吉林大学学士学位论文（设计）承诺书

本人郑重承诺：所呈交的学士学位毕业论文（设计），是本人在指导教师的指导下，独立进行实验、设计、调研等工作基础上取得的成果。除文中已经注明引用的内容外，本论文（设计）不包含任何其他个人或集体已经发表或撰写的作品成果。对本人实验或设计中做出重要贡献的个人或集体，均已在文中以明确的方式注明。本人完全意识到本承诺书的法律结果由本人承担。

学士学位论文（设计）作者签名：

2017 年 5 月 20 日

摘要

基于 Qt 的卷积神经网络辅助设计系统

以卷积神经网络为代表的神经网络方法近年来在图像识别等领域取得了传统方法难以企及的优秀成绩。神经网络的构成，本身具有模块化的特点，因此已经有数款可以用来搭建神经网络的深度学习框架问世。然而这些框架均采用基于命令行的工具集或者动态、静态库的形式发布，操作直观性差，学习周期较长，另外，由于对神经网络的理论解释尚不完善，尚无一个能够直接指导新网络结构该如何建立的理论，神经网络的设计和测试过程中需要进行频繁的参数调整等人工操作，进一步加重了开发人员的负担。为了解决上述问题，本文提出并设计实现了一套采用 C++语言，以 Qt 作为图形用户接口的集成化、图形化的卷积神经网络辅助设计系统。该系统提供了常用数据格式的转换和读取、神经网络的设计、测试、特征导出等操作，并支持核心组件的替换和更新，能够为相关开发人员减少时间成本。经过测试，系统达到了预期的设计要求。

关键字：卷积神经网络，深度学习框架，Qt，辅助设计，图形用户界面

Abstract

Qt Based Design System for Convolutional Neural Network

The neural network method, which is represented by convolution neural network, has achieved excellent results in the field of image recognition in recent years. The composition of the neural network has the characteristics of modularity, so there are several depth learning framework come out， which can be used to build the neural network. However, these frameworks are based on the command line tools and the form of release is dynamic or static library with poor operational intuition and a longer learning cycle. In addition, due to the theoretical explanation of the neural network is not perfect, no one can directly guide the new network structure how to build the theory. The neural network design and testing process requires frequent parameter adjustment and other manual operations, which further increasing the burden on the developers. In order to solve the above problems, this paper presents and designs a set of integrated graphical convolution neural network design system with C + + and Qt user interface. The system that can reduce the time for the relevant developers cost provides common data format conversion and reading, neural network design, testing, feature export and other operations, and support the replacement and update of the core components. After testing, the system meets the intended design requirements.

Keywords：Convolution neural network，Depth learning framework，Qt，Auxiliary design，Graphical user interface

目　录

第 1 章　绪论

1.1　研究背景和研究意义

本文描述了一种基于 Qt 的卷积神经网络辅助设计系统。在系统的开发过程中，涉及到了机器学习、神经网络、可视化编程以及神经网络的设计与调试等问题，因此，深入了解相关的背景以及研究现状对于毕业设计的完成具有实际意义。

1.1.1　研究背景

1. 机器学习

机器学习是一项致力于研究如何通过计算的手段，利用数据来增强系统自身性能的学科，机器学习研究的，是关于在计算机上从数据中产生“模型”的算法，即“学习算法”(Learning Algorithm)。有了学习算法，我们把经验数据提供给它，它就能基于这些数据产生模型，在面对新情况的时候，模型会给我们提供相应的判断。机器学习，研究的是关于“学习算法”的学问。

经过多年的发展，已经诞生了多种机器学习方法，例如支持向量机（Support Vector Machine，SVM)、提升方法（Boosting)、最大熵方法等。而与这些方法不同的是，以“神经网络”为代表的深度学习方法所获得的模型中，非线性操作的层级数[1]更多，凭借精度、处理速度等多方面的优势，近年来神经网络方法在语音识别、图像处理等多类应用中取得了突破性进展[2-9]。

2. 卷积神经网络

（1） 卷积神经网络的简介

神经网络方法处理问题的思路主要来源于对生物神经系统结构的研究。以图像数据为例，有研究表明，动物的的神经系统在处理视觉信号时，首先对初级特征和形状进行识别，然后组合成成更复杂的图像语义[10]。同样地，深度学习通过将原始图像分解为低层次的特征，把数据用分层的特征表示出来，进而通过特征对原始输入数据进行识别和分类。

前馈神经网络是最初的人工神经网络模型之一。这种系统的网络结构是一个有向无环图，在网络中没有封闭环路，信息从输入层通过一个或多个层到达输出层，

完成原始信号到高级语义的映射。典型的前馈神经网络有多层感知机[11]和卷积神经网络[12]等。

卷积神经网络通常可以看做是由很多结构类似的“网络模块”首尾相连形成的，每个“网络模块”由卷积阶段、激活阶段和下采样阶段等组成：

①卷积阶段。需要注意的是，这里的“卷积”与传统图像处理意义上的卷积稍有不同。在卷积神经网络中，“卷积”指的是一种“加权求和”操作，而这里的“权值”就是所谓的卷积核。在网络中，每次卷积操作就是通过将一定数量的卷积核作用于输入向量，在整个输入向量中寻找特定的一种特征。由于卷积神经网络对于图像的处理是像素级别的，为了防止出现网络参数过多以至于无法进行有效的训练的情况发生，在网络的设计上采用了权值共享规则[13]。

②激活阶段。生物神经元有“激活”和“抑制”两种状态，处于抑制状态的神经元节点不会对前驱节点传导来的信号做出反应，这种现象类似于阈值化处理，反映在卷积神经网络中，表现为对卷积阶段得到的特征按照一定原则进行筛选，这种筛选也因此被称为“激活函数”。为了减少线性模型的表达能力较差所带来的不良影响，激活函数通常采用非线性函数。常用的激活函数有sigmoid、tanh或softsign等饱和非线性（Saturating Nonlinearities）函数[14]。近年来，有越来越多的研究结果表明，不饱和非线性(Non-Saturating Nonlinearity)函数ReLU(Rectified Linear Units)[15-16]比传统的饱和非线性函数有更快的收敛速度。因此在训练整个网络时，采用ReLU激活函数的网络，其训练速度也比传统的方法快很多[2]。

③下采样阶段。下采样的目的是在尽可能的不破坏图片所描述的特征的情况下使输出特征图的分辨率降低，这对于输入图片的分辨率很高的网络意义十分重要，分辨率的降低意味着更少的参数，更快的训练速度。平均池化（Average Pooling）或者最大池化（Max Pooling）的操作是常用的下采样操作。另外，一些网络通过特殊设计的卷积（例如使用1×1的卷积核）来完成下采样操作。

在若干个“网络模块”之后，通常会连接若干个全连接层和一个输出层，通过分类器得出最终的分类结果，一个完整的卷积神经网络的示意图如图1-1所示。

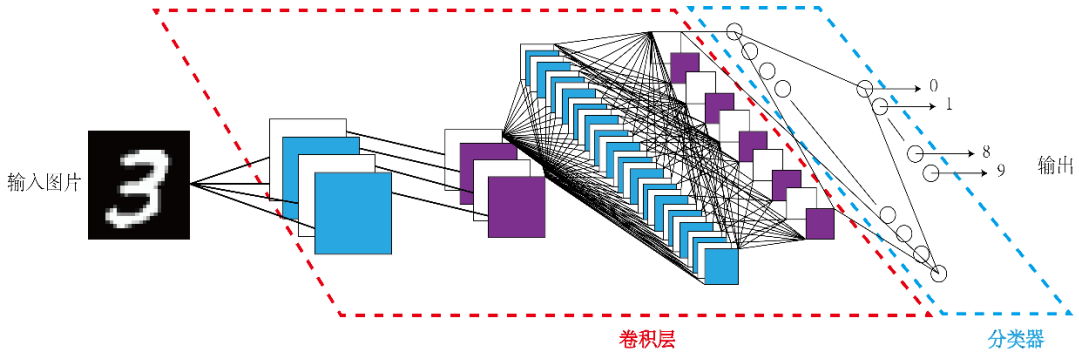


图 1-1 卷积神经网络

在实际的研究与开发过程中，通常使用反向传播算法以及有监督的训练方式对神经网络进行训练，算法流程如图 1-2 所示。网络中信号从输入特征向输出特征的方向传播。输入的信号，经过多个卷积层，得出最后一层输出的特征向量；将输出特征向量与期望的标签进行比较，生成误差项；沿着网络的反向路径，将误差逐层的传递到每个节点；根据权值更新公式，更新相应的权值。在训练开始阶段，网络中权值的初值可以通过随机分布函数进行初始化，或者通过加载预训练的网络来获得[17]。对于一个可用的卷积神经网络，网络的误差将会随迭代的进行而逐渐降低，并逐渐逼近极限值，此时称网络已经收敛。对于一个已经收敛的网络追加额外的训练迭代通常无法得到更好的效果。

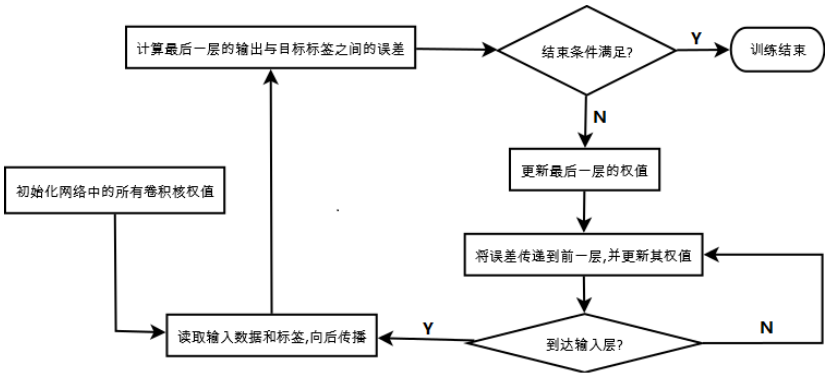


图 1-2 反向传播算法

近几年来，以卷积神经网络为代表的深度学习方法在图像处理领域取得了诸多传统方法望尘莫及的优秀成果，这也引发了学界和产业界的一致关注。不同于传统的图像处理方法，无论是处理何种任务的卷积神经网络，其结构都有相似之处。不同的研究人员提出的方法，都可以总结为把各种“网络模块”进行一些参数的变更并进行组合，这种做法类似于软件工程中的“模块化”、“组件化”编程思想，这也为通用的深度学习工具的设计与实现提供了可能性。

目前，已经出现了集中深度学习框架，其中比较常用的有 Caffe、TensorFlow 等，关于这些框架，后文会进行介绍和对比，此处不做详细说明。

（2） 卷积神经网络的设计和测试

卷积神经网络的研究过程中，一个经常被提起的问题是对这种方法做出“合理化的解释”。传统的图像处理方法中，对特征的处理和识别往往依赖于先验知识，而这些先验知识是人类根据自己的理解给出的，这就带来了一个问题，如果人类的理解是片面的，甚至是错误的，那么势必会影响以此为基础设计出来的算法的效果。显而易见的是，这样的算法只能看做是对人类自身知识的一个程序描述，它没有任何的“思考能力”，也称不上“智能”。

卷积神经网络在图像处理中获得了突破后，人们提出了一个问题：这种方法究竟识别到了图片中的什么特征？然而让人意想不到的是，通过对训练好的网络进行特征导出，人们发现，某些神经网络方法“发现”的图像特征是人类无法理解的，如图 1-3 所示。这就带来一个严重的问题：假如我们不知道神经网络为什么起作用，那又如何针对某一个全新的任务，从零开始设计一个有效的网络结构呢？

图 1-3 神经网络提取到的特征

目前，针对卷积神经网络的理论解释方面，已经有了很多工作，但是这些工作都是针对某一个特定的任务情景的，还达不到指导网络结构设计的程度。在完整的、普适的理论解释出现之前，卷积神经网络的设计过程主要包括以下步骤：

① 准备数据集。为了使用神经网络进行图片识别等工作，需要准备大量的有标注数据，一般来讲，数据量越大，设计成功的可能性就越高，例如常用的公开数据集中，较小的 Mnist 含有 60,000 个训练样本和 10,000 个测试样本，而目前最大的图片数据集 ImageNet 总共含有 14,197,122 个样本。

② 进行网络结构的设计。由于缺乏理论依据，网络结构的设计主要依靠的是经验。设计好的网络结构通常采用脚本语言或者编译式高级语言进行描述和保存。

③ 测试和调整。利用准备好的数据集和设计好的若干个网络结构，进行训练

和测试，通过程序的表现对网络的参数进行调整，由于神经网络所使用的数据集很庞大，将一个网络训练到收敛需要耗费很多时间，在这种情况下，进行参数微调和网络结构的优化，往往需要摸索很长时间。

更大的数据集、更快的设备和更高效的工具链是卷积神经网络的进一步发展所必备的条件。随着越来越多的公开数据库的建立和增强学习、无督导学习方法的发展，深度学习对大型标注数据库的依赖有望逐步降低；硬件的发展速度更是日新月异，截止 2017 年 5 月 20 日前，NVIDIA 公司刚刚发布了新一代帕斯卡架构的旗舰级显卡 Titan Xp 和 GeForce GTX 1080 Ti，Google、Intel 等业内巨头更是研发出了为深度学习设计的专用处理器芯片，配合使用 GPU 通用计算工具 CUDA 和深度学习工具 CUDNN，这些设备能够为研究人员提供前所未有的处理速度，并且随着硬件价格的逐步降低，组建多处理器的深度学习工作站甚至 GPU 集群的成本也进入到了可接受范围内；工具方面，虽然已有多款深度学习框架问世，但这些框架均是由一系列命令行工具甚至文本组成，掌握他们的使用方法需要一定的学习成本，且集成环境和工具链的缺位，使得这些工具的环境配置较为繁琐，一旦在训练和测试中出现问题，将会给本就漫长的设计流程添加不必要的时间成本。

