Syslab深度学习工具箱V2版本

**方案设计**



|  |  |  |  |
| --- | --- | --- | --- |
| **编制** | **曹宇宇、胡如明、梅再武** | **生效日期** |  |
| **审核** |  | **批准** |  |

**文件变更摘要**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **日期** | **版本** | **变更说明** | **修订** | **审核** | **批准** |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

目录

[1. 引言 1](#_Toc129125839)

[1.1 概述 1](#_Toc129125840)

[1.2 参考资料 1](#_Toc129125841)

[2. 需求分析 2](#_Toc129125842)

[2.1 功能要求 2](#_Toc129125843)

[2.1.1 基础功能 2](#_Toc129125844)

[2.1.2 神经网络DAG图 2](#_Toc129125845)

[2.1.3 GPU加速运算 2](#_Toc129125846)

[2.1.4 机理数据融合建模 2](#_Toc129125847)

[2.1.5 代码生成 3](#_Toc129125848)

[2.1.6 底层框架可扩展 3](#_Toc129125849)

[2.1.7 分布式并行计算 3](#_Toc129125850)

[2.1.8 丰富的应用案例库 4](#_Toc129125851)

[2.2 性能要求 4](#_Toc129125852)

[2.3 运行环境要求 4](#_Toc129125853)

[3. 开发目标 5](#_Toc129125854)

[3.1 指标要求 5](#_Toc129125855)

[3.2 函数表 5](#_Toc129125856)

[4. 总体设计 8](#_Toc129125857)

[4.1 总体架构 9](#_Toc129125858)

[4.2 总体方案 10](#_Toc129125859)

[4.3 应用流程 11](#_Toc129125860)

[4.4 预期效果 12](#_Toc129125861)

[5. 技术路线 13](#_Toc129125862)

[5.1 基础功能 13](#_Toc129125863)

[5.2 神经网络DAG图 13](#_Toc129125864)

[5.3 机理数据融合 14](#_Toc129125865)

[5.4 分布式并行计算 15](#_Toc129125866)

[5.5 代码生成 16](#_Toc129125867)

[5.6 底层框架可扩展 17](#_Toc129125868)

[5.7 GPU加速 18](#_Toc129125869)

[6. 验证方案 18](#_Toc129125870)

[6.1 性能验证 18](#_Toc129125871)

[6.2 应用验证 18](#_Toc129125872)

[6.2.1 图像处理 19](#_Toc129125873)

[6.2.2 音频处理 24](#_Toc129125874)

[6.2.3 文本分析 25](#_Toc129125875)

[6.2.4 复杂系统预测性维护 26](#_Toc129125876)

[6.2.5 机理-数据融合建模 27](#_Toc129125877)

[6.2.6 模型降阶 28](#_Toc129125878)

[6.2.7 综合展示系统 28](#_Toc129125879)

[7. 研发计划 29](#_Toc129125880)

[7.1 人员安排 29](#_Toc129125881)

[7.2 进度计划 30](#_Toc129125882)

# 引言

## 概述

Syslab 深度学习工具箱V1版本基于华为MindSpore深度学习框架开发，对标Matlab深度学习工具箱，目前已开发基本功能函数100个。

深度学习工具箱V2版本基于V1版本继续开发，进一步扩充基本功能函数至215个。并基于Julia语言实现多种深度学习底层框架自由选择；完善GPU运算功能及分布式并行计算能力；探索代码生成逻辑；在此基础上实现与MWORKS.Sysplorer相结合的机理数据融合建模。最终旨在构建多硬件平台、多框架、快速运算、功能完备的新一代深度学习工具箱。

深度学习工具箱V2版本以完成国家项目为最低要求，与华为MindSpore深度合作，同时为其他深度学习框架提供集成接口。更重要的目标在于探索AI技术在复杂装备数字化建模领域的创新应用，为复杂系统建模中的难点问题提供AI解决方案，满足客户需求，同时积累AI技术能力，为智能建模仿真平台预研提供技术支撑。

## 参考资料

1. MindSpore官方文档

<https://www.mindspore.cn/docs/zh-CN/r2.0.0-alpha/index.html>

1. PyTorch官方文档

<https://pytorch-cn.readthedocs.io/zh/latest/>

1. MatLab深度学习工具箱帮助文档

<https://ww2.mathworks.cn/help/deeplearning/index.html>

# 需求分析

## 功能要求

### 基础功能

在V1版本的基础上支持更多类型的神经网络模型构建和训练，训练参数更加多样化，网络构建更加灵活化，训练过程更加便捷化，训练结果更加可视化，模型读取更加实用化。

### 神经网络DAG图

利用DAG网络创建神经网络，也属于使用函数创建神经网络的方式。DAG网络用于创建具有更为复杂体系结构的神经网络（残差网络就是一种典型的DAG网络），其中，网络中的每一层均为有向无环图，各层可以有来自其他层的多个输入和多个输出。通过使用神经网络DAG图，使得网络构建更加简单，逻辑更加清晰。

### GPU加速运算

对于深度学习训练来说，GPU已经成为加速器的最佳选择。大多数计算本质上是并行的浮点计算，即大量的矩阵乘法，其最佳性能需要大量的内存带宽和大小，这些需求与HPC的需求非常一致，GPU正好可以提供高精度浮点计算、大量VRAM和并行计算能力。

### 机理数据融合建模

MWORKS.Sysplorer是面向多领域工业产品的系统建模与仿真验证环境，完全支持多领域统一建模规范Modelica，遵循现实中拓扑结构的层次化建模方式，支持物理建模、框图建模和状态机建模等多种建模方式，提供嵌入代码生成功能，支持设计仿真和实现的一体化，是数字化时代国际领先的建模仿真通用软件。

机理数据融合模型是基于MWORKS.Sysplorer的物理仿真模型与深度学习工具箱的深度学习模型联合仿真建模模型，在工程领域实际使用中可以弥补由于物理系统尚不明确或物理系统比较复杂时导致的建模困难与模型臃肿的问题。

### 代码生成

深度学习工具箱V2版本支持全场景的端到端模型部署功能。

1. 极致性能。高效的内核算法和汇编级优化，支持CPU、GPU、NPU异构调度，最大化发挥硬件算力，最小化推理时延和功耗。
2. 轻量化。提供超轻量的解决方案，支持模型量化压缩，模型更小跑得更快，使能AI模型极限环境下的部署执行。
3. 全场景支持。支持iOS、Android等手机操作系统以及LiteOS嵌入式操作系统，支持手机、大屏、平板、IoT等各种智能设备上的AI应用。
4. 高效部署。支持MindSpore/TensorFlow Lite/Caffe/Onnx模型，提供模型压缩、数据处理等能力，统一训练和推理IR，方便用户快速部署。

### 底层框架可扩展

深度学习工具箱V1版本基于华为MindSpore深度学习框架开发，市面上常用的深度学习框架多种多样，用户的个人习惯也各不相同。深度学习工具箱V2将添加底层框架扩展功能，将常用深度学习框架融合进工具箱中，让用户能够只改动少量函数参数的情况下更改深度学习工具箱调用的底层框架。

### 分布式并行计算

在深度学习中，当数据集和参数量的规模越来越大，训练所需的时间和硬件资源会随之增加，最后会变成制约训练的瓶颈。分布式并行训练，可以降低对内存、计算性能等硬件的需求，是进行训练的重要优化手段。根据并行的原理及模式不同，业界主流的并行类型有以下几种：

1. 数据并行（Data Parallel）：对数据进行切分的并行模式，一般按照batch维度切分，将数据分配到各个计算单元（worker）中，进行模型计算。
2. 模型并行（Model Parallel）：对模型进行切分的并行模式。模型并行可分为：算子级模型并行、流水线模型并行、优化器模型并行等。
3. 混合并行（Hybrid Parallel）：指涵盖数据并行和模型并行的并行模式。

### 丰富的应用案例库

深度学习工具箱V2版本将深度学习融入计算机视觉、图像处理、自动驾驶、信号处理、音频、文本分析和计算金融学领域的应用，构建了丰富的应用案例库。用户在使用深度学习工具箱做实际工程项目时，可以借鉴应用案例库中相似应用场景的案例进行开发。

## 性能要求

深度学习工具箱V2版本支持GPU加速运算以及分布式并行计算，在网络模型训练速度上有显著提升。

## 运行环境要求

|  |  |
| --- | --- |
| 软件环境 | 版本 |
| Linux | Linux-x86\_64，处理器需要大于3个 |
| MindSpore | 1.8.1及以上 |
| MindVision | 0.1.0 |
| MindSpore\_Hub | 1.8.0及以上 |
| PyTorch | 1.11.0及以上 |
| Python | 3.7及以上 |

|  |  |
| --- | --- |
| 硬件环境 | 说明 |
| 处理器 | 大于3个 |

# 开发目标

## 指标要求

表 3‑1 深度学习工具箱V2版本开发目标

|  |  |  |
| --- | --- | --- |
| 内容项 | 指标要求 | 说明 |
| 新增函数数量 | 115个 |  |
| 应用案例数量 | 20个以上 |  |
| 宣传公众号文章数量 | 5篇以上 |  |
| 综合展示系统 | 1套 |  |
| 专利数量 | 3篇 |  |
| 论文数量 | 2篇 |  |

