一位算法工程师从30+场秋招面试中总结出的超强面经—文本检测与GAN篇（含答案）

Original CV开发者都爱看的 极市平台 Today

作者丨灯会

来源丨极市平台

编辑丨极市平台

极市导读

作者灯会为21届中部985研究生，凭借自己整理的面经，去年在腾讯优图暑期实习，七月份将入职百度cv算法工程师。在去年灰飞烟灭的算法求职季中，经过30+场不同公司以及不同部门的面试中积累出了CV总复习系列，此为文本检测与GAN篇。 >>加入极市CV技术交流群，走在计算机视觉的最前沿

系列文章：

深度学习三十问！一位算法工程师经历30+场CV面试后总结的常见问题合集（含答案）

深度学习六十问！一位算法工程师经历30+场CV面试后总结的常见问题合集下篇（含答案）

一位算法工程师从30+场秋招面试中总结出的超强面经—语义分割篇（含答案）

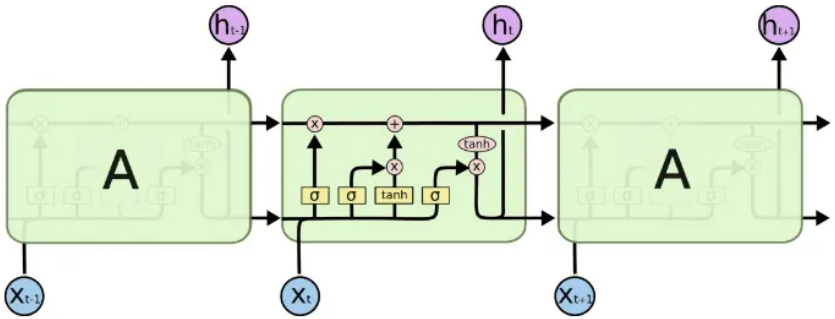
一位算法工程师从30+场秋招面试中总结出的超强面经——目标检测篇（含答案）

图像处理知多少？准大厂算法工程师30+场秋招后总结的面经问题详解

1.LSTM(长短期记忆)原理，其中的参数是否相同/画出LSTM的结构图/写一下LSTM的公式

Lstm由输入门,遗忘门,输出门和一个cell组成。第一步是决定从cell状态中丢弃什么信息,然后在决定有多少新的信息进入到cell状态中,最终基于目前的cell状态决定输出什么样的信息。

$$
\begin{aligned}
i &= \sigma(W_{ii}x + b_{ii} + W_{hi}h + b_{hi}) \\
f &= \sigma(W_{if}x + b_{if} + W_{hf}h + b_{hf}) \\
g &= \tanh(W_{ig}x + b_{ig} + W_{hg}h + b_{hg}) \\
o &= \sigma(W_{io}x + b_{io} + W_{ho}h + b_{ho}) \\
c' &= f * c + i * g \\
h' &= o * \tanh(c')
\end{aligned}
$$



LSTM一共有三个门，输入门，遗忘门，输出门，i,f,o分别为三个门的程度参数，g是对输入的常规RNN操作。公式里可以看到LSTM的输出有两个，细胞状态c‘　和隐状态h'，c'是经输入、遗忘门的产物，也就是当前cell本身的内容，经过输出门得到h'，就是想输出什么内容给下一单元。

LSTM中有哪些激活函数

LSTM中的三个门是用的**sigmoid**作为激活函数，生成候选记忆时候用的才是**tanh**，门j的激活函数如果用relu的话会有个问题，就是relu是没有饱和区域的，那么就没法起到门的作用。候选记忆用tanh是因为tanh的输出在-1~1，是0中心的，并且在0附近的梯度大，模型收敛快。

LSTM这两个激活函数的作用分别是什么 sigmoid将一个实数输入映射到[0,1]范围内,tanh函数将一个实数输入映射到[-1,1]范围内;

LSTM每个门的计算公式

遗忘门:

$$\mathrm{f} = \sigma(W_f[h_{t-1}, x_t] + b_f)$$

输入门:

$$\mathrm{i} = \sigma(W_i[h_{t-1}, x_t] + b_i)$$

输出门:

$$\mathrm{o} = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

2.推导LSTM正向传播和单向传播过程

前向推导过程：

Forget Gates

$$a_\phi^t = \sum_{i=1}^t w_{i\phi} x_i^t + \sum_{h=1}^H w_{h\phi} b_h^{t-1} + \sum_{c=1}^C w_{c\phi} s_e^{t-1}$$
$$b_\phi^t = f\left(a_\phi^t\right)$$