1.1.2 研究意义

深度学习，尤其是神经网络在图像处理，自然语言处理方面取得了传统方法难以企及的良好效果，具有较高的研究价值。随着硬件价格的逐步降低和性能额逐渐提升，训练和测试更大的网络成为了可能，ImageNet 等公开训练集的建成，以及无督导学习，增强学习，迁移学习等不依赖大规模库的方法的出现，使得数据来源对神经网络方法发展的制约逐渐减小，然而，当前的深度学习工具存在着集成化程度低，学习成本高，环境配置繁琐等问题，这些问题的存在引入了大量时间成本，因此，一个集成化的、图形化的，能够为开发人员节省时间和精力的研究开发环境，对深度学习的发展有一定的意义。

1.2 国内外研究现状分析

1.2.1 深度学习框架的研究现状

目前，常用的深度学习框架有 Caffe, TensorFlow, MXNet, Torch, Theano 等，下面将会对这几种框架进行简介和对比。

Caffe[26]是第一个主流的工业级深度学习工具。它的建设开始于 2013 年底，由 UC Berkely 的 Yangqing Jia 编写和维护。程序架构强大且合理。在计算机视觉领域，Caffe 是最流行的工具包。许多新的网络设计方案都拥有 Caffe 的实现，例如 ResNet、SSD 等。它拥有大量的扩展，但是在长期迭代中遗留的架构问题积重难返，这些状况使得Caffe的灵活性不足且不能完美地支持RNN(递归神经网络)的建模。

TensorFlow[27]是 Google 开源的深度学习工具。在 Google 搜索、图像识别以及 Gmail 邮箱中都使用了与该框架相关的研究成果。TensorFlow 能以理想的方式实现 RNN（递归神经网络），它使用“符号图”这一概念来进行向量运算，封装使得新网络的指定很容易。TensorFlow 支持快速开发,但是其运行速度较慢，内存占用较大。

MXNet[28]是由分布式机器学习社区开发和维护的开源的机器学习框架，是分布式机器学习通用工具包 DMLC 的重要组成部分。MXNet 的灵活性和效率较高，社区化开发的形式使得它具备详细的文档，较高的内存使用效率使得 MXNet 甚至能在智能手机上运行图像识别等任务。

Torch[29]是由 Facebook 开发并开源的的深度学习框架，使用 C 和 Lua 语言开发。它有较好的灵活性和速度，实现并且优化了基本的计算单元，使用者可以很简单地在此基础上实现自己的算法，计算优化问题由框架处理，核心计算单元使用 CUDA 优化，不需要用户考虑。网络的结构描述采用 Lua。缺点是作为接口的 lua 语言需要额外的学习时间。

Theano[30]是蒙特利尔理工学院于 2008 年开发的，主要开发语言是 Python。以 Theano 为基础，一大批深度学习 Python 软件包接连诞生，著名的 Blocks 和 Keras 便是其中的代表。Theano 适合于学术研究性质的实验，且对递归网络和语言建模有较好的支持，缺点是速度较慢。

表 1-1 对这些常用的机器学习框架进行了总结。

表 1-1　常用机器学习框架对比

框架名称	开发语言	运行速度	灵活性	文档支持	适合模型	平台支持	上手难易
Caffe	C++/CUDA	快	一般	全面	CNN	全平台	一般
TensorFlow	C++/CUDA/Python	一般	好	一般	CNN/RNN	Linux/OS	难
MXNet	C++/CUDA	快	好	全面	CNN	全平台	一般
Torch	C/Lua/CUDA	快	好	全面	CNN/RNN	Linux/OS	一般
Theano	Python/C++/CUDA	一般	好	一般	CNN/RNN	Linux/OS	易

在综合考虑了上手难易度、适用平台、编程语言和编译环境之后，本文拟采用 Caffe 作为核心框架。

1.2.2　深度学习相关的可视化工具

由于深度学习，特别是神经网络的相关研究中一直存在着抽象化程度高、可理解性差等问题，针对神经网络的理解、可视化、理论解释等研究一直是领域内的研究热点，下面列举了一些这方面的相关工作。

在描述神经网络方面，纽约大学的 Matthew 等人提出了所谓的“反卷积网络”[18]，采用与卷积神经网络并联的反卷积网络将深层次特征向输入层重映射，给出了一种尝试理解卷积神经网络深层次特征的思路和方案[19]。Caffe 的作者贾杨清也编写了网络结构示意图绘制工具[20]，该工具使用 Python 编写，绘制效果如图 1-4。

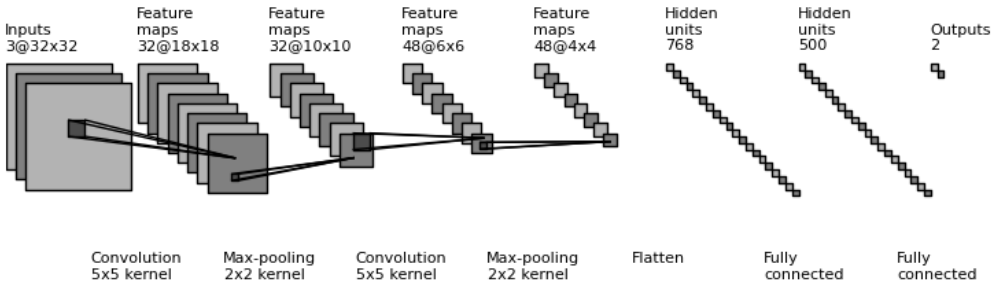


图 1-4　网络结构示意图绘制工具的绘制效果

Netscope 是一个在线工具[21]，这个工具实现了在线的 prototxt 辅助编写，而且可以以其特有的流程图的形式绘制出网络的结构，效果如图 1-5。

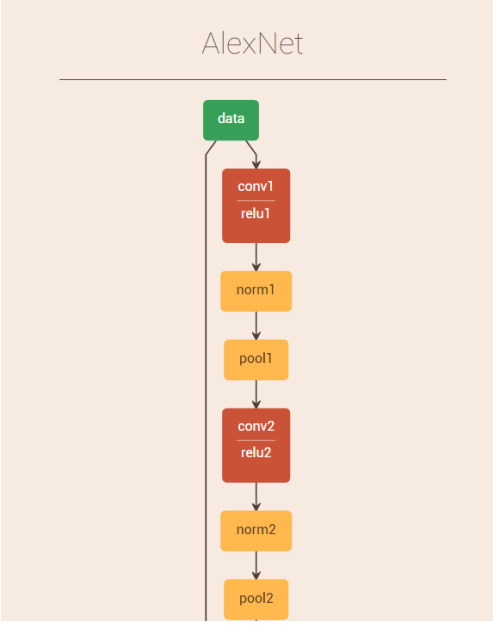


图 1-5 Netscope 绘制的卷积神经网络（AlexNet）示意图

Keras-vis 可以实现特征导出的效果[22]，不过它仅仅适用于 Keras 网络（图 1-6）；类似的工作还有基于 GoogleNet 使用 Mxnet 实现的特征融合工具[23]（图 1-7）。

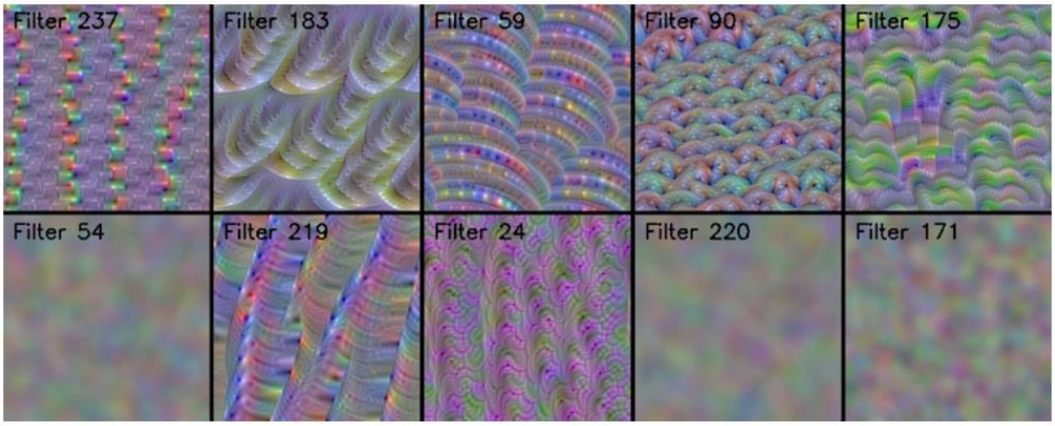


图 1-6 用于 Keras 的特征导出工具的效果

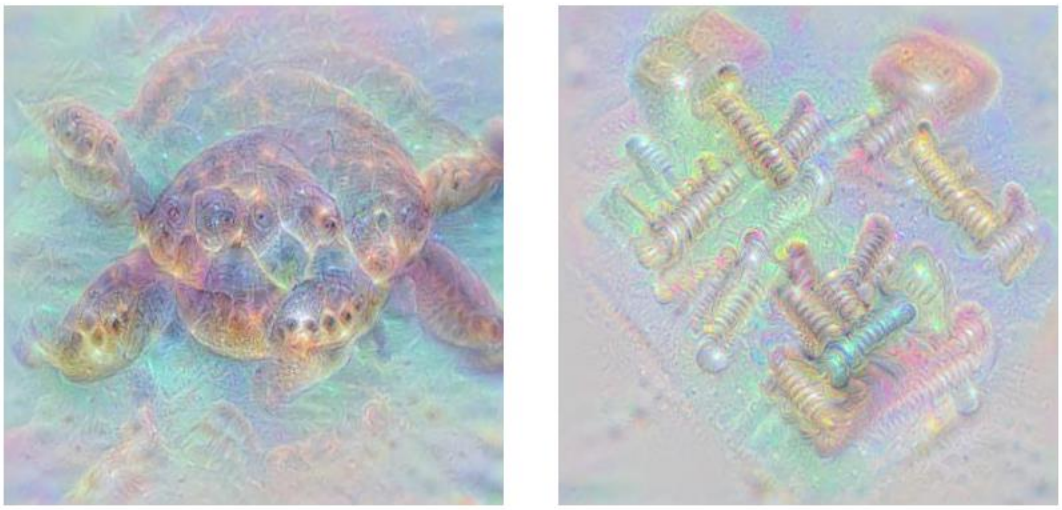


图 1-7 基于 GoogleNet 的特征融合工具的效果

还有一些工作致力于卷积神经网络的工作原理或工作过程的可视化，以期帮助入门者快速理解神经网络的一些内部特点。Convnetvis 是一个在线工具[24]，它使用 3D 视图描述了一个基于 Mnist 的手写数据识别网络的工作过程（图 1-8）。CNNVis 是一个较为先进的在线工具[25]，它内置了一些数据集和网络模型的选项，以较为美观的形式展现出神经网络的内部连接形式（图 1-9）。

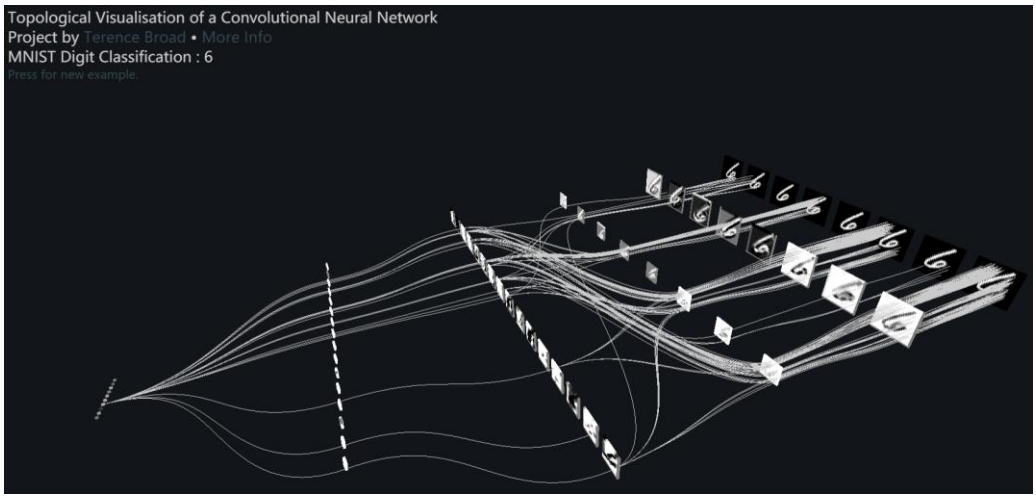


图 1-8 基于 Mnist 的手写数据识别网络的工作过程可视化工具

图 1-9 CNNVis 神经网络可视化工具的效果

在人机交互方面，目前绝大多数主流的深度学习工具均使用控制台工具，NVIDIA 的 DIGITS 框架自带了一个基于 Web 的用户界面，通过界面可以进行简单的参数设置和训练监控，但是遗憾的是该框架的使用并不广泛且只支持 Linux 系统，相关技术资料和社区不完善导致其使用难度较大。

无论是对卷积神经网络的理解、描述，还是人机交互，目前都有较多的工作在做各种尝试，对神经网络的理论解释在神经网络的相关研究中具有重大的意义，但是由于深度学习方法与传统方法解决问题的思路完全不同，该研究困难重重，一个

普适性的、广为接受的理论解释至今没有诞生。另一方面，在深度学习工具的建设方面，很多开源项目和社区的活跃侧面证明了其需求，但同时我们注意到，这些工具的通用性很差，它们大多数依赖于某种特定的网络结构和数据集，有些甚至只是显示后台事先渲染好的数据，这样的工具远远不能构成工具链，对于神经网络开发工作的辅助作用也较为有限。

1.2.3 图像化编程框架

用户界面（User Interface）是指对软件的人机交互、操作逻辑、界面美观的整体设计。好的UI设计要让软件的操作变得简洁、明确、方便，充分体现软件的主要功能，降低使用软件所需的时间成本。目前，有很多支持用户界面设计与实现的程序设计框架，常用的有MFC、Java Swing、C# WinForm、WPF和Qt等，下面将会对这几种常用的图形用户界面应用程序开发框架进行介绍和对比。