## 函数表

表 3‑2 深度学习工具箱V2版本预计开发的函数表

|  |  |  |
| --- | --- | --- |
| **序号** | **函数名** | **函数功能** |
| **1** | activations | 计算深度学习网络层激活 |
| **2** | resetState | 重置循环神经网络的状态 |
| **3** | [cascadeforwardnet](https://www.mathworks.com/help/deeplearning/ref/cascadeforwardnet.html) | 级联神经网络 |
| **4** | [trainscg](https://www.mathworks.com/help/deeplearning/ref/trainscg.html) | 比例共轭梯度反向传播 |
| **5** | [plottrainstate](https://www.mathworks.com/help/deeplearning/ref/plottrainstate.html) | 绘制训练状态值 |
| **6** | [genFunction](https://www.mathworks.com/help/deeplearning/ref/genfunction.html) | 生成用于模拟浅层神经网络的MATLAB函数 |
| **7** | [network](https://www.mathworks.com/help/deeplearning/ref/autoencoder.network.html) | 将Autoencoder对象转换为网络对象 |
| **8** | [patternnet](https://www.mathworks.com/help/deeplearning/ref/patternnet.html) | 生成模式识别网络 |
| **9** | [lvqnet](https://www.mathworks.com/help/deeplearning/ref/lvqnet.html) | 学习矢量量化神经网络 |
| **10** | [plotconfusion](https://www.mathworks.com/help/deeplearning/ref/plotconfusion.html) | 情节分类混淆矩阵 |
| **11** | [plotroc](https://www.mathworks.com/help/deeplearning/ref/plotroc.html) | 情节接收机的工作特性 |
| **12** | [selforgmap](https://www.mathworks.com/help/deeplearning/ref/selforgmap.html) | 自组织地图 |
| **13** | [plotsomhits](https://www.mathworks.com/help/deeplearning/ref/plotsomhits.html) | 绘制自组织地图样本命中 |
| **14** | [plotsomnc](https://www.mathworks.com/help/deeplearning/ref/plotsomnc.html) | 绘制自组织地图邻居连接 |
| **15** | [plotsomnd](https://www.mathworks.com/help/deeplearning/ref/plotsomnd.html) | 绘制自组织地图邻居距离 |
| **16** | [plotsomplanes](https://www.mathworks.com/help/deeplearning/ref/plotsomplanes.html) | 绘制自组织地图权重平面 |
| **17** | [plotsompos](https://www.mathworks.com/help/deeplearning/ref/plotsompos.html) | 绘制自组织地图权重位置 |
| **18** | [plotsomtop](https://www.mathworks.com/help/deeplearning/ref/plotsomtop.html) | 绘制自组织地图拓扑 |
| **19** | [competlayer](https://www.mathworks.com/help/deeplearning/ref/competlayer.html) | 竞争层 |
| **20** | [plotWeights](https://www.mathworks.com/help/deeplearning/ref/autoencoder.plotweights.html) | 绘制可视化的自动编码器的编码器权重 |
| **21** | seq2con | 将顺序向量转换为并行向量 |
| **22** | nndata | 创建神经网络数据 |
| **23** | nndata2gpu | 格式化神经数据以进行有效的GPU训练或模拟 |
| **24** | logsig | 对数sigmoid函数 |
| **25** | satlins | 对称饱和线性传递函数 |
| **26** | radbas | 径向基传递函数 |
| **27** | mapminmax | 通过将行最小值和最大值映射到 [-1 1] 来处理矩阵 |
| **28** | plotes | 绘制单输入神经元的误差面 |
| **29** | getwb | 以单个向量的形式获取网络权重和偏差值 |
| **30** | plotep | 在误差面上绘制权重偏差位置 |
| **31** | numsamples | 神经网络数据中的样本数 |
| **32** | sumabs | 矩阵的绝对元素之和 |
| **33** | gpu2nndata | 从GPU重新格式化神经数据 |
| **34** | compet | 竞争传递函数 |
| **35** | cellmat | 创建矩阵的单元格数组 |
| **36** | meansqr | 矩阵的平方元素的平均值 |
| **37** | formwb | 将偏差和权重形成单个向量 |
| **38** | plotvec | 使用不同颜色绘制矢量 |
| **39** | con2seq | 将并行向量转换为顺序向量 |
| **40** | newlind | 设计线性层 |
| **41** | setwb | 使用单个向量设置所有网络权重和偏差值 |
| **42** | confusion | 分类混淆矩阵 |
| **43** | getsamples | 获取神经网络数据样本 |
| **44** | vec2ind | 将矢量转换为索引 |
| **45** | tansig | 双曲正切 sigmoid 传递函数 |
| **46** | midpoint | 中点权重初始化功能 |
| **47** | rands | 对称随机权重/偏置初始化函数 |
| **48** | gadd | 广义加法 |
| **49** | staticderiv | 静态导数函数 |
| **50** | plotpc | 在感知器矢量图上绘制分类线 |
| **51** | newrb | 设计径向基网络 |
| **52** | gdivide | 广义除法 |
| **53** | gsubtract | 广义减法 |
| **54** | hardlim | 硬极限传递函数 |
| **55** | tonndata | 将数据转换为标准神经网络单元阵列形式 |
| **56** | errsurf | 单输入神经元误差面 |
| **57** | adapt | 使神经网络适应模拟数据 |
| **58** | normc | 标准化矩阵列 |
| **59** | normr | 规范化矩阵行 |
| **60** | separatewb | 从权重/偏差向量中分离偏差和权重值 |
| **61** | perceptron | 简单单层二元分类器 |
| **62** | ind2vec | 将索引转换为矢量 |
| **63** | getelements | 获取神经网络数据元素 |
| **64** | gsqrt | 广义平方根 |
| **65** | nncell2mat | 将神经网络单元数据组合成矩阵 |
| **66** | plotwb | 绘制重量和偏差值的Hinton图 |
| **67** | purelin | 线性传递函数 |
| **68** | randtop | 随机层拓扑函数 |
| **69** | poslin | 正线性传递函数 |
| **70** | gnegate | 广义否定 |
| **71** | sumsqr | 矩阵的平方元素之和 |
| **72** | sse | 和平方误差性能函数 |
| **73** | plotsom | 绘制自组织图 |
| **74** | perform | 计算网络性能 |
| **75** | numfinite | 神经网络数据中有限值的数量 |
| **76** | gmultiply | 广义乘法 |
| **77** | hardlims | 对称硬极限传递函数 |
| **78** | configure | 配置网络输入和输出，以最佳匹配输入和目标数据 |
| **79** | hextop | 六边形层拓扑函数 |
| **80** | minmax | 矩阵行的范围 |
| **81** | numsignals | 神经网络数据中的信号数 |
| **82** | nnsize | 神经数据元素、样本、时间步长和信号的数量 |
| **83** | gridtop | 网格层拓扑功能 |
| **84** | randnr | 标准化行权重初始化函数 |
| **85** | maxlinlr | 线性层的最大学习率 |
| **86** | linearlayer | 创建线性层 |
| **87** | plotpv | 绘制感知器输入/目标向量 |
| **88** | [timedelaynet](https://www.mathworks.com/help/deeplearning/ref/timedelaynet.html) | 时延神经网络 |
| **89** | [narxnet](https://www.mathworks.com/help/deeplearning/ref/narxnet.html) | 具有外部输入的非线性自回归神经网络 |
| **90** | [narnet](https://www.mathworks.com/help/deeplearning/ref/narnet.html) | 非线性自回归神经网络 |
| **91** | [layrecnet](https://www.mathworks.com/help/deeplearning/ref/layrecnet.html) | 层递归神经网络 |
| **92** | [distdelaynet](https://www.mathworks.com/help/deeplearning/ref/distdelaynet.html) | 分布式延迟网络 |
| **93** | [gensim](https://www.mathworks.com/help/deeplearning/ref/gensim.html) | 生成Simulink块以进行浅层神经网络仿真 |
| **94** | [removedelay](https://www.mathworks.com/help/deeplearning/ref/removedelay.html) | 消除对神经网络响应的延迟 |
| **95** | [closeloop](https://www.mathworks.com/help/deeplearning/ref/closeloop.html) | 将神经网络开环反馈转换为闭环 |
| **96** | [plotinerrcorr](https://www.mathworks.com/help/deeplearning/ref/plotinerrcorr.html) | 绘制输入到错误时间序列互相关 |
| **97** | [ploterrcorr](https://www.mathworks.com/help/deeplearning/ref/ploterrcorr.html) | 绘制误差时间序列的自相关 |
| **98** | [getsiminit](https://www.mathworks.com/help/deeplearning/ref/getsiminit.html) | 获取Simulink神经网络块的初始输入和层延迟状态 |
| **99** | preparets | 为网络模拟训练准备输入和目标时间序列数据 |
| **100** | prune | 删除大小为零的神经输入、层和输出 |
| **101** | numtimesteps | 神经网络数据中的时间步数 |
| **102** | prunedata | 修剪数据以保持与修剪网络的一致性 |
| **103** | tapdelay | 抽头延迟的移位神经网络时间序列数据 |
| **104** | nncorr | 神经网络时间序列之间的互相关 |
| **105** | [dims](https://www.mathworks.com/help/deeplearning/ref/dlarray.dims.html) | dlarray的尺寸标签 |
| **106** | [finddim](https://www.mathworks.com/help/deeplearning/ref/dlarray.finddim.html) | 查找具有指定标签的尺寸 |
| **107** | [extractdata](https://www.mathworks.com/help/deeplearning/ref/dlarray.extractdata.html) | 从dlarray提取数据 |
| **108** | clearCache | 清除加速深度学习功能跟踪缓存 |
| **109** | next | 从minibatchqueue获取下一个小批量数据 |
| **110** | confusionmat | 计算分类问题的混淆矩阵 |
| **111** | Layer | 用于深度学习的网络层 |
| **112** | SeriesNetwork | 深度学习系列网络 |
| **113** | compile | 编译工作流对象 |
| **114** | average | 计算多类问题中平均接收机工作特性（ROC）曲线的性能指标 |
| **115** | calibrate | 使用已知水平的校准音校准仪表 |

# 总体设计

深度学习工具箱V1版本已实现功能函数100个，V2版本拟扩充功能函数至215个，深度学习工具箱V2可使用的功能模块包括：

1. 图像深度学习
2. 时序深度学习
3. 预训练网络
4. 神经网络训练
5. 函数逼近与聚类
6. 深度学习功能加速
7. 神经网络框架切换
8. 深度学习代码生成

