


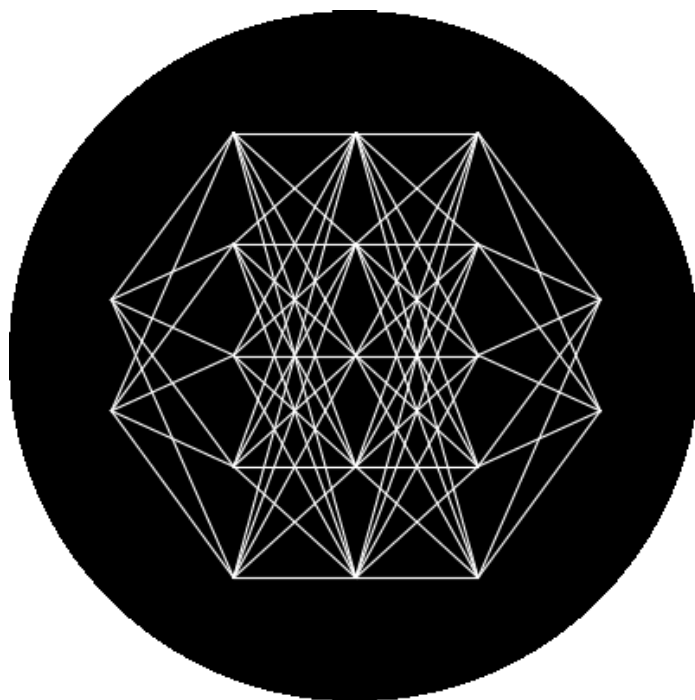


[首页](#) / [社区](#) / [单样本学习 \(One shot learning\) 和孪生网络 \(Siamese Network\) 简介](#)

# 单样本学习 (One shot learning) 和孪生网络 (Siamese Network) 简介

 bluefish 发布于4月前 阅读885次 |  4 人点赞  0 条评论

社区推荐:掘金是一个面向开发者的技术社区,内容涵盖了前后端,Android 和 iOS 开发,每天更新深度文章,最新技术资讯,优质开源库推送。无论是入门还是进阶,掘金都可以帮助你成为更优秀的开发者。



这篇博客翻译自 *One Shot Learning and Siamese Networks in Keras*, 翻译后投稿到了新智元【深度神经网络 One-shot Learning】孪生网络少样本精准分类, 本文算是授权转载。

## 背景

传统观点一般认为深度神经网络通常比较擅长从高维数据中学习,例如图像或者语言,但这是建立在它们有大量标记的样本来训练的情况下。然而,人类却拥有单样本学习的能力--如果你找一个从来没有见过小铲刀的



bluefish

被 92人 关注, 获得了132个 喜欢

+关注

真正的敏捷是一件很有价值的事。因为时间是衡量事业的标准,如金钱是衡量货物的标准。—— 弗·培根\*



 4人点赞

 收藏

共有0 条点评, 0人 收藏

我要点评



 主题目录 [只显示目录](#)

- 背景
- 定义这个问题- N类别单样本学习 (One-shot Learning)
- 关于数据集Omniglot !
- 一个单样本学习的baseline--1近邻
- 使用深度神经网络来做单样本学习? !
- 孪生网络(Siamese networks)
- 网络架构
- 观察一下: 逐对训练的有效的数据集大小
- 代码
- 结果
- 讨论
- 下一步是啥?
- [结论](#)



人, 给它们一张小铲刀的图片, 它们应该就能很成功的将它从其他厨房用具里面鉴别出来。



(从来没有进过厨房?现在你有机会来测试一下你的单样本学习能力了!右边图像中, 哪个是与左边大图片相同类别的?)