Cells

$$a_c^t = \sum_{i=1}^t w_{ic} x_i^t + \sum_{h=1}^H w_{hc} b_h^{t-1}$$
$$s_c^t = b_\phi^t s_c^{t-1} + b_4^t g\left(a_e^t\right)$$

Output Gates

$$a_\omega^t = \sum_{i=1}^I w_{i\omega} x_i^t + \sum_{h=1}^H w_{h\omega} b_h^{t-1} + \sum_{c=1}^C w_{c\omega} s_c^t$$
$$b_\omega^t = f\left(a_\omega^t\right)$$

Cell Outputs

$$b_c^t = b_\omega^t h\left(s_c^t\right)$$

反向推导过程：

$$\epsilon_e^t \overset{\text{def}}{=} \frac{\partial \mathcal{L}}{\partial b_c^t} \quad \epsilon_s^t \overset{\text{det}}{=} \frac{\partial \mathcal{L}}{\partial s_c^t}$$

Cell Outputs

$$\epsilon_\varepsilon^t = \sum_{k=1}^K w_{ck} \delta_k^t + \sum_{g=1}^G u_{cg} \delta_g^{t+1}$$

Output Gates

$$\delta_\omega^t = f'\left(a_\omega^t\right) \sum_{c=1}^C h\left(s_c^t\right) \epsilon_e^t$$

States

$$\epsilon_x^t = b_\omega^t h'\left(s_c^t\right) \epsilon_c^t + b_\phi^{t+1} e_s^{t+1} + w_{ce} \delta_k^{t+1} + w_{c\phi} \delta_\phi^{t+1} + w_{c\omega} \delta_\omega^t$$

Cells

$$\delta_c^t = b_c^t g'\left(a_c^t\right) \epsilon_s^t$$

Forget Gates

$$\delta_{\phi}^{t}=f'\left(a_{\phi}^{t}\right)\sum_{c=1}^{C}s_{e}^{t-1}\epsilon_{n}^{t}$$

Input Gates

$$\delta_{i}^{t}=f'\left(a_{i}^{t}\right)\sum_{c=1}^{C}g\left(a_{c}^{t}\right)\epsilon_{s}^{t}$$

3.cnn、lstm区别、文本里怎么应用　cnn和lstm各自的区别和应用场景

CNN 专门解决图像问题的，可用把它看作特征提取层，放在输入层上，最后用MLP 做分类。(**CNN可能更加适合分类问题**)

RNN 专门解决时间序列问题的，用来提取时间序列信息，放在特征提取层（如CNN）之后。

LSTM（Long Short-Term Memory）是长短期记忆网络，是一种时间递归神经网络，适合于处理和预测时间序列中间隔和延迟相对较长的重要事件。为了解决RNN中时间上的梯度消失，机器学习领域发展出了长短时记忆单元LSTM，通过门的开关实现时间上记忆功能，并防止梯度消失。

4. rnn原理　循环神经网络，为什么好?

循环神经网络模型（RNN）是一种节点定向连接成环的人工神经网络，是一种反馈神经网络，RNN利用内部的记忆来处理任意时序的输入序列，并且在其处理单元之间既有内部的反馈连接又有前馈连接，这使得RNN可以更加容易处理不分段的文本等。

5.什么是RNN

一个序列当前的输出与前面的输出也有关,在RNN网络结构中,隐藏层的输入不仅包括输入层的输出还包含上一时刻隐藏层的输出,网络会对之前的信息进行记忆并应用于当前的输入计算中。

RNN梯度消失问题,为什么LSTM和GRU可以解决此问题

RNN由于网络较深,后面层的输出误差很难影响到前面层的计算,RNN的某一单元主要受它附近单元的影响。而LSTM因为可以通过阀门记忆一些长期的信息,相应的也就保留了更多的梯度。而GRU也可通过重置和更新两个阀门保留长期的记忆,也相对解决了梯度消失的问题。

RNN容易梯度消失，怎么解决?

