

机器学习纳米学位毕业项目

猫狗大战

Udacity 丁小刚

2018 年 03 月 30 日

目录

一、 问题定义	3
1. 项目概览	3
2. 问题说明	3
3. 指标.....	3
二、 问题分析	4
1. 数据研究	4
2. 探索可视化	5
3. 算法与方法	6
4. 基准测试	7
三、 问题解决方法	7
1. 数据预处理	7
2. 过程实施	10
3. 改进措施	12
四、 实验结果	12
1. 模型评估与验证	12
2. 结果分析	13
五、 结论	14
1. 结果可视化	14
2. 思考	15
3. 改进	15
参考文献	16

一、问题定义

1. 项目概览

猫狗大战[1]最开始是 Kaggle 在 2013 年举办的一个关于图像识别的比赛，目的是尽可能准确地识别出图片中的内容是猫还是狗。在此之前有计算机视觉专家表示，人们所用方法的准确率一般都很难超过 60%，基本就是比瞎猜稍微能强一点。近年来以神经网络为代表的一系列机器学习方法使得该类问题的准确率有了大幅提升。尤其是深度卷积神经网络在图片识别方面的应用。得益于此，几年前很难解决的图片识别问题，现在可以得到比较好的结果了，在 Kaggle 的猫狗大战公开算法排名中，有许多算法已经可以得到 90% 以上的准确率了。

2. 问题说明

这个比赛提供了两个数据集，一个是训练集，一个是测试集。训练集中包含了 25000 张已经标记了的图片，一半是猫，一半是狗，各 12500 张。测试集中包含了 12500 张未标记的图片。这些图片都是由微软研究院的 Asirra[2] 提供，内容均为日常生活中所拍摄的照片，所以图片背景非常多样，其中的猫和狗也是姿态各异，品种繁多，照片的光线有明有暗，有远有近，这些因素导致准确识别猫狗具有一定难度。

我们需要设计一个二元分类机器学习算法，利用训练集提供的数据，提取出猫狗这样的知识，来区分测试集中所有图片中包含狗的概率，如果图片中是狗，则算法得到的概率应该尽可能接近 1.0。

3. 指标

为了区分不同算法识别准确度，kaggle 提供了一个交叉熵的计算方式，用来量化算法最终的得分。

评估标准采用 Kaggle 上提供的交叉熵计算方法：

$$LogLoss = -\frac{1}{n} \sum_{i=1}^n [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

其中， n 表示测试集中所有图片的数量， \hat{y}_i 表示测试结果为狗的概率， y_i 的值在图片是狗时取 1，是猫时则取 0， $\log()$ 表示自然对数。

LogLoss 的值越小越好，最终的结果应该达到 kaggle 公开排名的前 10%，也就是小于目标 0.06127。

二、问题分析

1. 数据研究

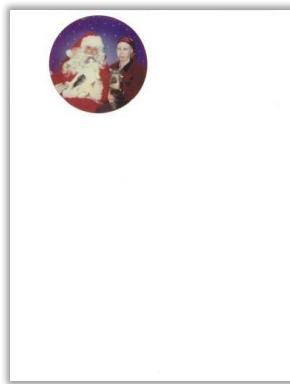


图 1. cat. 10029. jpg

训练数据集中包含有 25000 张图片，全部为彩色 jpg 格式，图片尺寸不一致，大多为 300x500 不等。其中 12500 张图片名含有 cat，并且绝大部分确实是猫，发现其中一张图片 cat. 10029. jpg 并不是猫，这应该属于误标的图片。剩余 12500 张图片名都包含有 dog，绝大部分都是狗。整体来看，这是一个已经标注了的猫狗图片集，并且包含有异常数据，即非猫非狗的图片。

测试数据集中包含有 12500 张图片，同样为彩色 jpg 格式，尺寸不统一，全部以数字命名。算法需要将最终的预测结果对应文件名存储为 csv 文件提交到 kaggle 来得到成绩。