MFC是Microsoft Foundation Classes（微软基础类库）的简称。微软公司实现了该C++类库，以C++类的形式封装了Windows API，包含大量Windows句柄封装类和很多Windows的内建控件和组件的封装类。使用MFC包含的应用程序框架可以减少工作量。使用MFC能够在一定程度上避免直接使用Win32 API带来的编程工作量大，上手困难，程序安全性差等问题，Microsoft Visual Studio集成开发环境为MFC定制了图形化编程工具，通过使用窗口设计器，编程人员可以进行“所见即所得”的快速开发。然而，由于MFC诞生于年代比较早，受制于当时的编程思想和C++特性，其内部实现有很多历史遗留问题的痕迹，存在着类型不安全，线程不安全等诸多弊端，并且由于MFC是Windows API的高层封装，使用MFC编写的图形化程序只能运行在Windows平台上且难以进行跨平台移植。随着微软公司的.NET平台战略逐步成型，MFC已经于2011年3月停止了更新。

Java是面向对象的编程语言，具有功能强大和简单易用两个特征。Java源代码不能直接编译为机器码，而是生成一种被称为字节码的中间语言，这种中间语言面向Java虚拟机（JVM, Java Virtual Machine），配合不同平台上对应的JVM，Java可以实现跨平台。Swing包含在JAVA基础类库当中，它包括了一些基本的GUI（图形用户界面）器件，如：文本编辑区域、按钮、窗口和表。Swing用纯Java写成，所以同Java本身一样可以跨平台运行，可以更换面板和主题。然而Swing不是真的使用原生平台的图形界面，而是模仿它们的外观。Swing是一种轻量级组件，

虽然可以非常容易的进行跨平台开发，但是执行速度较慢，而且受制于 Java 本身的效率问题，Swing 不适合用于大型软件的开发。

C#是微软公司发布的一种面向对象的高级程序设计语言，同时也是在.NET Framework 之上进行程序开发的主力语言。C#看起来与 Java 有着惊人的相似：不允许多继承、接口的定义与使用、与 Java 几乎同样的语法和编译成中间代码再运行的过程。得益于.NET Framework 与 Windows 系统出色的运行时服务，C#拥有较高的运行效率。Microsoft Visual Studio 集成开发环境对 C#的支持和辅助非常强大，借助其中附带的窗口设计器和针对 C#增强的智能感知和编码辅助功能可以很容易地设计基于 Winform 的 GUI 程序,事实上 Winform 程序的架构与 Java Swing 程序极为相似，二者都是采用高级语言控制界面控件的生成和摆放，这种开发模式可以在完全没有图形化编程辅助工具的情况下通过文本编码直接生成界面，但是由于控制界面生成的代码冗余度很高，繁琐且不容易使用，这种开发模式已经逐渐不被采用，取而代之的是使用标记语言或者脚本语言控制界面元素布局的新方法。

WPF（Windows Presentation Foundation）是微软推出的基于 Windows 的用户界面框架，属于.NET Framework 的一部分。WPF 使用统一的编程模型、C#和 XAML 语言和 MVC 框架，分离了美工人员与编程人员的工作；同时它提供了现代化的多媒体交互用户图形界面。WPF 抛弃了陈旧的 GDI/GDI+，而是直接基于 DirectX 图形引擎进行界面的绘制，支持 GPU 硬件加速，提高了运行效率。使用配套的开发工具，编程人员可以使用基于 XML 语言的 XAML 标记性语言或者窗口设计器来创建更加美观的视觉效果和更友好的交互方式。在 Windows 平台上，WPF 已经取代了 Winform 和 MFC 成为了微软最重视的桌面应用开发框架。

Qt 于 1991 年由奇趣科技开发，是一个跨平台的 C++图形用户界面应用程序开发框架，经历了几次收购之后，Qt 仍然保持着更新。Qt 可 O 以开发 GUI 程序、控制台工具和服务器等非 GUI 程序。Qt 是面向对象的，在原始 C++代码的基础上，增添了代码生成扩展、元对象编译器（Meta Object Compiler, moc）以及一些宏；易于扩展，允许组件编程。Qt 既支持直接用高级语言生成界面元素，也支持使用基于 XML 扩展的 QML 标记性语言进行界面的制作，在官方提供的 QtCreater 的辅助下可以快速创建个性化的 UI。

表 1-2 对上述的几种常用图形用户界面应用程序开发框架进行了对比。

表 1-2 常用图形界面应用程序开发框架对比

框架名称	开发语言	是否跨平台	布局控制	是否更新	开发复杂度
MFC	C++	否	C++	否	高
Java Swing	Java	是	Java	是	一般
C# WinForm	C#	否	C#	是	一般
WPF	C#/VB.Net	是	XAML	是	低
Qt	C++/Python	是	C++/QML	是	一般

本文拟采用 Caffe 作为核心组件，由于 Caffe 采用 C++编码，为了减少程序开发的复杂度，避免引入交叉编译等额外问题，且综合考虑了技术的先进性和交互效果，本文采用 Qt 作为图形用户界面应用程序开发框架。

1.2.4 总结

以神经网络为代表的深度学习方法，近年来在图像处理、模式识别等领域取得了诸多突破性进展，相关方法受到了研究人员的广泛关注。以卷积神经网络为例，在效果得到了学界的一致认可的同时，仍然存在着理论解释缺乏，新网络的设计和调整依靠经验等问题，同时，仍有诸如减少数据依赖、进一步提高精度和效率、拓展应用领域等问题亟待解决，因此，以神经网络为代表的深度学习方法具有较高的研究价值和研究空间。

对国内外相关领域的研究现状进行分析后发现，目前已经有多款深度学习框架问世，基于这些框架，已经诞生了许多值得关注的研究成果。但是，这些框架多以开源库、工具集等形式出现，没有形成完整的工具链和集成环境，存在着开发环境的搭建和配置难度高、命令复杂、学习成本高等问题，给相关的开发过程增添了额外的成本。集成化、图像化的开发环境，对于神经网络的开发具备实际意义。

在对现有的深度学习框架和图形用户界面应用程序开发框架进行了对比和梳理之后，决定采用 Qt 和 Caffe，开发一款集成化、图形化的卷积神经网络辅助设计系统。

1.3 主要研究内容

以 Caffe 为核心框架，以 Qt 为图形用户界面应用程序开发框架，以 C++为主要开发语言，开发一款集成化的卷积神经网络辅助设计软件，该软件将神经网络设计中的网络设计，保存，数据转换，训练，监视，测试，组件升级等一系列功能封装为一个集成系统，为卷积神经网络的开发流程提供完整工具链支持。具体研究内容包括：

（1） 管理神经网络开发过程中使用和形成的文档。该系统能够组织文件目录结构，存储网络描述文件、部署文件、数据集文件等各种文件，提供可视化的文件树供用户进行手动管理。

（2） 为网络描述文件提供带有辅助功能的编辑器。系统提供一个文本编辑器，能够对网络部署描述和结构描述文件进行语法高亮提示，为网络结构的设计提供辅助。

（3） 为网络结构的创建提供可视化蓝图编辑器。参考 Unreal Engine 4 中提供的虚幻编辑器的脚本可视化编辑功能（如图 1-10，Unreal Engine 4 中的蓝图编辑器），系统提供一个蓝图编辑器，该编辑器能够将文本化的网络结构描述文件可视化为结构图，且可以通过编辑结构图实现对描述文本的修改和创建。

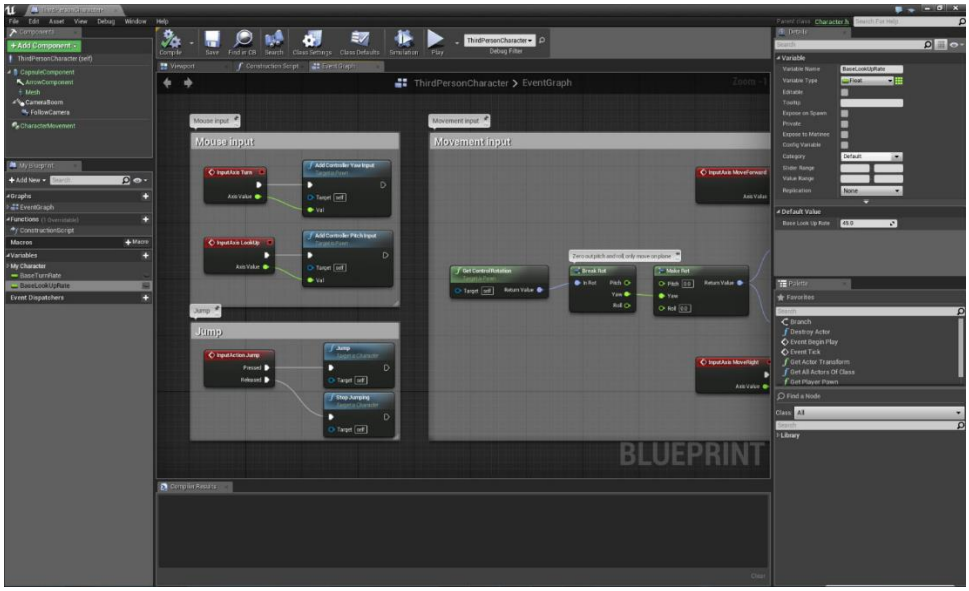


图 1-10 Unreal Engine 4 中的蓝图编辑器

（4） 集成化工具。系统提供配置管理功能，通过配置管理，可以在不修改集成工具的源代码的情况下切换系统使用的核心框架，对内部命令和参数进行维护，实现系统的扩展和更新，同时，配置管理模块维护一个文本化的配置文件，用户可以通过直接对配置文件进行内容更新来影响系统的行为。除此之外，系统还提供数据格式装换，数据集浏览，特征导出等常用工具，并全部实现图形化。

1.4 技术路线及方案

1.4.1 研究过程和技术路线

依照软件工程的理论和实践经验，本文遵循需求分析、概要设计、详细设计、编程，单元测试和部署测试等步骤与顺序，同时考虑到所使用的技术的掌握情况，在毕业设计开始的初期，进行了有计划的资料查阅和基础知识学习，在进行了需求分析和概要设计之后，确立了如下技术路线：

（1） 文档化管理。参考 Microsoft Visual Studio 的界面布局方案，使用 Windows 平台的“文档视图模型”进行文档化管理，将对文件的直接操作入口绑定在文件树的右键上下文菜单上。为了在整个界面内合理排布各种功能，采用“边缘浮动窗口+中心 MDI 视图”的设计方案。

（2） 调用核心组件。本系统采用 Caffe 深度学习框架作为核心组件，通过对 Caffe 组件进行预编译，提供头文件、动态库和静态库供外部调用，同时参考集成开发环境（IDE）“编译核心+辅助功能壳”的设计思路，将不需要后期扩展变更的功能封装成独立的可执行文件，而主程序通过启动额外进程的形式对该组件进行调用，采用输入输出重定向的方法实现线程之间的值传递。

（3） 图形化的蓝图编辑器。通过对 Qt 的图形组件进行重载和改写，实现碰撞检测，曲线绘制和上下文相应，并通过栈式的语法管理实现蓝图和文本的互相转换。

（4） 配置管理。程序的不存在源代码级别的内部数据，所有运行过程中需要的参数都是通过外部的数据库或文件系统存储的，配置管理模块负责管理这些

文件，并向需要参数的其他模块提供参数访问服务，用户可以通过图形界面或者直接编辑配置文件的形式更改这些参数，进而变更程序的行为，通过修改核心组件的位置，可以实现可新组建的替换，这个过程不需要更改任何源代码和重新编译生成。

1.4.2 重点和难点以及解决方案

在本毕业设计的开发过程中，有以下重点和难点，相应的解决方案也一并列出。

（1） **Qt 和 Caffe 同时使用。**由于 Qt 的窗口设计器“QtDesigner”根据编程人员设计的界面形式，会生成采用 qml 描述的 ui 文件，这种文件是不能直接参与 C++的编译过程的，Qt 通过 moc.exe 对 ui 文件进行转换，生成对应的 C++头文件，然后，所有带有“QObject”标记的类都会经过 Qt 的处理，转换为纯 C++实现，最终参与整个工程的编译连接等生成过程，所以，带有“QObject”标记的 Qt 类的生成工具必须是 Qt 的生成工具；另一方面，由于 Caffe 采用了 Google 的 Protobuf 作为数据交换格式，在 Caffe 的源代码中包含了一个协议定义文件“caffe.pb.h”，这个文件是使用“protoc.exe”将协议定义文本转换形成的 C++文件，所以包含或间接包含这个文件的 C++文件的生成工具必须是 protoc.exe。如果通过图形界面调用 Caffe 相关的功能，势必会引入既含有 caffe.pb.h 头文件，又带有“QObject”标记的类，由于生成工具的冲突，这样的类势必无法编译成功。

解决方案：通过引入另一个类隔离 Qt 类和 Caffe 类。为了解决上述生成工具不兼容的问题，本文中大量使用了如图 1-11 所示的类结构，可见，通过中间类的隔离，不需要既含有 caffe.pb.h 头文件，又带有“QObject”标记的类就可以实现从图形界面调用核心功能。同时，中间类的存在可以封装参数完整性验证等与核心功能关系不大且篇幅比较长、影响业务逻辑的可读性的代码。

