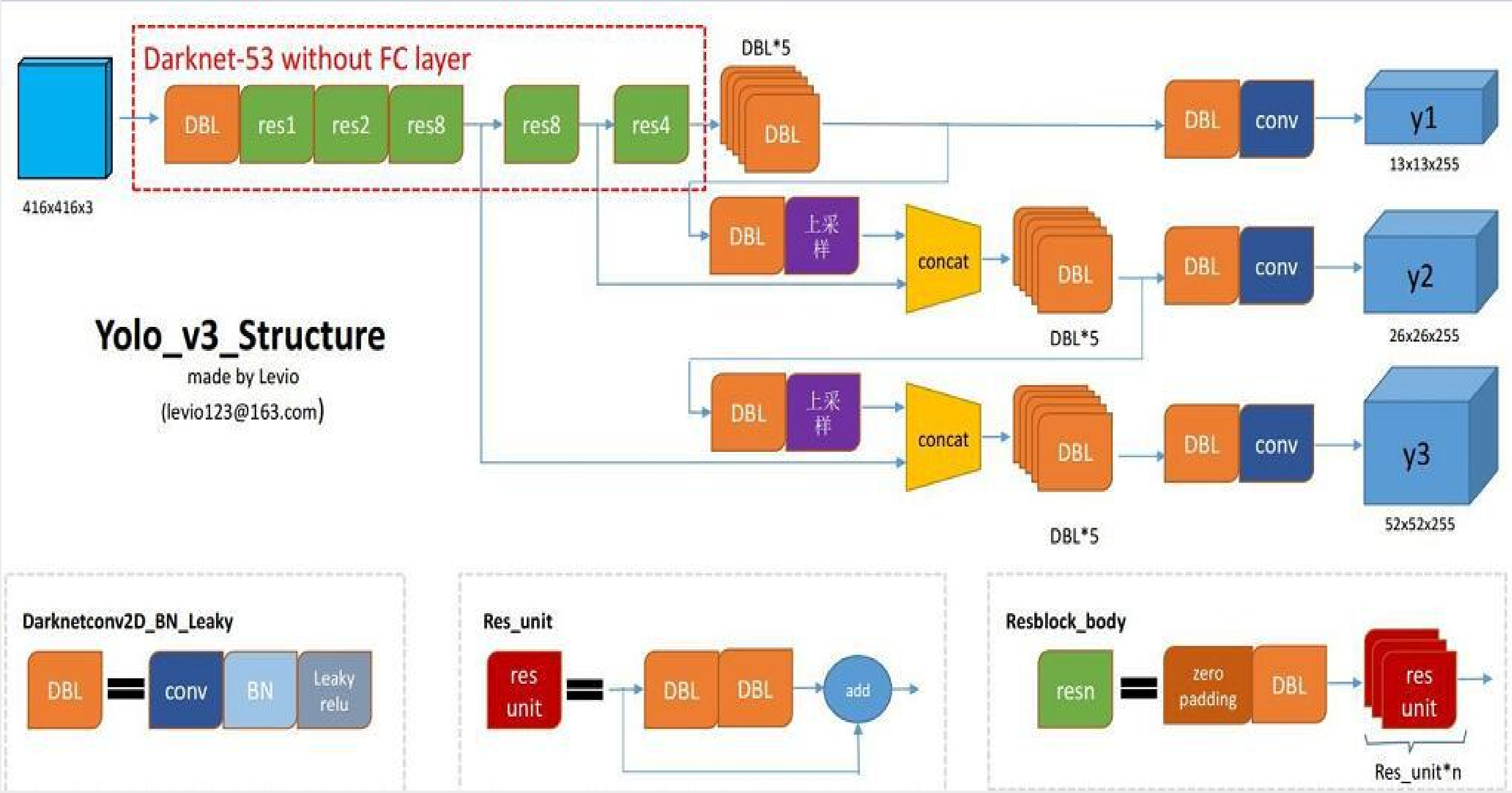
The concepts national income and national product have roughly the same value and can be used interchangeably if our interest is in their sum total which is





Structure

The structure of FPN



Add The Title

我们对该模型的细节通过注释代码和注释cfg文件来解释:

细节链接地址:

https://github.com/Eric3911/Dakrnet-YOLOv3/blob/master/%E4%BC%98%E5%8C%96%E8%AE%AD%E7%BB%83%E5%8F%82%E6%95%B0%E8%A7%A3%E9%87%8A