由于训练数据集中包含有异常数据，所以这里准备采用基于 ImageNet 的预训练模型 Inception v3[3]、Xception[4]初步筛选出训练集中的非猫非狗图片，这些预训练模型已经能够分类 126 种狗和 7 种猫[5]，最后将这些非猫非狗图片从训练集当中剔除。还有一部分图片尺寸过小，清晰度不够，也将其剔除。这样有利于算法最终准确率的提升。

2. 探索可视化

随机查看几张训练集与测试集中的图片：



图 2. 训练集中的图片



图 3. 测试集中的图片

训练集与测试集中的猫狗都是姿态各异，背景复杂，光线明暗也不同，图片的尺寸也不固定，猫狗的品种也很多。

训练集中图片的尺寸分布：

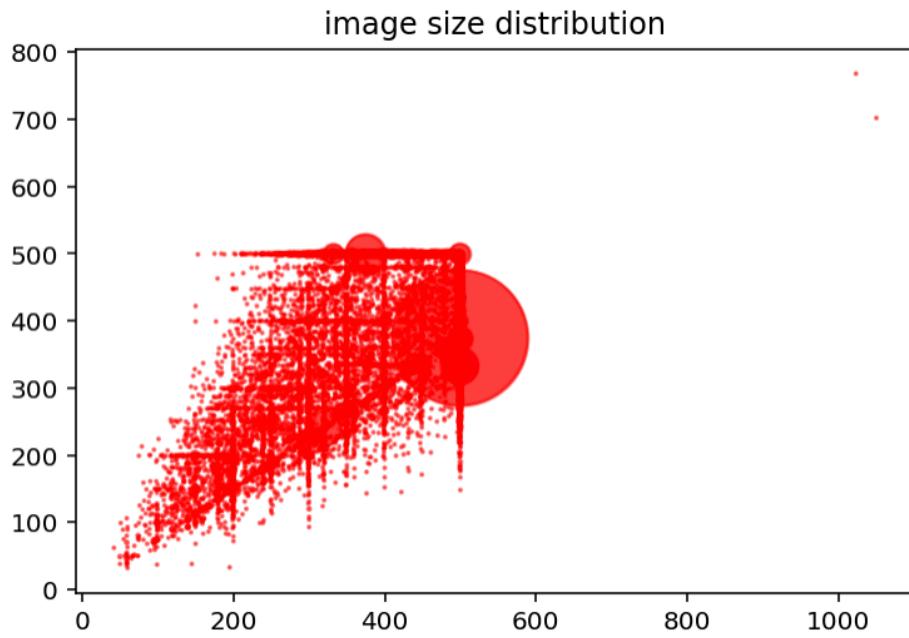


图 4. 训练集图片尺寸分布

图 3 中，每个红点的横纵坐标表示了图片的宽高尺寸，以红点为圆心的饼图大小代表了该尺寸下图片的数量。可以看出，大部分图片的尺寸宽高都在 300–500 范围内，有少数图片的宽高都小于 100，还有两张图片的宽高都大于 600。

3. 算法与方法

近年来卷积神经网络在图片识别领域里取得了十分显著的成就，我们准备采用卷积神经网络来处理这个问题。利用的工具则是基于 TensorFlow 的 Keras 框架[6]，TensorFlow 提供了许多机器学习的基础工具，Keras 可以迅速基于 TensorFlow 搭建一个完整的神经网络机器学习算法。

卷积神经网络包含有许多层次结构，原始数据可以是图像、文字序列、音频序列等，卷积层会不断从前一层提取特征信息，池化层会将特征信息的维度减小，同时强化不同特征信息所代表的特征，卷积层的堆叠可以得到越来越抽象的特征图序列，再利用池化层减小其尺寸，再拼接卷积层提取特征，再池化，特征图序列的长度越来越大，特征图尺寸越来越小，最后经过全局池化层，就可以得到一个一维向量，利用非线性激活函数将其映射为最终结果，利用 Dropout 防止过拟合，在猫狗大战中，最终就是得到一个概率值，如果在其他问题中将会得到一个结果向量，代表了解空间中的某一个结果。至此神经网络的前馈过程结束。再将所得到的结果与真实结果做差值，将误差反向传播回网络中，更新每一层的相关系数。将所有系数运算表示的结果与真实值之差的某种形式作为损失函数，用优化器寻找损失函数的最小值，以确定最优的网络参数组合。