1）梯度裁剪（Clipping Gradient）

既然在BP过程中会产生梯度消失（就是偏导无限接近0，导致长时记忆无法更新），那么最简单粗暴的方法，设定阈值，当梯度小于阈值时，更新的梯度为阈值。

优点：简单粗暴

缺点：很难找到满意的阈值

2）LSTM（Long Short-Term Memory）

一定程度上模仿了长时记忆，相比于梯度裁剪，最大的优点就是，自动学习在什么时候可以将error反向传播，自动控制哪些是需要作为记忆存储在LSTM cell中。一般长时记忆模型包括写入，读取，和忘记三个过程对应到LSTM中就变成了input_gate,output_gate,

forget_gate,三个门，范围在0到1之间，相当于对输入输出进行加权的学习，利用大量数据来自动学习加权的参数（即学习了哪些错误可以用BP更新参数）。具体的公式表达：

优点：模型自动学习更新参数

- **LSTM跟RNN有啥区别**

LSTM与RNN的比较

RNN在处理long term memory的时候存在缺陷，因此LSTM应运而生。LSTM是一种变种的RNN，它的精髓在于引入了细胞状态这样一个概念，不同于RNN只考虑最近的状态，LSTM的细胞状态会决定哪些状态应该被留下来，哪些状态应该被遗忘。

下面来看一些RNN和LSTM内部结构的不同：

RNN

LSTM

由上面两幅图可以观察到，LSTM结构更为复杂，在RNN中，将过去的输出和当前的输入concatenate到一起，通过tanh来控制两者的输出，它只考虑最近时刻的状态。在RNN中有两个输入和一个输出。

而LSTM为了能记住长期的状态，在RNN的基础上增加了一路输入和一路输出，增加的这一路就是细胞状态，也就是途中最上面的一条通路。事实上整个LSTM分成了三个部分：

1）哪些细胞状态应该被遗忘

2）哪些新的状态应该被加入

3）根据当前的状态和现在的输入，输出应该是什么

下面来分别讨论：

1）哪些细胞状态应该被遗忘

这部分功能是通过sigmoid函数实现的，也就是最左边的通路。根据输入和上一时刻的输出来决定当前细胞状态是否有需要被遗忘的内容。举个例子，如果之前细胞状态中有主语，而输入中又有了主语，那么原来存在的主语就应该被遗忘。concatenate的输入和上一时刻的输出经过sigmoid函数后，越接近于0被遗忘的越多，越接近于1被遗忘的越少。

2）哪些新的状态应该被加入

继续上面的例子，新进来的主语自然就是应该被加入到细胞状态的内容，同理也是靠sigmoid函数来决定应该记住哪些内容。但是值得一提的是，需要被记住的内容并不是直接

concatenate的输入和上一时刻的输出，还要经过tanh，这点应该也是和RNN保持一致。并且需要注意，此处的sigmoid和前一步的sigmoid层的w和b不同，是分别训练的层。

细胞状态在忘记了该忘记的，记住了该记住的之后，就可以作为下一时刻的细胞状态输入了。

3）根据当前的状态和现在的输入，输出应该是什么

这是最右侧的通路，也是通过sigmoid函数做门，对第二步求得的状态做tanh后的结果过滤，从而得到最终的预测结果。

事实上，LSTM就是在RNN的基础上，增加了对过去状态的过滤，从而可以选择哪些状态对当前更有影响，而不是简单的选择最近的状态。

在这之后，研究人员们实现了各种LSTM的变种网络。不变的是，通常都会用sigmoid函数做门，筛选状态或者输入。并且输出都是要经过tanh函数。具体为什么要用这两个函数，由于刚接触还不能给出一定的解释，日后理解了再补充。

6. GRU的原理

GRU（Gate Recurrent Unit）是循环神经网络（Recurrent Neural Network, RNN）的一种。和LSTM（Long-Short Term Memory）一样，也是为了解决长期记忆和反向传播中的梯度等问题而提出来的。

GRU由重置门和跟新门组成,其输入为前一时刻隐藏层的输出和当前的输入,输出为下一时刻隐藏层的信息。重置门用来计算候选隐藏层的输出,其作用是控制保留多少前一时刻的隐藏层。跟新门的作用是控制加入多少候选隐藏层的输出信息,从而得到当前隐藏层的输出。

$$
\begin{aligned}
z_t &= \sigma(W_z \cdot [h_{t-1}, x_t]) \\
r_t &= \sigma(W_r \cdot [h_{t-1}, x_t]) \\
\tilde{h}_t &= \tanh(W \cdot [r_t * h_{t-1}, x_t]) \\
h_t &= (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t
\end{aligned}
$$