Three future

What we hope to achieve in the short and long run

53 = 2 + 1*2 + 1 + 2*2 + 1 + 8*2 + 1 + 8*2 + 1 + 4*2 + 1　　/ Darknet网络含有5组重复的resblock_body()单元

A DarkNet53下采样特征提取

```
darknet = Model(inputs, darknet_body(inputs))


def darknet_body(x):
    '''Darknent body having 52 Convolution2D layers'''
    x = DarknetConv2D_BN_Leaky(32, (3,3))(x)
    x = resblock_body(x, 64, 1)
    x = resblock_body(x, 128, 2)
    x = resblock_body(x, 256, 8)
    x = resblock_body(x, 512, 8)
    x = resblock_body(x, 1024, 4)
    return x
```

B 新的激活函数Leakrelu

```
def DarknetConv2D(*args, **kwargs):
    """Wrapper to set Darknet parameters for Convolution2D."""
    darknet_conv_kwargs = {'kernel_regularizer': l2(5e-4)}
    darknet_conv_kwargs['padding'] = 'valid' if kwargs.get('strides')==(2,2) else 'same'
    darknet_conv_kwargs.update(kwargs)
    return Conv2D(*args, **darknet_conv_kwargs)
def DarknetConv2D_BN_Leaky(*args, **kwargs):
    """Darknet Convolution2D followed by BatchNormalization and LeakyReLU."""
    no_bias_kwargs = {'use_bias': False}
    no_bias_kwargs.update(kwargs)
    return compose(
        DarknetConv2D(*args, **no_bias_kwargs),
        BatchNormalization(),
        LeakyReLU(alpha=0.1))
```

这个全新的激活函数相比softmax可以识别更多类别，这个也是解决fasterrcnn没法识别一张图片中多种类别而yolo系列可以的根本原因

Padding Style

Darknet uses left and top padding instead of 'same' mode

darknet每块之间使用了（1，0，1，0）的PADDING层

```
def resblock_body(x, num_filters, num_blocks):
    '''A series of resblocks starting with a downsampling Convolution2D'''
    # Darknet uses left and top padding instead of 'same' mode
    x = ZeroPadding2D(((1,0),(1,0)))(x)
    x = DarknetConv2D_BN_Leaky(num_filters, (3,3), strides=(2,2))(x)
    for i in range(num_blocks):
        y = compose(
                DarknetConv2D_BN_Leaky(num_filters//2, (1,1)),
                DarknetConv2D_BN_Leaky(num_filters, (3,3)))(x)
        x = Add()([x,y])
    return x
```

C note

darknet使用了向左和向上的填充替代的same模式

每个darknet块之间使用了1010的填充方式会导致卷积输入的尺寸不一样，w，h为img的参数，f为filter的size，s为步长stride

这种做法可以保证gridcell更加稳定

valid:

$$Output\_height = Output\_width = [\frac{W-F+1}{s}]$$

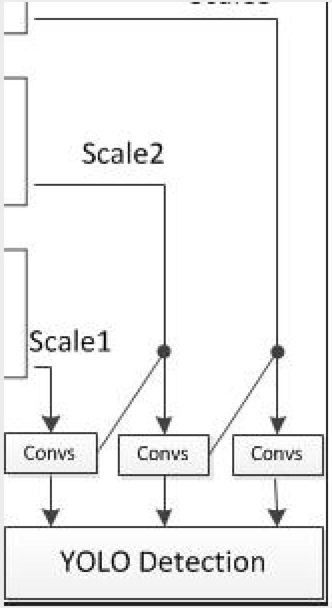
same:

$$Output\_height = Output\_width = [\frac{W}{S}]$$

The Detection Objectives

YOLOV3的借鉴了特征金字塔（FPN）的概念实现了物体定位


```
x, y1 = make_last_layers(darknet.output, 512, num_anchors*(num_classes+5))
x = compose(
            DarknetConv2D_BN_Leaky(256, (1,1)),
            UpSampling2D(2))(x)
x = Concatenate()([x,darknet.layers[152].output])
x, y2 = make_last_layers(x, 256, num_anchors*(num_classes+5))
x = compose(
            DarknetConv2D_BN_Leaky(128, (1,1)),
            UpSampling2D(2))(x)
x = Concatenate()([x,darknet.layers[92].output])
x, y3 = make_last_layers(x, 128, num_anchors*(num_classes+5))
return Model(inputs, [y1,y2,y3])
```

