机器学习工程师纳米学位毕业项目

猫狗大战

董进贤

2018 年 4 月 3 日

1 问题定义

1.1 项目概述

本项目源于 kaggle 竞赛，解决的问题属于计算机视觉智能的研究领域。本项目要求训练出的模型，在给定的图片中识别出猫狗，并且达到指定的识别率。为方便标准的评估，所涉及的数据集亦是源自 kaggle 的原始数据集。这个数据集包含 25000 张已作出标记的用于训练的图片，其中猫狗各占比一半。另外，还有 12500 张的未标记图片，为测试数据集。

卷积神经网络(Convolutional Neural Network，CNN)，以其共享超参等方式，极大的提高了传统神经网路的训练效率，目前是图像识别领域公认的解决办法。所以我们的模型也以 CNN 为实现。这种共享超参数，作为卷积核与输入数据作用计算获得结果的方式，等效于矩阵中的卷积运算，故这个神经网络如此得名。

1.2 问题陈述

数据集中的图像均是来自于现实世界中未经过处理的图像，大小不一，图像中的场景也各不相同，另外具体到猫狗的姿态也有非常大的差异，其品种更是各不相同。相较于传统的机器学习算法，实在难以处理此类问题（比如，我们知道在历史上，第一个成功的卷积神经网络 LeNet-5 虽然成功完成了数字识别，但在综合表现上反而被后来的 SVM 所超越）。

虽然我们最终的 CNN 是经过优化处理，增加了深度的模型。但是归结起来，我们还是采用了四个步骤：一是输入阶段，预处理数据到方便训练的形态；二是卷积层训练，以提取数据在不同通道不同向量方向上的特征数据；三是池化筛选特征数据，用作下一层的卷积训练或者分类处理；最后则是全联接综合所有的维度特征作出最终的分类预测。

1.3 评价指标

评价指标采用 kaggle 官方的评价公式，其得分值越低，效果越好，公式如下：

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

其中：

1) n 为测试集中图片的数量；

2) $\hat{y}_i$为猜测图片是 dog 的概率；

3) y_i为 1 时图片为 dog，为 0 时图片为 cat；

4) log()为自然对数。

目前 CNN 计算较为完善，各个模型的预测率均较高。由公式计算出的对数损失评价预测效果，而非正确率是合理的。而且在模型在过分肯定预测结果时，在公式的计算下，也不一定得到较好的结果。这点能够发现过度拟合的模型，评价更为客观合理。

2 分析

2.1 数据的探索

我们采用的数据集来自 kaggle，下载后得到三个文件，test.zip、train.zip 和 sample_submission.csv。其中 test.zip 为测试数据集，大小为 271MB，train.zip 为训练数据集，大小为 544MB，sample_submission.csv 应该是提交样例。解压压缩包，我们发现数据集中均是 jpg 图片，命名格式为“类型.序号.log”的形式，如：cat.1.log，dog.1.log。

进一步探索数据，发现训练集包含 25000 张图片，猫狗各半 12500 张，测试集 12500 张图片（测试集图片只有序号，是没有标记的）。浏览图片，发现图片的大小也是不一致的，清晰度，分辨率也有差别。打开图片，发现图片中的场景各异，而且有错误标注的图片如图 1 所示。

cat . 7564. j pg　　cat . 10712. j pg

图1 错误标注

另外图片中的猫狗也是形态各异，也有其它主体的干扰，这些都增加了识别难度，如图 2 所示。示例中也可以看出，图片大小尺寸，也有不小的差异，其中 dog.9076.jpg 甚至还包含大量的留白。

cat . 10700. j pg

cat . 10778. j pg

dog. 5476. j pg

dog. 9045. j pg

dog. 9076. j pg

图2 形态背景各异

2.2 探索性可视化

在这里我们首选对数据集的大小特性作一定的分析，通过散点图绘制，能够比较直观的发现其中的规律。首先我们对 dog 和 cat 的图片，分别以宽为 x 坐标，高为 y 坐标绘制了散点图如图 3 所示。