这是一个人们认为很容易的飞起, 但是直到我们想写一个算法让它去做这件事。。。 (那就GG了) 很明显, 机器学习系统很希望拥有这种快速从少量样本中去学习的能力, 因为收集和标记数据是一个耗时费力的工作。我认为这是通往通用人工智能的漫漫长路中很重要的一步。

最近涌现出来很多有趣的基于神经网络的单样本样本学习论文, 它们已经得到了一些不错的结果。这是一个让我激动的新领域, 所以我想去对它做一个简要介绍, 来让深度学习新手更好的认识它。

在这篇博客中, 我想:

- 介绍并定义单样本学习问题
- 描述单样本分类问题的基准, 并给出一个其性能的 baseline
- 给出一个少样本学习的例子并部分的实现 这篇论文 中提到的模型
- 指出一些大家通常不会想到的小点子



定义这个问题- N类别单样本学习(One-shot

## Learning)

在我们解决任何问题之前, 我们应该精准的定义处这个问题到底是什么, 下面是单样本分类问题的符号化表示: 我们的模型只获得了很少的标记的训练样本  $S$ , 它有  $N$  个样本, 每个相同维度的向量有一个对应的标签  $y$

$$S = \{(x_1, y_1), \dots, (x_N, y_N)\}$$

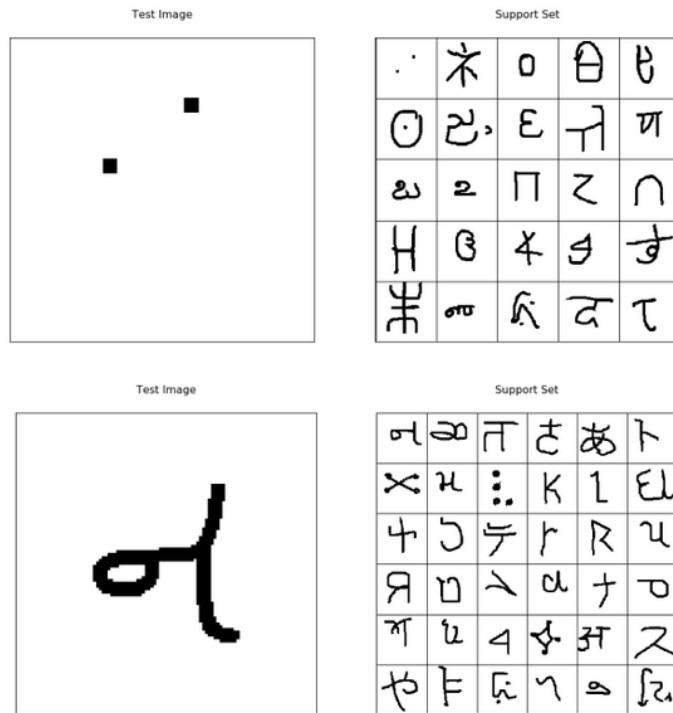
再给出一个待分类的测试样例。因为样本集中每个样本都有一个正确的类别, 我们的目标是正确的预测  $y \in S$  中哪一个是 的正确标签。

这里有很多种定义问题的方式, 但上面是我们的定义, 注意这里有一些事项需要记录一下:

- 现实生活中可能约束更少, 可能一张图片并不见得只有一个正确的类别。
- 这个问题很容易泛化到k-shot 学习, 我们只需要把每个类别 $y_{\{i\}}$   $y_i$  的单个样本换成k个样本就可以了。
- 当N很的时候, 可能 有更多可能的类别, 所以正确预测类别更难。
- 随机猜的正确率是  $\frac{100}{n} \%$ 。

这里有一些在Omniglot数据集上单样本学习的例子, 我会在下一部分介绍它。

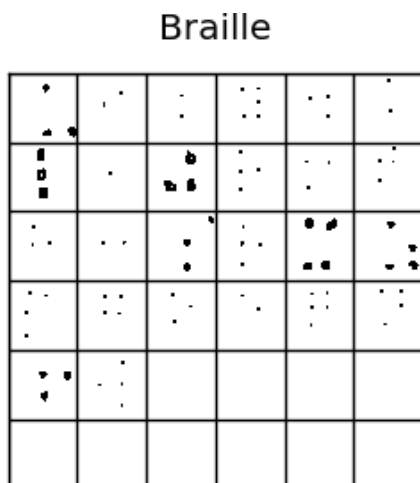




图示分别为9类, 25类, 36类的单样本学习任务

### 关于数据集Omniglot!

Omniglot数据集 拥有50种, 1623类手写字符。对于每类字符仅有20个样本, 每个样本分别由不同的人绘制而成, 分辨率为105\*105



### Bengali

ঐ	ঐ	আ	ন	ট	ল	ঊ
ঔ	ক	য়	অ	ও	ট	ব
দ	থ	ষ	ঝ	এ	ই	জ
শ	ছ	ভ	ড	ম	ণ	য়
ঙ	ত	হ	স্ব	য	উ	থ
চ	গ	ঢ	ল	ঝ	ঞ	ষ
ঠ	ফ	ব	ব			

