

Apple Vision Pro: 智能设备的耗电，速度和隐私



轻量化的人脸识别模型

自动化所 蔡怀广 2023.06.07

目录

CONTENTS

-
- 01.** LightCNN
 - 02.** On-Device Face Recognition
 - 03.** 对MFM的改进尝试
 - 04.** Take Home Message

LightCNN

- A Light CNN for Deep Face Representation With Noisy Labels, 2018年自动化所孙哲南老师团队发表在IEEE Transactions on Information Forensics and Security。
- achieves state-of-the-art results on various face benchmarks without fine-tuning。
- LightCNN的核心操作是Max-Feature-Map (MFM) 。
- 除了MFM外, LightCNN的网络设计还融合了Network in Network、小卷积滤波器的思想
- 在训练中还通过一种语义自举 (semantic bootstrapping) 的方法通过预训练的网络方法重新标记训练数据, 从而抵抗标记的噪声。

[A light CNN for deep face representation with noisy labels](#)

X Wu, R He, Z Sun, T Tan

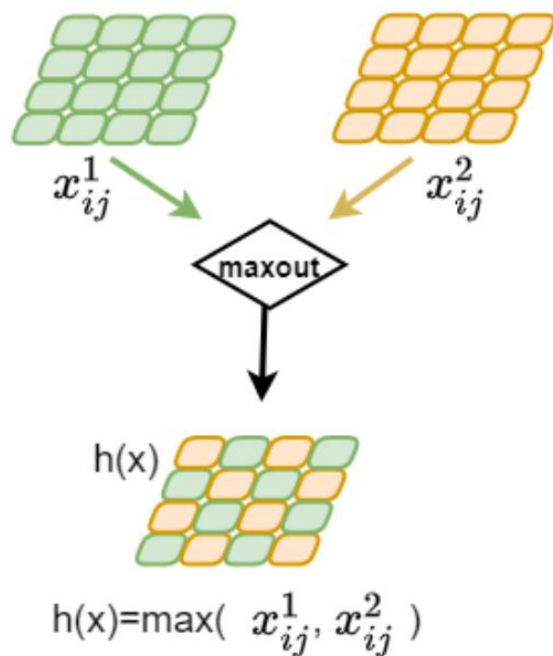
IEEE Transactions on Information Forensics and Security 13 (11), 2884-2896

1095

2018

MFM(Max-Feature-Map)

- MFM 将两个特征图融合并逐点输出最大值。如下图所示，同一个样本通过网络得到的第一个特征图（绿色）和第二个特征图（黄色），经过MFM后，得到一个绿黄相间的特征图。
- $out[0]$, $out[1]$ 从 x 中按照第二维度分离得到，比如 x 是 $[N, 96, 32, 32]$ （ N 表示批大小， N 张图片），那么 $out[0]$, $out[1]$ 的维度就是 $[N, 48, 32, 32]$ ，两者之间进行逐元素的max得到的输出维度就是 $[N, 48, 32, 32]$ 。




```
class mfm(nn.Module):  
    def forward(self, x):  
        x = self.filter(x)  
        out = torch.split(x, self.out_channels, 1)  
        return torch.max(out[0], out[1])
```

Layer (type)	Output Shape	Param #
Conv2d-1	$[-1, 96, 32, 32]$	7,296
mfm-2	$[-1, 48, 32, 32]$	0
MaxPool2d-3	$[-1, 48, 16, 16]$	0
Conv2d-4	$[-1, 96, 16, 16]$	4,704
mfm-5	$[-1, 48, 16, 16]$	0
Conv2d-6	$[-1, 192, 16, 16]$	83,136
mfm-7	$[-1, 96, 16, 16]$	0
group-8	$[-1, 96, 16, 16]$	0

On-Device Face Recognition

- Towards On-Device Face Recognition in Body-worn Cameras, IEEE International Workshop on Biometrics and Forensics (IWBF) 2021
- 来自Wichita State University的作者们比较了代表性轻量化模型 (MobileNet-V2, EfficientNetB0、LightCNN-9、LightCNN-29) 和经典 VGG-16、ResNet-50在BWCface数据集 (由可穿戴式摄像头进行搜集) 的人脸识别的准确性和实时性。
- BWCface: 用 Boblov 1296P WiFi Body Mounted Camera对132个人在室内和阳光下进行拍照, 得到178K张面部图像。