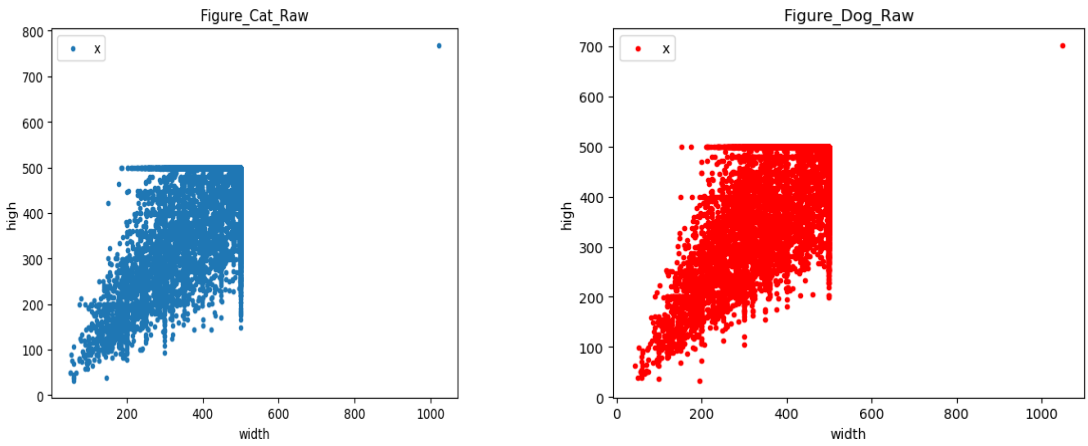

图3 数据集原始尺寸分布

我们发现，数据集中图片的尺寸，大部分都集中在 500 以下，但是 dog 和 cat 的数据集中，均存在异常的离散点。很显然这个数据属于异常值，并不利于模型的训练，需要去除。

经过排查处理，我们发现，异常的图片分别是 dog.2317.jpg 和 cat.835.jpg。我们将这两张异常的图片删除后，重新统计分析数据集后，绘制散点图如图 4 所示。

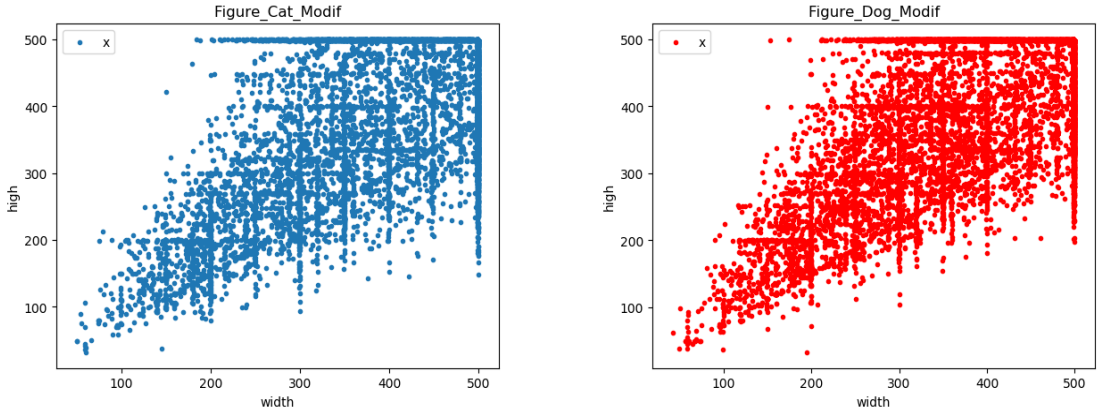

图4 清洗异常图片后尺寸分布

2.3 算法和技术

猫狗识别问题属于机器视觉的领域，具体到本项目涉及的算法和技术，需要说明的核心概念是，深度学习、卷积神经网络和迁移学习。其中迁移学习的模型，包括 InceptionV3、Xception 和 Inception ResnetV2，后续也会做一定的说明和阐述。

2.3.1 深度学习

深度学习的概念源于人工神经网络（Artificial Neural Network，ANN），深度学习通过组合低层特征形成更加抽象的高层表示（属性类别或特征），以发现数据的分布式特征[1]。深度学习是相较于传统机器学习的叫法。普通机器学习通过人工经验抽取样本特征，模型学习后获得单层特性[2]。深度学习通过对原始数据进行逐层特征提取变换，将样本空间的特征表示变换到新的特征空间，自动地学习得到层次的特征表示，从而更有利于分类或特征的可视化[3]。

说道深度学习，必须要理清深度神经网络（Deep Neural Network，DNN）的概念。深度学习所得到的模型结构，包含大量的神经元（感受器、回归函数），每个神经元与大量的其他神经元相互连接，神经元连接之间传递信号的强度（权重），会在学习训练的过程不断修正。这种深层的网络结构符合神经网络的特性[4]，因而命名深度神经网络（Deep Neural Network，DNN）。