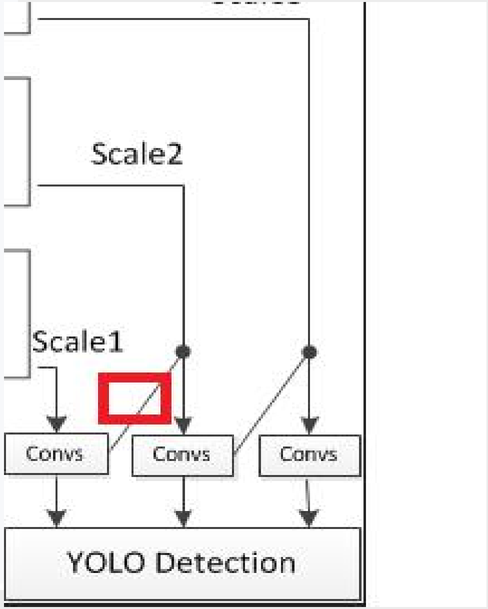


The Detection Objectives

Convs的实现

Convs由make_last_layers函数来实现。

```
def make_last_layers(x, num_filters, out_filters):
    '''6 Conv2D_BN_Leaky layers followed by a Conv2D_linear layer'''
    x = compose(
            DarknetConv2D_BN_Leaky(num_filters, (1,1)),
            DarknetConv2D_BN_Leaky(num_filters*2, (3,3)),
            DarknetConv2D_BN_Leaky(num_filters, (1,1)),
            DarknetConv2D_BN_Leaky(num_filters*2, (3,3)),
            DarknetConv2D_BN_Leaky(num_filters, (1,1)))(x)
    y = compose(
            DarknetConv2D_BN_Leaky(num_filters*2, (3,3)),
            DarknetConv2D(out_filters, (1,1)))(x)
    return x, y
```



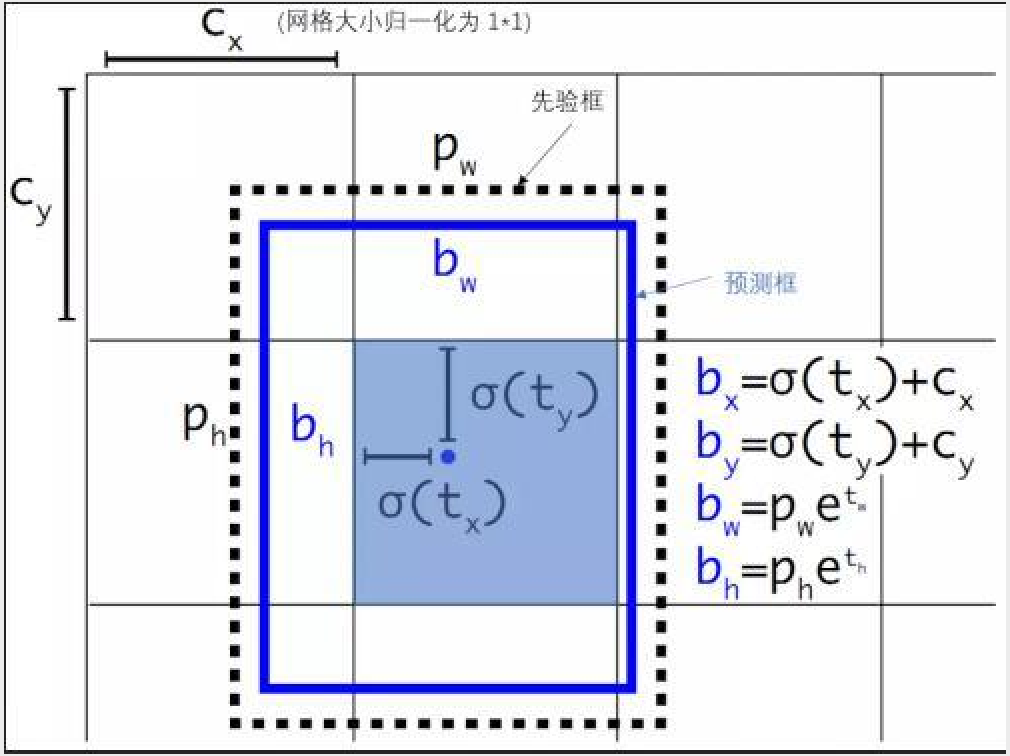
The Compose

Compose arbitrarily many functions, evaluated left to right.


```
x = compose(
            DarknetConv2D_BN_Leaky(256, (1,1)),
            UpSampling2D(2))(x)
x = Concatenate()([x,darknet.layers[152].output])

def compose(*funcs):
    """Compose arbitrarily many functions, evaluated left to right.
  Reference: https://mathieularose.com/function-composition-in-python/
  """
    # return lambda x: reduce(lambda v, f: f(v), funcs, x)
    if funcs:
        return reduce(lambda f, g: lambda *a, **kw: g(f(*a, **kw)), funcs)
    else:
        raise ValueError('Composition of empty sequence not supported.')
```