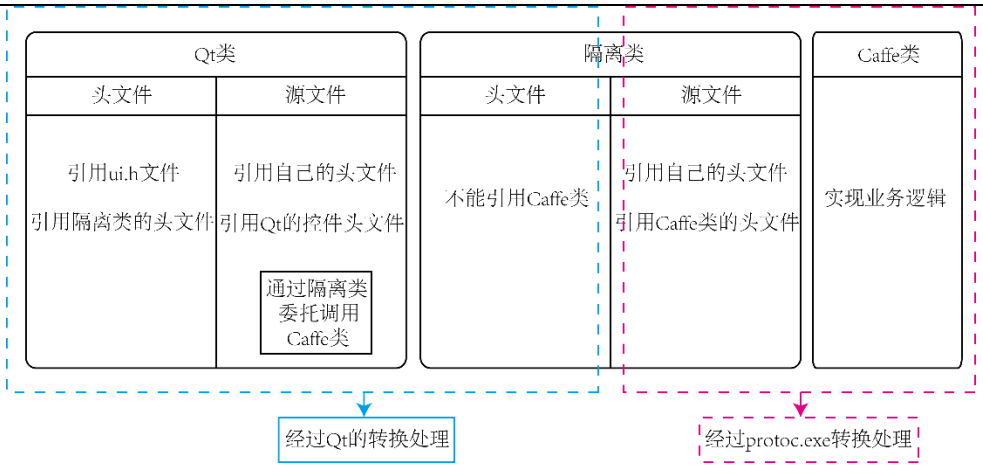


图 1-11 通过中间类隔离 Qt 类和 Caffe 类

（2） **蓝图编辑器中图形的绘制。**参考虚幻编辑器中“模块”之间的链接曲线，需要实现一种能够自动根据起止点变换曲线形状的曲线生成方法。现有的曲线生成函数不能直接实现这个功能。

解决方案：分析具体的绘制过程，以贝塞尔曲线为基础，定义“灵敏系数”和“变距因数”，通过一个自定义函数调控曲线控制点的位置，动态生成贝塞尔曲线，从而实现上述功能。

1.5 论文的整体结构安排

全文共分为 5 章，结构安排如下：

第 1 章：绪论。首先对本课题的背景和意义进行了介绍，然后分析了相关领域的国内外研究现状，从而确定出本文所采用的技术，最后明确项目的主要研究内容和技术路线以及方案。

第 2 章：相关工作。首先对本文采用的技术进行了分析。详细分析了 Caffe 工程的源代码结构和 Qt 框架程序的特点，这是技术方案制定的基础和依据。

第 3 章：对技术方案进行了细化介绍，其中重点介绍了使用带有中间层的类结构解决生成工具的冲突问题和自定义的曲线生成算法，随后按照软件工程工作流程，进行了需求分析和概要设计，最后给出详细设计和实现。

第 4 章：测试和评价。按照软件工程的理论和实践经验，对系统进行单元测试和部署测试，分别给出了测试的方案和结果。

第 5 章：总结与展望。首先对本文的研究结果做出总结，对最终实现的系统给出评价，分析了当前研究中的不足，并对未来深度学习框架和可视化方面的研究做出了展望。

第 2 章　相关工作

2.1　技术分析

本文所描述的系统在开发过程中涉及到对 Caffe 的源代码的扩展和定制，并且涉及到 Qt 框架的深度使用，因此，对于 Caffe 工程的源代码和 Qt 进行分析具有实践价值。

2.1.1　Caffe 工程源码结构分析

Caffe 作为框架级别程序，其代码量较大，结构复杂，但是了解其整体架构和实现思想是进行基于 Caffe 的开发所必须的条件，下面对 Caffe 的架构进行简要分析。

（1） Caffe 的依赖

Caffe 主要有以下几种依赖：Glog：用于输出日志；Protobuf ：用于定义数据交换格式；Boost：相当于后备的 C++标准库，Caffe 主用使用其提供的有关智能指针和线程方面的实现；Gflags：命令行工具；levelDB&lmDB：提供基于文件映射的 IO，本质上是使用 B+树的 key-value 数据库；Opencv：图形处理库，只在数据表层使用，用于处理图片的色彩变换，几何变换等操作。

（2） Caffe 解决方案的结构

Caffe 解决方案的结构如图 2-1 所示，整个解决方案中分为多个工程其中有一个名为“caffe”的工程生成 lib 和 dll，其他的工程生成的都是可执行文件，通过对 caffe.lib 等动态库和静态库的调用实现功能。

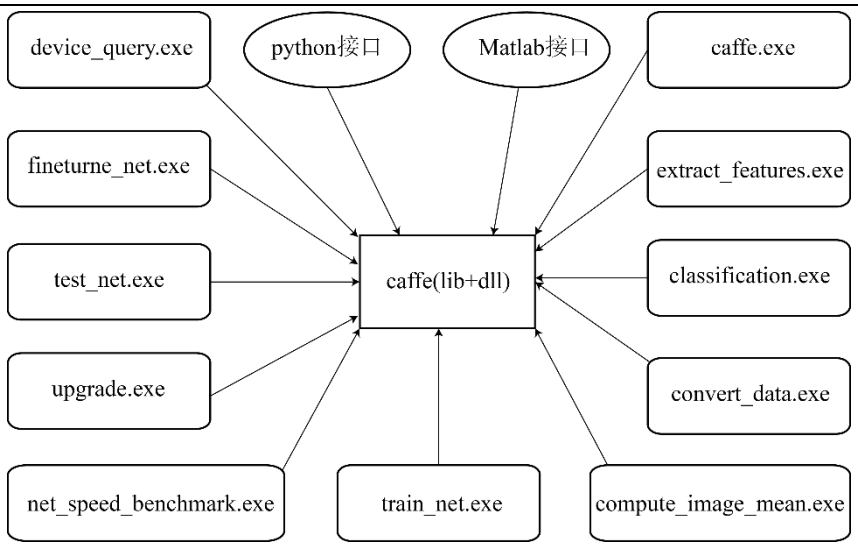


图 2-1 caffe 解决方案的结构

（3） Caffe 的核心结构

Caffe 的核心类分为四大种类，分别是：IO 核心类、数据核心类、业务核心类和辅助类，下面分别给出这些类的简要介绍：

IO 核心：训练数据和测试数据是由大量的磁盘文件构成的，直接读取磁盘文件的 IO 开销会十分巨大，Caffe 将这些零散的数据聚合成 LevelDB 或 LmDB，避免了频繁的开关文件，以提高 IO 效率。数据库的管理主要由三个类簇实现，如图 2-2 所示：

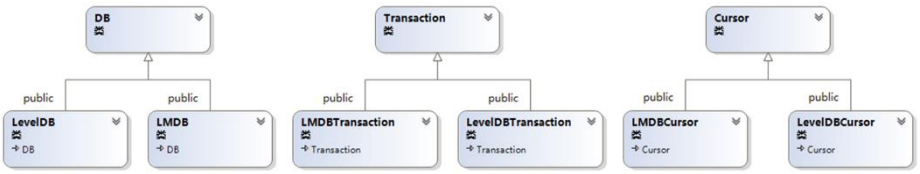


图 2-2 IO 核心类簇

数据核心：用来表示网络中各种各样的数据，包括训练数据，网络各层自身的参数，包括权值、偏置以及梯度，网络之间传递数据都是通过 Blob 来实现的。Blob 数据同时支持在 CPU 与 GPU 上的存储和交换，能够在两者之间同步。这个类非常大，它含有 53 个方法，是整个 Caffe 的数据核心。Blob 类的部分字段定义见图 2-3。

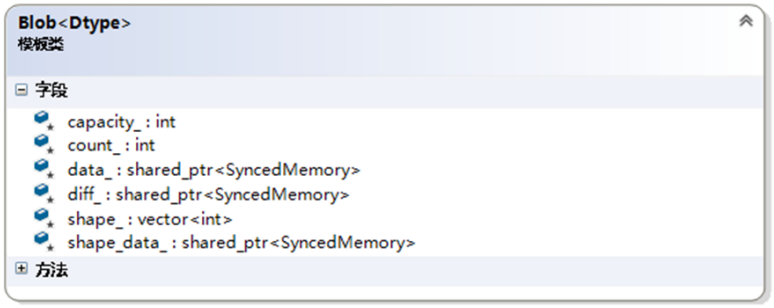


图 2-3 Blob 类的字段定义

业务核心：Caffe 的业务核心分为三个部分，分别是 Solver 类簇、Filler 类簇和 Layer 类簇。其中，Solver 类簇定义了针对不同网络模型的求解方法，并负责记录网络的训练过程，保存网络模型参数，中断并恢复网络的训练。通过继承超类 Solver，能够自定新的网络求解方式。Solver 类簇的示意图为 2-4。

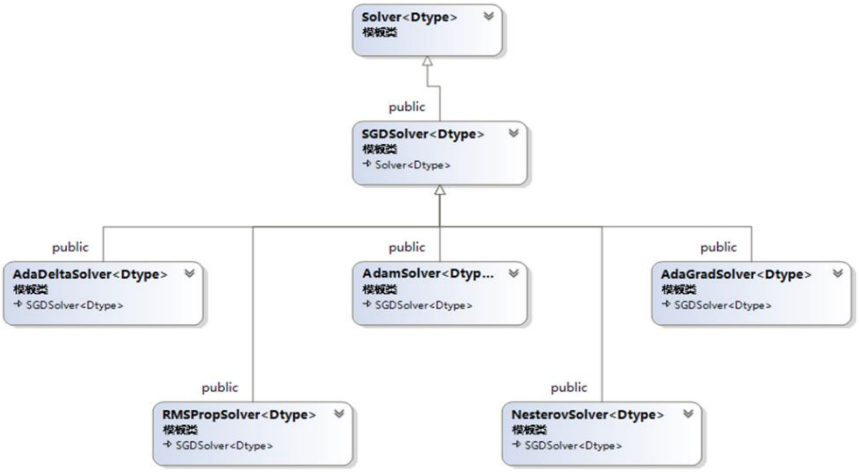


图 2-4 Solver 类簇

Filler 类簇作用是根据 prototxt 中给出的参数对权重进行初始化，初始化的方式有很多种，例如常量初始化（Constant）、高斯分布初始化（Gaussian）、Positive Unitball 初始化、均匀分布初始化（Uniform）、Xavier 初始化、Msra 初始化、双线性初始化（Bilinear）等。Filler 的示意图见 2-5。

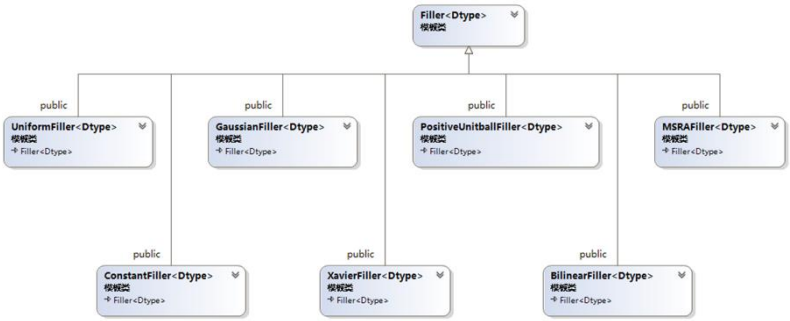


图 2-5 Filler 类簇

Layer 类簇是对神经网络中各种层的实现，包括卷积层、下采样层、全连接层等。在 Caffe 中，激活操作也被定义为一个单独的层。同时每种 Layer 都实现了前向传播和反向传播方法，并通过 Blob 来传递数据。Layer 类簇较为庞大，占据了 Caffe 的绝大多数编码，图 2-6 展示了 layer 类簇的大体结构，其中右上角为超类。

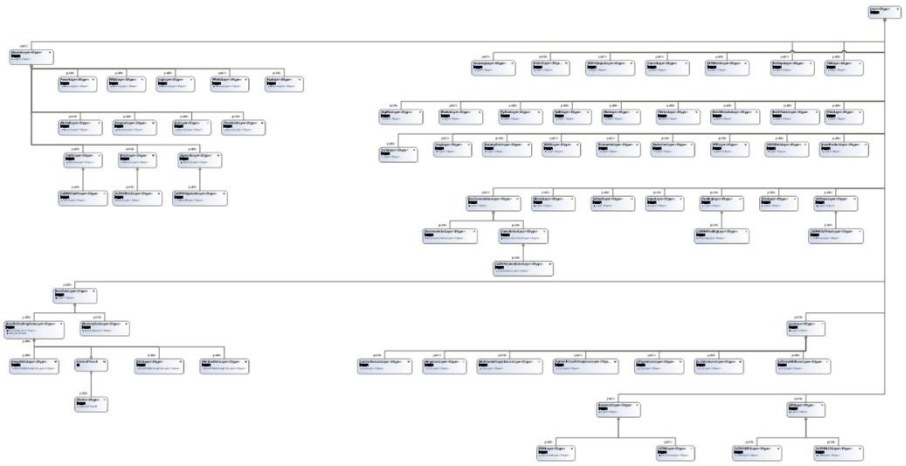


图 2-6 Layer 类簇

辅助类：这些类负责程序的时钟服务，基本类型定义，网络结构保存等，图 2-7 给出辅助类的示意图。

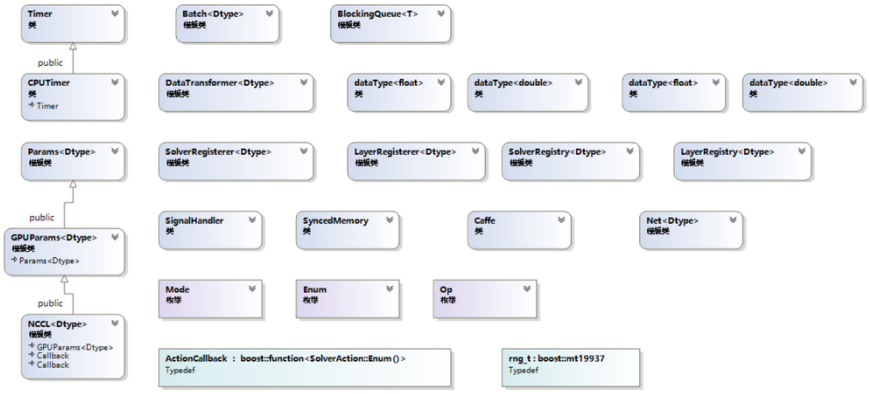


图 2-7 辅助类

（4） Caffe 的运行过程

初始化：首先构造 Solver 对象。Solver 的构造会触发 Net 类的构造，分别构造训练和测试网络示例。Net 类的构造函数调用 LayerRegistry 类的静态成员函数 CreateLayer 得到一个指向 Layer 类的 shared_ptr 类型指针，并把每一层的指针存放在 vector 中，具体来讲，这里保存了三种指针：层的输入 Blob、输出 Blob 和参数。在建立每层时，调用相应层的构造函数，为本层申请空间并进行权值初始化。