### Sanskrit

प	झ	झ	ष	म	ल	घ
ट	ठ	क	त्र	फ	अ	व
ड	ए	न	ज	ग	थ	स
द	आ	भ	ओ	य	उ	त
र	छ	ण	ड	ल	थ	ढ
क्व	च	इ	ब	ह	श	क

### Greek

ϕ	Λ	β	δ	λ
μ	α	κ	χ	ν
υ	θ	γ	ι	σ
ω	π	η	ο	ε
ρ	ξ	ζ	ψ	



Futurama



Hebrew



上面是omniglot数据集的一些例子, 如图所示, 这里有很多  
 多种字符 如果你喜欢机器学习, 你肯定听说过 MNIST  
 数据集 .Omniglot有时被成为mnist的转置, 因为它有  
 1623类字符, 每类只有20个样本, 相比mnist的10个类  
 别, 每个类别都有上千样本, 正好相反.omniglot还有创  
 作的笔画数据, 但是我们这里用不到它。通常, 我们把样  
 本分为30类训练样本, 剩下20类作为评估。所有这些不  
 同的字符可以组成很多种单样本学习任务, 所以它确实  
 是一个单样本学习的一个很好的评估标准。

### 一个单样本学习的**baseline--1**近邻

最简单的分类方式是使用k近邻方法, 但是因为每个类  
 别只有一个样本, 所以我们需要用1近邻。这很简单, 只  
 需要计算测试样本与训练集中每个样本的的欧式距离,



然后选择最近的一个就可以了：

$$C(\hat{x}) = \operatorname{argmin}_{c \in S} \|\hat{x} - x_c\|$$

根据Koch等人的论文, 在omniglot数据集中的20类上, 单样本分类, 1-nn可以得到大约28%的精度, 28%看起来很差, 但是它已经是随机猜测(5%)的6倍精度了。这是一个单样本学习算法最好的baseline或者“合理性测试”了。

Lake等人的 Hierarchical Bayesian Program Learning, 层次贝叶斯程序学习(以下简称HBPL) 得到了大约95.2%的精度, 非常不错。我只看懂了30%, 但它非常有趣, 它与深度学习直接从原始像素上训练相比, 是风马牛不相及的, 因为：

- HBPL使用笔画数据, 而不是仅仅用原始像素
- HBPL在omniglot数据集上学习一个笔画的生成模型, 这个算法需要更加复杂的标注, 所以不像深度学习能直接从狗、卡车、大脑扫描图以及小铲子等图片的原始像素上去做单样本学习, 这种图片也不是由笔画构成的。Lake等人也指出, 人类可以在omniglot数据集20类样本上达到95.5%的精度, 仅仅比HBPL高一点。在钻牛角尖思想的引导下, 我亲自试验了一下20类任务, 达到了97.2%的精度。我并不是做的真正的单样本学习, 因为很多符号我本来就认识, 因为我熟悉希腊字母、平假名和片假名, 我把这些我本来就认识的移除, 我还是得到了96.7%的精度。我认为是我从自己吓人的字迹中练就了超人般的字符识别能力。

## 使用深度神经网络来做单样本学习?!

如果我们单纯的训练一个用交叉熵损失的softmax分类器神经网络来做单样本学习, 很明显, 网络会严重过拟合。即便是每类给出上百个样本, 现代的神经网络依然会过拟合。深度网络有百万级别的参数来拟合训练数据, 所以它们可以学习到一个巨大的函数空间(正式来说, 是因为它们有一个很高的 VC维, 这就是为什么它们为什么可以很好的从复杂的高维数据中学习的部分



原因)。很不幸的是,神经网络这个优势又成为了它们做单样本学习的一大障碍。当有百万级的参数需要做梯度下降,有这么多可能学习到的映射关系,我们怎么能设计一个网络,让他可以从单个样本去学习呢?

