# 基于项目驱动的Pytorch与计算机视觉入门学习计划

## 一、学习目标

通过 3 个月（约12周）的项目驱动学习，掌握 Pytorch 基础语法与核心功能，熟悉计算机视觉常见任务（图像分类、目标检测、图像分割等）的流程与方法，具备独立完成小型计算机视觉项目的实践能力，并通过 GitHub 记录学习过程与成果，形成完整的学习日志。

## 二、学习阶段与项目安排

### 第一阶段：基础搭建（第 1 - 2 周）

#### 项目：Pytorch 基础入门实践

* **学习任务**：
  1. 学习 Python 基础语法（变量、数据类型、控制流、函数等），若已有 Python 基础，可快速复习，预计 1 天。
  2. 安装 Pytorch 及相关库（如 torchvision），了解其安装环境配置，预计 1 天。
  3. 学习 Pytorch 张量（Tensor）的创建、操作（加减乘除、索引、切片等）、维度变换等知识，通过官方文档和简单示例练习，预计 2 天。
  4. 学习 Pytorch 自动求导机制（Autograd），理解计算图原理，掌握如何通过张量的requires\_grad属性实现自动求导，预计 2 天。
  5. 学习 Pytorch 神经网络模块（nn.Module）的使用，搭建简单的线性回归模型并进行训练，预计 2 天。
* **GitHub 学习日志记录**：
  1. **创建仓库**：在 GitHub 上创建名为pytorch-cv-learning的仓库，添加 README 文件，简要说明学习计划与目标。
  2. **每日记录**：在仓库中创建week1文件夹，每天学习结束后，在文件夹内创建以日期命名的 Markdown 文件（如[2025-06-05.md](http://2025-06-05.md" \t "_blank)），记录当天学习的知识点、遇到的问题及解决方法。例如：
     + 记录 Pytorch 张量的操作代码片段和理解心得；
     + 详细描述安装 Pytorch 过程中遇到的环境问题及解决步骤。
  3. **上传代码**：将当天编写的线性回归模型代码上传至week1文件夹，添加清晰的注释，并在日志文件中说明代码功能与实现思路。
* **成果**：完成一个基于 Pytorch 的线性回归模型，能够对数据进行训练和预测；GitHub 仓库中形成第一周的学习日志和代码记录。

### 第二阶段：图像分类入门（第 3 - 5 周）

#### 项目：MNIST 手写数字图像分类

* **学习任务**：
  + 学习计算机视觉基本概念，如图像的表示（像素、通道）、数据增强（翻转、旋转、缩放等），预计 1 天。
  + 学习 torchvision 中的数据集（如 MNIST）和数据加载器（DataLoader）的使用，了解如何加载和预处理数据，预计 2 天。
  + 学习卷积神经网络（CNN）的基本结构，如卷积层（Conv2d）、池化层（MaxPool2d）、全连接层（Linear），理解各层的作用和参数含义，预计 3 天。
  + 使用 Pytorch 搭建简单的 CNN 模型用于 MNIST 手写数字分类，如 LeNet - 5 模型，预计 3 天。
  + 学习模型训练过程中的损失函数（如交叉熵损失函数 CrossEntropyLoss）、优化器（如随机梯度下降 SGD、Adam）的选择与使用，进行模型训练和评估，预计 3 天。
  + 学习模型保存与加载，能够将训练好的模型保存，并在后续使用时加载进行预测，预计 1 天。
* **GitHub 学习日志记录**：
  1. **新建文件夹**：在仓库中创建week2 - week5文件夹，继续以日期命名创建 Markdown 日志文件，记录每周学习内容。
  2. **记录学习过程**：详细记录 CNN 模型搭建的每一个步骤，包括网络结构设计思路、参数设置依据；记录数据增强的具体方法和效果对比；记录不同损失函数和优化器对模型训练结果的影响。
  3. **上传成果**：将完整的 MNIST 手写数字分类项目代码（包括数据加载、模型搭建、训练、评估、保存与加载等部分）上传至对应文件夹，在日志中附上模型训练过程中的准确率和损失值变化曲线截图（可使用 Matplotlib 生成）。
  4. **问题与解决**：遇到如模型过拟合、训练不收敛等问题时，在日志中记录分析过程和尝试的解决方法，包括调整网络结构、改变超参数等操作。
* **成果**：搭建并训练完成 MNIST 手写数字分类模型，测试集准确率达到 95% 以上，能够保存和加载模型进行预测；GitHub 仓库完整记录项目开发全过程。