LayerSetUp 是 Layer 超类的虚方法。初始化流程：Solver->Net->Layer->Blob，所以最后我们得到了一个 Solver 类型的指针，初始化结束。

训练：通过 Solver 类型指针调用 solver()，从这里开始训练的流程。solver()函数控制迭代次数，每次迭代都是调用 solver 类的 step()函数。Step()函数控制 batch 的大小，每次执行 ForwardBackward()函数都会返回一个损失值，并累加在 loss 之中，循环一个 batch 之后，对 loss 进行平均，再调用 update()函数更新权值。Net 类的 ForwardBackward()函数先调用 Forward(&loss)，再调用 Backward()函数，而 Net 类的 Forward(&loss)和 Backward()函数实际上是在 layer 上实现的。上述平均操作由 UpdateSmoothedLoss()实现，带有平滑功能，将当前 Loss 和历史过程中更新的 Loss 进行平均，能够减少 Loss 的震荡问题。update()是 Solver 的纯虚函数，每个派生的 Solver 都必须有自己的更新方法，这就体现了不同网络结构的不同设计策略。

2.1.2 Qt 分析

（1） Qt 的组件。

Qt 作为一个成熟的 C++图形界面应用开发框架，拥有较为完善的工具链和组件，且拥有足够的扩展性。除了 Qt 的源代码资源和动态库、静态库、调试数据库和示例程序之外，Qt 还包含了一个用于 Qt 开发的轻量级跨平台集成开发环境“Qt Creator”（图 2-8）。Qt Creator 集成了帮助查看器、带有上下文感知的高级 C++编辑器和源代码管理，项目构建工具的常用工具，且在各平台上的界面和表现几乎一致，是一个易用性强的集成开发工具；同时，Qt 还包含了专门用于界面设计的 Qt Designer（图 2-9），专门用于读取翻译文件的 Qt Linguist（图 2-10）等工具，通过安装相应的动态库和静态库，可以为 Qt Designer 的工具箱添加新的界面控件。

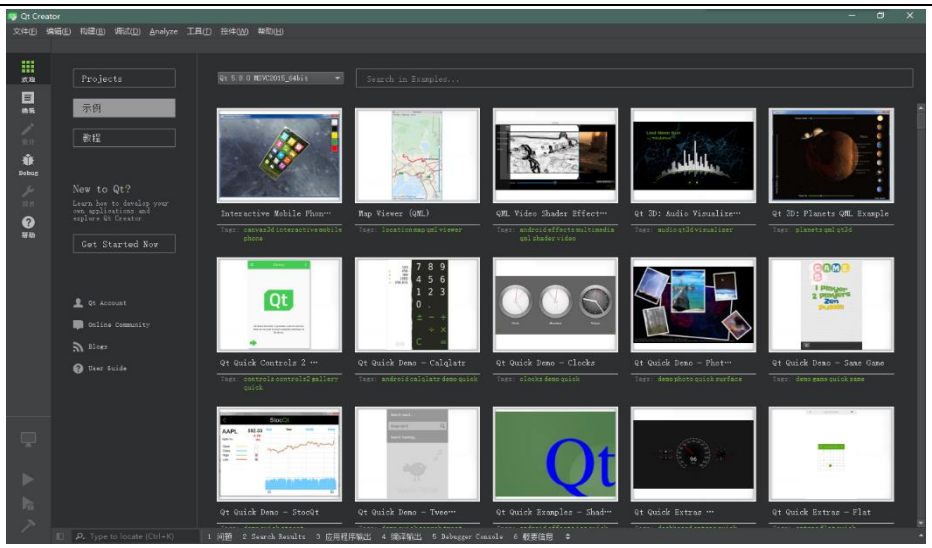


图 2-8 Qt Creator

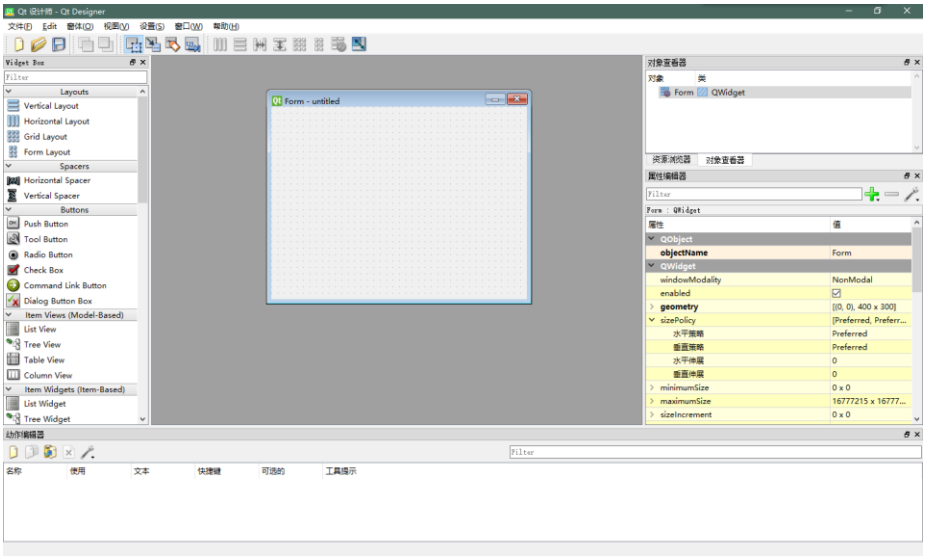


图 2-9 Qt Designer

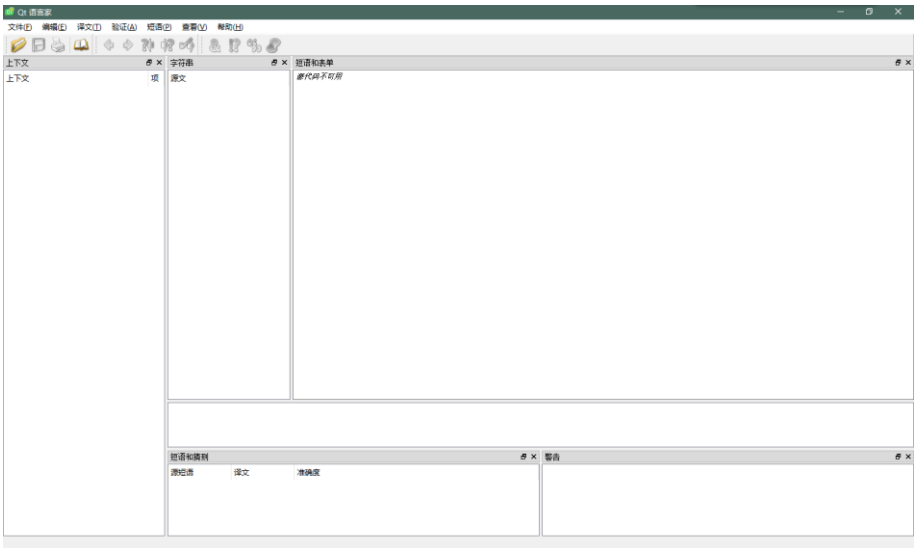


图 2-10 Qt Linguist

（2） Qt 开发环境的配置

Qt 的开发环境的配置相对简单，以 Windows 平台为例，直接下载所需版本的安装包进行安装即可，安装完毕之后就可以使用内置的 Qt Creator 进行程序的开发，如果需要进行复杂工程的开发，还可以安装 Microsoft Visual Studio 和对应版本的 Qt for Visual Studio 插件，无需进行额外的配置，就可以通过 Visual Studio 进行 Qt 程序的开发了。

（3） Qt 编写的 GUI 程序的结构和生成过程。

使用 Qt 开发的 C++用户界面应用程序的基本结构如图 2-11 所示。工程主要由外部依赖项、Generated Files、界面文件、头文件、源文件、资源文件和类图等组成。其中，外部依赖项中存储的是工程的外部依赖库所需要的头文件；Generated Files 中存储的是 Qt 在 make 过程中产生的文件，即界面文件和资源文件生成的头文件和源文件；头文件和源文件文件夹中存储的是 Qt 类和 C++类的定义和声明；资源文件中存储的是和界面相关的图片等资源文件的 Qt 描述；类图是对整个工程进行分析时所用的、由 VS 生成的类图。程序编制完成后，启动编译生成过程，Qt 的相关组件会首先阅读整个工程，首先将资源文件和界面文件进行 Uic 操作，转换为对应的 C++文件，其次对所有带有“QObject”宏标记的类进行 MOC 操作，将 Qt 类中的信号槽传递等操作转换为 C++实现，随后，MSVC 编译器开始对所有的 C++文件进行编译、链接和生成，如果整个工程中没有编译阶段错误和链接错误，生成过程到此结束，否则将会中断生成过程并显示错误原因。

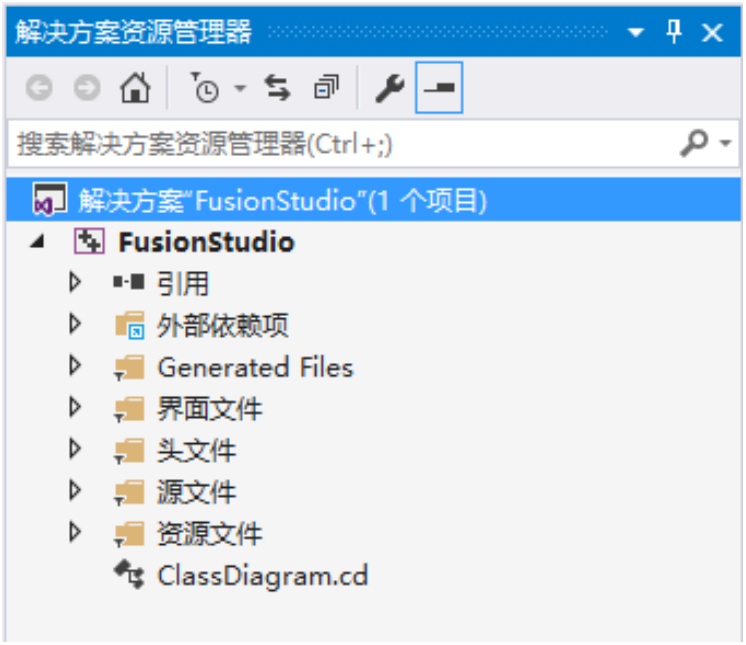


图 2-11 使用 Qt 编写的图形应用界面程序

第 3 章 系统实现

3.1 技术方案

在对 Qt 和 Caffe 进行了深入的研究测试后，决定采用“内部调用+外部委托+重定向”的方式进行本系统的开发，技术方案如下：

（1） 内部调用：对于深度学习框架中与网络结构、系统架构等易变组件无关的部分，采用内部调用的方式进行开发，具体形式有直接编码和外部依赖项。这部分功能与数据处理有关，需要较高的速度和可靠性，且没有后期更新的必要，故直接采用编码方式内化到程序主体中。

（2） 外部委托：对于测试、分类等与具体网络形式有关或者有较大的可能进行后期升级的组件，采用外部委托的方式，将其预编译为动态库，以供程序主体进行调用。由于没有与程序的主体产生代码层面的耦合，这部分功能可以通过替换核心组件的方式进行升级。

（3） 重定向：对于训练功能，由于这个功能有极强的效率和并行度要求，且其状态提示不容许对程序的运行速度有过大的要求，上述两种方案无法达到要求，这种情况与 Visual Studio 和 MSVC 编译器的关系极为类似，故而模仿其实现方案，采用输入输出重定向的方法，直接通过程序主体构造参数，传送给指定的外部进程，并将该进程的输出重定向到主程序的主线程上，从而实现相应功能的可视化，且避免了图形界面程序为了实现消息循环和事件驱动而产生的额外开销。

3.2 需求分析

3.2.1 需求描述

（1）对于模型的训练，测试，提供图形操作界面。各种参数的设置，文件的读取载入，训练过程的提示，测试用的图片，都应通过 UI 的形式呈现，如果需要使用外部程序，不应该出现用户需要打开另一个可执行文件的情况，所有的外部调用都应是主程序自动完成的。

（2）对网络结构的建立提供辅助，实现形式可以是语法高亮等。网络结构采用文本化描述，应该对该文本提供必要的语法高亮，编辑器应该能够实现文本的打开、修改、保存，具备行号显示，注释显示等功能。

（3）对于常用数据形式，提供可视化的格式转换工具和浏览工具。数据集的读取、加载和转换，应当是图形化的，转换的过程的成功与失败以及转换的进度应当在界面上给出提示，程序应有一个工具可以实现数据集的浏览，包括显示图片内容，标签，对图片进行缩放等操作。

（4）能够以可视化的方式显示网络模型提取到的特征。对于预训练好的网络和给定的输入图片，程序应当能给出这张图片在神经网络的某一层中的特征表示，提取到的特征可以保存为图片，也可以存入数据库，但程序应能够实现特征的显示。

（5）支持核心组件的升级和替换。本程序使用Caffe框架作为内核，如果Caffe发布了新版本，本程序要在不修改主程序源码的情况下进行核心组件的替换和升级，而保持功能的正常运行。

3.2.2 系统边界

在以下的说明中，将 “由 Caffe 实现或依赖于 Caffe 实现的、完成卷积神经网络的核心计算功能的程序部分”统一简称为“内核”，将本文所论述的“基于 Qt 的卷积神经网络辅助设计系统”简称为统一“系统”。