gridcell计算问题



THE TITLE

Yolo v3采用直接预测相对位置的方法。预测出b-box中心点相对于网格单元左上角的相对坐标。直接预测出（tx，ty，tw，th，t0），然后通过以下坐标偏移公式计算得到b-box的位置大小和confidence。

$$b_x = \sigma(t_x) + c_x$$
$$b_y = \sigma(t_x) + c_x$$
$$b_w = p_w e^{t_w}$$
$$b_h = p_h e^{t_h}$$
$$p_r(object) * IOU(b, object) = \sigma(t_o)$$

tx、ty、tw、th就是模型的预测输出。cx和cy表示grid cell的坐标，比如某层的feature map大小是13 × 13，那么grid cell就有13 × 13个，第0行第1列的grid cell的坐标cx就是0，cy就是1。pw和ph表示预测前bounding box的size。bx、by、bw和bh就是预测得到的bounding box的中心的坐标和size。在训练这几个坐标值的时候采用了sum of squared error loss（平方和距离误差损失），因为这种方式的误差可以很快的计算出来。

Yolo v3使用逻辑回归预测每个边界框的分数。如果边界框与真实框的重叠度比之前的任何其他边界框都要好，则该值应该为1。如果边界框不是最好的，但确实与真实对象的重叠超过某个阈值(Yolo v3中这里设定的阈值是0.5)，那么就忽略这次预测。Yolo v3只为每个真实对象分配一个边界框，如果边界框与真实对象不吻合，则不会产生坐标或类别预测损失，只会产生物体预测损失。

Loss Function

关键信息是需要确定的:(x,y),(w,h),class,confidence

yolov3的损失函数采用误差的平方和整合了预测框定位误差与有无目标的IOU误差以及分类误差。

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

坐标误差

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

IOU误差

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

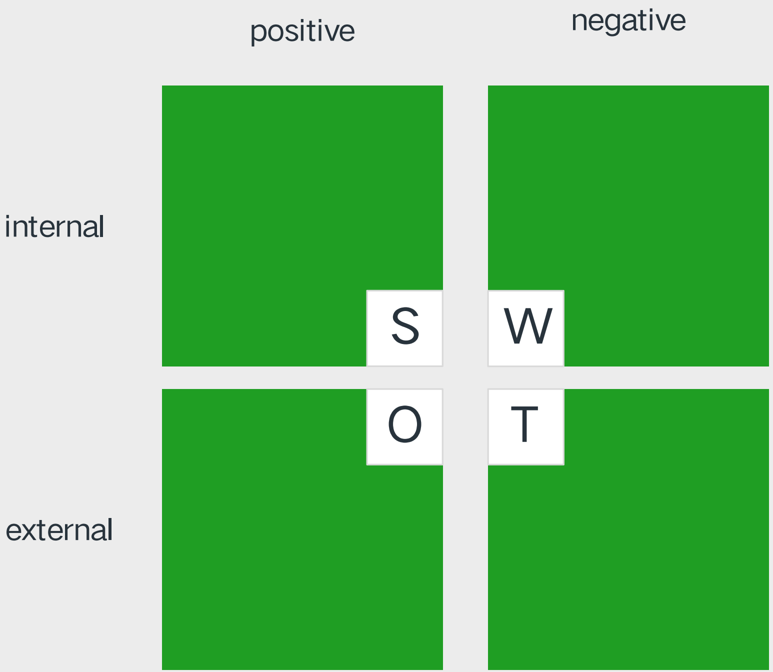
$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