BOBLOV KJ21 Body Camera, 1296P Body Wearable Camera Support Memory Expand Max 128G 8-10Hours Recording Police Body Camera Lightweight and Portable Easy to Operate Clear NightVision (KJ21 Only)

[Visit the BOBLOV Store](#)
4.4 ★★★★★ 1,602 ratings | 180 answered questions

-6% \$89⁹⁹
List Price: \$95.99

[Join Prime to buy this item at \\$80.99](#)
No Import Fees Deposit & **FREE Shipping** to Hong Kong [Details](#)

Sign in to redeem. **Save 5%** on 5 select item(s) promo code: YYH8MHYX [Shop items](#)

实验设定：基于模板的识别

- 开集评估：神经网络模型会在MS1Mv2（85,742个人的5.8M面部图像）上预训练好后，在BWCface进行开集评估。具体来说，BWCface分为模板集和探测集，模板集由每个人的12张图片组成，探测集由每个人的随机选择的100张图片构成。评估准确性的方法是模型在探测集上的准确率（此时模板集样本和类别已知）。
- 基于模板的识别：拿探测集图片的特征和每个人的模板集图片的特征算相似度，把这个探测集图片在某个人的模板集图片的平均相似度当作属于这个人的概率。
- 图片特征的提取：将图片输入到神经网络，把神经网络某一全连接层的、大小为512的输出当作这个图片的特征。
- 模板和探测集处于相同和不同域的实验：右图中Office VS. Day表示模板集是在室内获得的，探测集是在室外获得的。

TABLE (III) Face identification accuracy of deep CNN models on BWCFace dataset [20] in same and cross-lighting conditions at rank-1, rank-5 and rank-10.

Light Condition	Rank-1 [%]	Rank-5 [%]	Rank-10 [%]
LightCNN-9			
Office vs. Office	93.35	97.78	98.56
Day. vs. Day	78.70	90.47	93.76
Day vs. Office	22.24	35.63	50.65
Office vs. Day	22.41	32.36	44.75
LightCNN-29			
Office vs. Office	94.48	98.27	99.17
Day. vs. Day	78.07	89.41	93.52
Day vs. Office	22.82	32.86	46.29
Office vs. Day	22.93	37.32	47.18
MobileNet-V2			
Office vs. Office	85.50	97.38	98.99
Day. vs. Day	72.43	93.98	97.72
Day vs. Office	25.18	33.30	44.60
Office vs. Day	22.35	32.30	44.60
EfficientNet-B0			
Office vs. Office	80.95	94.93	97.61
Day. vs. Day	68.24	91.18	96.13
Day vs. Office	27.23	42.23	56.08
Office vs. Day	27.92	41.92	55.59
VGG-16			
Office vs. Office	81.65	95.32	97.71
Day. vs. Day	69.49	88.73	94.53
Day vs. Office	31.43	49.92	68.08
Office vs. Day	27.22	44.48	58.34
ResNet-50 [20]			
Office vs. Office	96.33	98.47	99.00
Day. vs. Day	99.36	99.81	99.85
Day vs. Office	84.25	95.61	98.06
Office vs. Day	94.45	98.99	99.63

实验结论

- 准确性：轻量型模型中LightCNN表现优于 MobileNet-V2, EfficientNetB0；在模板集和探测集都是室内的条件下，Rank-1指标下，LightCNN-29 准确率比RestNet-50低1.85%，但是参数量降低了一半。
- 域泛化：轻量型模型都效果差（Rank-1指标下，准确性20%到30%左右），VGG-16也不行，只有RestNet-50准确性降低得不多。
- 实时性：在iphone 6上，LightCNN-29的推理时间是RestNet-50的一半。
- 实验结论：LightCNN在准确性和实时性（参数量）上实现了最好的平衡。

TABLE (III) Face identification accuracy of deep CNN models on BWCFace dataset [20] in same and cross-lighting conditions at rank-1, rank-5 and rank-10.

Light Condition	Rank-1 [%]	Rank-5 [%]	Rank-10 [%]
LightCNN-9			
Office vs Office	93.35	97.78	98.56

TABLE (II) Real-time Inference: Deep feature extraction time per sample measured in milliseconds (ms) on three real mobile devices for each CNN model.