GRU有两个门：更新门，输出门

● LSTM原理，与GRU区别

LSTM算法全称为Long short-term memory，是一种特定形式的RNN（Recurrent neural network，循环神经网络），而RNN是一系列能够处理序列数据的神经网络的总称。

RNN在处理长期依赖（时间序列上距离较远的节点）时会遇到巨大的困难，因为计算距离较远的节点之间的联系时会涉及雅可比矩阵的多次相乘，这会带来梯度消失（经常发生）或者梯度膨胀（较少发生）的问题，这样的现象被许多学者观察到并独立研究。为了解

决该问题，研究人员提出LSTM。

LSTM是门限RNN，其单一节点的结构如下图1所示。LSTM的巧妙之处在于通过增加输入门限，遗忘门限和输出门限，使得自循环的权重是变化的，这样一来在模型参数固定的情况下，不同时刻的积分尺度可以动态改变，从而避免了梯度消失或者梯度膨胀的问题。

图1 LSTM的CELL示意图

根据LSTM网络的结构，每个LSTM单元的计算公式如下图2所示，其中Ft表示遗忘门限，It表示输入门限，Ct表示前一时刻cell状态、Ct表示cell状态（这里就是循环发生的地方），Ot表示输出门限，Ht表示当前单元的输出，Ht-1表示前一时刻单元的输出。

$$
\begin{aligned}
f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\
i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\
\tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \\
C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \\
o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o) \\
h_t &= o_t * \tanh(C_t)
\end{aligned}
$$

与GRU区别：1）GRU和LSTM的性能在很多任务上不分伯仲。2）GRU 参数更少因此更容易收敛，但是数据集很大的情况下，LSTM表达性能更好。3）从结构上来说，GRU只有两个门（update和reset），LSTM有三个门（forget，input，output），GRU直接将hidden state 传给下一个单元，而LSTM则用memory cell 把hidden state 包装起来。

7.LSTM和Naive RNN的区别

RNN和LSTM内部结构的不同：

RNN

LSTM

由上面两幅图可以观察到，LSTM结构更为复杂，在RNN中，将过去的输出和当前的输入concatenate到一起，通过tanh来控制两者的输出，它只考虑最近时刻的状态。在RNN中有两个输入和一个输出。

而LSTM为了能记住长期的状态，在RNN的基础上增加了一路输入和一路输出，增加的这一路就是细胞状态，也就是途中最上面的一条通路。事实上整个LSTM分成了三个部分：

1）哪些细胞状态应该被遗忘

2）哪些新的状态应该被加入

3）根据当前的状态和现在的输入，输出应该是什么

下面来分别讨论：

1）哪些细胞状态应该被遗忘

这部分功能是通过sigmoid函数实现的，也就是最左边的通路。根据输入和上一时刻的输出来决定当前细胞状态是否有需要被遗忘的内容。举个例子，如果之前细胞状态中有主语，而输入中又有了主语，那么原来存在的主语就应该被遗忘。concatenate的输入和上一时刻的输出经过sigmoid函数后，越接近于0被遗忘的越多，越接近于1被遗忘的越少。

2）哪些新的状态应该被加入

继续上面的例子，新进来的主语自然就是应该被加入到细胞状态的内容，同理也是靠sigmoid函数来决定应该记住哪些内容。但是值得一提的是，需要被记住的内容并不是直接concatenate的输入和上一时刻的输出，还要经过tanh，这点应该也是和RNN保持一致。并且需要注意，此处的sigmoid和前一步的sigmoid层的w和b不同，是分别训练的层。细胞状态在忘记了该忘记的，记住了该记住的之后，就可以作为下一时刻的细胞状态输入了。

3）根据当前的状态和现在的输入，输出应该是什么

这是最右侧的通路，也是通过sigmoid函数做门，对第二步求得的状态做tanh后的结果过滤，从而得到最终的预测结果。事实上，LSTM就是在RNN的基础上，增加了对过去状态的过滤，从而可以选择哪些状态对当前更有影响，而不是简单的选择最近的状态。

8.CTC的原理

CTC是计算一种损失值，主要的优点是可以对没有对齐的数据进行自动对齐。CTC是序列标注问题中的一种损失函数。

9.如何解决OCR文本过长的问题

分段

10.文本检测中，出现多个box重叠如何处理

（NMS）

11. GAN网络的思想

GAN用一个生成模型和一个判别模型,判别模型用于判断给定的图片是不是真实的图片,生成模型自己生成一张图片和想要的图片很像,开始时两个模型都没有训练,然后两个模型一起进行对抗训练,生成模型产生图片去欺骗判别模型,判别模型去判别真假,最终两个模型在训练过程中,能力越来越强最终达到稳态。