分类误差

```
xy_loss = object_mask * box_loss_scale * K.binary_crossentropy(raw_true_xy, raw_pred[...,0:2], from_logits=True)
wh_loss = object_mask * box_loss_scale * 0.5 * K.square(raw_true_wh-raw_pred[...,2:4])
# 置信度
confidence_loss = object_mask * K.binary_crossentropy(object_mask, raw_pred[...,4:5], from_logits=True)+ (1-object_mask)
* K.binary_crossentropy(object_mask, raw_pred[...,4:5], from_logits=True) * ignore_mask
# 分类
class_loss = object_mask * K.binary_crossentropy(true_class_probs, raw_pred[...,5:], from_logits=True)
xy_loss = K.sum(xy_loss) / mf
wh_loss = K.sum(wh_loss) / mf
confidence_loss = K.sum(confidence_loss) / mf
class_loss = K.sum(class_loss) / mf
loss += xy_loss + wh_loss + confidence_loss + class_loss
```

The Business Objectives

What we hope to achieve in the short and long run



设计目标

业务场景主要用在相关高速公路对证件核查录库

技术上限

需要核实将数据和地理坐标信息录入相关系统

模型选择

需要在inference过程计算量较少的确保一张证件在无GPU加速情况下实现1s以内

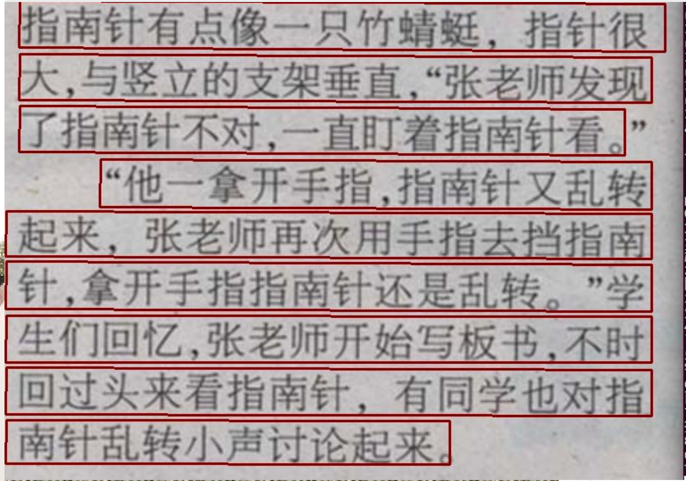
结论：

不能使用主流的文字定位模型，因此我们使用yolov3同时为了对模型稳定性我们需要抵抗背景干扰添加通道注意力

Case Study

What we hope to achieve in the short and long run

定位出现误差和我们通过单应矩阵纠正后结果如下图驾驶证有明显提高


指南针有点像一只竹蜻蜓，指针很大，与竖立的支架垂直，"张老师发现了指南针不对，一直盯着指南针看。"

"他一拿开手指，指南针又乱转起来，张老师再次用手指去挡指南针，拿开手指指南针还是乱转。"学生们回忆，张老师开始写板书，不时回过头来看指南针，有同学也对指南针乱转小声讨论起来。

证号	44030119800101****
姓名	李小明
地址	广东省深圳市南山区龙井路 xxx 号
准驾车型	C1
有效期	2010-05-23 至 2016-05-23

Text recognition

3.3. FAN Training

We combine a ResNet-based feature extractor, AN and FN into one network, as shown in Fig. 4. The details are given in Section 4.2. AN uses the extracted features to generate alignment factors and glimpse vectors, with which FN focuses the attention of AN on the proper target character regions in the images.

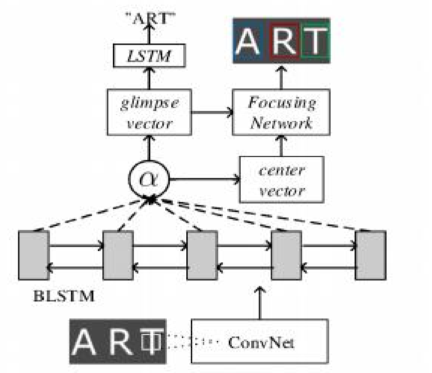
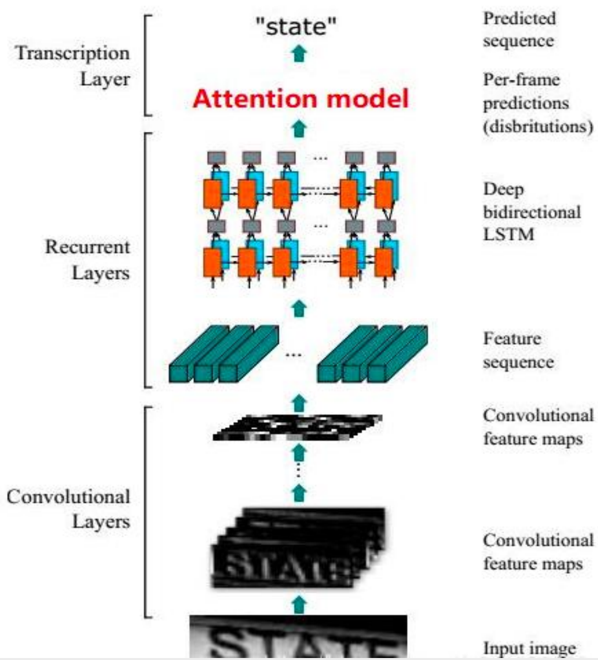