（1）对于模型的训练，测试，提供图形操作界面。对于训练和测试当中发生的数据传递和参数显示等问题，应当由系统处理，而调集 CPU 和 GPU 等运算能力进行具体计算，在内存中组建神经网络的具体实例等操作由内核完成，不属于系统的功能。

（2）对网络结构的建立提供辅助，实现形式可以是语法高亮等。网络结构描述文本的读取、显示、编码辅助应当由系统完成，而解析网络结构并将其转换为 C++ 实现等步骤由内核完成，不属于系统的功能。

（3）数据格式转换工具和浏览工具。“常用的数据集”包括 Mnist、Cifar 和图片文件与标注文件构成的数据集合。加载、读取、转换这些形式的数据为系统的功能，但是系统的功能不包括反向转换，即将采用系统内部数据库 lmdb 存储的数据转换为采用 Mnist、Cifar 或者图片形式存储；系统对于采用 lmdb 形式存储的数据提供浏览功能，但对于采用非 lmdb 的其他形式存储的训练、验证或测试数据

集，系统不提供浏览功能；浏览功能中所述的“显示图片所对应的标签”，指的是显示出数据库中直接存储的标签，有些数据集带有标签词典文件，对整型数表示的标签进行意义对照解释，而显示解释之后的标签含义不为系统的功能。

（4）能够以可视化的方式显示网络模型提取到的特征。使用该功能必须提供的数据有预训练好的网络、输入图片、网络部署描述文件，系统不具备自动生成部署描述的能力，在不提供足够的正确文件的情况下，系统的该功能将不能运行。提取到的特征存储在数据库中，可以通过系统的相关功能进行读取，但是系统不提供对特征图片的编辑功能，仅输出原始图片。

（5）支持核心组件的升级和替换。需求分析中指出的“当 Caffe 发布新版本时”中的 Caffe 版本，指的是由 Berkeley Vision and Learning Center（BVLC）在其官方 GitHub 主页上发布的 Caffe 版本，对新替换的组件进行检测和导入是系统的功能，但是生成 Caffe 的动态、静态链接库需要额外操作并需要借助编译器，此功能不属于系统的功能范畴。系统所支持的升级和替换，不包括对 Caffe 整体架构和外部接口的大规模变更，对于不进行上述变更的自定义版本，系统也可以支持。

3.3 系统设计

3.3.1 输入和输出

（1）训练

输入：保存训练和验证数据集的 lmdb 数据库、网络结构描述。

输出：网络快照、预训练网络模型、训练过程统计数据。

（2）编辑

输入：prototxt 代码。

输出：prototxt 文件。

（3）测试

输入：网络部署描述、预训练网络、图片均值文件、标签词典、测试图片。

输出：该张图片属于词典中的各个类别的可能性。

（4）特征导出

输入：网络部署描述、预训练网络、导出位置、数据库名、预导出的层的 ID、mini_batches。

输出：保存着指定特征的 lmdb 数据库。

（5）数据转换

输入：转换前的数据库存储位置、输出位置、转换参数、文件名。

输出：存储着数据集的 lmdb 数据库。

（6）数据集浏览

输入：数据集位置。

输出：数据集中的图片和标签。

3.3.2 运行环境

（1）软件环境

系统运行所需的软件环境为：

a. 如需使用 Python 接口，需要安装 Python，系统支持 Python2.7 以及 Python3.5；

b. 如需使用 CUDA，需要安装 NVIDIA CUDA Toolkit；

c. 如需使用 CuDNN，需要安装 NVIDIA CUDA Deep Neural Network library。

（2）硬件环境

系统运行所需的最低硬件环境要求为：

a. Windows 7 64 位家庭普通版

b. Intel i3-530 2.93Ghz 或 AMD Phenom II X4 940 3.0GHz

c. 内存 4 GB

d. 硬盘可用空间 5 GB

神经网络的训练是高时间复杂度和空间复杂度的任务，且对并行化要求高，上述硬件环境仅为维持系统的正常安装和运行所需的最低要求，在以下的推荐配置上，可以获得更好的效率：

a. Windows 10 64 位专业版

b. Intel Core i7-3770 3.5GHz 或 AMD FX-8350 4.0GHz 及以上

c. 内存 8 GB 或更高

d. 硬盘 50 GB

e. 支持 CUDA 和 CUDNN 的 NVIDIA 显示卡

本系统测试和开发过程中所使用的硬件环境如下：

a. Windows 10 64 位专业版

b. Intel Core i7-6700K 4.0GHz

c. 8GB 内存

d. 1T 硬盘/大于 320GB 的可用空间

e. NVIDIA GeForce GTX 1080 显卡

3.3.3 处理流程

系统共分为八个模块，另有三个独立视图和一个核心组件。其中，数据集浏览模块、数据转换模块和特征导出模块采用基于中间层的内部委托调用形式调用核心组件中的相关接口；训练模块和测试模块通过输入输出重定向模块，采用外部进程的形式调用核心组件，并通过控制台视图进行交互；首选项模块可以对该过程中所使用的命令和组件进行配置；文件视图，开始视图是独立的，仅连接了一些浅层接口；编辑器模块依赖于系统的图形界面，且不依赖与核心组件。系统的整体结构见图 3-1。

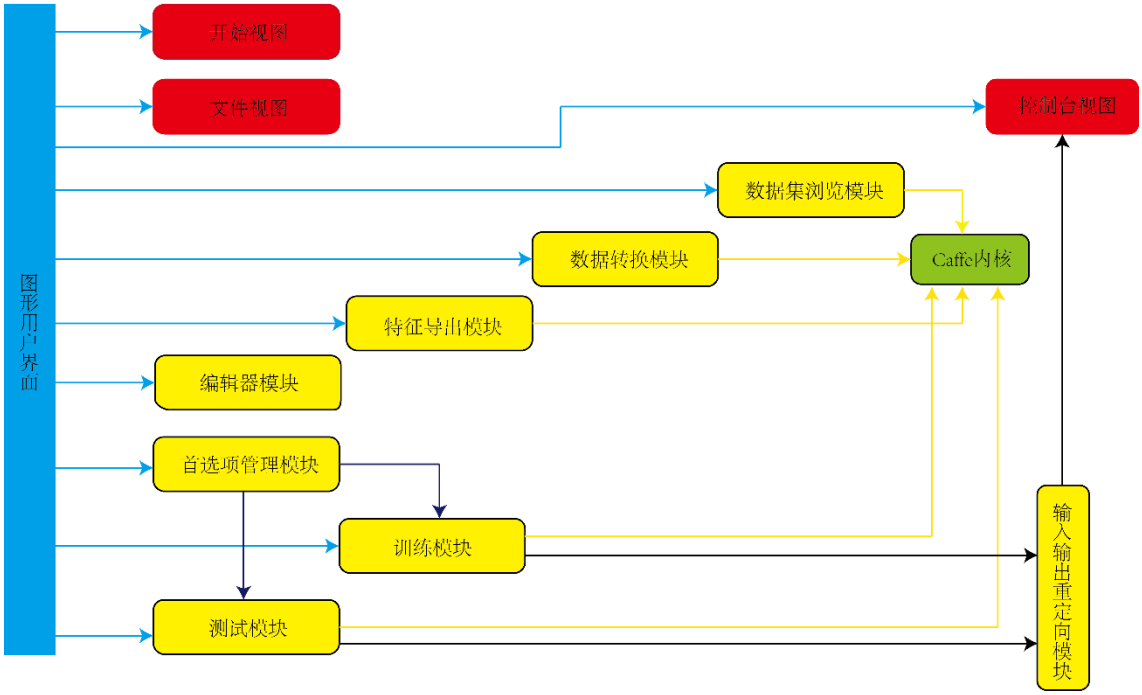


图 3-1 “基于 Qt 的卷积神经网络辅助设计系统”系统结构图

3.3.4 功能与模块划分

以下表格显示了系统的主要功能和模块的关系。

表 3-1 主要功能和模块

功能 / 模块	工程目录管理	数据集浏览	数据集转换	网络结构设计	网络训练	网络测试	特征导出	帮助信息
开始视图								√
文件视图	√							
控制台视图					√	√		
编辑器模块				√				
特征导出模块							√	
数据转换模块			√					
数据集浏览模块		√						
首选项管理模块					√	√		
测试模块						√		
训练模块					√			
输入输出重定向模块					√	√		
Caffe 内核		√	√		√	√	√	

3.3.5 接口设计

（1）外部接口

本系统的外部接口有图形用户界面和 Python 交互接口两种，其中，使用图形用户界面可以完成卷积神经网络的设计、调整、训练、测试等整套流程，并提供了数据转换和浏览等实用工具；Python 交互接口继承自 Caffe 内核，本系统保留了原有的全部 Python 接口。

（2）内部接口

本系统采用模块化设计，内部接口主要有两种形式，第一种为“中间层委托”，即使用一个中间类完成 Qt 界面类和业务逻辑类的桥接同时支持两种互不兼容的生成工具的处理；第二种为“外部委托”，使用输入输出重定向，在 Qt 界面类中完成命令的构造，以此调用外部进程，并将该进程的输出重定向到 UI 上，实现支持替换的组件与主程序的桥接。

3.3.6 人工处理过程

由于现有技术方案的限制，系统中存在以下几处不得不进行人工干预及处理的操作：

（1） **将图片集转换为采用 lmdb 数据库存储。**在这个过程中需要一个记录了所有图片文件名的文本文件作为参数，虽然可以使用诸如目录服务类之类的方法读取某个目录下的全部文件，但考虑到，作为神经网络的训练集，图片的数量往往是数万张乃至数十万张，读取如此庞大的目录将会造成不可避免的卡顿，所以本系统采用了文件列表的方式，用户需要使用 DOS 命令来生成这个文件并加载到系统中。

（2） **针对预训练网络的分类测试。**基于（1）中提到的类似的考虑，在使用这个功能的时候同样需要使用 DOS 命令生成一个文件列表文件，以提高系统的 IO 可靠性。

（3） **特征导出。**在导出预训练网络的某一层的特征时，需要给出这一层的 ID，这个 ID 是层的唯一识别，是一个在定义网络结构的时候由用户给出的字符串。由于底层依赖没有提供反向查找一个网络中所有的层 ID 的方法，所以系统无法给出 ID 的备选项，使用特征导出的时候，用户需要自行找到这个字符串并填写在指定的界面控件处。

3.4 系统实现

3.4.1 实现概述

系统的图形用户界面采用 Qt 开发，主界面采用 MDI 文档为核心，浮动窗口为辅助的视图形式，首选项管理器采用模态对话框实现，文件树视图和终端视图使用浮动窗口实现，其余的功能如数据转换，训练、测试等功能均为 MDI 文档视图，在编辑器视图中，实现了语法高亮和可视化编辑功能。

图 3-2 展示了系统最终实现的界面效果。

图 3-2 系统界面效果

3.4.2 具体实现

（1）支持语法高亮的文本编辑器

图 3-3 展示实现语法高亮文本编辑器的类结构。其中，Ui_NetEditor 类实现了编辑器的界面，具体界面见图 3-4，Editor 类继承自 QPlainTextEdit，具备基本的文本编辑功能，并实现了文本编辑器的保存、另存为、载入、新建等功能，同时使用一个槽函数，连接到内容变更信号上，每当编辑器界面中的内容发生了变更，就触发 highlightCurrentLine()函数，该函数根据 MyHighLighter 中定义的语法规则对当前行使用正则表达式进行匹配，并根据匹配结果使用对应的颜色和字体进行语法标注。另外，通过对 QWidget 的继承，实现了行号显示条，能够自动标注行号。语法高亮的效果如图 3-4 所示。