不同的卷积层网络结构最终的性能不尽相同，目前比较成功的几个模型有 VGG16、ResNet50、InceptionV3、Xception、InceptionResNetV2 等[4, 7-9]。这些模型在 ImageNet 数据集上都得到了非常好的成绩。

VGG16 模型提出利用 3x3 的较小卷积核增加卷积层深度到 16-19 层的方法来提高模型性能。ResNet50 模型提出了一种新的更易于优化的差值连接网络，并且在更深的网络层结构（是 VGG 网络结构深度的 8 倍）中也能得到突出的准确度。InceptionV3 提出了一种分解卷积以及增强归一化的方法尽可能的提高网络计算效率，利用较少的参数数量同样得到了很好的成绩。InceptionResNetV2 模型将 ResNet 中的差值连接应用在 Inception 模型中，进一步用更少的参数来训练更深的网络。Xception 模型是在 Inception 模型的基础上设计出的更深的网络结构。

我们将在这几个模型的基础上做迁移学习来解决猫狗问题。基本思路是利用这几个模型的卷积网络部分得到原始图片的特征图序列，经过全局池化后将这些特征图非线性映射到最终类别，我们重点训练这个分类器。因为这几个模型已经能够在 ImageNet 上得到非常不错的成绩，可以认为这几个模型可以得到图片中真实有效的特征图，而根据特征图来做二元分类问题，难度将大大减小。

4. 基准测试

为了证明迁移学习的有效性以及测试不同方法最终的结果优劣，我们选择的基准是 Kaggle 提供的猫狗大赛公开排行榜前 10% 的成绩，也即是 LogLoss <= 0.06127。

三、问题解决方法

1. 数据预处理

- (1) 根据图片尺寸分布图可以看到，有一部分图片宽高都小于 100，这样的清晰度可以认为是一种干扰因素，所以将所有宽高都小于 100 的图片集合 tiny_pics 76 张剔除。
- (2) ImageNet 中已经包含有 126 种狗以及 7 种猫[5]，利用训练好的 ResNet50 以及 InceptionV3 直接对训练集中的猫狗做预测。我们选择了 top-25 的标准，也就是说，模型预测结果中前 25 个类别只要包含 126 种狗或者 7 种

猫就算预测正确，按照这个标准将所有标记为猫的图片筛选出预测不是猫的图片，以及将所有标记为狗的图片预测结果不是狗的图片，两种模型的结果求交集，得到 cat_outlier 101 张，dog_outlier 18 张，这些图片相对于整个训练集 25000 张来说比较小，可以剔除。



图 5. 猫异常图片

可以看到虽然有些图片被误判为不是猫，但真正的非猫图片已经成功筛选出来。



图 6. 狗异常图片

所有的狗异常图片只有 18 张，可以认为都不是狗。

由于 ImageNet 中包含的狗种类多，猫种类少，所以 top-25 的标准对猫异常图片误判率较高，而对狗异常图片漏判率较高。这里为了简便我们统一采用 top-25 的标准。

最终得到的训练集为 cat-12369 张、dog-12440 张。

(3) 对于不同模型，输入的图片还要进一步处理。

Xception 模型的输入尺寸 299x299

VGG16 模型的输入尺寸 224x224

ResNet50 模型的输入尺寸 224x224

InceptionV3 模型的输入尺寸 299x299

InceptionResNetV2 模型的输入尺寸 299x299

调整好尺寸后，还要进行减 ImageNet 均值、变换彩色通道等不同的前处理步骤，这些前处理方法已经由 Keras 框架给出，如无必要，不需要自己实现，为了预防错误，使用 keras 提供的方法即可。

(4) 构建图片生成器

由于硬件限制，一次不能训练太多的数据，需要分批加载，这里利用 ImageDataGenerator 来作为图片生成器，按照 batch size 来调整每次加载数据的多少，按照需要对要生成的数据进行混洗，猫狗的训练集数据按照 0.2 的比例随机取出作