Figure 4. The FAN network architecture. Here, the *CNN-BLSTM* encoder transforms an input image $\mathcal{I}$ into high level sequence of features, the RNN decoder generates each target character, and FN focuses the attention of AN on the right target character regions in the input images. FN and AN are trained simultaneously.

整体流程为：encoder+decoder

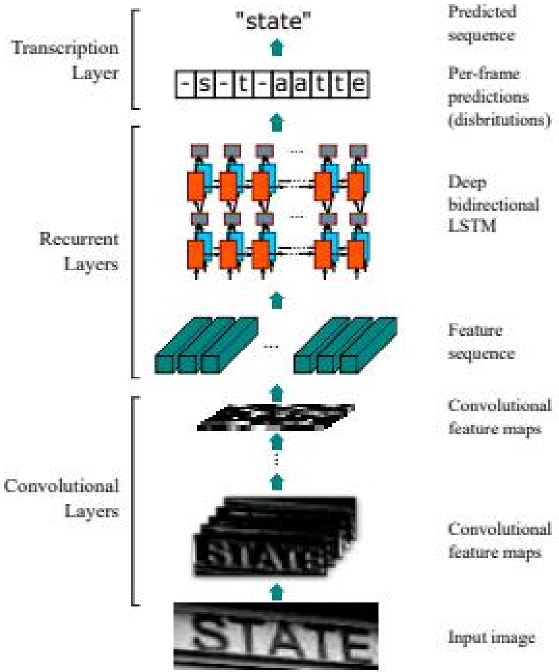
encoder采用CNN+biLSTM模型

decoder采用Attention模型



crnn网络设计：

CRNN由CNN+BiLSTM+CTC构成：



网络结构：

Type	Configurations
Transcription	-
Bidirectional-LSTM	#hidden units:256
Bidirectional-LSTM	#hidden units:256
Map-to-Sequence	-
Convolution	#maps:512, k:2 × 2, s:1, p:0
MaxPooling	Window:1 × 2, s:2
BatchNormalization	-
Convolution	#maps:512, k:3 × 3, s:1, p:1
BatchNormalization	-
Convolution	#maps:512, k:3 × 3, s:1, p:1
MaxPooling	Window:1 × 2, s:2
Convolution	#maps:256, k:3 × 3, s:1, p:1
Convolution	#maps:256, k:3 × 3, s:1, p:1
MaxPooling	Window:2 × 2, s:2
Convolution	#maps:128, k:3 × 3, s:1, p:1
MaxPooling	Window:2 × 2, s:2
Convolution	#maps:64, k:3 × 3, s:1, p:1
Input	$W \times 32$ gray-scale image

- input： 输入文字块，归一化到 `32*w` 即height缩放到32，宽度按高度的比率缩放，也可以缩放到自己想要的宽度，训练时为批次训练，缩放到[32,Wmax]），示例为（32,128）
- 经过两个conv层和两个poling层，conv3层时数据大小为256*8*32，两个pooling层步长为2
- pooling2层步长为（2，1），（个人看法：作者使用的英文训练，英文字符的特征是高大于宽的特征，倘若使用中文训练，建议使用（2,2），我的代码中默认为（2,2），示例以（2，1）为例,所以此时输出为256*4*33
- bn层不改变输出的大小（就是做个归一化，加速训练收敛），p3层时,w+1,所以pooling3层时，输出为512*2*34
- conv7层时，kernel 为22，*stride(1,1) padding(0,0)*
  Wnew = (2 + 2 padW - kernel) / strideW + 1 = 1
  Hnew = 33
  所以conv7层输出为512*133*
- 后面跟两个双向Lstm,隐藏节点都是256
  Blstm1输出33*1*256*
  *Blstm2输出 33**1*5530 5530 = 字符个数 + 非字符 = 5529 + 1
  最终的输出结果直观上可以想象成将128分为33份，每一份对应5530个类别的概率