人类很容易从单个样本就能学会小铲刀或者字母\Theta $\Theta$ 的意思,因为我们一辈子一直都在从相似对象中观察和学习。把一个随机初始化的神经网络与人类这种花了一辈子时间去识别物体和符号相比,的确不太公平,因为随机初始化的神经网络对数据的映射结构缺乏先验。这也是为什么我看到的单样本学习论文都是采用的从其他任务上的知识迁移方法。

神经网络非常擅长从结构化的复杂/高维数据中(例如图像)提取特征。如果给神经网络与单样本学习任务相似的训练数据,它或许能够从这些数据中学习到有用的特征,这些特征可能不需要调整就能用到单样本学习。这样,我们仍旧能叫他单样本学习,因为辅助的训练数据与单样本测试的数据不是相同的类别。

(注意:这里的 特征 指的是“被用来训练的数据的映射数据”(译者注:例如经过CNN提取到的特征))

接下来以一个有趣的问题就是我们如何设计一个神经网络让他来学习特征?最显而易见的方法就是用迁移学习(如果有标记数据的话)来做这件事--在训练数据上训练一个softmax分类器,然后在单样本学习任务的数据集上微调最后一层的权重。实际上,神经网络分类器在omniglot数据集上不会有什么良好的表现,因为每类的样本仅有几个,即使是微调最后一层的权重,网络也会在训练集上过拟合。但这种方法也比使用L2距离的k近邻方法要好很多了(参考 Matching Networks for One shot learning 中对各种单样本学习方法的效果的比较)。

这里还是有一种方法来做单样本学习的!忘了1近邻方法?这个简单的,非参的单样本学习器计算测试集中的样本与训练集中每个样本的L2距离并选择最近的作为它的类别。这种方法是ok的,但是L2距离会陷入严重的维度灾难问题,所以它在成千维的数据上(像omniglot)





上表现不太好。另外,如果你有两个接近相同的图片,如果你把其中一张图片的像素向右移动一点,那么两张图片的L2距离会从0一下子变得非常高。L2距离在这种任务上是一个非常糟糕的度量。深度学习能奏效吗?我们可以使用深度卷积神经网络来学习一种非参的近邻分类器可以使用的相似性函数。

## 孪生网络(Siamese networks)



我原本打算放一张连孪双胞胎作为这一节的介绍图片呢,但是我最终认为孪生的小猫的图片可能更适合。

我在这篇教程中会实现 这篇极好的论文 中的方法。Koch等人的单样本学习方法是同时给神经网络两张图片以让他来猜测两张图片是否是同一个类别。当我们做上面提到的单样本分类任务的时候,网络可以比较测试集与训练集中的每张图片,然后挑选出哪一张与它最可能是同样类别。所以我们想让神经网络架构同时输入两张图片,输出它们属于同一个类别的概率。

假设  $x_1$  和  $x_2$  是数据集中的2个类别,我们让  $x_1 \circ x_2$  表示  $x_1$  和  $x_2$  是同一个类别。注意  $x_1 \circ x_2$  与  $x_2 \circ x_1$  是等价的--这意味这如果我们颠倒输入图片的顺序,输出的概率是完成相同的--  $p(x_1 \circ x_2)$  与  $p(x_2 \circ x_1)$  相等。这被成为 对称性,孪生网络就是依赖它设计的。

对称性是非常重要的,因为它要学习一个距离度量--  $x_1$



到  $x_2$  的距离应该等于  $x_2$  到  $x_1$  的距离。

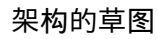
如果我们仅仅把两个样本拼接起来, 把它作为神经网络的单一的输入, 每个样本将会是与一个不同权重集合的矩阵相乘(或缠绕), 这会打破对称性。没问题, 这样子网络依然能成功的为每个输入学习到完全相同的权重, 但是对两个输入学习相同的权重集合会更容易一些。所以我们可以让两个输入通过完全相同共享参数的网络, 然后使用绝对差分作为线性分类器的输入--这是孪生网络必须的结构。两个完全相同的双胞胎, 共用一个头颅, 这就是孪生网络的由来。