### 第三阶段：目标检测初探（第 6 - 8 周）

#### 项目：基于 YOLO 的简单目标检测

* **学习任务**：
  + 深入学习目标检测相关概念，如边界框（Bounding Box）、锚框（Anchor Box）、交并比（IoU）等，预计 2 天。
  + 学习 YOLO（You Only Look Once）目标检测算法的基本原理，了解其网络结构和检测流程，预计 3 天。
  + 学习使用预训练的 YOLO 模型（如 YOLOv5），通过 torchvision 或官方提供的 Pytorch 实现，对自定义图片进行目标检测，预计 3 天。
  + 尝试在小型自定义数据集上微调 YOLOv5 模型，包括数据集的标注（使用 LabelImg 等工具）、数据格式转换、模型训练参数调整等，预计 5 天。
  + 学习目标检测任务的评估指标，如平均精度均值（mAP），并对训练好的模型进行评估，预计 2 天。
* **GitHub 学习日志记录**：
  1. **更新日志结构**：在week6 - week8文件夹内，按任务模块（原理学习、模型使用、数据集标注、模型微调、评估）创建子文件夹，每个子任务对应一个日志文件。
  2. **记录原理与实践**：在原理学习日志中，用文字和示意图解释边界框、锚框等概念；在模型使用日志中，记录调用预训练模型的代码和检测效果截图；在数据集标注日志中，说明标注工具的使用方法和数据集格式转换过程。
  3. **上传数据集与代码**：将标注好的自定义数据集（脱敏处理后）、微调模型的代码及训练日志（如训练过程中的 mAP 值变化）上传至对应文件夹，在日志中说明数据集的规模和模型微调的关键参数设置。
  4. **总结与反思**：项目结束后，在week8的总结日志中，分析模型在自定义数据集上的检测效果，对比预训练模型和微调后模型的性能差异，提出改进方向。
* **成果**：使用预训练 YOLO 模型完成自定义图片目标检测，在自定义数据集上微调 YOLOv5 模型，模型达到一定的检测效果，并能使用 mAP 等指标进行评估；GitHub 仓库完整呈现目标检测项目的学习和实践过程。

### 第四阶段：图像分割实践（第 9 - 11 周）

#### 项目：基于 U-Net 的医学图像分割

* **学习任务**：
  + 学习图像分割的基本概念和应用场景，了解语义分割、实例分割的区别，预计 1 天。
  + 学习 U - Net 网络结构，理解其编码器 - 解码器结构以及跳跃连接的作用，预计 3 天。
  + 学习医学图像数据的特点和预处理方法（如归一化、裁剪、旋转等），使用公开的医学图像数据集（如 BraTS 脑肿瘤分割数据集），预计 3 天。
  + 使用 Pytorch 搭建 U - Net 模型，并进行医学图像分割任务的训练，学习适合图像分割的损失函数（如 Dice Loss、交叉熵与 Dice Loss 结合等），预计 5 天。
  + 对训练好的模型进行评估，使用图像分割的评估指标，如 Dice 系数、交并比（IoU）等，分析模型性能，预计 2 天。
* **GitHub 学习日志记录**：
  1. **分类记录**：在week9 - week11文件夹下，创建 “概念学习”“网络搭建”“数据处理”“模型训练”“模型评估” 等子文件夹，分别记录对应任务的学习内容。
  2. **可视化展示**：在网络搭建日志中，使用 mermaid 绘制 U - Net 网络结构示意图；在模型训练日志中，插入训练过程中的损失值和 Dice 系数变化曲线；在模型评估日志中，附上分割结果的可视化图片（如原始医学图像、真实分割结果、模型预测结果对比图）。
  3. **代码与数据**：上传 U - Net 模型代码、医学图像数据集预处理代码、评估代码以及处理后的数据集（遵循开源协议），在日志中说明代码的核心功能和数据集的使用方法。
  4. **经验分享**：在学习过程中，若发现有效的医学图像预处理技巧或损失函数调整策略，在日志中详细记录并分享心得。
* **成果**：搭建并训练完成基于 U - Net 的医学图像分割模型，在测试集上达到一定的分割精度，能够使用相关指标评估模型效果；GitHub 仓库成为图像分割项目学习的完整知识库。