2.3.2 卷积神经网络

卷积神经网络（Convolutional Neural Network，CNN）是人工神经网络（Artificial Neural Network，ANN）中的一种。CNN 的基本结构有输入层、卷积层、取样层、全连接层和输出层构成。卷积层和取样层一般会有若干个，采用卷积层和取样层交替设置。由于卷积层中输出特征面的每个神经元与其输入进行局部连接，并通过对应的连接权值与局部输入进行加权求和再加上偏执值，得到该神经元的输入值，该过程等同于矩阵的卷积过程，故该类神经网络由此命名[5]。

在 CNN 结构中，深度越深，特征面数目越多，则网络能够表示的特征空间越大、网络学习能力也越强，然而也会使网络的计算更复杂，极易出现过度拟合现象。因而，在实际应用中，需要适当选取网络深度、特征面数目、卷积核的大小及卷积时滑动步长，以在较短的时间和计算量下，训练出更好的模型。

目前 CNN 在图像分类和识别中应用广泛，几乎是深度学习中，默认的处理方法。本项目也采用 CNN 进行猫狗的分类识别处理。

2.3.3 迁移学习

CNN 的核心在于对于图像进行特征提取（浅层的特征如边缘、纹理等），随着神经网络的层数越高，抽线层次也越高。最后的全连接层再基于训练中得到的特征数据，进行分类鉴别，得到最终的分类结果。

从最原始的图像开始训练神经网络，得到超参和特征数据，需要大量的数据，及大量的计算资源。好在猫狗分类问题，已经拥有可以利用的现有模型和预训练权重。通过此，我们可以直接提取到特征数据，在特征数据上附加以分类器，就可以得到非常优良的结果。

在本项目中，我门选用的预训练模型有三个 InceptionV3、Xception 和 Inception ResnetV2。

2.3.3.1 InceptionV3

InceptionV3 模型共 315 层，见附件 notebook/model_inceptionv3.png。图中对于模型结构，作了清晰的可视化展示。

该模型要求输入图片的分辨率为 299*299，模型的改进主要是增强了卷积模块功能。网络深度也有一定程度的增加。

2.3.3.2 Xception

Xception 模型共 136 层，该文档中不宜展示，可见附件 notebook/model_xception.png。图中对于模型结构，作了清晰的可视化展示。

Xception 是 google 继 Inception 后提出的对 Inception v3 的另一种改进，主要是采用 depthwise separable convolution 来替换原来 Inception v3 中的卷积操作。这个过程实际上是加宽了网络。这个模型也输出 2048 维的特征向量。

2.3.3.3 Inception ResnetV2

Inception ResnetV2 模型共 784 层，该文档中不宜展示，可见附件 notebook/model_inception_resnet_v2.png。图中对于模型结构，作了清晰的可视化展示。

Inception ResnetV2 模型的诞生源于 ResNet 和 GoogleLeNet，是早期的 InceptionV3 演化而来，其具有相当高的复杂度。该模型中存在 shortcuts，使得我们能训练出更深的神经网络，从而提高识别率。

2.4 基准模型

我们需要获得 kaggle 前百分之十的排名（即 131 名以前），也就是对数损失得分需要小于 0.06。由于迁移学习的效果，我们应该会得到远远优于此的得分。

3 方法

3.1 数据预处理

根据我们选择的解决方案，需要对图片数据进行一定程度的预处理。我将这个过程分成两个部分：一是可持久化阶段的数据处理；二是内存数据处理阶段。

首先，原始数据为两个压缩文件，test.zip 和 train.zip，分别是测试集和训练集数据。解压后图片分别存在于文件夹 test 和 train，然后创建 train_bad 文件夹，稍后用于存放删除的无效图片文件（对于小于 100*100 的图片，对于训练没有益处，我们做清洗处理）。具体的 shell 命令，再附件 README.md 文件中有详细的说明记录。

其次，由于我们选择的预训练模型，对于输入图片的要求都是 299*299，所以在读取图片到内存的过程中，我门调整图片的尺寸大小为 299*299。这个过程不做持久化处理，保存在磁盘上的图片文件，依然是原始大小。