CNN Model	Num. of Params (M)	Device	Extraction Time (ms)
ResNet-50	23.5	iPhone 6	2386
		iPhone X	941
		iPhone XR	834
VGG-16	138	iPhone 6	memory constraints
		iPhone X	4138
		iPhone XR	3747
MobileNetV2	3.4	iPhone 6	1022
		iPhone X	357
		iPhone XR	240
EfficientNet-B0	5.3	iPhone 6	1321
		iPhone X	450
		iPhone XR	329
LightCNN-29	12.6	iPhone 6	1426
		iPhone X	604
		iPhone XR	430
LightCNN-9	5.5	iPhone 6	562
		iPhone X	248
		iPhone XR	171

master 1 branch 0 tags

Go to file Add file Code

AlfredXiangWu update 7b38a6f on Feb 9, 2022 18 commits

.gitignore	Initial commit	6 years ago
LICENSE	Initial commit	6 years ago
README.md	update	last year
extract_features.py	release LightCNN-29v2	6 years ago
light_cnn.py	fix bug	6 years ago
light_cnn_v4.py	update	last year
load_imglist.py	first version	6 years ago
train.py	release LightCNN-29v2	6 years ago

About

A Light CNN for Deep Face Representation with Noisy Labels, TIFS 2018

arxiv.org/abs/1511.02683

pytorch face-recognition lightcnn

- Readme
- MIT license
- Activity
- 686 stars
- 26 watching
- 167 forks
- Report repository

Code

dataset	network	params	top1 err	top5 err
cifar100	mobilenet	3.3M	34.02	10.56
cifar100	mobilenetv2	2.36M	31.92	09.02
cifar100	squeezenet	0.78M	30.59	8.36
cifar100	shufflenet	1.0M	29.94	8.35
cifar100	shufflenetv2	1.3M	30.49	8.49
cifar100	vgg11_bn	28.5M	31.36	11.85
cifar100	vgg13_bn	28.7M	28.00	9.71
cifar100	vgg16_bn	34.0M	27.07	8.84
cifar100	vgg19_bn	39.0M	27.77	8.84
cifar100	resnet18	11.2M	24.39	6.95
cifar100	resnet34	21.3M	23.24	6.63
cifar100	resnet50	23.7M	22.61	6.04
cifar100	resnet101	42.7M	22.22	5.61

master 5 branches 1 tag

Go to file Add file Code

weiaicunzai docs: fix table format 11d8418 on Mar 27, 2022 243 commits

conf	feat: add resume training	3 years ago
models	fix: google large gpu memory usage	3 years ago
.gitignore	refactor: add runs and checkpoints folders to .gitignore file	3 years ago
README.md	docs: fix table format	last year
dataset.py	Remove trailing whitespaces	3 years ago
lr_finder.py	feat: remove argument 'w' and set num_works to 4	3 years ago
test.py	feat: change argument gpu to store_true	3 years ago
train.py	feat: print epoch during testing	3 years ago
utils.py	feat: add stochastic depth network	3 years ago

About

Practice on cifar100(ResNet, DenseNet, VGG, GoogleNet, InceptionV3, InceptionV4, Inception-ResNetv2, Xception, Resnet In Resnet, ResNext, ShuffleNet, ShuffleNetv2, MobileNet, MobileNetv2, SqueezeNet, NasNet, Residual Attention Network, SENet, WideResNet)

deep-learning pytorch image-classification densenet resnet squeezenet inceptionv3 googlenet resnext wideresnet cifar100 mobilenet inceptionv4 shufflenet xception nasnet inception-resnet-v2

Readme

对MFM的改进尝试

1. 将Max替换为Min将得到类似性能
2. 将Max替换为其光滑近似logsumexp不仅得到类似性能,还能提高训练速度和推理速度。
3. 还有大量失败的改进的结果

操作	准确性	测试集运行时间
LightCNN9(mfm): 1,579,204	0.6372	2.85s
min	0.6208	-
mean	0.5487	-
softmax	0.0100	-
logsumexp(K=1)	0.0100	-
logsumexp(K=10)	0.6229	1.60s
logsumexp(K=30)	0.6180	1.32s
logsumexp(K=100)	0.6371	1.53s
topk	0.0100	-
Feature map with large response	-	-
Feature map stack with large response	0.5338	-
mobilenetv2: 2,369,380	0.6778	8.30s
LightCNN29: 8,671,620	0.6612	5.01s