Part Three

模型部署

THE TITLE

1、采用opencvDNN调用模型的pb文件或者h5执行inference

2、服务端的tensorsever、flask、Django

这里最大的难点是部分模型打包成sdk出现问题

目前移动端的算法和芯片需要采用专用的方式实现和规避这个问题

模型压缩和部署

DNNDK User Guide

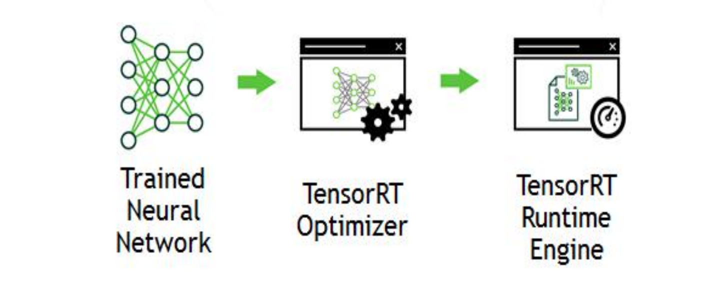
UG1327 (v1.4) April 29, 2019

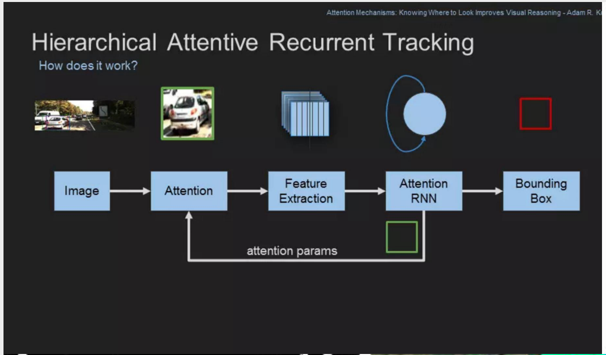
XILINX

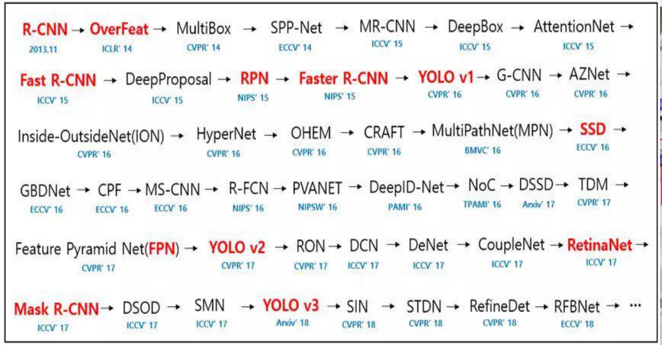
- 硬件平台选择
  - GPU（1080TI、P4、2080TI、TX1、TX2等）
  - Arm（海思平台、RockChip平台等）
  - FPGA
  - DSP（Movidius、Ceva等）
  - ASIC（通用：海思3559A等；专用：？）

- 基于网络结构改变的加速策略
  - 简单修改成熟网络（减少filter channel number、减少层数等）
  - 手工设计新网络结构（Depth-Wise、ShuffleNet）
  - 模型参数裁剪（模型参数压缩；如何使小网络更好收敛：Mimick）
  - NAS（例如：基于强化学习的模型结构搜索）

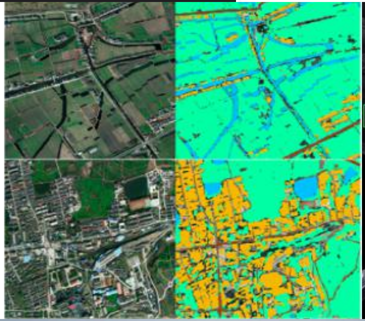
- 软件优化
  - 模型定点化
  - 优先将模型用TensorRT运行
  - GPU平台其它运算用CUDA实现，少使用CPU
  - ARM平台，利用NEON指令
  - 合理组Pipeline提高系统资源利用率（系统级优化）