## 总体架构

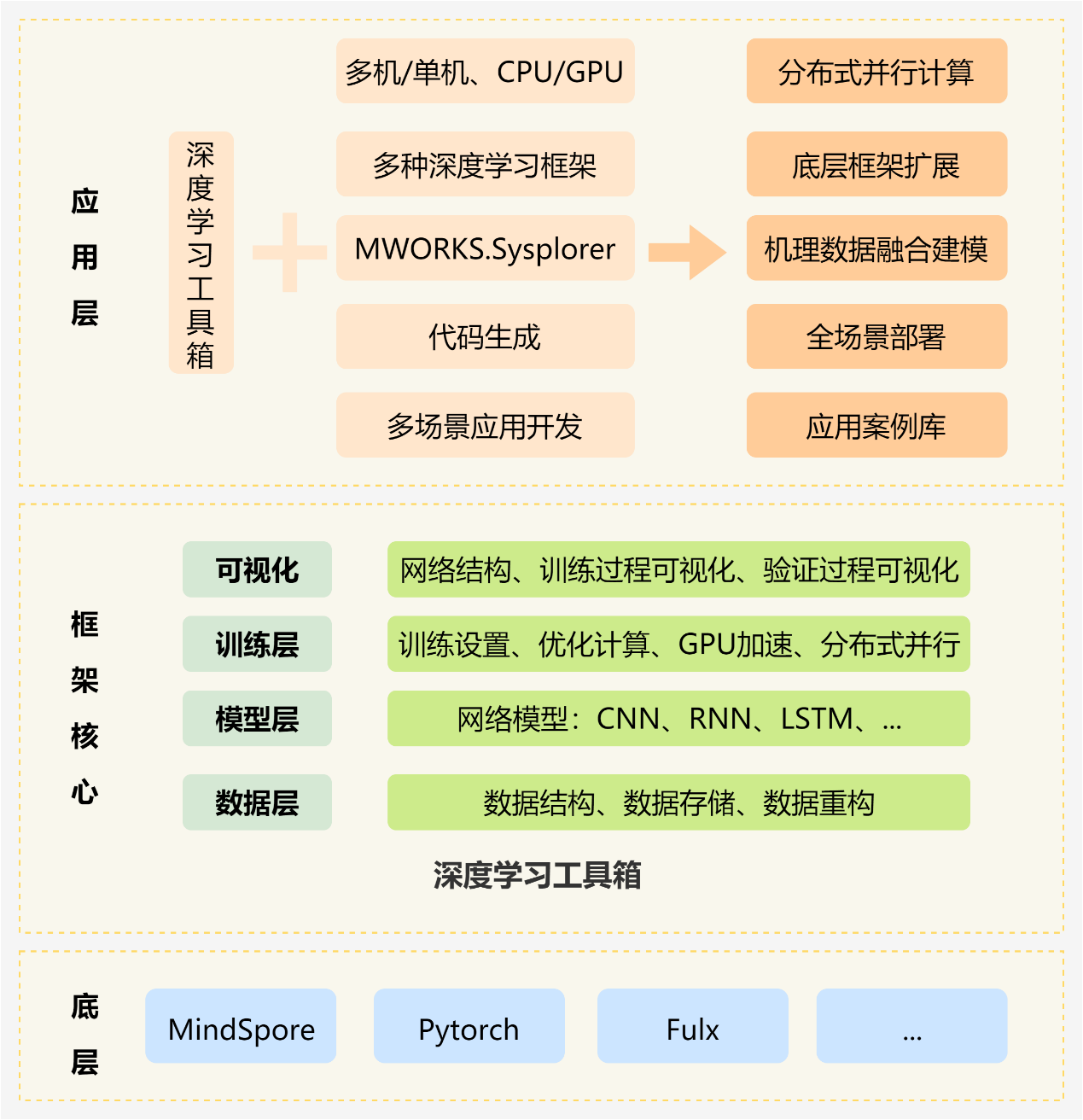


图3.1 深度学习工具箱总体架构图

1. 应用层

深度学习工具箱在已有版本的基础上，进一步扩充功能，集成MindSpore GPU以及分布式并行计算能力，提升性能，实现Sysplorer模型仿真数据端到端的网络训练，凭借代码生成功能完成全场景部署，并进一步开发多场景下的应用案例，完成应用案例库的搭建。

1. 框架核心
2. 深度学习工具箱依赖MindSpore、Pytorch框架开发，在模型层、训练层、可视化层与MindSpore、Pytorch深度结合。
3. 工具箱采用同MindSpore、Pytorch相同的应用流程，从数据层的采集到模型层的模型构建，再到训练层的模型训练，最后通过可视化层和应用层将模型可视化和部署。
4. 底层

底层框架可扩展，凭借Julia语言的兼容性，使得深度学习工具箱可以切换多种深度学习框架建模、训练。

## 总体方案

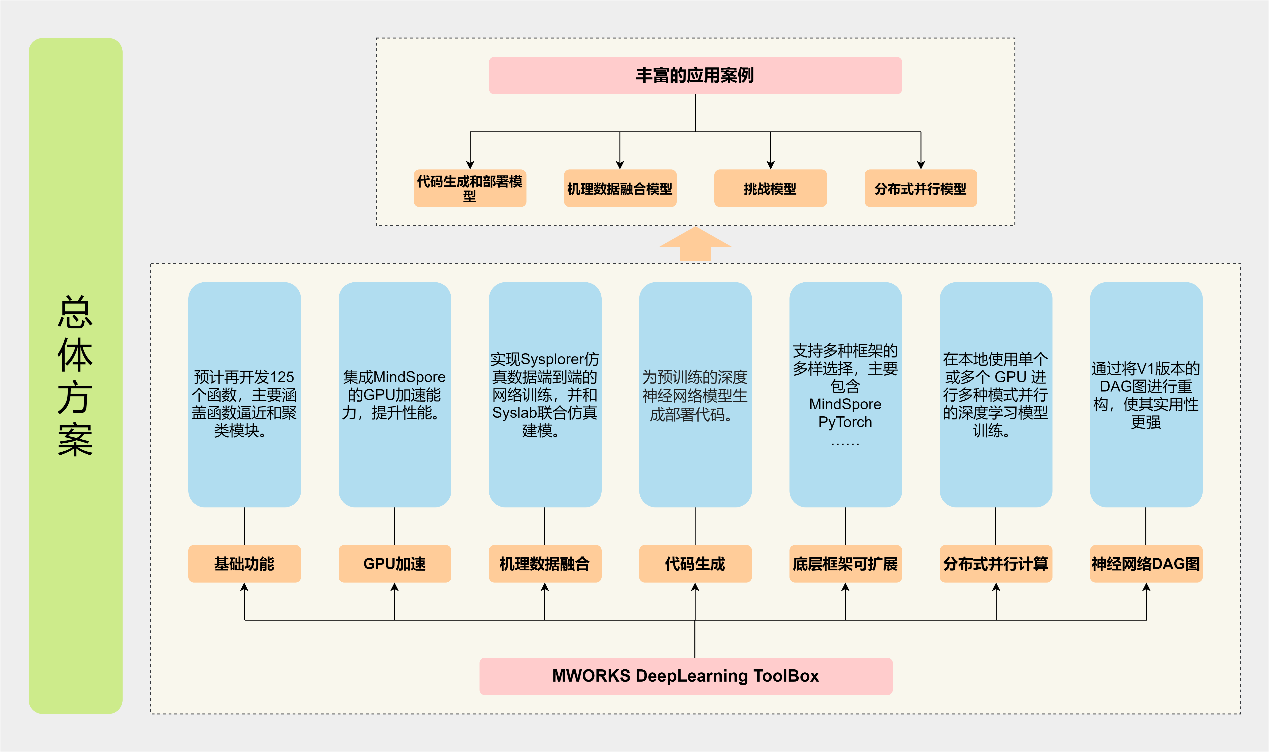


图3.2 深度学习工具箱总体方案图

深度学习工具箱主要包含七大技术路线：基础功能函数扩充、GPU运算加速、与Sysplorer相结合的机理数据融合建模、应用于全场景部署的代码生成、多主机/GPU硬件环境下的分布式并行计算、基于Julia API的底层框架扩展、神经网络DAG图重构。

## 应用流程

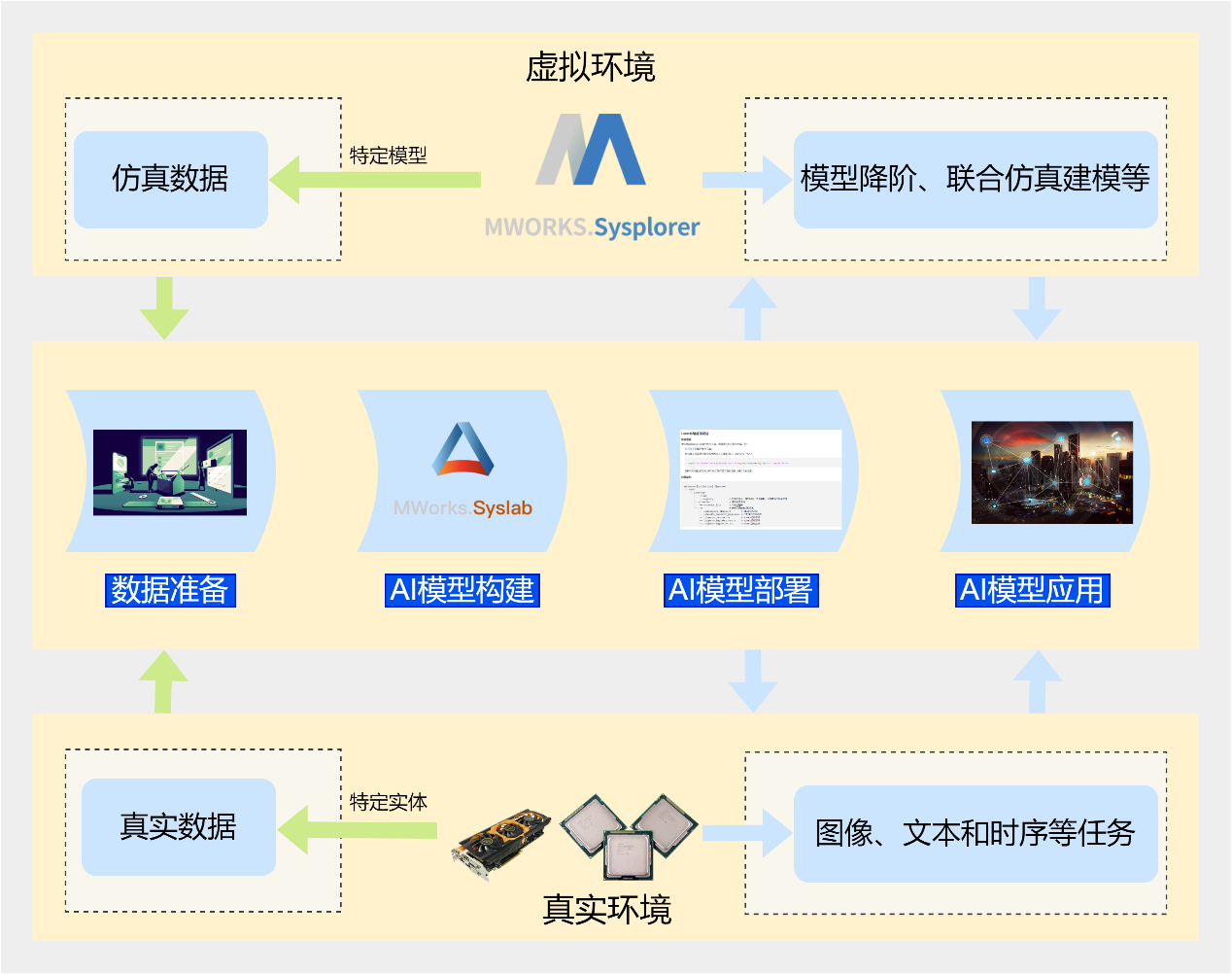


图3.3 深度学习工具箱应用流程图

深度学习工具箱应用分为以下四步：

1. 数据准备。该阶段数据获取的来源可分为虚拟仿真数据和真实数据，虚拟仿真数据从MWORKS.Sysplorer特定数学物理模型中产生，真实数据从特定实体中产生，例如汽车发动机、飞机舵系统、现实图像声音等。
2. AI模型构建。从各方获得的数据经过其他工具箱（机器学习工具箱、基础工具箱等）的数据清洗等预处理手段，导入构建好的模型进行训练、调参等工作后，将模型进行保存以待后用。
3. AI模型部署。根据部署环境的不同，对所需部署的模型参数进行设置使其能契合当前环境。可以考虑两大块部署环境，一是部署在虚拟环境，即MWORKS.Sysplorer，通过模型降阶等工作将数据模型和数学物理模型进行联合建模，实现完整系统或模块的建模仿真工作；另一是部署在真实环境，即端边云，通过以上设备即可实现训练好的模型直接推理，应用更为便利。
4. AI模型应用。一方面通过在虚拟环境部署，实现虚拟影射现实的前瞻性应用；另一方面通过在真实环境部署，实现现实应用更为便捷，效果更为直观的能力。亦或是可将两者结合实现数字孪生，现实与虚拟互为表里，实现智能生活、智能工业等一系列AI愿景。