### 第五阶段：综合项目与总结（第 12 周）

#### 项目：自选计算机视觉综合项目

* **学习任务**：
  + 根据个人兴趣，从图像分类、目标检测、图像分割等任务中选择一个方向，或结合多个任务，确定一个综合项目主题，如 “基于深度学习的交通场景分析（包含车辆检测、行人检测与道路分割）”，预计 1 天。
  + 进行项目规划，包括数据收集（或使用公开数据集）、数据预处理、模型选择与搭建、训练与优化等步骤的设计，预计 2 天。
  + 实施项目，整合之前学习的知识和技能，完成项目开发，预计 5 天。
  + 对项目进行总结，撰写项目报告，包括项目背景、方法、实验结果、遇到的问题及解决方法等，预计 2 天。
* **GitHub 学习日志记录**：
  1. **新建项目分支**：在仓库中创建以项目名称命名的分支（如traffic - scene - analysis），在分支下创建项目相关文件夹（如data存放数据集，code存放代码，report存放项目报告）。
  2. **全程记录**：在项目实施过程中，每天在code文件夹内的日志文件中记录当天的开发进度、遇到的技术难题及解决方案；在data文件夹的日志中说明数据收集和处理情况。
  3. **项目报告**：将完整的项目报告以 Markdown 或 PDF 格式上传至report文件夹，在 README 文件中对项目进行简要介绍，附上项目代码运行说明和关键成果展示。
  4. **合并与总结**：项目完成后，将分支合并到主分支，在仓库的 README 文件中更新学习计划完成情况，总结整个学习过程中的收获与不足，附上 GitHub 学习日志的导航索引，方便他人查看学习。
* **成果**：完成一个自选的计算机视觉综合项目，提交完整的项目代码和项目报告；GitHub 仓库形成系统的学习记录，涵盖从基础学习到综合项目实践的全过程。

## 三、学习资源推荐

1. **书籍**：

* 《深度学习入门之 PyTorch》：系统讲解 Pytorch 基础知识和深度学习应用。
* 《计算机视觉：算法与应用》：全面介绍计算机视觉的基本概念和算法。

1. **在线课程**：

* 李沐老师的《动手学深度学习》（基于 Pytorch）：通过代码实例讲解深度学习和 Pytorch 知识。
* Coursera 上的《Convolutional Neural Networks》：深入学习卷积神经网络在计算机视觉中的应用。

1. **官方文档**：

* Pytorch 官方文档：[https://pytorch.org/docs/stable/index.html](https://pytorch.org/docs/stable/index.html" \t "_blank)
* torchvision 官方文档：[https://pytorch.org/vision/stable/index.html](https://pytorch.org/vision/stable/index.html" \t "_blank)

1. **开源代码与项目**：

* GitHub 上搜索相关 Pytorch 和计算机视觉项目，如 YOLOv5、U - Net 的 Pytorch 实现，学习优秀代码结构和实现方法。

## 四、希望

1. **注重实践**：每个项目都要亲自动手编写代码，遇到问题多调试，通过实践加深对知识的理解。同时，及时将实践过程和成果记录到 GitHub 学习日志中。
2. **及时总结**：每阶段学习结束后，对当前阶段学习的内容进行总结，记录遇到的问题和解决方法；每个项目完成后，对项目进行复盘，梳理知识点和技术要点，更新到 GitHub 学习日志，形成知识沉淀。
3. **交流讨论**：加入相关学习社区或论坛，与其他学习者交流经验，分享遇到的问题，获取更多学习资源和建议。可以将 GitHub 仓库分享给他人，互相学习和点评。
4. **回顾展望**：每周或每个项目阶段结束后，回顾 GitHub 学习日志，查漏补缺，强化薄弱知识点，优化学习方法和项目代码；培养文献阅读的习惯，通过知乎和一些CV公众号来学习跟踪最新的文献和开源代码，逐步培养针对性优化和创新的能力。