Min

1. 希望两个特征图都能比较好地提取特征
2. 性能损失不大。

```
class lmfm(nn.Module):  
    def forward(self, x):  
        x = self.filter(x)  
        out = torch.split(x, self.out_channels, 1)  
        return torch.min(out[0], out[1])
```

操作	准确性	测试集运行时间
LightCNN9(mfm): 1,579,204	0.6372	2.85s
min	0.6208	-
mean	0.5487	-
softmax	0.0100	-
logsumexp(K=1)	0.0100	-
logsumexp(K=10)	0.6229	1.60s
logsumexp(K=30)	0.6180	1.32s
logsumexp(K=100)	0.6371	1.53s
topk	0.0100	-
Feature map with large response	-	-
Feature map stack with large response	0.5338	-
mobilenetv2: 2,369,380	0.6778	8.30s
LightCNN29: 8,671,620	0.6612	5.01s

Mean

1. 两个特征图的意见平均
2. 性能一定程度的损失。

```
def forward(self, x):  
    x = self.filter(x)  
    out = torch.split(x, self.out_channels, 1)  
    return (out[0]+out[1])/2
```

操作	准确性	测试集运行时间
LightCNN9(mfm): 1,579,204	0.6372	2.85s
min	0.6208	-
mean	0.5487	-
softmax	0.0100	-
logsumexp(K=1)	0.0100	-
logsumexp(K=10)	0.6229	1.60s
logsumexp(K=30)	0.6180	1.32s
logsumexp(K=100)	0.6371	1.53s
topk	0.0100	-
Feature map with large response	-	-
Feature map stack with large response	0.5338	-
mobilenetv2: 2,369,380	0.6778	8.30s
LightCNN29: 8,671,620	0.6612	5.01s

Softmax

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

1. 失去了全局信息, (0.1,0.1), (0.9,0.9) 经过Softmax后结果一样都是(0.5,0.5)
2. 性能完全不行。

```
def forward(self, x):  
    x = self.filter(x)  
    out = torch.split(x, self.out_channels, 1)  
    output_shape = out[0].shape  
    a= out[0].reshape(output_shape[0], -1)  
    b= out[1].reshape(output_shape[0], -1)  
    a = torch.stack((a,b), dim=1)  
    a = F.softmax(a, dim=1)  
    a = torch.max(a, dim=1)[0]  
    out = a.reshape(output_shape)  
    return out
```

操作	准确性	测试集运行时间
LightCNN9(mfm): 1,579,204	0.6372	2.85s
min	0.6208	-
mean	0.5487	-
softmax	0.0100	-
logsumexp(K=1)	0.0100	-
logsumexp(K=10)	0.6229	1.60s
logsumexp(K=30)	0.6180	1.32s
logsumexp(K=100)	0.6371	1.53s
topk	0.0100	-
Feature map with large response	-	-
Feature map stack with large response	0.5338	-
mobilenetv2: 2,369,380	0.6778	8.30s
LightCNN29: 8,671,620	0.6612	5.01s

Logsumexp

1. 对max的光滑近似
2. 性能差不多但早可能

max

后面谈到的大部分内容，基础点就是max操作的光滑近似，我们有：

$$\max(x_1, x_2, \dots, x_n) = \lim_{K \rightarrow +\infty} \frac{1}{K} \log \left(\sum_{i=1}^n e^{Kx_i} \right) \quad (1)$$

所以选定常数 K ，我们就有近似：

$$\max(x_1, x_2, \dots, x_n) \approx \frac{1}{K} \log \left(\sum_{i=1}^n e^{Kx_i} \right) \quad (2)$$

在模型中，很多时候可以设 $K = 1$ ，这等价于把 K 融合到模型自身之中，所以最简单地有：

$$\begin{aligned} \max(x_1, x_2, \dots, x_n) &\approx \log \left(\sum_{i=1}^n e^{x_i} \right) \\ &\triangleq \text{logsumexp}(x_1, x_2, \dots, x_n) \end{aligned} \quad (3)$$

这里出现了logsumexp，这是一个很常见的算子，在这里它是max函数的光滑近似。没错，max函数的光滑近似其实是logsumexp，而不是字面上很像的softmax。相关推导还可以参考我之前写的《寻求一个光滑的最大值函数》。