## 预期效果

本产品以MWORKS中的工具箱的形式存在，功能亮点包括但不限于：

1. 相较于V1版本，基础功能更为完善，支持更丰富的函数库，更多的神经网络和预训练模型，更多的训练参数设置，更美观的训练结果展示，更便捷的模型导入导出功能。
2. 神经网络DAG图更为直观，可以更清晰地展现模型构建的网络结构，增删网络层更为方便。
3. 支持GPU加速运算，运算能力和速度大为提升，可以支持训练规模更大、数据量更多的模型。
4. 支持与MWORKS.Sysplorer双向建模仿真，初步实现物理数学模型与数据模型的机理数据融合建模。
5. 支持代码生成，实现Syslab生成的模型全场景部署，不仅是现实环境，也可以是虚拟的建模环境。
6. 支持底层框架可扩展，实现MindSpore框架和PyTorch框架的自由切换，而不改变函数和参数配置，接口统一，语法一致。
7. 支持分布式并行计算，实现多卡、单卡的分布式并行训练，参数设置浅显易懂，训练效率大大提升。
8. 支持更丰富的应用案例，传统场景案例增加，新增工业拓展案例，故障预测、故障诊断、剩余寿命预测等案例增加。

# 技术路线

## 基础功能

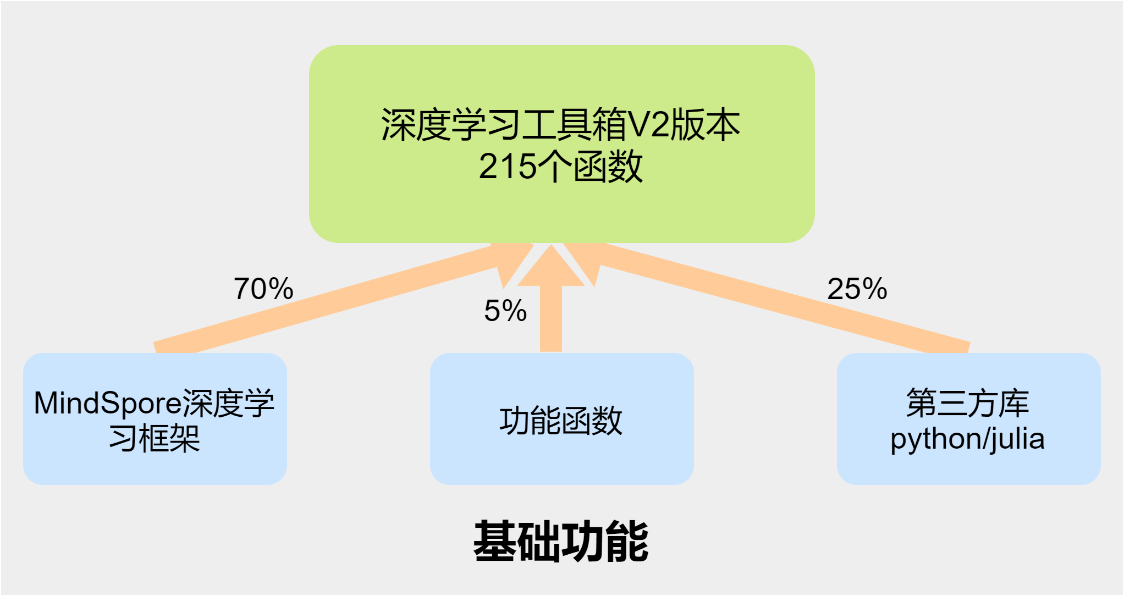


图4.1 深度学习工具箱基础功能图

深度学习工具箱V1版本已实现功能函数100个，V2版本拟扩充功能函数至215个。其中，使用MindSpore深度学习框架的函数占比约70%，功能函数约占5%，使用第三方库的函数约占25%。

## 神经网络DAG图

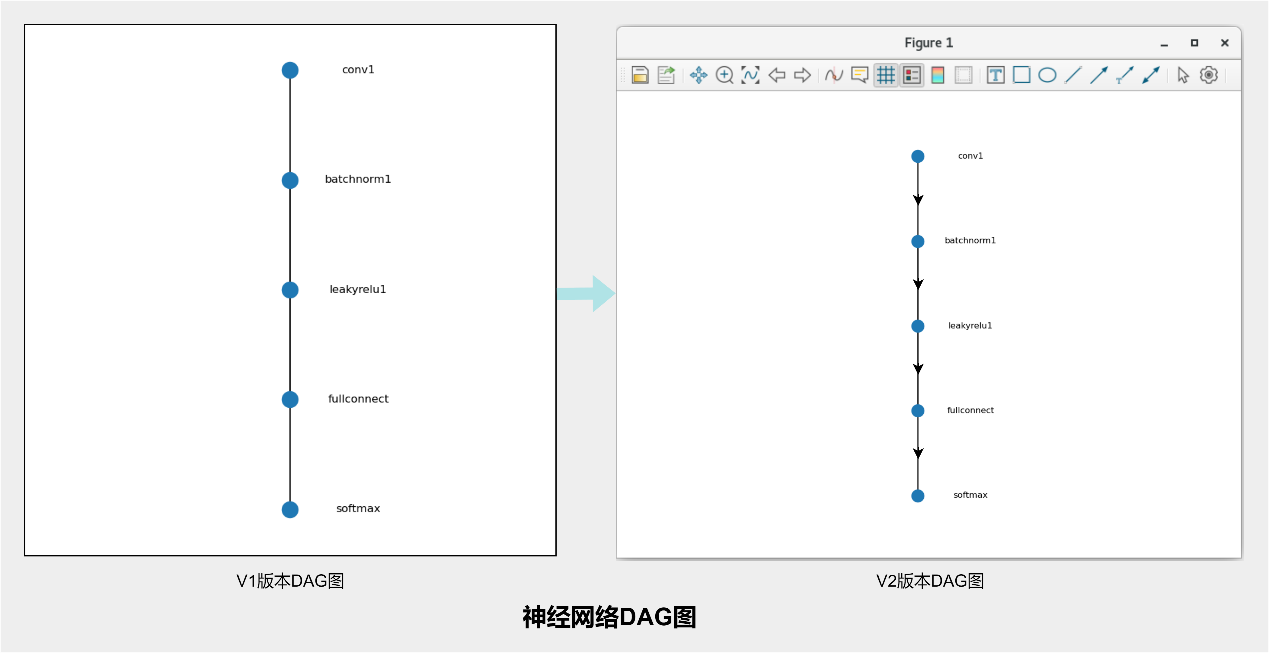


图4.2 深度学习工具箱神经网络DAG图

V1版本的神经网络DAG图采用Python已有的工具函数进行绘制，无法清楚地表示网络的输入输出。V2版本将进行功能改进，结合Syslab的TyPlot绘图工具箱和Netron完成更为复杂的DAG图展示，方便清晰地展示网络结构、输入输出。

## 机理数据融合

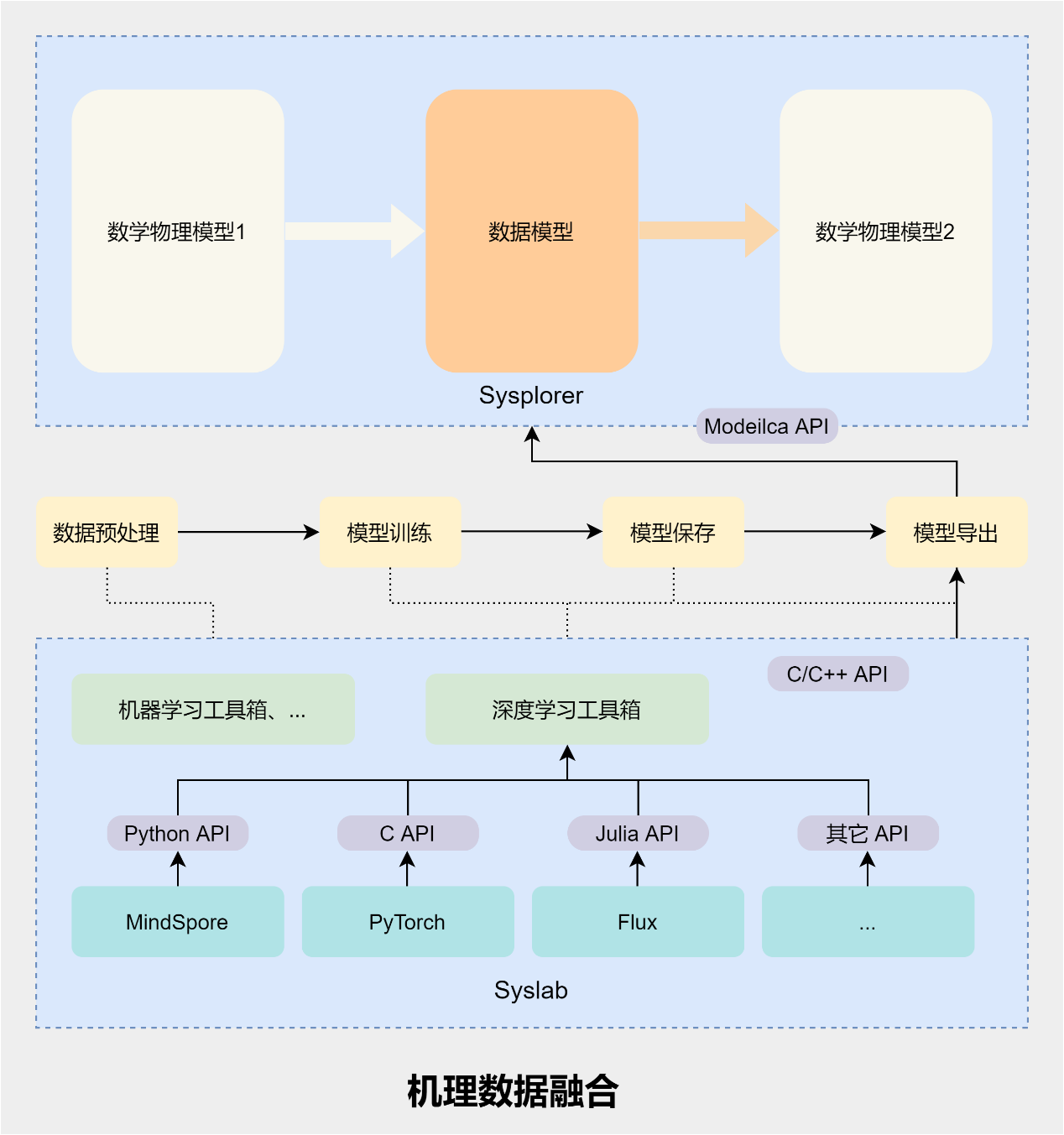


图4.3 机理数据融合建模技术路线图

通过在Syslab平台上的数据预处理、模型训练、模型保存、模型导出操作将已训练好的深度学习模型保存并导出，再通过Sysplorer平台的Modelica API导入模型并调用，实现使用数据模型与物理模型的融合建模。