GAN网络的损失函数说一下？它的优化目标是啥？

$$\min_G \max_D V(D,G) = E_{x\sim P_{\text{datta}})}[\log D(X)] + E_{z_{|Y}P_{d(z)}}[\log(1-D(G(z)))]$$

这是一个最大最小优化问题，先优化D，然后再优化G，本质上是两个优化问题，拆解后得到下面两个公式：

优化D:

$$\min_G \max_D V(D,G) = E_{x\sim P_{\text{dath}}\,x)}[\log D(X)] + E_{z\sim P_{\lambda}(z)}[\log(1_D D(G(z)))]$$

优化G:

$$\min_G V(D,G) = E_{z\sim p_z(z)}[\log(1-D(G(z)))]$$

【相关参数说明】E表示数学期望,比如表示log(D(x))的数学期望，其中x服从概率分布Pdata（x)。即等于

$$\int_x P_{\text{data}}\,(x)\log(D)(x)dx$$

关于损失函数的一些说明：

优化D(判别网络)时，不关生成网络的事，后面的G(z)相当于已经得到的假样本。优化D的公式的第一项，使的真样本x输入时，得到的结果越大越好，因为需要真样本的预测结果越接近于1越好。对于假样本，需要优化使其结果越小越好，也就是D(G(z))越小越好，因为它的标签为0。但是第一项越大，第二项就越小，这就矛盾了，所以把第二项改成1-D(G(z))，这样就是越大越好，两者合起来就是越大越好。

优化G(生成网络)时，不关真样本的事，所以把第一项直接去掉，只剩下假样本，这时希望假样本的标签是1，所以D(G(z))越大越好，但为了统一成1-D(G(z))的形式，就变成最小化1-D(G(z))，本质上没有区别，只是为了形式的统一。

这两个优化模型合并起来，就成了上面的最大最小目标函数了，里面既包含了判别模型的优化，同时也包含了生成模型的以假乱真的优化。

12.CycleGAN

原理介绍一下

CycleGAN其实就是一个A→B单向GAN加上一个B→A单向GAN。两个GAN共享两个生成器，然后各自带一个判别器，所以加起来总共有两个判别器和两个生成器。一个单向GAN有两个loss，而CycleGAN加起来总共有四个loss。CycleGAN论文的原版原理图和公式如下，其实理解了单向GAN那么CycleGAN已经很好理解。

X→Y 的判别器损失如下，字母换了一下，和上面的单向 GAN 是一样的：

$$L_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{dat}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{dat}}(x)}[\log(1 - D_Y(G(x)))]$$

同理，Y→X 的判别器损失为：

$$L_{GAN}(F, D_X, Y, X) = \mathbb{E}_{x \sim p_{\text{dat}}(x)}[\log D_X(x)] + \mathbb{E}_{y \sim p_{\text{dat}}(y)}[\log(1 - D_X(F(y)))]$$

而两个生成器的 loss 加起来表示为：

$$L_{cyc}(G, F) = \mathbb{E}_{x \sim p_{\text{dat}}(x))}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{dat}}(y)}[\|G(F(y)) - y\|_1]$$

最终网络的所有损失加起来为：

$$L(G, F, D_X, D_Y) = L_{GAN}(G, D_Y, X, Y) + L_{GAN}(F, D_X, Y, X) + L_{cyc}(G, F)$$

论文里面提到判别器如果是对数损失训练不是很稳定，所以改成的均方误差损失，如下：

$$L_{LSGAN}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{dat}}(y)}\left[(D_Y(y) - 1)^2\right] + \mathbb{E}_{x \sim p_{\text{dat}}(x)}\left[(1 - D_Y(G(x)))^2\right]$$

下面放一张CycleGAN示意图，比论文原版的更加直观。

13.训练 GAN 的时候有没有遇到什么问题

遇到GAN训练不稳定问题。通过Wasserstein GAN来解决这个问题。WGAN前作分析了Ian Goodfellow提出的原始GAN两种形式各自的问题，第一种形式等价在最优判别器下等价于最小化生成分布与真实分布之间的JS散度，由于随机生成分布很难与真实分布有不可忽略的重叠以及JS散度的突变特性，使得生成器面临梯度消失的问题；第二种形式在最优判别器下等价于既要最小化生成分布与真实分布直接的KL散度，又要最大化其JS散度，相互矛盾，导致梯度不稳定，而且KL散度的不对称性使得生成器宁可丧失多样性也不愿丧失准确性，导致collapse mode现象。