操作	准确性	测试集运行时间
LightCNN9(mfm): 1,579,204	0.6372	2.85s
min	0.6208	-
mean	0.5487	-
softmax	0.0100	-
logsumexp(K=1)	0.0100	-
logsumexp(K=10)	0.6229	1.60s
logsumexp(K=30)	0.6180	1.32s
logsumexp(K=100)	0.6371	1.53s
topk	0.0100	-
Feature map with large response	-	-
Feature map stack with large response	0.5338	-
mobilenetv2: 2,369,380	0.6778	8.30s
LightCNN29: 8,671,620	0.6612	5.01s

Topk

1. 同个样本的全体特征图上每个位置的最大out_channel个元素重新组成特征图。
2. 性能完全不行。

操作	准确性	测试集运行时间
LightCNN9(mfm): 1,579,204	0.6372	2.85s
min	0.6208	-
mean	0.5487	-
softmax	0.0100	-
logsumexp(K=1)	0.0100	-
logsumexp(K=10)	0.6229	1.60s
logsumexp(K=30)	0.6180	1.32s
logsumexp(K=100)	0.6371	1.53s
topk	0.0100	-

```
def forward(self, x):  
    x = self.filter(x)  
    out = torch.topk(x, self.out_channels, dim=1, largest=True, sorted=True)  
    return out.values
```

Feature map with large response

1. 希望保持特征图的完整性，两个特征图中选择响应大的那个
2. 效率太低完全跑不起来，1小时多连1个epoch都没跑完，我暂时没想到可以快速实现这个想法的代码。

```
def forward(self, x):
    x = self.filter(x)
    maxfeaturemap2 = x[:, :, self.out_channels]
    size_half = int(maxfeaturemap2[0].numel()/2)
    for index in range(x.shape[0]):
        for i in range(self.out_channels):
            a = x[index][i]
            b = x[index][i+self.out_channels]
            counts = int(torch.sum(a > b))
            maxfeaturemap2[index][i] = a if counts > size_half else b
    return maxfeaturemap2
```

操作	准确性	测试集运行时间
LightCNN9(mfm): 1,579,204	0.6372	2.85s
min	0.6208	-
mean	0.5487	-
softmax	0.0100	-
logsumexp(K=1)	0.0100	-
logsumexp(K=10)	0.6229	1.60s
logsumexp(K=30)	0.6180	1.32s
logsumexp(K=100)	0.6371	1.53s
topk	0.0100	-
Feature map with large response	-	-
Feature map stack with large response	0.5338	-
mobilenetv2: 2,369,380	0.6778	8.30s
LightCNN29: 8,671,620	0.6612	5.01s

Feature map stack with large response

1. 根据具备更大元素个数，选择out[0]还是out[1]。
2. 性能一定程度的损失。性能损失不大。

```
def forward(self, x):  
    x = self.filter(x)  
    out = torch.split(x, self.out_channels, 1)  
    size_half = int(out[0].numel()/2)  
    return out[0] if int(torch.sum(out[0] > out[1])) > size_half else out[1]
```

操作	准确性	测试集运行时间
LightCNN9(mfm): 1,579,204	0.6372	2.85s
min	0.6208	-
mean	0.5487	-
softmax	0.0100	-
logsumexp(K=1)	0.0100	-
	0.6229	1.60s
	0.6180	1.32s
	0.6371	1.53s
	0.0100	-
	-	-
Feature map stack with large response	0.5338	-
mobilenetv2: 2,369,380	0.6778	8.30s
LightCNN29: 8,671,620	0.6612	5.01s

Take Home Message

1. 由于移动设备电量有限、用户对AI应用实时性要求高、用户自身的隐私保护需求的原因，如何设计出更加轻量化的神经网络模型从而使得部署到移动设备上的模型能够耗电量少地、快速地、本地地完成任务是工业界和研究者一直追求的目标。
2. LightCNN在准确性和实时性（参数量）上实现了较好的平衡。如果有轻量化网络设计的需求，可以考虑LightCNN或者MFM操作。
3. Max-Feature-Map的原理。
4. <https://github.com/weiaicunzai/pytorch-cifar100> 很好用。
5. 将Max替换为Min将得到类似性能。
6. 将Max替换为其光滑近似logsumexp不仅得到类似性能，还能提高训练速度和推理速度。
7. 如何对特征图进行融合，还有很多值得尝试的方向，具体原理也还待进一步阐明。