## 分布式并行计算

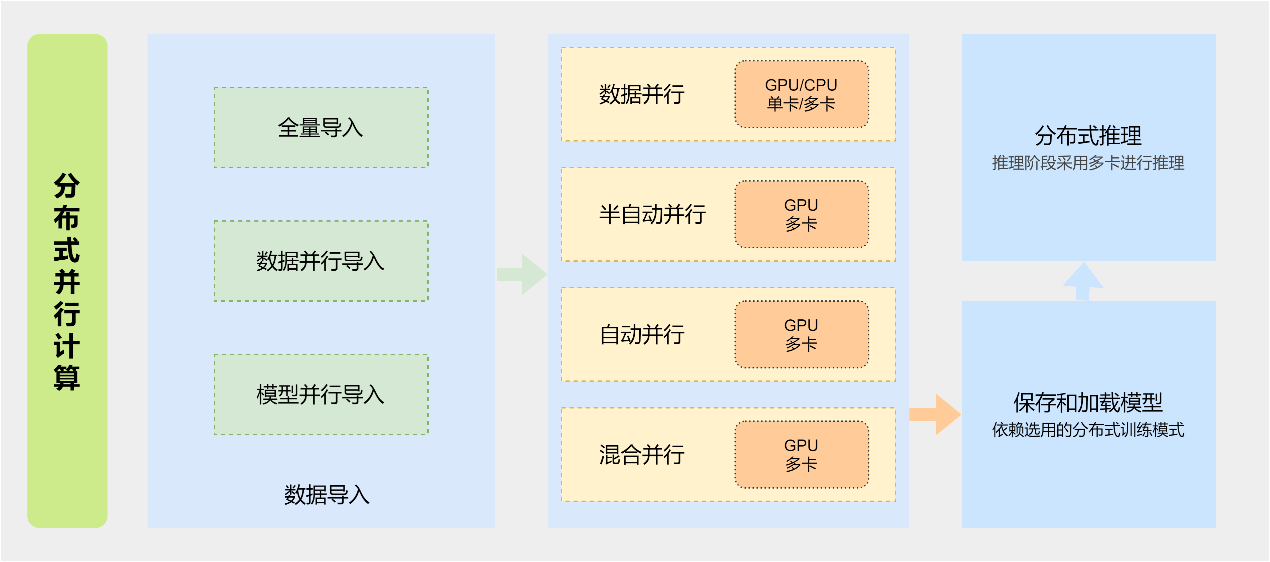


图4.4 分布式并行计算技术路线图

在分布式并行训练中，支持三种数据的导入方式：

1. 全量导入。仅在半自动和全自动并行模式下生效。用户可以通过set\_auto\_parallel\_context(full\_batch=True)开启。开启全量导入之后，在自动并行流程中认为读入的batch是一个网络输入的完整shape。例如，在8卡训练的情况下，假设每张卡dataset返回的shape是[32, 8]，那么当前一个迭代训练的训练的数据即为[32, 8]。因此，用户需要保证每卡在每轮迭代输入的数据是一致的。例如，确保每卡数据集的shuffle的顺序是一致的。
2. 数据并行导入。用户不设置 full\_batch的情况下，每卡读入的数据是当前训练迭代的一个分片，因此要求每卡读入的数据内容不一样。例如8卡训练的情况下，每卡读入数据的shape为[32,8]，那么当前一个迭代训练的数据总量为[32\*8, 8]。
3. 模型并行导入。模型并行导入的方式主要针对图像领域中图像尺寸太大无法在单卡进行计算时，直接在输入流程上就对图像进行切分。

数据导入后，可以采取四种并行计算模式：

1. 数据并行：用户的网络参数规模在单卡上可以计算的情况下使用。这种模式会在每卡上复制相同的网络参数，训练时输入不同的训练数据，适合大部分用户使用。
2. 半自动并行：用户的神经网络在单卡上无法计算，并且对切分的性能存在较大的需求。用户可以设置这种运行模式，手动指定每个算子的切分策略，达到较佳的训练性能。
3. 自动并行：用户的神经网络在单卡上无法计算，但是不知道如何配置算子策略。用户启动这种模式，MindSpore会自动针对每个算子进行配置策略，适合想要并行训练但是不知道如何配置策略的用户。
4. 混合并行：完全由用户自己设计并行训练的逻辑和实现，用户可以自己在网络中定义AllGather等通信算子。适合熟悉并行训练的用户。