为验证集。

2. 过程实施

主要完成了以下五种试验：

(1) 单预训练模型拼接分类器

将 Xception、VGG16、InceptionResNetV2 的卷积部分拼接全连接池化层，Dropout 层，加上激活函数再输出。模型的初始参数加载预训练模型后，将所有卷积层参数都锁定，训练的时候只训练最后拼接的分类器。每一种模型都训练 5 代，Dropout 取值 0.25，batch size 设置为 128，优化器使用 Adadelta 方法。

(2) 模型微调

重新生成一个与上一个步骤相同的 VGG16 模型结构，并且直接加载上一步该模型训练 5 代后的参数，将优化器替换成 sgd，学习率从 0.01 开始，自适应递减，将模型的最后一个卷积层修改为可训练状态，可训练参数由 513 个变为 2359808 个，Dropout 仍然取 0.25，训练 3 代。

(3) 模型融合

这里采用了三种融合方式。

a. 利用 ResNet50、Xception、InceptionV3 以及 InceptionResNetV2，四种模型，导出所有训练集包括验证集，以及测试集的特征向量，然后拼接这些特征向量，连接一个分类器，进行训练。此时依然需要划分验证集并混洗，不过现在混洗的是训练集生成的特征向量。batch size 取 128，Dropout 取 0.2，训练 8 代。预测时的输入也不再是测试集图片，而是测试集生成的特征向量。

b. 直接利用 Xception、InceptionV3、InceptionResNetV2、ResNet50 四种模型带全局池化层的结构，串联起四种输出，拼接一个分类器，模型共 13 层，可训练参数 7681 个，batch size 选择为 50，Dropout 取 0.2，训练 3 代。

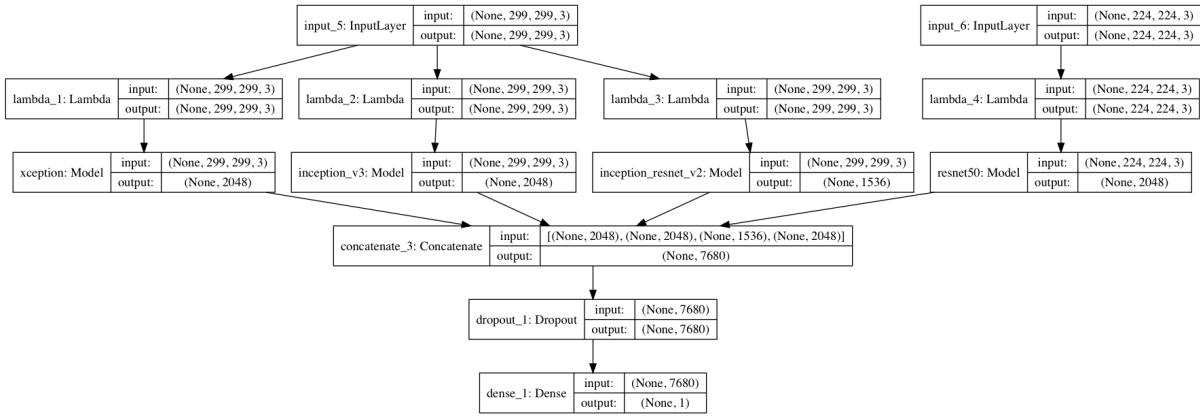


图 7. 融合结构 1

c. 与上一步相同的操作，只是融合的结构选择为 Xception、VGG16、InceptionResNetV2、ResNet50，模型同样为 13 层，可训练参数 6144 个，其他参数相同，训练 3 代。