## 网络架构

不幸的是, 如果我们想介绍清楚为什么卷积神经网络可以工作, 那这篇博客就会长了去了。如果你想理解卷积神经网络, 我建议你去学习 CS231 然后去 colah .对于没有深度学习经验的读者, 我只能这样概况CNN: 一张图像是3D像素矩阵, 一个卷积层是一个神经元连接到一个小网格的像素或者神经元, 然后使用与这个神经相同的连接在一张图片或者特征块上滑动一遍, 生成另外一个3d的神经元。一个最大池化层让特征图在空间上变小。很多这样的层按照顺序堆叠到一起就可以用梯度下降来训练了, 它们在图像任务上表现良好。

我仅对CNN做一个精简介绍, 因为这不是本文的重点。Koch等人使用卷积孪生网络去分类成对的omniglot图像, 所以这两个孪生网络都是卷积神经网络。这两个孪生网络每个的架构如下: 64通道的 $10 \times 10$ 卷积核,  $\text{relu} \rightarrow \text{max pool} \rightarrow 128$ 通道的 $7 \times 7$ 卷积核,  $\text{relu} \rightarrow \text{max pool} \rightarrow 128$ 通道的 $4 \times 4$ 卷积核,  $\text{relu} \rightarrow \text{max pool} \rightarrow 256$ 通道的 $4 \times 4$ 卷积核。孪生网络把输入降低到越来越小的3d张量上, 最终它们经过一个4096神经元的全连接层。两个向量的绝对差作为线性分类器的输入。这个网络一共有38,951,745个参数--96%的参数属于全连接层。这个参数量很大, 所以网络有很高的过拟合风险, 但是成对的训练意味着数据集是很大的, 所以过拟合问题不出现。




$$L(x_1, x_2, t) = t \cdot \log(p(x_1 \circ x_2)) + (1 - t) \cdot \log(1 - p(x_1 \circ x_2)) + \lambda \cdot \|w\|_2$$
$$C(\hat{x}, S) = \underset{c}{\operatorname{argmax}} P(\hat{x} \circ x_c), x_c \in S$$

观察一下:逐对训练的有效的数据集大小

我注意到，采用逐对训练的话，将会有平方级别对的图像用来训练模型，这让模型很难过拟合，好酷。假设我们有  $E$  类，每类有  $C$  个样本。一共有  $C \cdot E$  张图片，总共可能的配方数量可以这样计算：



$$N_{pairs} = \binom{C \cdot E}{2} = \frac{(C \cdot E)!}{2!(C \cdot E - 2)!} \text{ 对于omniglot中}$$

的964类(每类20个样本),这会有185,849,560个可能的配对,这是巨大的!然而,孪生网络需要相同类的和不同类的配对都有。每类  $E$  个训练样本,所以每个类别有  $\binom{E}{2}$  对,这意味着这里有  $N_{same} = \binom{E}{2} \cdot C$  个相同类别的配对。--对于Omniglot有183,160对。及时183,160对已经很大了,但他只是所有可能配对的千分之一,因为相同类别的配对数量随着E平方级的增大,但是随着C是线性增加。这个问题非常重要,因为孪生网络训练的时候,同类别和不同类别的比例应该是1:1.--或许它表明逐对训练在那种每个类别有更多样本的数据集上更容易训练。