## 代码生成

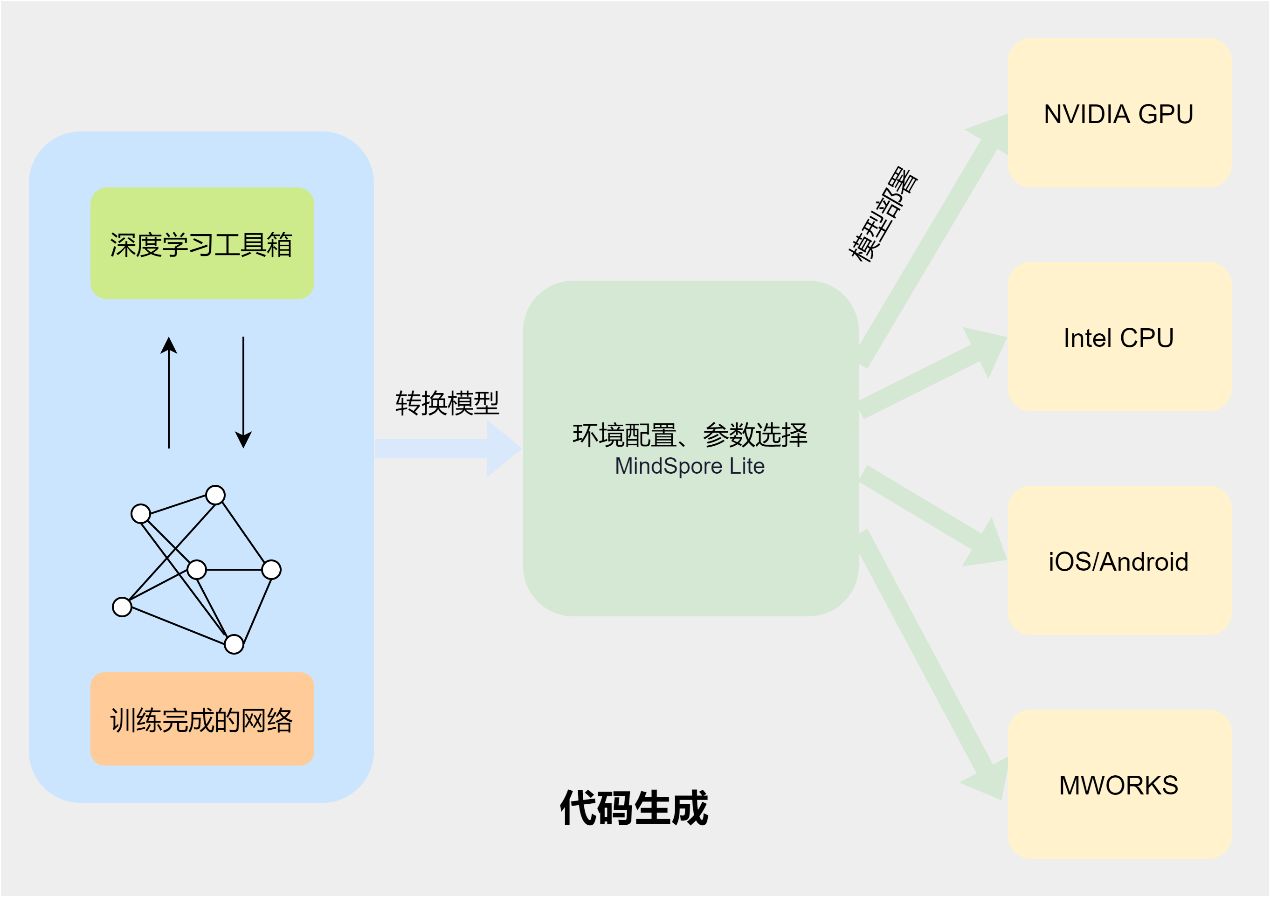


图4.5 代码生成技术路线图

代码生成功能将深度学习工具箱已训练完成的模型转换为可在轻量化深度学习框架运行推理的模型，并可将轻量化框架部署到多场景运行环境中，为用户提供端到端的解决方案。

## 底层框架可扩展

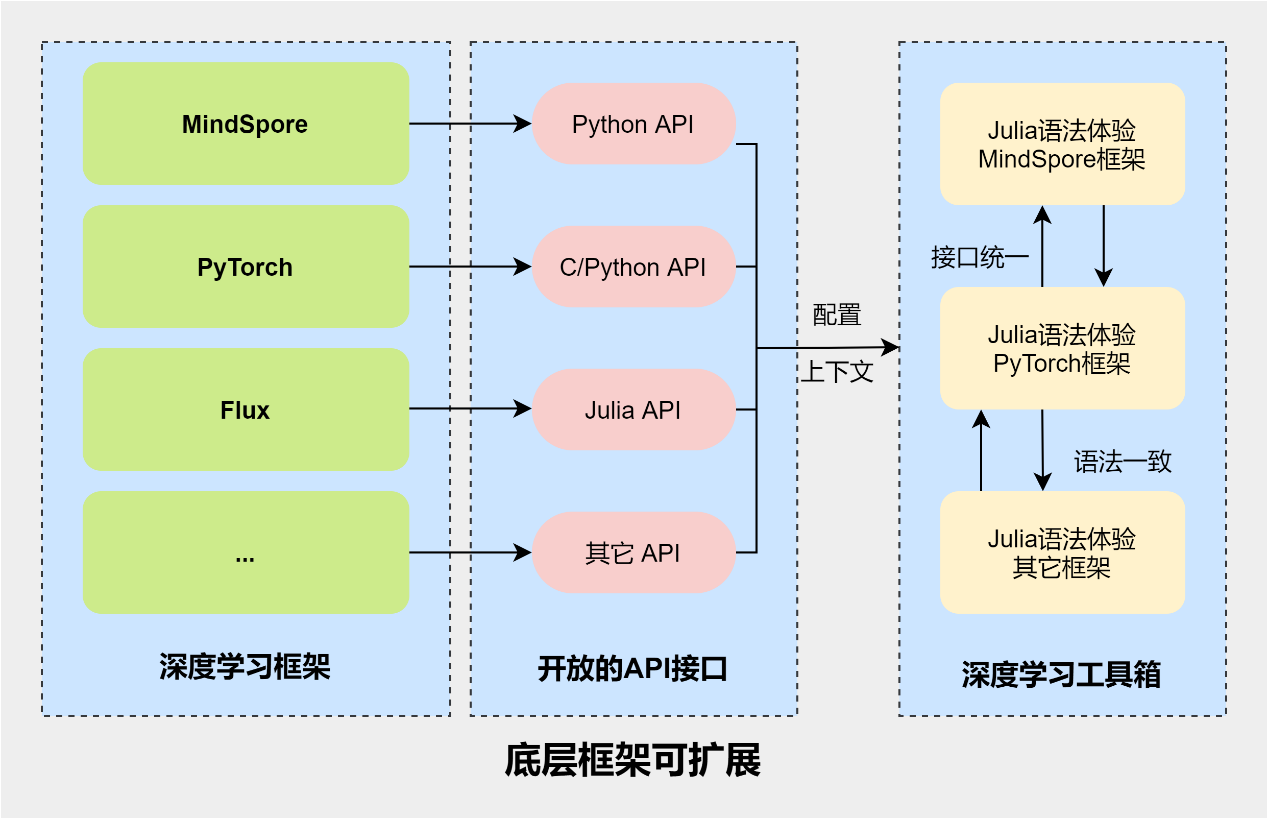


图4.6 底层框架可扩展技术路线

深度学习工具箱基于Julia语言开发，由于Julia语言具有优异的兼容性，可与多种语言相互调用，所以目前常用的深度学习框架工具箱皆可兼容。用户可通过配置上下文管理进行深度学习框架的随意切换。

## GPU加速

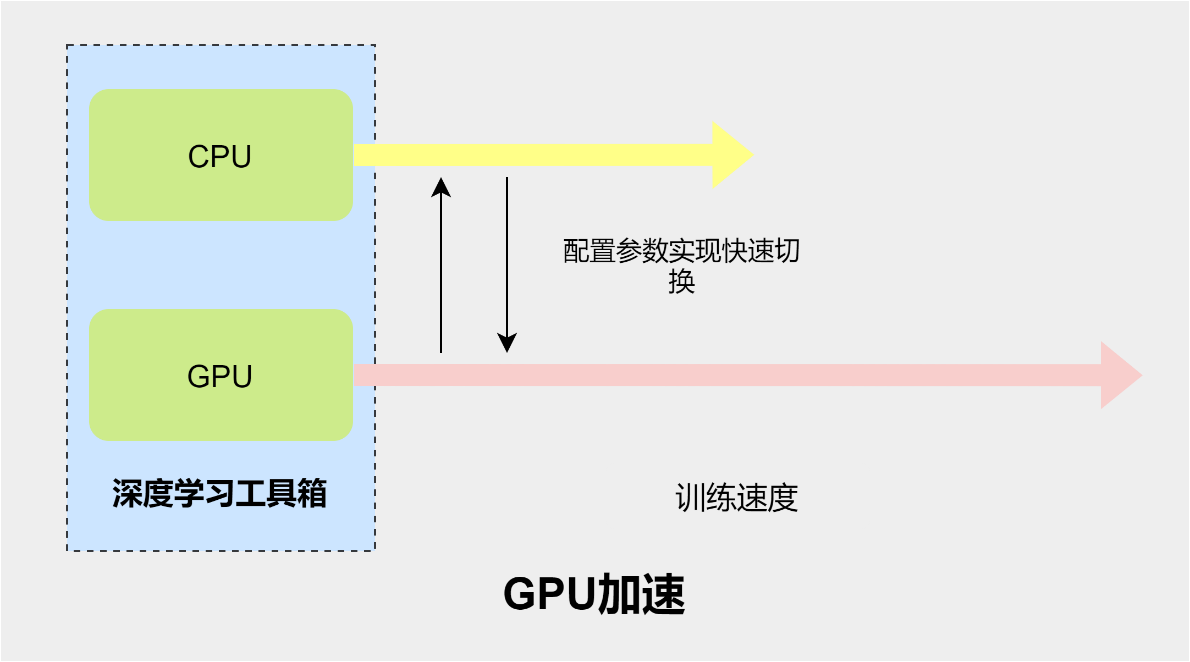


图4.7 GPU加速运算技术

根据客户端硬件环境，通过配置参数实现切换不同的处理器，在GPU环境下可极大的加快模型的训练速度。

# 验证方案

## 性能验证

Syslab深度学习工具箱性能验证主要通过两个维度进行对比：

1. 与主流深度学习框架对比

相同神经网络在Syslab深度学习工具箱与主流深度学习框架以相同的数据、参数进行训练，对其求解结果进行统计和分析，包括精度和速度等方面。

1. 与Mtalab对比

Syslab 深度学习工具箱V2版本与 Matlab 进行横向性能比较。

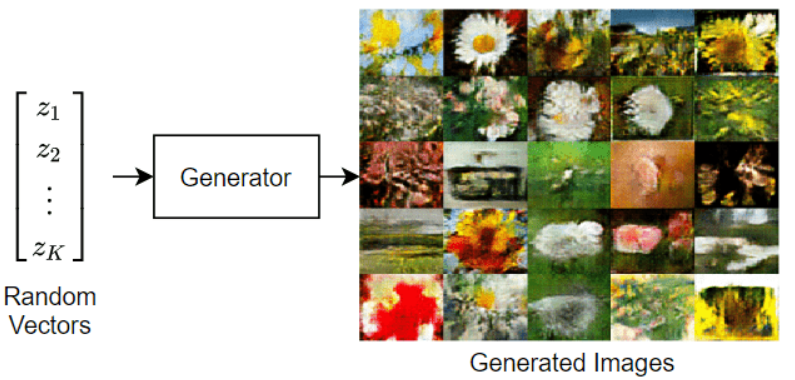
## 应用验证

在图像处理、音频处理、文本分析、复杂系统预测性维护、机理数据融合建模、模型降阶等领域开展应用研究，开发相应的应用案例。此外，构建一套包含图像识别、目标检测、机器人机理数据融合建模技术的综合展示物理系统。

### 图像处理

#### 使用GAN网络生成图像

生成式对抗网络 (GAN) 是一种深度学习网络，它能够生成与真实输入数据具有相似特征的数据。此示例说明如何训练生成对抗网络来生成图像，原理过程如下图所示：



#### 使用深度学习对多光谱图像进行语义分割

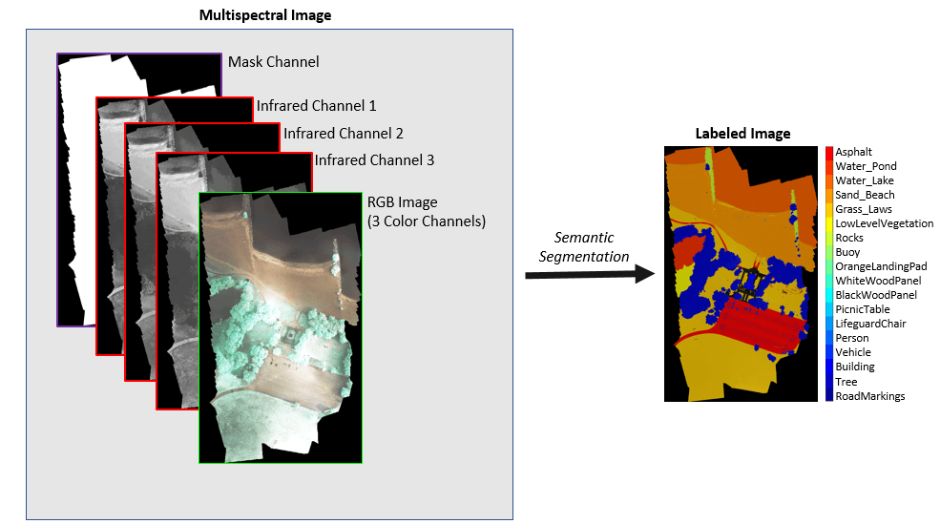


图5.1 Hamlin Beach 州立公园森林覆盖图

此示例说明如何使用 U-Net 对包含七个通道的多光谱图像执行语义分割。

语义分割涉及用类标注图像中的每个像素。语义分割的一个应用是跟踪森林采伐，即森林植被随时间的变化。环保机构需要跟踪森林采伐，以评估和量化某个地区的环境和生态健康。

基于深度学习的语义分割可以通过高分辨率航拍照片精确测量植被覆盖度。一个挑战是区分具有相似视觉特性的类，例如尝试将绿色像素分类为草、灌木或树。为了提高分类准确度，一些数据集包含多光谱图像，这些图像提供关于每个像素的附加信息。例如，Hamlin Beach 国家公园数据集用三个近红外通道为彩色图像补充信息，以提供更清晰的分类。