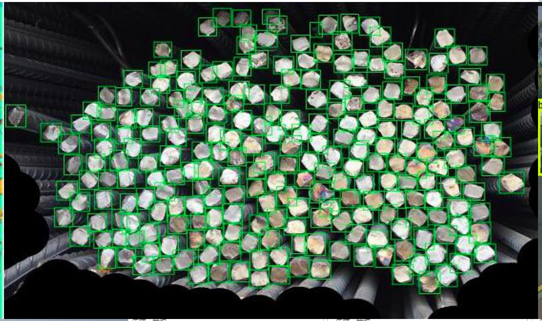


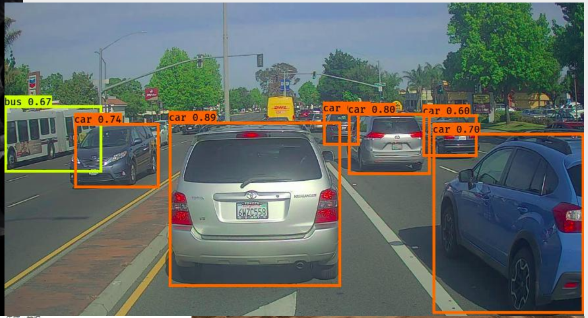

Attention Mechanisms: Knowing Where to Look Improves Visual Reasoning - Adam R. K
Hierarchical Attentive Recurrent Tracking
How does it work?
Image
Attention
Feature Extraction
Attention RNN
Bounding Box
attention params


R-CNN → OverFeat → MultiBox → SPP-Net → MR-CNN → DeepBox → AttentionNet →
2013.11 ICLR' 14 CVPR' 14 ECCV' 14 ICCV' 15 ICCV' 15 ICCV' 15
Fast R-CNN → DeepProposal → RPN → Faster R-CNN → YOLO v1 → G-CNN → AZNet →
ICCV' 15 ICCV' 15 NIPS' 15 NIPS' 15 CVPR' 16 CVPR' 16 CVPR' 16
Inside-OutsideNet(ION) → HyperNet → OHEM → CRAFT → MultiPathNet(MPN) → SSD →
CVPR' 16 CVPR' 16 CVPR' 16 CVPR' 16 BMVC' 16 ECCV' 16
GBDNet → CPF → MS-CNN → R-FCN → PVANET → DeepID-Net → NoC → DSSD → TDM →
ECCV' 16 ECCV' 16 ECCV' 16 NIPS' 16 NIPSW' 16 PAMI' 16 TPAMI' 16 Arxiv' 17 CVPR' 17
Feature Pyramid Net(FPN) → YOLO v2 → RON → DCN → DeNet → CoupleNet → RetinaNet →
CVPR' 17 CVPR' 17 CVPR' 17 ICCV' 17 ICCV' 17 ICCV' 17 ICCV' 17
Mask R-CNN → DSOD → SMN → YOLO v3 → SIN → STDN → RefineDet → RFBNet → ...
ICCV' 17 ICCV' 17 ICCV' 17 Arxiv' 18 CVPR' 18 CVPR' 18 CVPR' 18 ECCV' 18


airplane 0.996
airplane 0.994
airplane 0.999
airplane 0.998
airplane 0.997
airplane 1.000






bus 0.67
car 0.74
car 0.89
car 0.80
car 0.60
car 0.70


具体目标类别参考
BMP2 0.992
BMP2 0.994
BMP2 0.991
BMP2 0.919
BTR70 0.958
BTR70 0.928
BTR70 0.956
BTR70 0.922
T72 0.908
T72 0.985
T72 0.931
T72 0.903
T72 0.954


价税合计: 叁万伍仟贰佰零柒圆壹角肆分
发票代码: 1100153130
发票号码: 26451036
合计税额: ¥1992.86
合计金额: ¥33214.28
开票日期: 2016年04月06日
税率: 6%
购买方名称: 深圳市腾讯计算机系统有限公司
购买方识别号: 440300708461136
销售方名称: 北京阅读在线科技有限公司
销售方识别号: 110105599620504
收款人: 李明
开票人: 张三


告警：抽烟
时间:2018-06-20 10:41:25
时间:2018-06-20 10:32:35
时间:2018-06-20 10:13:26
车牌：浙DD6772
时间:2018-06-20 09:53:33
车牌：浙DD6769
时间:2018-06-20 09:49:28
车牌：浙DC9111
时间:2018-06-20 09:39:40


THANK YOU
Thought for service "is the sacred mission of hi design.
谢谢各位聆听同时有问题请在github上互相提交后续有机会
我将放出医学全景分割工程。