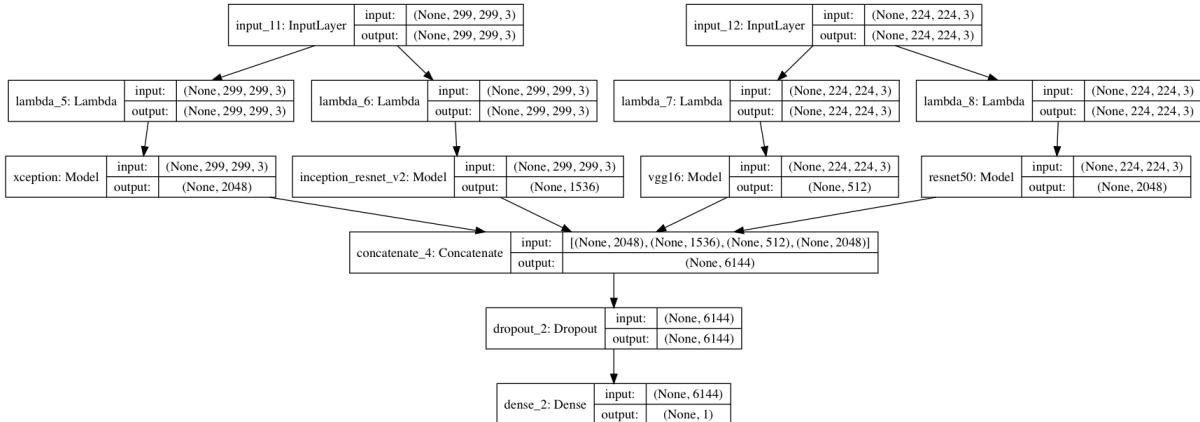


图 8. 融合结构 2

(4) 数据增强

仍然采用 VGG16 模型，重新生成后，直接加载训练 5 代后的参数，但是现在的图片生成器更换为数据增强型，具体是将图片宽高缩小 0.2 倍，并进行水平翻转，然后让 VGG16 模型继续学习，batch size 为 128，Dropout 为 0.25，训练 3 代。

(5) 类激活图

以 VGG16 模型为 base model，新建一个 cam 模型，cam 的输入与 VGG16 相同，输出比 VGG16 增加了一个 VGG16 模型最后一个卷积层的输出，也即是特征图序列，shape=(7, 7, 512)，将特征图序列与最后的全连接层系数 weights，shape=(512,)，相乘，最后就会得到一个堆叠在一起的特征图(7x7)，将其缩放到输入图片的尺寸就得到了该输入图片的类激活图，绝对值越大的地方，就是模型越关心的地方。

3. 改进措施

改进的地方有两点：

- (1) 筛选训练集中的异常值时，利用预训练模型直接进行预测，一开始选择的 top-10 的标准，筛选出的猫异常值中包含的正常图片过多，狗异常值图片中也包含有一些正常图片，这是由于 ImageNet 中的猫狗种类差异较大造成的结果不平衡，所以最终选择了 top-25 的标准，舍弃更多的猫，而保留更多的狗，总共舍弃掉的图片控制在 200 张以内。
- (2) 针对 Kaggle 提供的交叉熵计算方法，将最终的结果 clip 到 0.005~0.995 之间。

$$\text{LogLoss} = -\frac{1}{n} \sum_{i=1}^n [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

其中， n 表示测试集中所有图片的数量， \hat{y}_i 表示测试结果为狗的概率， y_i 的值在图片是狗时取 1，是猫时则取 0， $\log()$ 表示自然对数。

- 当图片是猫时， $y_i = 0$ ：
如果预测正确 $\hat{y}_i = 0$ ，则 $\text{LogLoss} = 0$
如果预测错误 $\hat{y}_i = 1$ ，则 $\text{LogLoss} = -\ln(0) = -\ln(10^{-15}) = 34.5$ ，因为 double 最小为 10^{-15} 。而 $-\ln(1 - 0.995) = 5.3$ 。
- 当图片是狗时， $y_i = 1$ ：
如果预测正确 $\hat{y}_i = 1$ ，则 $\text{LogLoss} = -\ln(1) = 0$ ，而 $-\ln(0.995) = 0.005$
如果预测错误 $\hat{y}_i = 0$ ，则 $\text{LogLoss} = -\ln(0) = -\ln(10^{-15}) = 34.5$ ，因为 double 最小为 10^{-15} 。 $(\text{LogLoss} \text{ 此时的理论值} + \infty)$