另外需要说明的是，我们采取的标签是 0 和 1，分部代表猫和狗。这里需要识别的类型并不繁多。我们简单的设置标签即可，没有另行定义归一化函数。

整个处理过程，分别由函数 load_train_data 和 load_test_data 完成，至此完成数据的预处理过程。

3.2 执行过程

对于完成了预处理的数据，我们通过 sklearn 的 train_test_split 方法，将数据随机拆分成训练集和测试集（验证集）。为了做最后的陈述分析以及模型的展示分析，我们首先分别使用三个预训练模型训练处理，然后预测结果，分别保存至对应的 csv 文件中。

3.2.1 导出特征向量

对于 InceptionV3、Xception 和 Inception ResetnetV2 三个模型的处理方式，基本一致。估计我们定义函数 pick_features 完成此项任务。我们使用 model.predict 配合原始数据，生成特征向量。GlobalAveragePooling2D 将卷积层输出的每个激活图直接求平均值。提取出的特征向量，实际上就是 numpy 的三个数组数据，分别保存到 feature_Xception.h5、feature_InceptionV3.h5 和 feature_InceptionResNetV2.h5 三个文件中。

每个模型的特征向量导出，都相当耗时，附件的 notebook 中，我们对每一个导出过程放在单独的 checkpoint 里面进行。另外，最终生成的三个文件过于庞大，我并没有提交到 git 工程，如果感兴趣可以运行我的代码，重新生成。

3.2.2 构建模型

接下来就是构建我们的模型结构，由于采用了预训练模型提取的特征向量。我们实际的工作，其实就是最后的分类处理。构建最后的全连接层即可完成这个目标。实际使用的模型可视化后如图 5 所示。

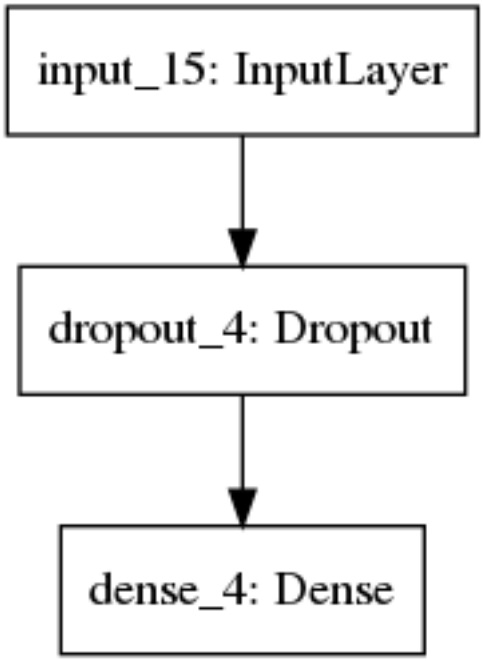

图5 自定义模型

然而这个可视化的模型结构，并不能展示我们整个项目的解决方案要义，我作了额外的工作。将整个解决方案的连接综合展示，得到如图 6 所示的模型结构图。这张图清晰的展示了，我们这个项目的结构逻辑。即，将原始图片数据预处理后，分别输入三个预训练模型，通过这三个千锤百炼的模型，提取到特征向量后。将便于分类处理的特征数据，输入到最后的全连接层作分类训练。