## 代码

如果你更喜欢用jupyter notebook?这里是传送门

下面是模型定义,如果你见过keras,那很容易理解。我只用Sequential()来定义一次孪生网络,然后使用两个输入层来调用它,这样两个输入使用相同的参数。然后我们使用绝对距离合并起来,添加一个输出层,使用二分类交叉熵损失来编译这个模型。



```
from keras.layers import Input, Conv2D, Lambda
from keras.models import Model, Sequential
from keras.regularizers import l2
from keras import backend as K
from keras.optimizers import SGD, Adam
from keras.losses import binary_crossentropy
import numpy.random as rng
import numpy as np
import os
import dill as pickle
import matplotlib.pyplot as plt
from sklearn.utils import shuffle

def W_init(shape, name=None):
    """Initialize weights as in paper"""
    values = rng.normal(loc=0, scale=1e-2, size=shape)
    return K.variable(values, name=name)
#//TODO: figure out how to initialize layer
def b_init(shape, name=None):
    """Initialize bias as in paper"""
    values = rng.normal(loc=0.5, scale=1e-2, size=shape)
    return K.variable(values, name=name)

input_shape = (105, 105, 1)
left_input = Input(input_shape)
right_input = Input(input_shape)
#build convnet to use in each siamese 'leg'
convnet = Sequential()
convnet.add(Conv2D(64, (10, 10), activation='relu',
                  kernel_initializer=W_init))
convnet.add(MaxPooling2D())
convnet.add(Conv2D(128, (7, 7), activation='relu',
                  kernel_regularizer=l2(2e-4)))
convnet.add(MaxPooling2D())
convnet.add(Conv2D(128, (4, 4), activation='relu'))
convnet.add(MaxPooling2D())
convnet.add(Conv2D(256, (4, 4), activation='relu'))
convnet.add(Flatten())
convnet.add(Dense(4096, activation="sigmoid"),
#encode each of the two inputs into a vector
encoded_l = convnet(left_input)
encoded_r = convnet(right_input)
#merge two encoded inputs with the l1 distance
```



```
L1_distance = lambda x: K.abs(x[0]-x[1])
both = merge([encoded_l,encoded_r], mode = 'concat')
prediction = Dense(1,activation='sigmoid',bias_initializer='zeros')(both)
siamese_net = Model(input=[left_input,right_input],output=prediction)
#optimizer = SGD(0.0004,momentum=0.6,nesterov=True)

optimizer = Adam(0.00006)
#//TODO: get layerwise learning rates and momentum
siamese_net.compile(loss="binary_crossentropy",optimizer=optimizer)

siamese_net.count_params()
```

原论文中每个层的学习率和冲量都不相同--我跳过了这个步骤, 因为使用keras来实现这个太麻烦了, 并且超参数不是该论文的重点。Koch等人增加向训练集中增加失真的图像, 使用150,000对样本训练模型。因为这个太大了, 我的内存放不下, 所以我决定使用随机采样的方法。载入图像对或许是这个模型最难实现的部分。因为这里每个类别有20个样本, 我把数据重新调整为  $N\_classes \times 20 \times 105 \times 105$  的数组, 这样可以很方便的来索引。



```

class Siamese_Loader:
    """For loading batches and testing task"""
    def __init__(self, Xtrain, Xval):
        self.Xval = Xval
        self.Xtrain = Xtrain
        self.n_classes, self.n_examples, self.n_val, self.n_ex_val, _, _ = Xval

    def get_batch(self, n):
        """Create batch of n pairs, half support, half query
        categories = rng.choice(self.n_classes, n//2, replace=False)
        pairs = np.zeros((n, self.h, self.w, self.c))
        targets = np.zeros((n,))
        targets[n//2:] = 1
        for i in range(n):
            category = categories[i]
            idx_1 = rng.randint(0, self.n_examples)
            pairs[0][i, :, :, :] = self.Xtrain[idx_1, :, :, :]
            idx_2 = rng.randint(0, self.n_examples)
            #pick images of same class for query
            category_2 = category if i >= n//2 else categories[i]
            pairs[1][i, :, :, :] = self.Xtrain[idx_2, :, :, :]
        return pairs, targets

    def make_onehot_task(self, N):
        """Create pairs of test image, support image
        categories = rng.choice(self.n_val, N, replace=False)
        indices = rng.randint(0, self.n_ex_val, N)
        true_category = categories[0]
        ex1, ex2 = rng.choice(self.n_examples, 2, replace=False)
        test_image = np.asarray([self.Xval[ex1, :, :, :], self.Xval[ex2, :, :, :]])
        support_set = self.Xval[categories, :, :, :]
        support_set[0, :, :, :] = self.Xval[true_category, :, :, :]
        support_set = support_set.reshape(N, self.h, self.w, self.c)
        pairs = [test_image, support_set]
        targets = np.zeros((N,))
        targets[0] = 1
        return pairs, targets

    def test_onehot(self, model, N, k, verbose=0):
        """Test average N way onehot learning"""
        pass
        n_correct = 0