可以看出 clip 对于减小 LogLoss 的重要性，及必要性。

四、实验结果

1. 模型评估与验证

通过计算可以得到以下的结果：

Model	LogLoss	Rank
Xception	0. 08306	>250
VGG16	0. 06959	<166
InceptionResNetV2	0. 07800	>200
微调 VGG16	0. 06095	<130
ResNet50、Xception、 InceptionV3、 InceptionResNetV2 特征融合	0. 03712	<7
ResNet50、Xception、 InceptionV3、 InceptionResNetV2 结构融合	0. 04966	<59
ResNet50、Xception、VGG16、 InceptionResNetV2 结构融合	0. 04564	<34
图像增强 VGG16	0. 06450	<142

表 1.各模型最终 LogLoss

[gap_pred.csv](#)
2 days ago by Ding Xiaogang
gap e8b128d20

0.03712



图 9. 特征融合模型最终 LogLoss

可以看到特征融合的模型效果最好，其次是结构融合与微调模型，单模型与图像增强的效果都有限。

2. 结果分析

单模型的训练结果都无法下降到目的 LogLoss，经过微调后 VGG16 模型能够刚好达到预期目的，而图像增强并没有提高模型的最终成绩，整体来看融合模型的效果都要比单个模型的效果好，其中特征融合模型又相比结构融合模型效果更好，而且根据实际计算情况，可以发现，特征融合模型比结构融合模型的计算量也要小，训练更迅速。

单模型没有融合模型的效果好，可以说明不同的模型获得的图像特征有差别，融合模型更有利于综合不同的特征，最终得到更好的结果。图像增强对模型性能提升不大，可以说明该模型的能力有限，增加训练集也无法提高其性能。