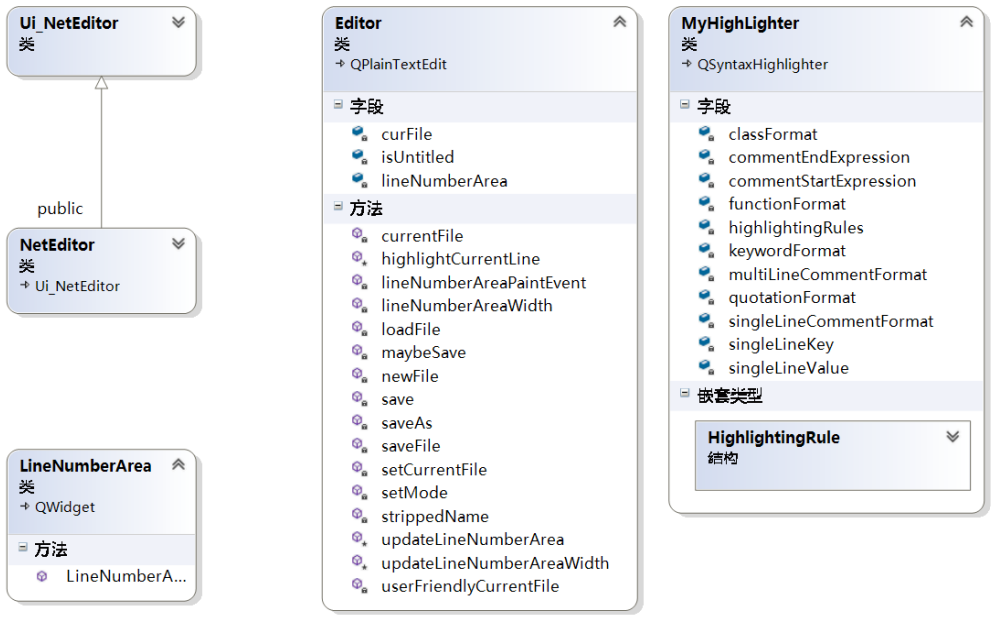


图 3-3 实现语法高亮文本编辑器的类结构

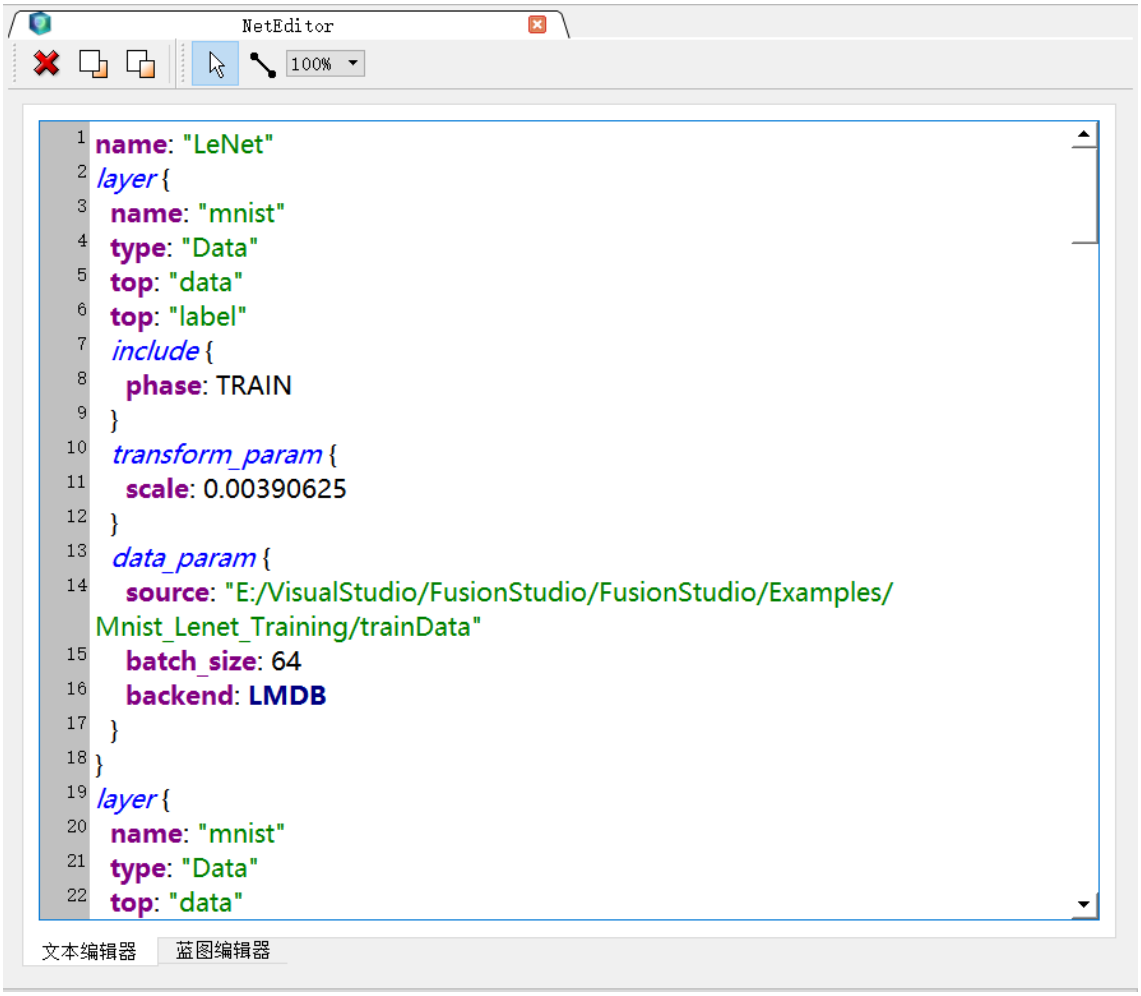


图 3-4 文本编辑器的显示效果

（2）图形化网络结构设计器

图 3-5 展示实现蓝图编辑器的类结构。通过继承 QGraphicsScene 实现

DiagramScene 类，实现了插入图元、删除图元和图元颜色编辑等操作，重写父类中的鼠标响应函数来响应鼠标的拖动和点击，以实现图元的拖动；继承 QGraphicsPixmapItem 实现 DiagramItem 类，用于存储几种既定图元，图元从矢量图加载，能够向绘图区域进行绘制，并支持碰撞检测；继承 QGraphicsLineItem 实现 Arrow 类，用于存储和绘制网络块之间的连接曲线，该曲线要连接两个网络块且能够随着网络块的位置变化而自动调整形状，为了实现该功能，必须对标准曲线绘制函数进行扩展。

Qt 提供了贝塞尔曲线的绘制函数 cubicTo()，该函数绘制的贝塞尔曲线需要指定起点、终点和两个控制点，其基本形状如图 3-6 所示，符合系统对曲线形状的要求，但是这样的曲线无法实现根据起止点位置而调整自身形状，为此，我们定义了灵敏度参数 *theta* 和变距因数 *m*，通过这两个参数来控制曲线的形状以达到自动调整的目的。其中，*theta* 是一个常数，通过调整这个值来调整曲线变形的速度，*m* 的值由以下的公式给出：

$$m = \frac{theta}{（|y_2 - y_1| + 1）\times（|x_2 - x_1| + 1）}$$

其中，点（x_1, y_1）是曲线的起点，（x_2, y_2）是曲线的终点，变距因数 m 的作用是根据曲线的起点和终点拉开两个控制点之间在 X 轴上的距离，根据贝塞尔曲线的特点，拉开这两个控制点的距离会使得曲线的两个拐点更加“锋利”，从而出现一种“绕开”的风格。两个控制点的坐标使用以下的方法生成：

$$x_1^k = \frac{x_1 + x_2}{2} + m,\ y_1^k = y_1 + \frac{m}{p}$$

$$x_2^k = \frac{x_1 + x_2}{2} - m,\ y_2^k = y_2 - \frac{m}{p}$$

其中，（x_1^k，y_1^k）和（x_2^k，y_2^k）为曲线的控制点，p 为纵向灵敏度参数，由于曲线在纵向上的变形不需要像横向变形那样反应灵敏，所以要用一个参数调控纵向的变距量。曲线的绘制结果如图 3-7 和图 3-8 所示。可见随着起止点的变化，曲线的形状进行了自动调整。

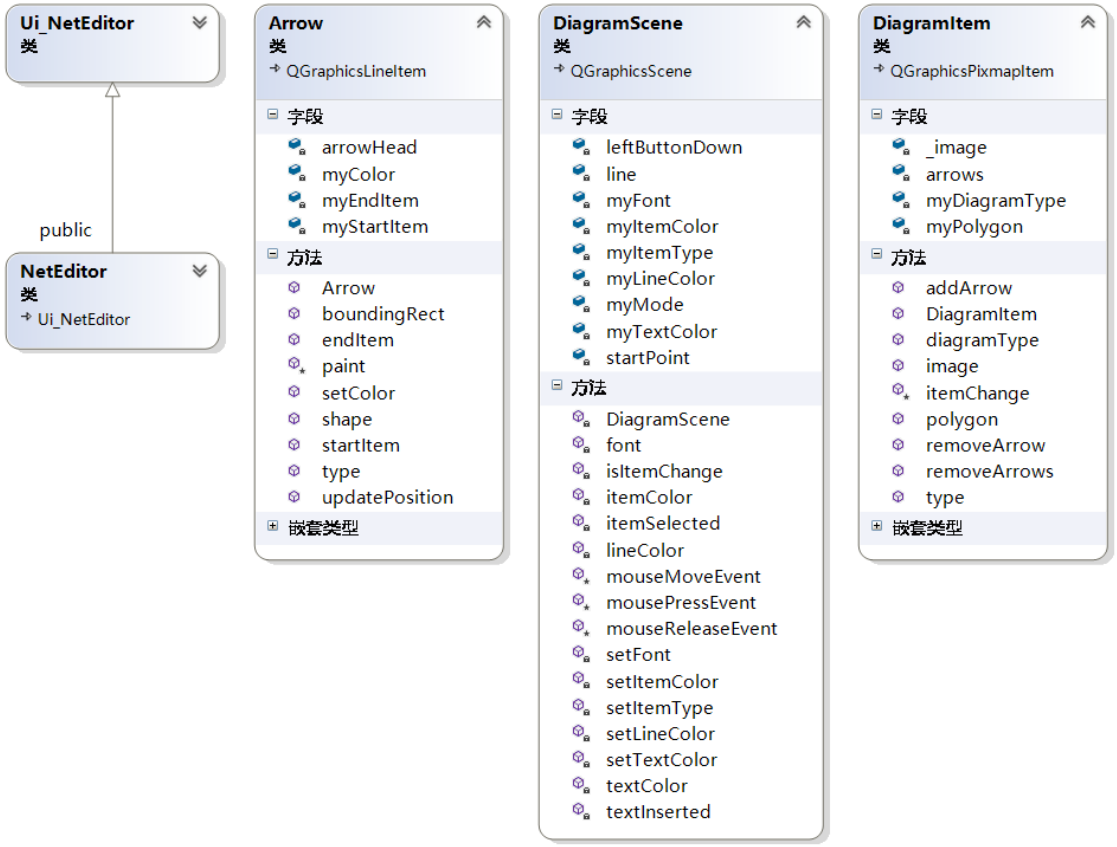
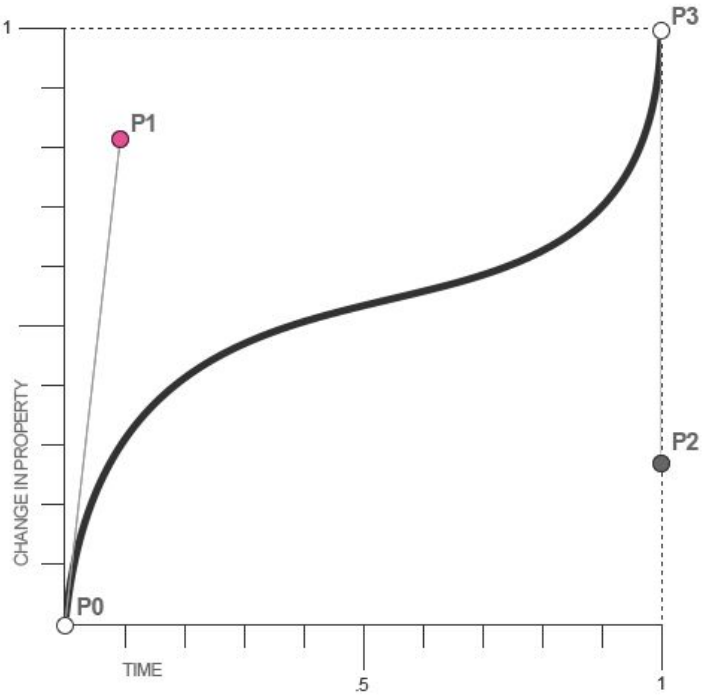