```



```

    if verbose:
        print("Evaluating model on {}".format(k))
    for i in range(k):
        inputs, targets = self.make_one_shot_examples(targets)
        probs = model.predict(inputs)
        if np.argmax(probs) == 0:
            n_correct+=1
    percent_correct = (100.0*n_correct/k)
    if verbose:
        print("Got an average of {}% {}".format(percent_correct, k))
    return percent_correct

```

下面是训练过程了。没什么特别的, 除了我监测的是验证机精度来测试性能, 而不是验证集上的损失。

```

evaluate_every = 7000
loss_every=300
batch_size = 32
N_way = 20
n_val = 550
siamese_net.load_weights("PATH")
best = 76.0
for i in range(900000):
    (inputs,targets)=loader.get_batch(batch_size,N_way)
    loss=siamese_net.train_on_batch(inputs,targets)
    if i % evaluate_every == 0:
        val_acc = loader.test_oneshot(siamese_net,n_val)
        if val_acc >= best:
            print("saving")
            siamese_net.save('PATH')
            best=val_acc

    if i % loss_every == 0:
        print("iteration {}, training loss: {}".format(i, loss))

```

## 结果

一旦学习曲线变平整了, 我使用在20类验证集合上表现最好的模型来测试。我的网络在验证集上得到了大约83%的精度, 原论文精度是93%。或许这个差别是因为我没有实现原论文中的很多增强性能的技巧, 像逐层的学习率/冲量, 使用数据失真的数据增强方法, 贝叶斯超参





数优化, 并且我迭代的次数也不够。我并不担心这个, 因为这个教程侧重于简要介绍单样本学习, 而不是在那百分之几的分类性能上钻牛角夹。这里不缺乏这方面的资源。

我很好奇, 模型的精度是怎么随样本的类别数目N变化的, 所以我把它画了出来, 与1近邻, 随机猜测以及模型在训练集上的精度的比较。



## 结果

如图所示, 验证集上的精度要比训练集上差一些, 尤其是当N的数量很多的时候, 这里面肯定有过拟合的问题。我们也想测试一下传统的正则化方法(像dropout)在验证集与训练集完全不同的时候的表现。对于较大的N, 它比我期待中的要好, 在50-60种类别上, 仍旧有65%的平均精度。

## 讨论

现在我们只是训练了一个来做鉴别相同还是不同的二分类网络。更重要的是, 我们展现了模型能够在没有见过的字母表上的20类单样本学习的性能。当然, 这不是使用深度学习来做单样本学习的唯一方式。

正如我前面提到的, 我认为这个孪生网络的最大缺陷是它要拿测试图像与训练集中图像逐个比较。当这个网络将测试图像与任何图像  $x_1$  相比, 不管训练集是什么,  $p(\hat{x} \circ x_1)$  都是相同的。这很愚蠢, 假如你在做单样本学



习任务,你看到一张图片与测试图像非常类似。然而,当你看到训练集中另外一张图片也与测试集非常相似,你就会对它的类别没那么自信了。训练目标与测试目标是不同的,如果有一个模型可以很好的比较测试图片与训练集,并且使用仅仅有一个训练图片与之拥有相同类别的限制,那模型会表现的更好。

Matching Networks for One Shot learning 这篇论文就是做这个的。它们使用深度模型来端到端的学习一个完整的近邻分类器,而不是学习相似度函数,直接在单样本任务上训练,而不是在一个图像对上。Andrej Karpathy's notes 很好的解释了这个问题。因为你正在学习机器分类,所以你可以把他视为元学习(meta learning)。One-shot Learning with Memory-Augmented Neural Networks 这篇论文解释了单样本学习与元学习的关系,它在omniglot数据集上训练了一个记忆增强网络,然而,我承认我看不懂这篇论文。