此示例预计使用深度学习工具箱中的二维U-Net预训练网络进行训练。

#### 使用深度学习进行三维脑肿瘤分割

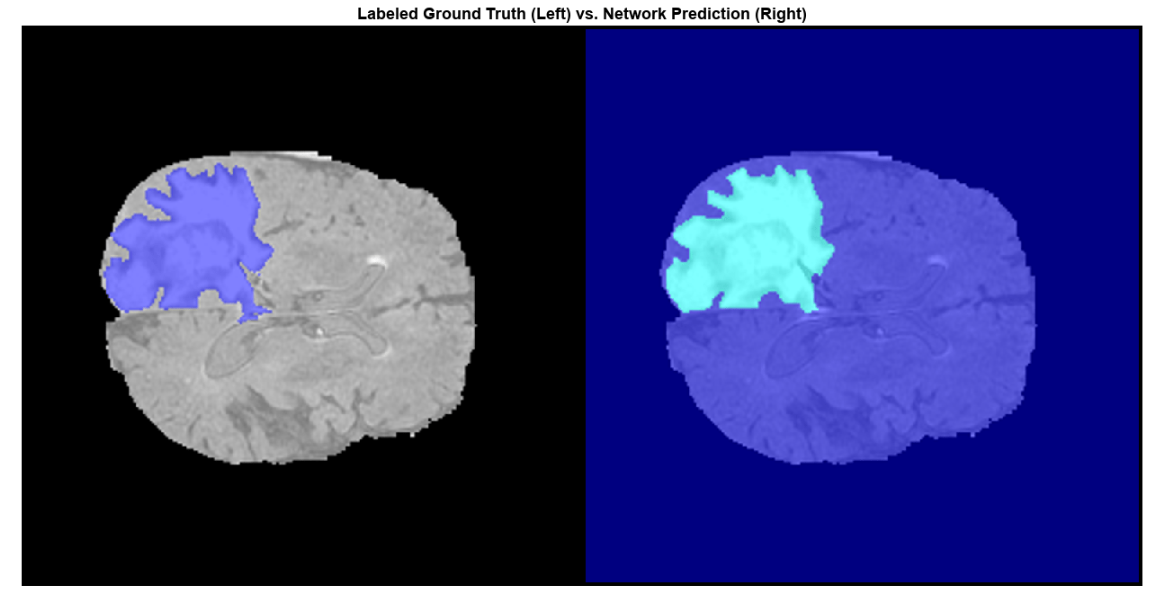


图5.2 三维医学脑肿瘤图像

此示例说明如何基于三维医学图像执行脑肿瘤的语义分割。

语义分割中，图像中的每个像素或三维体的每个体素都被标注为某一类。此示例说明如何在磁共振成像 (MRI) 扫描中使用三维 U-Net 深度学习网络执行脑肿瘤二元语义分割。U-Net 是一种快速、高效、简单的网络，在语义分割领域 非常流行。

医学图像分割面临的另一挑战是当使用常规的交叉熵损失时，数据中的类不平衡会妨碍训练。此示例通过使用加权多类 Dice 损失函数解决此问题。对类进行加权有助于抵消较大区域对 Dice 分数的影响，使网络更容易学习如何分割较小区域。

此示例预计使用深度学习工具箱中的三维U-Net预训练网络和加权多类 Dice 损失函数进行训练。

#### 使用深度学习增加图像分辨率

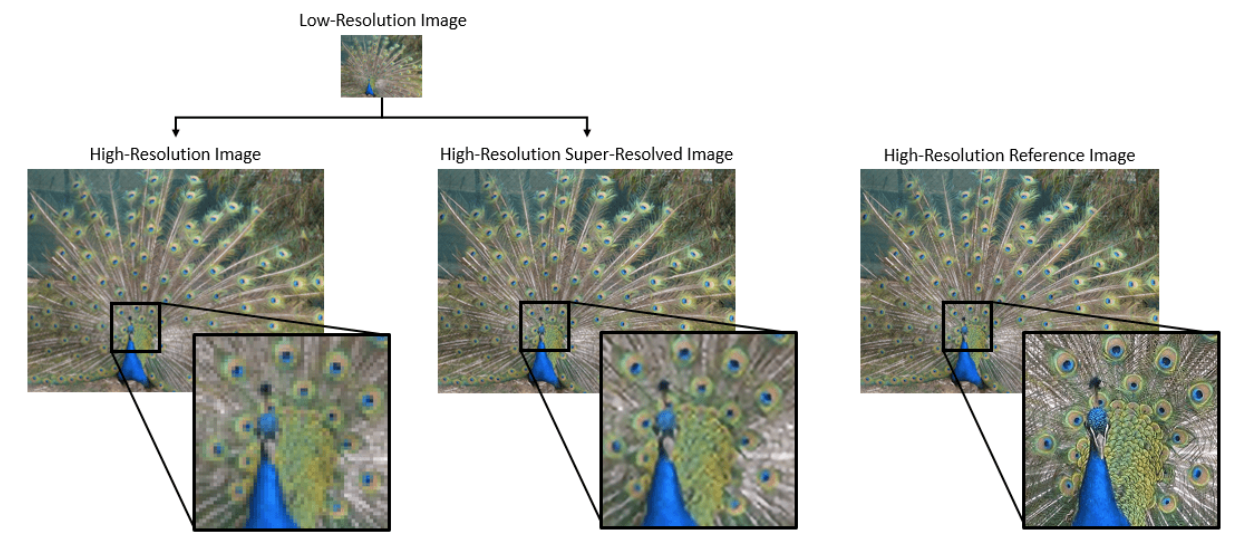


图5.3 静态自然图像

此示例说明如何使用超深超分辨率 (VDSR) 神经网络从低分辨率图像创建高分辨率图像。

VDSR 是一种用于执行单图像超分辨率的卷积神经网络架构。VDSR 网络学习低分辨率图像和高分辨率图像之间的映射。这种映射是有可能获得的，因为低分辨率和高分辨率图像具有相似的图像成分，主要区别在于高频细节。

VDSR 采用一种残差学习策略，这意味着网络会学习估计残差图像。在超分辨率中，残差图像是高分辨率参考图像和低分辨率图像之间的差异，低分辨率图像已使用双三次插值扩增以匹配参考图像的大小。残差图像包含关于图像高频细节的信息。

此示例预计使用深度学习工具箱中的 41 个单独层定义 VDSR 网络，这些层包括：

* convolution2dLayer - 卷积神经网络的二维卷积层
* reluLayer - 修正线性单元 (ReLU) 层

#### 基于神经网络的海上杂波去除

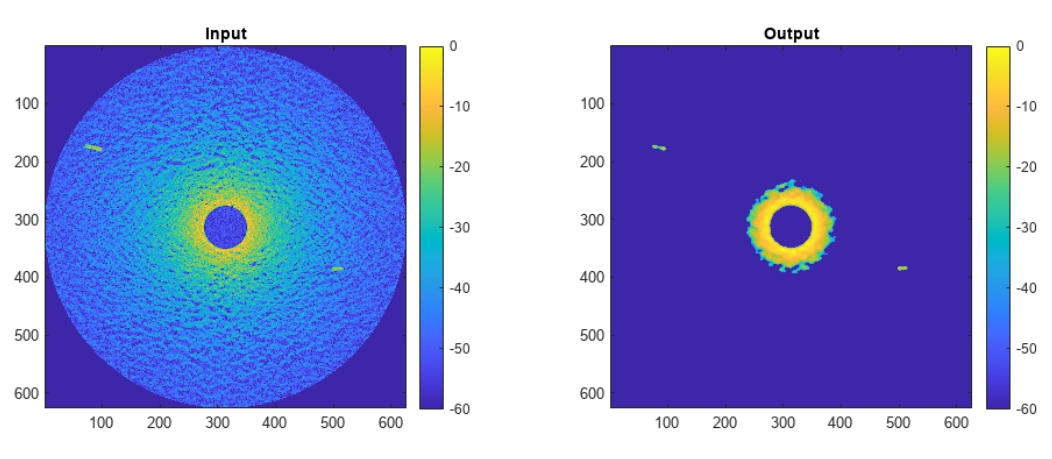


图5.4 合成雷达图像

此示例展示了如何训练和评估卷积神经网络，以使用深度学习工具箱从海上雷达PPI图像中去除杂波回波。

数据集包含84对合成雷达图像。每一对都由一个输入图像和一个期望的响应图像组成，输入图像既有海杂波，也有扩展的目标回波。使用雷达收发机和旋转均匀线性阵列（ULA）的雷达场景模拟创建图像。每个图像包含两个不重叠的扩展目标，一个表示小型集装箱船，另一个表示大型集装箱船。这些船是由长方体表面上的一组点散射体建模的。

此示例预计使用深度学习工具箱中的以下网络层，这些层包括：

* convolution2dLayer - 卷积神经网络的二维卷积层
* reluLayer - 修正线性单元 (ReLU) 层
* batchNormalizationLayer – 批标准化层

### 音频处理

#### 基于音频的机器健康监测异常检测

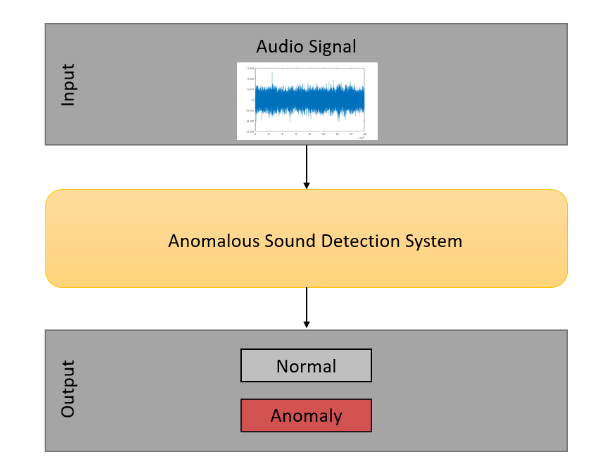


图5.5 机器声音异常检测流程

此示例显示了如何设计自动编码器神经网络，以使用无监督学习对机器声音执行异常检测。在本例中，您将使用对数梅尔谱图下载和处理数据，设计和训练自动编码器网络，并通过将统计模型应用于训练的网络输出进行样本外预测。

基于音频的异常检测是识别物体产生的声音是否异常的过程。这适用于工业部件故障的自动检测，因为发出异常声音的机器可能发生故障。

将声音分类为正常或异常的问题可以被视为一个标准的监督学习任务，其中模型在两种声音类型的样本上进行训练，并学习区分它们。然而，在实践中，通常无法获得异常声音的数据集，因为机器故障发生的频率不够高或持续时间不够长，无法正确记录。此外，不可能创建代表每种异常类型的数据集，因为机器可能会因各种原因而故障。

自动编码器对于异常检测任务非常有用，因为它们仅对正常样本进行训练。自动编码器网络执行无监督学习任务，即 找到输入的低维编码以及从其低维表示精确重构输入的规则。这迫使自动编码器学习专门用于压缩和解压缩正常样本的过程。激励原理是，当异常样本被馈送到自动编码器中时，重建误差将比训练集的预期大得多，因为网络学习的信号压缩和解压缩方案仅预期对正常样本有效。为了对不可见样本进行预测，基于正常样本的重建误差的预期分布来选择误差阈值，并且任何误差大于阈值的输入都被分类为异常。

在此示例中，自动编码器首先使用与输入维度相同数量级的多个节点将输入通过完全连接的层的编码部分。然后，数据馈送到瓶颈层，该瓶颈层具有比输入大小小得多的节点，这迫使网络将输入信号压缩成低维表示。该压缩表示馈送到解码部分，该解码部分通常镜像与编码器部分相同的架构。最后，解码器输出被传递到具有与输入相同维数的最终输出层。网络损耗被视为原始输入和重建信号之间的回归误差。