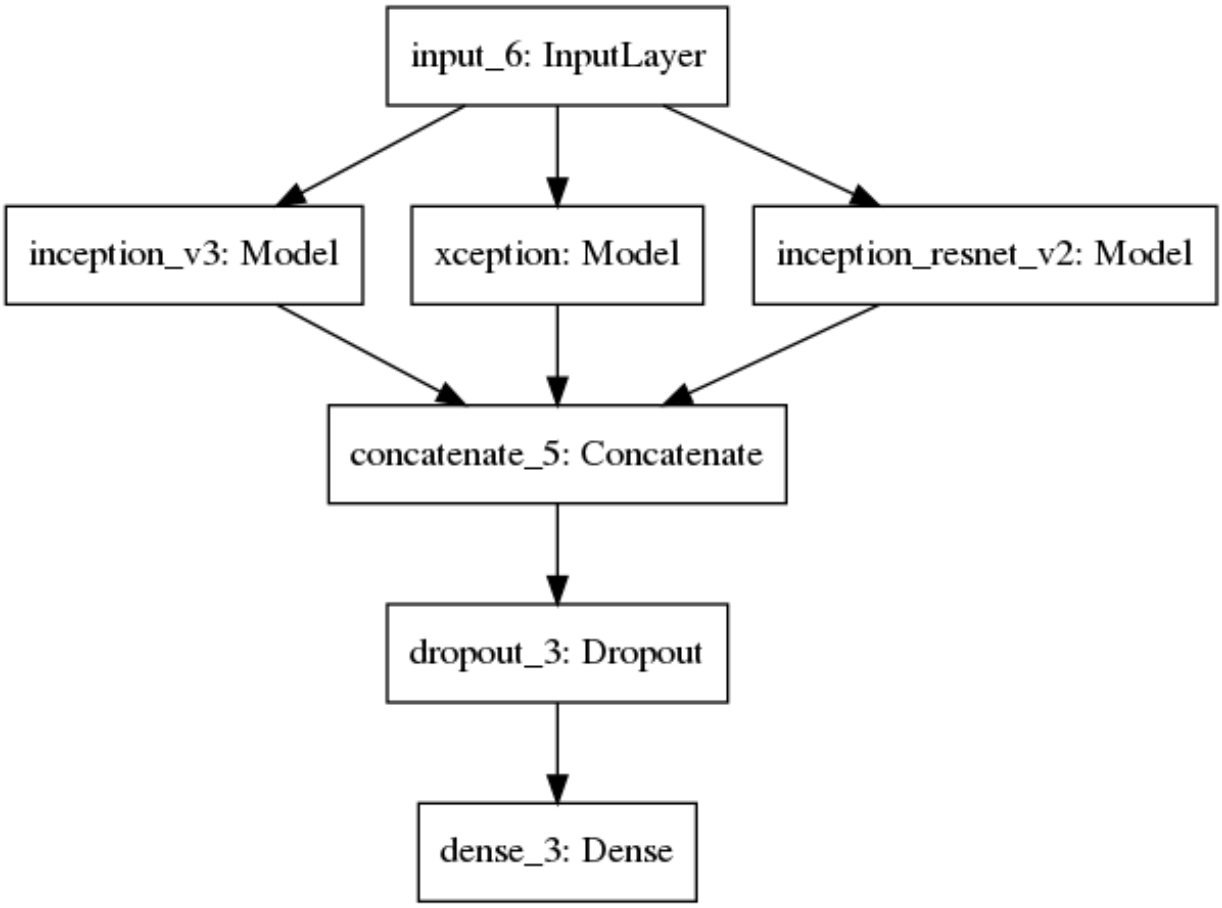

图6 解决方案模型概览

3.2.3 训练

模型构建完成后，我们以特征数据为输入，训练模型。由于模型层次较浅，发现训练速度非常快。并且，由于特征数据质量非常高，我们发现对数损失下降非常快，10 个周期后，可以达到 0.0081。准确率及对数损失变化如图 7 所示。