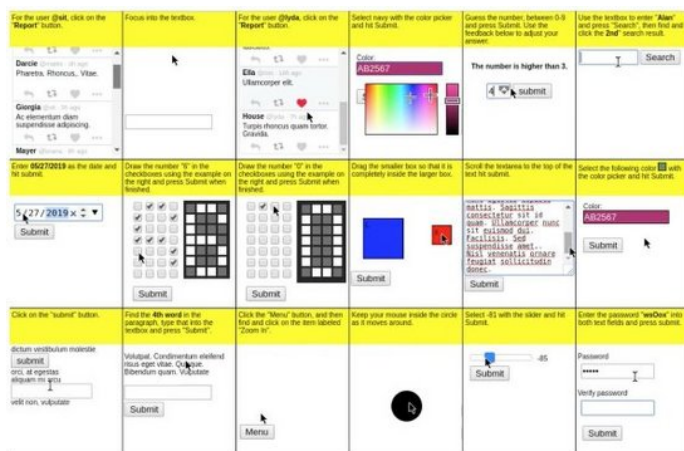
下一步是啥?

---

omniglot数据集是2015年的了,现在已经有了可拓展机器学习算法在特定的单样本学习上达到了人类的水平。希望未来有一天omniglot数据集会像mnist之于监督学习那样,成为单样本学习的标准检测数据集。

图像分类挺酷的,但我并不认为它是机器学习界最有趣的问题。现在我们知道了深度单样本学习有了不错的效果,我想如果尝试把单样本学习应用到更有挑战性的任务上,那样才是真酷。单样本学习的想法可以被用到样本效率更高的增强学习上面,尤其是像OpenAI's Universe这样的问题,这些问题有很多马尔科夫决策过程或者环境,它们拥有类似的视觉和动态信息。如果有一个增强学习机制能以类似马尔科夫决策过程学习后可以有效的探索新环境,那样简直酷毙了。





## OpenAI的比特世界

单样本模仿学习 是我最喜欢的单样本学习论文。它的目标是建立一个可以学习鲁棒的策略的机制,它在人类展示一次任务后就能解决这个任务。它是这样做的:

1. 建立一个神经网络,它可以映射当前状态和一个序列状态(人类示范)到一个动作
2. 把这个模型在人类示范动作和微小的变动任务对上训练,目标是让模型可以基于第一个示范来复现第二个示范动作。

这真是震惊我了,这提供了一种通往制造可以广泛应用的,可以学习的机器人的康庄大道啊。

把单样本学习引入到NLP也是一个很酷的idea。

Matching Networks在单样本语言模型上进行了尝试,仅给出很小的训练集,在测试集上填充缺失单词,这看起来工作的不错。棒!

## 结论

总而言之,谢谢你的阅读!我希望你已经通过单样本学习的方式掌握了单样本学习的概念;)如果没有学会,我很乐于听到你们的反馈和问题。

查看原文:单样本学习(One shot learning)和孪生网络(Siamese Network)简介

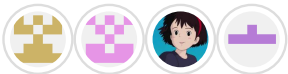




⬅️ 上一篇

下一篇 ➡️

感兴趣的用户



需要

登录

后回复方可回复, 如果你还没有账号你可以

注册

一个

帐号。

🕒 最新项目

在 MacOS 中自动安装微信防撤回插件并打开微信。

一个在线nginx配置生成器

ARK Chains的一个小型部署脚本, 可让您在几分钟内创建...

用于快速为本地开发环境生成HTTPS证书的一组脚本

rak8s - 利用Ansible搭建一个基于树莓派的Kubernetes群集

Ubuntu环境下使用NWJS对web应用封装

Antigen是shell ( zsh ) 插件的管理器

使用ARP协议在您的网络中检测新的未知设备的命令行工具

📖 相关主题

Linux 系统双网卡绑定配置实现负载均衡和故障转移



如何使用Kubernetes GPU集群自动化深度学习训练

OpenStack大规模部署详解

Linux下的hosts文件和network文件区别

Docker学习总结之Docker与Vagrant之间的特点比较

对于想学习Unix/Linux的人，需要掌握的一些很常用命令

Cobbler实现自动化安装操作系统

Linux网络源代码学习