此示例中编码器部分由2个完全连接的层组成，输出大小为128。瓶颈层将网络约束为原始640维输入的8维表示。当输入被重构并馈送到输出层时，解码器部分镜像编码器架构。使用半均方误差作为损失函数来训练网络并量化重建误差。

### 文本分析

#### 使用深度学习对文本数据进行分类

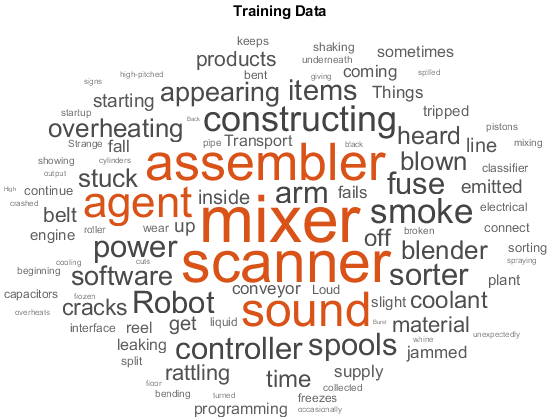


图5.6 词云

此示例说明如何使用深度学习长短期记忆 (LSTM) 网络对文本数据进行分类。

文本数据本身就是有序的。一段文本是一个单词序列，这些单词之间可能存在依存关系。要学习和使用长期依存关系来对序列数据进行分类，请使用 LSTM 神经网络。LSTM 网络是一种循环神经网络 (RNN)，可以学习序列数据的时间步之间的长期依存关系。

要将文本输入到 LSTM 网络，首先将文本数据转换为数值序列。您可以使用将文档映射为数值索引序列的单词编码来实现此目的。为了获得更好的结果，还要在网络中包含一个单词嵌入层。单词嵌入将词汇表中的单词映射为数值向量而不是标量索引。这些嵌入会捕获单词的语义细节，以便具有相似含义的单词具有相似的向量。它们还通过向量算术运算对单词之间的关系进行建模。

此示例预计使用深度学习工具箱中的以下网络层和函数，包括：

* wordEncoding - 单词编码
* doc2sequence - 文档转换为数值索引序列
* wordEmbeddingLayer – 词嵌入层
* lstmLayer – LSTM层
* fullyConnectedLayer – 全连接层
* softmaxLayer – softmax层

### 复杂系统预测性维护

针对复杂装备与系统，开展基于深度学习的故障诊断、故障预警以及剩余使用寿命预测等研究，开发相关应用示例。

#### 汽车剩余使用寿命预测

乘用车作为一个零部件众多、系统间耦合复杂的系统，其可能发生的故障种类众多且形态复杂。乘用车系统的故障预测对于降低风险、提升经济性具有重要的意义。由于乘用车系统的生命周期较长、整车价格较为昂贵，使用物理试验的方式获取乘用车系统全生命周期运行数据的可行性较差。本案例将研究一种乘用车系统全生命周期故障仿真方法，在系统全生命周期运行过程中，通过对其部件注入时变故障的方式，仿真获取乘用车系统全生命周期下的性能退化运行数据，并使用该数据训练乘用车系统故障预测模型。

#### 航天器舵系统故障诊断

舵系统是飞行器控制系统的执行机构，是不可缺少的重要部分，尤其是作为主要操纵执行器的升降舵和方向舵。舵系统的性能可靠性与飞行器整体的可靠性密切相关，一旦它们发生故障，轻则造成飞行器的稳定性能下降，出现虚警和乘员恐慌，重则直接导致飞行器坠毁，出现重大的人员财产损失和负面舆情。

本案例将通过故障建模及仿真的方式获得大量丰富的故障仿真样本，并在此基础上设计从大量仿真数据中迁移诊断知识到实际诊断场景的舵系统智能故障诊断模型，将此诊断模型应用于实际诊断问题中，验证故障诊断模型的有效性并对其诊断机理进行分析，为解决实际诊断场景中的小样本故障诊断问题提供新思路。

#### 液体火箭发动机故障预警

液体火箭发动机作为运载火箭系统飞行动力的核心动力，决定着运载火箭的整体性能，直接关系到航天任务的成败。液体火箭发动机是一个复杂的热-水动力系统，由许多不同的、相互独立的动态链接交叉耦合而成。工作环境往往是在高压、高温、高速气流烧蚀、推进剂成分侵蚀、高频振荡等极端恶劣条件下。在极端环境中，工作条件接近材料的物理极限，任何微小的异常都可能迅速发展为破坏性故障，导致发射任务失败，造成巨大的经济损失。本案例将研究基于卷积神经网络和LSTM的氢氧液体火箭发动机启动瞬态故障预测及预警方法。首先，利用MWorks对某大型氢氧火箭发动机进行建模，采用参数故障注入仿真发动机启动阶段的正常/故障运行状态。其次通过卷积神经网络（CNN）从输入数据中捕获局部特征，利用长短期记忆（LSTM）从特征中提取长期依赖关系，对潜在故障进行预测；最后，定位故障部件，实现发动机的故障预警。

### 机理-数据融合建模

#### 机器人动力学机理-数据融合建模

使用传统的深度学习，解决机器人的逆问题，构建机器人逆动力学的纯数据模型，可能会出现以下问题：就算训练数据满足所有的物理规律，训练好的人工神经网络仍然可能做出非物理的预测，外插无约束。此外，有一些场景很难用物理公式计算，比如：1）柔性关节，关节之间用弹簧或绳子连接，而弹性系数并无数值流体身躯；2）体内有半满的液体，内部质量分布随身躯的姿态而变化；3）摩擦力，关节之间存在静摩擦力与动摩擦力，基座与地面之间存在静摩擦力与动摩擦力等。

本案例将使用LSTM神经网络解决柔性关节与摩擦力不好理论计算的问题，构建机器人动力学系统的机理-数据融合模型，使得融合模型预测结果更物理，对新的径迹预测也更健壮。

### 模型降阶

#### 传动系统动力学模型降阶

降阶模型（ROM）可替换物理模型，它允许在不损害原始物理模型准确性的情况下减少所需计算量。对于需要大量计算的工作流，可以使用ROM代替原始物理系统。

本案例将通过训练长短期记忆神经网络（LSTM）来创建降阶模型，以替换Sysplorer传动系统模型中组件模型。

### 综合展示系统

搭建一套以工业机器人为主体的综合展示系统，基于机理-数据融合建模方法构建机器人数字孪生系统，模型仿真结果与实验测量结果实时对比。机器人末端安装摄像头，训练好的神经网络模型，通过代码生成，下载至摄像头终端，实现图像识别与目标检测。

展示要点：数字孪生、机理-数据融合建模、图像识别与目标检测

# 研发计划

## 人员安排

|  |  |  |
| --- | --- | --- |
| 工作内容 | 人员安排 | 备注 |
| 总体设计 | 梅再武 | 总体把控，人员招聘与管理 |
| 函数开发（基于MindSpore） | 胡如明 | 函数开发，测例搭建，撰写帮助文档 |
| 函数开发（基于PyTorch） |  | 暂缺 |
| 函数开发（基于Flux（julia）） |  | 暂缺 |
| 核心功能开发 | 曹宇宇 | GPU加速、分布式并行计算、代码生成等核心功能开发 |
| 基础应用案例 | 胡如明 | 图像、音频、文本等深度学习基础应用案例开发 |
| 预测性维护案例 | 曹宇宇 | 故障诊断、故障预测等方向案例 |
| 应用创新案例 | 梅再武 | 机理数据融合相关案例，探索深度学习在多领域建模领域的创新应用 |
| 展示系统搭建 | 梅再武，曹宇宇 | 调研、设计方案，若条件成熟搭建展示系统 |

## 进度计划

Syslab 深度学习工具箱产品路线图将深度学习工具箱研发分为四个阶段，第一阶段完成深度学习工具箱V2版本的设计方案。第二阶段在V1版本的基础上继续进行基础功能函数的开发，扩充至215个，并完成深度学习工具箱机理数据融合建模、GPU加速运算、分布式并行计算功能。第三阶段完成深度学习工具箱底层框架扩展、代码生成功能，并开始应用案例的开发，初步完成深度学习应用案例库的搭建。第四阶段是集成测试阶段，并搭建综合展示系统，完成115个功能函数与应用案例帮助文档撰写，完成测试并完成工具箱集成工作。

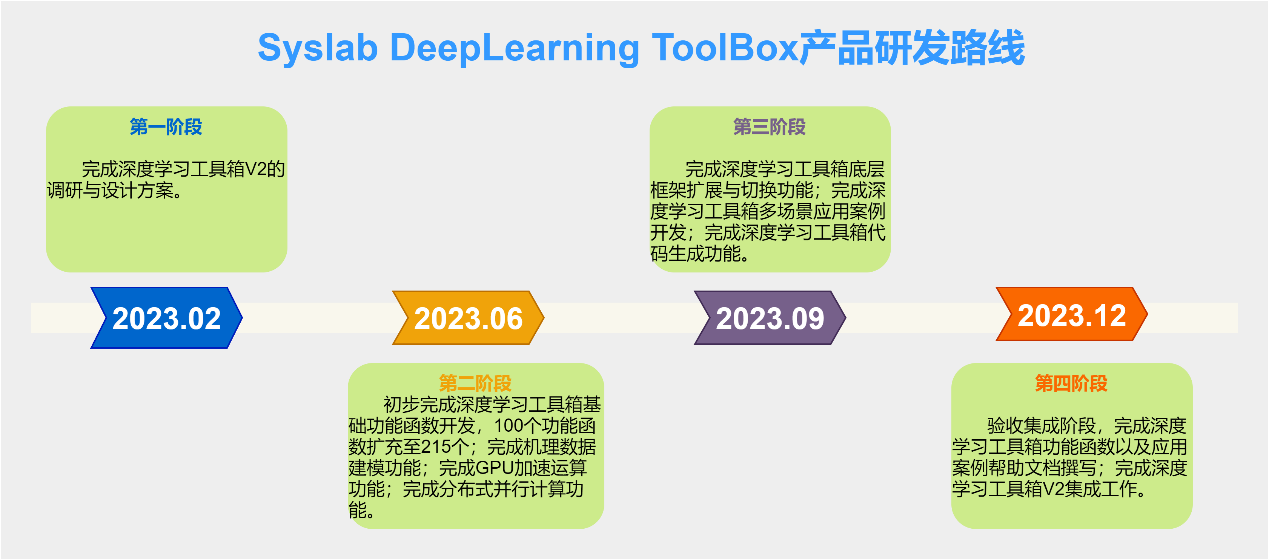


图6.1 深度学习工具箱进度计划图