图 3-5　图形化网络结构编辑器（蓝图编辑器）的类结构



3-6 贝塞尔曲线

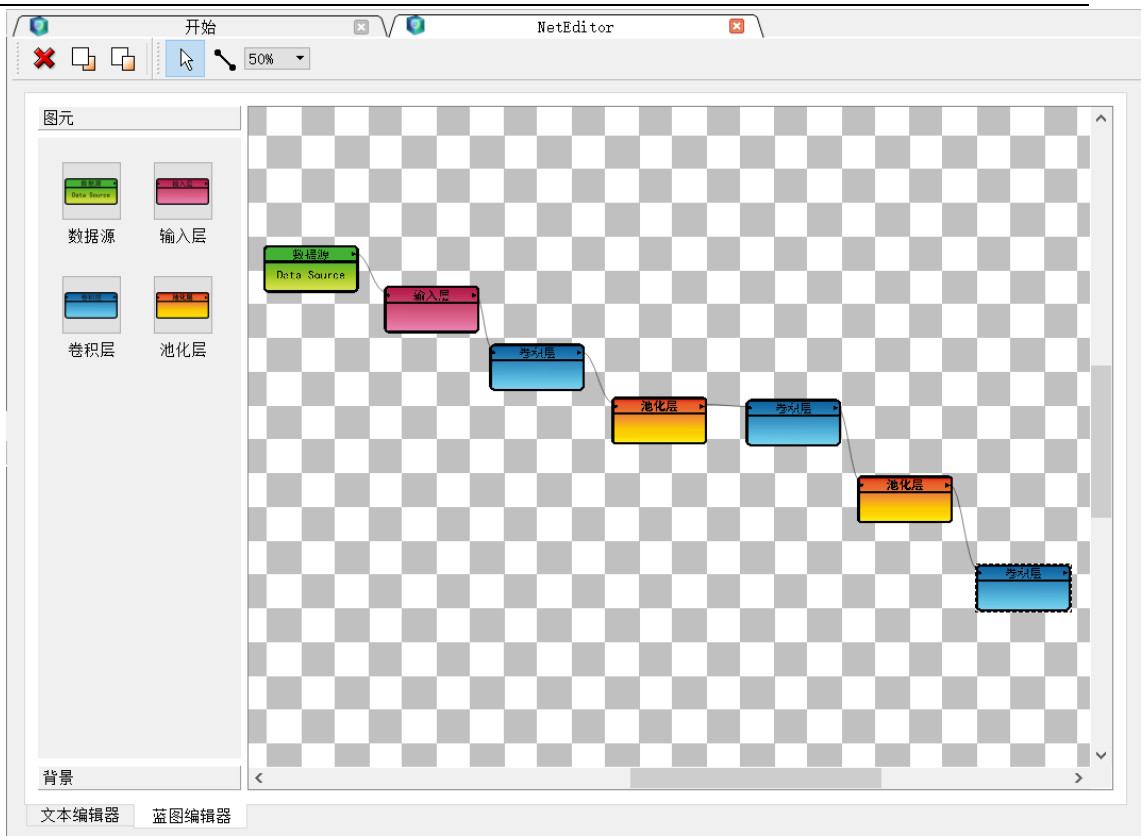


图 3-7 蓝图编辑器 （1）

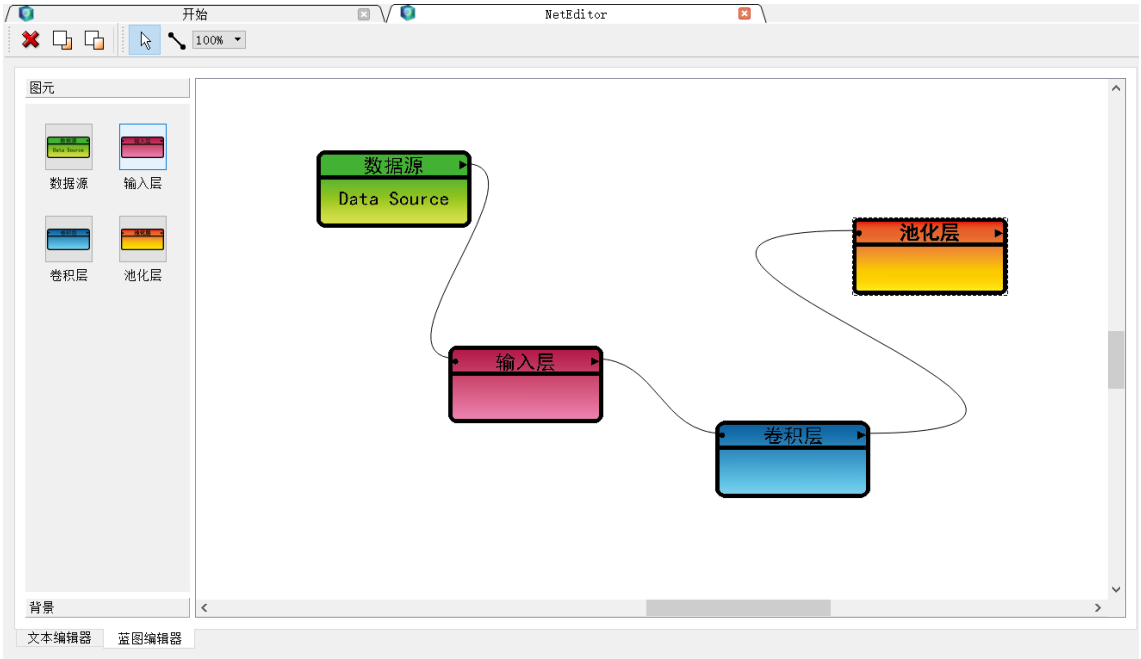


图 3-8 蓝图编辑器（2）

第 4 章　测试和评价

4.1 测试要点

测试方法：黑盒测试。

测试手段：采用手动测试。

4.2 模块功能测试

对所有的功能模块进行功能测试，由于其中的输入输出重定向模块和图形界面无法作为一个单独得到组件工作，所以这两项的测试在与其有关的模块中完成。

4.2.1 编辑器模块

针对编辑器模块的测试点如表 4-1 所示。

表 4-1 编辑器模块测试点

序号	功能点
1	编写 prototxt 文件，关键字、语句块、注释等语法元素要有高亮显示。
2	将编写好的文档保存到磁盘。
3	蓝图编辑器能够在绘图区域放置网络元素单元块。
4	蓝图编辑器的菜单条中所有按钮功能正常。
5	在蓝图编辑器的绘图区域能够使用连线连接两个网络结构块，且曲线自然流畅。

测试结果：所有功能点均达到要求。

4.2.2 特征导出模块

针对特征导出模块的测试点如表 4-2 所示。

表 4-2 特征导出模块测试点

序号	功能点
1	能够正常载入所有的参数和所需文件。
2	进度条等其他界面元素显示正常。
3	能够在指定位置生成存储有特征的 lmdb 数据库文件。
4	当参数和所需文件输入异常或者有空项目的时候，能够提示用户进行更正而不崩溃。
5	能够将提取到的特征显示在图片浏览窗口中。

测试结果：所有功能点均达到要求。

4.2.3 数据转换模块

针对数据转换模块的测试点如表 4-3 所示。

表 4-3 数据转换模块测试点

序号	功能点
1	能够正常载入所有的参数和所需文件。
2	界面元素显示正常。
3	能够在指定位置生成存储有数据集的 lmdb 文件，且可读取。
4	当参数和所需文件输入异常或者有空项目的时候，能够提示用户进行更正而不崩溃。

测试结果：所有功能点均达到要求。

4.2.4 数据集浏览模块

针对数据集浏览模块的测试点如表 4-4 所示。

表 4-4 数据集浏览模块测试点

序号	功能点
1	能够正常载入 lmdb 数据库
2	界面元素显示正常。
3	能够显示数据库中的图片和标签值。
4	能够对图片进行缩放和播放下一张等操作。
5	如果加载的数据库有问题或者为空，应当有所提示而不引起程序崩溃。

测试结果：所有功能点均达到要求。

4.2.5 首选项管理模块

针对首选项管理模块的测试点如表 4-5 所示。

表 4-5 首选项管理模块测试点

序号	功能点
1	能够正常打开，并能从配置文件中加载正确的默认参数。
2	通过对界面上提示的参数进行修改，能够写入到配置文件中。
3	如果配置文件不存在，能够建立配置文件。
4	组件校验功能正常。
5	命令设置功能正常。

测试结果：所有功能点均达到要求。

4.2.6 测试模块

针对测试模块的测试点如表 4-6 所示。

表 4-6 测试模块测试点

序号	功能点
1	能够正常载入所有的参数和所需文件。
2	界面元素显示正常。
3	当参数和所需文件输入异常或者有空项目的时候，能够提示用户进行更正而不崩溃。
4	控制台视图能够正常显示测试的结果。
5	能够显示正在测试的图片。

测试结果：所有功能点均达到要求。

4.2.7 训练模块

针对训练模块的测试点如表 4-7 所示。

表 4-7 训练模块测试点

序号	功能点
1	能够正常载入训练求解器文件，并读取出其中的参数显示在界面上。
2	界面元素显示正常，对于界面上元素的修改能够正常保存在求解器文件中。
3	当参数和所需文件输入异常或者有空项目的时候，能够提示用户进行更正而不崩溃。
4	控制台窗口能够正常显示训练过程的输出信息。
5	能够正确保存网络快照，网络模型等训练输出信息。

测试结果：所有功能点均达到要求。

4.3 部署测试

本系统使用 Installshield 进行打包，并在数台不同的计算机上进行安装测试，为了证明本系统完成安装后不需要进一步进行环境配置，选取了若干可能影响系统可用性的参变量，设计了对比试验，实验结果如表 4-8。

表 4-8 部署测试

	GPU	Python	Qt	CUDA/CUDNN	安装结果
1	NVIDIA Gtx 1060	2.7	有	有	可用
2	NVIDIA Gtx 1060	2.7	有	无	可用
3	NVIDIA Gtx 1060	2.7	无	有	可用
4	NVIDIA Gtx 1060	2.7	无	无	可用
5	NVIDIA Gtx 1060	3.5	有	有	可用
6	NVIDIA Gtx 1060	3.5	有	无	可用
7	NVIDIA Gtx 1060	3.5	无	有	可用
8	NVIDIA Gtx 1060	3.5	无	无	可用
9	NVIDIA Gtx 1060	无	有	有	可用
10	NVIDIA Gtx 1060	无	有	无	可用
11	NVIDIA Gtx 1060	无	无	有	可用
12	NVIDIA Gtx 1060	无	无	无	可用
13	AMD	2.7	有	无	可用
14	AMD	2.7	无	无	可用
15	AMD	3.5	有	无	可用
16	AMD	3.5	无	无	可用
17	AMD	无	有	无	可用
18	AMD	无	无	无	可用

经过严格的对比试验，结果表明本系统运行的和安装不需要依赖于目标系统上的其他环境，由于已经将所有的依赖库打包，安装到目标计算机上之后，即可立即运行，不需要进行额外的配置。

第 5 章 总结与展望

5.1 总结

基于 Qt 的卷积神经网络辅助设计系统采用 C++为编程语言，以 Qt 为图形界面编程框架，以深度学习框架 Caffe 为核心组件，设计并实现了一款图形化、集成化的卷积神经网络设计环境，通过安装释放，用户可以方便的搭建神经网络开发所需的环境。该系统为神经网络的设计、训练、测试等流程建立了一套工具链，提供了常用的工具，对网络结构的设计进行了优化设计，且支持核心组件的升级替换。通过使用这套环境，可以为相关开发人员节省时间成本。

经过完善的测试流程，本系统满足需求分析所提出的软件需求，形成了一款完整的软件产品，但同时，该系统仍与许多不足之处，由于系统架构和所采用的技术的限制，仍存在很多需要人工处理的过程，网络结构设计器的蓝图编辑功能仍有很多不完善之处，例如缺少从文本向蓝图的反向解析功能等，由于使用了大量外部依赖项，导致本系统的体积达到 2.6GB，代码尚缺乏进一步优化。

综上，基于 Qt 的卷积神经网络辅助设计系统基本达成了最初的设计要求，仍有部分功能值得进一步改进和研究。

5.2 展望和预测

随着深度学习，特别是以卷积神经网络为代表的神经网络方法在图像处理、自动驾驶、自然语言处理等领域取得越来越多的成果，并不断拓展新的适用方向，深度学习框架作为其研究学习的必备工具必将获得越来越多得到关注。更好的工具意味着更低的入门门槛，也意味着研究人员可以把有限的时间投入到更有价值的研究工作中而不是浪费在对工具和环境的反复调整上。可以预见，未来的深度学习工具，将会向着集成化程度更高、工具链更完整、支持平台更多、安装使用更加方便、界面图形化、迭代快速化的方向发展。

参考文献

[1]Bengio Y. Learning deep architectures for AI[J]. Foundations and trends in Machine Learning, 2009, 2(1): 1-127.

[2]rizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[C] Advances in Neural Information Processing Systems. 2012: 1097-1105.

[3]Dahl G E, Yu D, Deng L, et al. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition[J]. IEEE Transactions on Audio, Speech, and Language Processing, 2012, 20(1): 30-42.

[4]Sun Y, Wang X, Tang X. Deep learning face representation from predicting 10,000 classes[C] Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014: 1891-1898.

[5]Taigman Y, Yang M, Ranzato M A, et al. Deepface: Closing the gap to human-level performance in face verification[C] Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014: 1701-1708.

[6]Karpathy A, Toderici G, Shetty S, et al. Large-scale video classification with convolutional neural networks[C] Proceedings of theIEEE conference on Computer Vision and Pattern Recognition. 2014: 1725-1732.

[7]Ji S, Xu W, Yang M, et al. 3D convolutional neural networks for human action recognition[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2013, 35(1): 221-231.

[8]Dong C, Loy C C, He K, et al. Learning a deep convolutional network for image super-resolution[C] European Conference on Computer Vision. Springer International Publishing, 2014: 184-199.

[9]Roth H R, Lu L, Liu J, et al. Improving Computer-Aided Detection Using Convolutional Neural Networks and Random View Aggregation[J]. IEEE Transactions on Medical Imaging, 2016, 35(5): 1170-1181.

[10]Serre T, Kreiman G, Kouh M, et al. A quantitative theory of immediate visual recognition[J]. Progress in brain research, 2007, 165: 33-56.

[11]Hornik K, Stinchcombe M B, White H, et al. Multilayer feedforward networks are

universal approximators[J]. Neural Networks, 1989, 2(5): 359-366.

[12]Cun Y L, Boser B E, Denker J S, et al. Handwritten digit recognition with a back-propagation network[C]. Conference and Workshop on Neural Information Processing Systems, 1990: 396-404.

[13]Redmon J, Divvala S, Girshick R, et al. You only look once: Unified, real-time object detection[C] Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016: 779-788.

[14]Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks.[J]. Journal of Machine Learning Research, 2010: 249-256

[15]Dahl G E, Sainath T N, Hinton G E, et al. Improving deep neural networks for LVCSR using rectified linear units and dropout[C]. International Conference on Acoustics, Speech, and Signal Processing, 2013: 8609-8613.

[16]Nair V, Hinton G E. Rectified linear units improve restricted boltzmann machines[C] Proceedings of the 27th International Conference on Machine Learning (ICML-10). 2010: 807-814.

[17]Jarrett K, Kavukcuoglu K, Ranzato M, et al. What is the best multi-stage architecture for object recognition[C]. International Conference on Computer Vision, 2009: 2146-2153.

[18]Zeiler M D, Krishnan D, Taylor G W, et al. Deconvolutional networks[C] IEEE Conference on Computer Vision and Pattern Recognition, 2010: 2528-2535.

[19]Zeiler M D, Fergus R. Visualizing and understanding convolutional networks[C] European Conference on Computer Vision. Springer International Publishing, 2014: 818-833.

[20]github: https://github.com/gwding/draw_convnet, 2017-4-25

[21]github: https://github.com/ethereon/netscope, 2017-3-1

[22]github: https://github.com/raghakot/keras-vis, 2017-3-25

[23]http://auduno.com/post/125362849838/visualizing-googlenet-classes, 2017-4-15

[24]http://terencebroad.com/convnetvis/vis.html, 2017-4-8

[25]http://shixialiu.com/publications/cnnvis/demo/, 2017-5-15

[26] Jia Y, Shelhamer E, Donahue J, et al. Caffe: Convolutional architecture for

fast feature embedding[C]Proceedings of the 22nd ACM interna

[27] https://www.tensorflow.org,2017-5-19

[28] http://mxnet.io/index.html,2017-5-19

[29] http://torch.ch,2017-5-19

[30] http://deeplearning.net/software/theano,2017-5-19

致　　谢

本论文是在导师王欣老师的细心指导下完成的。导师渊博的专业知识，严谨的治学态度，精益求精的工作作风对我影响深远。本论文从选题到完成，倾注了导师大量的心血。在此谨向导师表示崇高的敬意和衷心的感谢。

本论文写作过程中所使用的场地和设备是由吉林大学计算机科学与技术学院图形学与数字媒体实验室提供的，在此对实验室的老师和学长们致以谢意。

本论文的顺利完成，离不开各位同学和朋友的关心和帮助。在此感谢杨竣翔、徐斌、薛潺涓、孙坤博的帮助，同窗之间的友谊永远长存。