WGAN前作针对分布重叠问题提出了一个过渡解决方案，通过对生成样本和真实样本加噪声使得两个分布产生重叠，理论上可以解决训练不稳定的问题，可以放心训练判别器到接近最优，但是未能提供一个指示训练进程的可靠指标，也未做实验验证。

WGAN本作引入了Wasserstein距离，由于它相对KL散度与JS散度具有优越的平滑特性，理论上可以解决梯度消失问题。接着通过数学变换将Wasserstein距离写成可求解的形式，利用一个参数数值范围受限的判别器神经网络来最大化这个形式，就可以近似Wasserstein距离。在此近似最优判别器下优化生成器使得Wasserstein距离缩小，就能有效拉近生成分布与真实分布。WGAN既解决了训练不稳定的问题，也提供了一个可靠的训练进程指标，而且该指标确实与生成样本的质量高度相关。

参考链接

- http://www.mianshigee.com/question/12442yxc
- https://blog.csdn.net/lanmengyiyu/article/details/79941486
- https://blog.csdn.net/weixin_42111770/article/details/80900575
- https://www.nowcoder.com/ta/review-ml/review?page=159
- https://linzhenyuyuchen.github.io/2020/04/03/Pytorch-%E6%A2%AF%E5%BA%A6%E8%A3%81%E5%89%AA/

本文亮点总结

1.LSTM结构更为复杂，在RNN中，将过去的输出和当前的输入concatenate到一起，通过tanh来控制两者的输出，它只考虑最近时刻的状态。在RNN中有两个输入和一个输出。

2.WGAN前作针对分布重叠问题提出了一个过渡解决方案，通过对生成样本和真实样本加噪声使得两个分布产生重叠，理论上可以解决训练不稳定的问题，可以放心训练判别器到接近最优，但是未能提供一个指示训练进程的可靠指标，也未做实验验证。

如果觉得有用，就请分享到朋友圈吧！

极市干货

YOLO教程：一文读懂YOLO V5 与 YOLO V4｜大盘点｜YOLO 系目标检测算法总览｜全面解析YOLO V4网络结构

实操教程：PyTorch vs LibTorch：网络推理速度谁更快？｜只用两行代码，我让Transformer推理加速了50倍｜PyTorch AutoGrad C++层实现

算法技巧（trick）：深度学习训练tricks总结（有实验支撑）｜深度强化学习调参Tricks合集｜长尾识别中的Tricks汇总（AAAI2021）

最新CV竞赛：2021 高通人工智能应用创新大赛｜CVPR 2021｜Short-video Face Parsing Challenge｜3D人体目标检测与行为分析竞赛开赛，奖池7万+，数据集达16671张！

极市原创作者激励计划

极市平台深耕CV开发者领域近5年，拥有一大批优质CV开发者受众，覆盖微信、知乎、B站、微博等多个渠道。通过极市平台，您的文章的观点和看法能分享至更多CV开发者，既能体现文章的价值，又能让文章在视觉圈内得到更大程度上的推广。

对于优质内容开发者，极市可推荐至国内优秀出版社合作出书，同时为开发者引荐行业大牛，组织个人分享交流会，推荐名企就业机会，打造个人品牌 IP。

投稿须知：

1.作者保证投稿作品为自己的原创作品。

2.极市平台尊重原作者署名权，并支付相应稿费。文章发布后，版权仍属于原作者。

3.原作者可以将文章发在其他平台的个人账号，但需要在文章顶部标明首发于极市平台

投稿方式：

添加小编微信Fengcall（微信号：fengcall19），备注：**姓名-投稿**

△**长按添加极市平台小编**

觉得有用麻烦给个在看啦~

Read more

People who liked this content also liked

Transformer杀疯了！竟在图神经网络的ImageNet大赛中夺冠，力压DeepMind、百度.....

AI科技评论

快手开源斗地主AI，入选ICML，能否干得过「冠军」柯洁?

机器之心

20年磨一剑！南大周志华团队力作「演化学习」重磅首发

新智元