五、 结论

1. 结果可视化

在训练过程中，训练集的 loss 都是先下降，后平缓，acc 不断提升，而验证集 loss 也是一直下降，acc 不断提高，然后由于过拟合可能会下降。

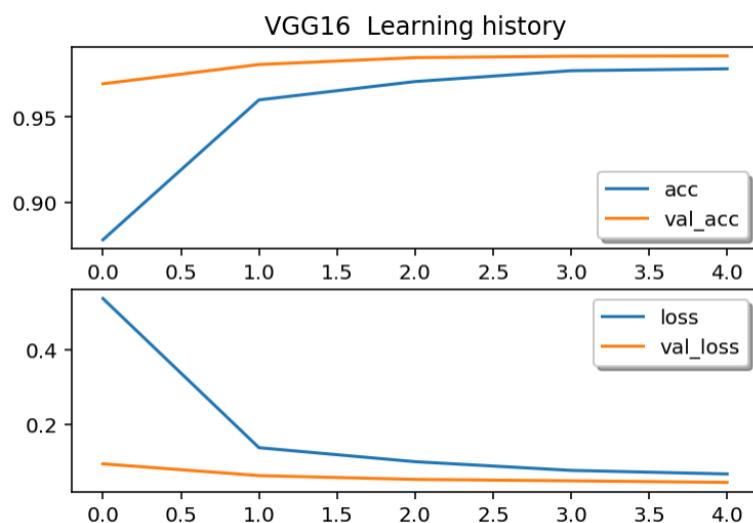


图 10. VGG16 训练图

通过类激活图来验证模型是否关注到了图片中的特征，越是特征明显的部位，应该是模型越关注的地方。

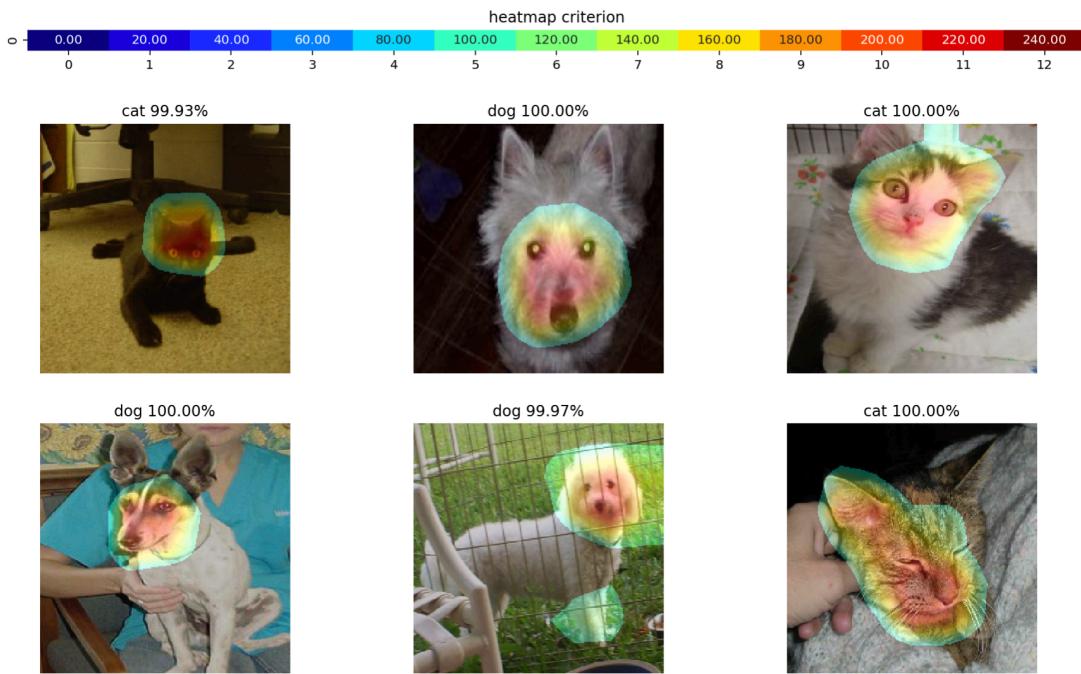


图 11. 类激活图

2. 思考

1). 通过观察不同模型的训练过程，发现训练完成 1 代以后，loss 就已经开始很难再继续下降了，或者下降很有限。这与模型的可训练参数数量以及问题复杂度有关，可训练参数越多，解空间越复杂，模型越难寻找到全局最优点，问题越复杂，需要有效的参数当然也会越多，但是参数多了却很容易过拟合。所以微调时，可训练的参数数量应该一点一点增加，不能一次增加太多，并且参数的初始化也非常影响模型的训练难度。

2). 多模型的融合，更能全面而准确的得到特征向量，更加有利于后续的分类器发挥更好的能力。单个模型得到的特征向量，准确度有限，后面的分类器效果再好，整个模型的性能仍然有限。

3. 改进

鉴于微调能够提高单模型的最终性能，所以有理由相信微调也能提升融合模型的性能，由于融合模型的微调会增加非常多的可训练参数，计算量将会比原先还要大很多，这里就不进行了。另外还可以通过增加训练集，寻找更多标注过的猫狗图片来进行学习，这将进一步提高融合模型的性能。

参考文献

- [1] Kaggle. (2017). *Dogs vs. Cats Redux: Kernels Edition*. Available: <https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition>
- [2] J. Elson, J. Douceur, J. Howell, and J. Saul, "Asirra: A CAPTCHA that Exploits Interest-Aligned Manual Image Categorization," 2007.
- [3] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," *ArXiv e-prints*, vol. 1512, Accessed on: December 1, 2015Available: <http://adsabs.harvard.edu/abs/2015arXiv151200567S>
- [4] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," *ArXiv e-prints*, vol. 1610, Accessed on: October 1, 2016Available: <http://adsabs.harvard.edu/abs/2016arXiv161002357C>
- [5] Keras. (2015). *ImageNet Class Index*. Available: https://s3.amazonaws.com/deep-learning-models/image-models/imagenet_class_index.json
- [6] F. Chollet and others, "Keras," ed: GitHub, 2015.
- [7] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *ArXiv e-prints*, vol. 1409, Accessed on: September 1, 2014Available: <http://adsabs.harvard.edu/abs/2014arXiv1409.1556S>
- [8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *ArXiv e-prints*, vol. 1512, Accessed on: December 1, 2015Available: <http://adsabs.harvard.edu/abs/2015arXiv151203385H>
- [9] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," *ArXiv e-prints*, vol. 1602, Accessed on: February 1, 2016Available: <http://adsabs.harvard.edu/abs/2016arXiv160207261S>