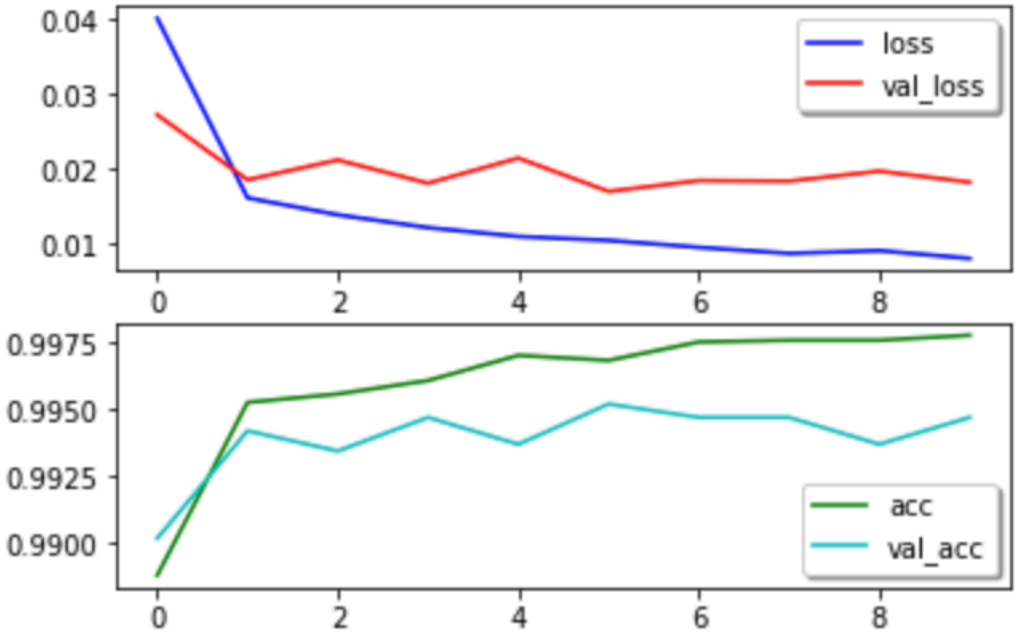

图7 训练曲线

3.2.4 预测结果

根据训练好的模型，我们对测试数据进行预测处理。结果保存到final_submission.csv。提交 kaggle 后，得到成绩结果如表 1 所示。这个需要注意的是首创预测非常失败，并不成功。具体的改进，我在模型微调中说明。

表1. 预测结果

模型	首次预测	调整后
InceptionV3	2.35763	缺省
Xception	2.26968	缺省
Inception ResnetV2	2.40522	缺省
Final_model	2.61388	缺省

3.2.5 改进处理

首次完成这个项目的结果，我发现结果非常的不理想。远远低于我的预期。得到的结果如表 1 所示。

我总结原因并尝试改进如下：

1) 首先数据的预处理阶段，虽然对于不合尺寸的图片进行了删除处理，但是对于错误标注的图片，没有引起足够的重视，予以删除。

2) 对于训练好的模型，还可以进行一定的微调。比如通过 fine-tune 来解冻靠近输出的层，改善结果。

3) 另外我吧 Dropout 的参数设置为 0.5，防止过度拟合。

4 结果

4.1 模型的评价与验证

经过前面的测试和实践，我们发现预训练模型确实能够降低我们的设计难度和训练量，但是所得到的结果排名也并非特别优秀。综合几种模型的特征数据，并进行微调处理，才能得到我们可靠结果。

经过微调处理后，我们模型得分达到了预期的水平。证明我们的推测和处理是可靠的。另外，如果想得到更好的结果，我们还可以寻找更优秀的预训练模型，以及进一步微调模型。

4.2 合理性分析

经过反复的调整参数，我的得分始终徘徊在 2 点多左右，未达到预期。

5 项目结论

5.1 对项目的思考

猫狗大战是非常成熟的项目，选用预训练模型也是常规的做法。每一个步骤我都反复确认检查过很多次。但确实没有得到预期的得分，但是整个解决方案，应该是正确有效的。

5.2 需要作出的改进

本项目采用模型融合到方式，借用了预训练模型的特征向量，再进行分类训练。如果需要进一步改进，可以采用更优秀的模型，导出的特征数据。另外对模型进行对预训练模型进行微调（fine-tune），也是一种方式。

** 在提交之前， 问一下自己... **

- 你所写的项目报告结构对比于这个模板而言足够清晰了没有？
- 每一个部分（尤其**分析**和**方法**）是否清晰，简洁，明了？有没有存在歧义的术语和用语需要进一步说明的？
- 你的目标读者是不是能够明白你的分析，方法和结果？
- 报告里面是否有语法错误或拼写错误？
- 报告里提到的一些外部资料及来源是不是都正确引述或引用了？
- 代码可读性是否良好？必要的注释是否加上了？
- 代码是否可以顺利运行并重现跟报告相似的结果？

参考文献：

[1] BENGIO Y,DELALLEAU O. On the expressive power of deep architectures[C] // Proc of the 14th International Conference on Discovery Science. Berlin:Springer-Verlag,2011:18-36.

[2] DALAL N, TRIGGS B. Histograms of oriented gradients for human detection[C]// Computer Vision and Pattern Recognition，2005. IEEE Computer Society Conference on. Piscataway, NJ:IEEE Computer Society Conference on. Piscataway, NJ:IEEE,2005:886-893.

[3] HINTON G E, OSINDERO S, TEH Y W. A fast learning algorithm for deep belief ntes[J]. Neural Computation, 2006 ,18(7):1527-1554.

[4] PSALTIS D, SIDERIS A, YAMAMURA A. A multilayered neural network controller[J]. IEEE Control Systems Magazine, 1988, 8(2): 17-21.

[5] Yann LeCun, Leon Bottou, Yoshua Bengio, et al. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 1998, 86(11):2278-2324.