



山东大学
SHANDONG UNIVERSITY

毕业论文（设计）

论文（设计）题目： 使用深度学习在 SNO+ 实验进行位置重建

姓 名	秦祥语
学 号	202122161288
学 院	泰山学堂
专 业	物理学
年 级	2021 级
指导教师	张洋

2025 年 4 月 15 日

摘要

SNO (Sudbury Neutrino Observatory, 萨德伯里中微子观测站)+ 实验是开创性的 SNO 的后继实验, 坐落于加拿大萨德伯里地下约 2 公里处。它的主要科学目标是探测极为罕见的无中微子双贝塔衰变 ($0\nu\beta\beta$)。如果观测到这一现象, 将对中微子物理和基本粒子理论产生深远的影响。为实现这一雄心勃勃的目标, SNO+ 在一个直径 12 米的丙烯酸容器中装载了 780 吨液体闪烁体, 并计划向其中加入约 1.3 吨的同位素碲-130(^{130}Te)。要从背景事例中分辨出 $0\nu\beta\beta$ 的微弱信号, 准确的事例重建(包括对事例位置、方向、时间以及能量的精确测定) 至关重要。

在传统的中微子实验中, 事例重建通常依赖基于最大似然估计的解析方法。虽然这些方法应用广泛, 但在面对复杂的事例、大量噪声或庞大数据时, 它们往往受到内在限制, 表现出重建精度不足及计算效率较低等问题。近年来, 深度学习 (deep learning) 技术在粒子物理研究中崭露头角, 凭借其从大规模数据中捕捉复杂非线性关系的卓越能力, 为提升重建性能提供了重要契机。

然而, 尽管深度学习方法取得了显著成就, 评估模型预测的可靠性和鲁棒性仍是一个关键挑战。尤其需要关注全面的不确定性量化, 以确保基于深度学习的重建算法在科学上具有可信度。对不确定性的定量分析 (包括源于数据的随机不确定性和源于模型的认知不确定性) 不仅能够避免预测过度自信或信心不足的问题, 也对后续的物理分析至关重要。

在本文中, 我们提出了一种专门针对 SNO+ 实验特点而设计的全新深度学习事例位置重建算法。该模型采用了基于 Transformer 的神经网络架构, 因其在处理序列数据和注意力机制方面具有强大的能力而备受关注。为训练和验证该深度学习模型, 我们使用了基于 SNO+ 实验中 RAT(reactor analysis tool, 反应堆分析工具) 软件的蒙特卡洛模拟数据, 该数据能较好地反映真实探测器环境和事例分布。此外, 我们还提出了一种新的基于闪烁体中自然放射性的 $^{214}\text{Bi} - ^{214}\text{Po}$ 事例符合信号的验证方法。该方法提供了一个由数据驱动的有力验证手段, 用于直观地评估模型在空间和时间上的重建准确度, 从而与传统的基于模拟的评估形成互补。

通过系统的评价, 我们的结果显示, 这种基于 Transformer 的深度学习模型在位置重建精度、计算效率以及对探测器噪声与不确定性的鲁棒性方面, 均基本达到了传统的

MLE 方法。更为重要的是，通过引入蒙特卡洛 Dropout 技术，能够同时量化随机和认知不确定性，从而全面表征重建结果的可信度。

总体而言，本研究不仅提出并验证了一种结合不确定性估计的新型深度学习位置重建框架，而且对中微子物理实验的数据分析方法做出了一定贡献。所提出的方法具有广泛适用性，后续可以通过修改迁移到其他中微子或稀有事例探测实验中。

关键词：双贝塔衰变；中微子物理；重建；深度学习；不确定性估计；液体闪烁体探测器

ABSTRACT

The SNO+ experiment is the successor to the pioneering Sudbury Neutrino Observatory (SNO). It is located approximately 2km underground in Sudbury, Canada. Its primary scientific goal is to detect the exceptionally rare event, neutrinoless double-beta decay ($0\nu\beta\beta$). This phenomenon would provide profound implications for neutrino physics and fundamental particle theory if observed. To achieve this ambitious goal, SNO+ employs a detector filled with 780 tonnes of liquid scintillator, which will subsequently be loaded with approximately 1.3 tonnes of the isotope ^{130}Te in its 12-meter diameter acrylic vessel. Precise event reconstruction, including accurate determination of event location, direction, timing, and energy, is crucial for distinguishing the subtle signals of $0\nu\beta\beta$ from background events.

Traditionally, event reconstruction in neutrino experiments relies on analytical methods, often based on maximum likelihood estimation. Although widely employed, these methods face inherent limitations when dealing with complex events, substantial noise, or large datasets, leading to constrained reconstruction accuracy and computational inefficiencies. Recently, deep learning techniques have emerged prominently within particle physics research, demonstrating significant potential for enhancing reconstruction performance due to their exceptional capability of capturing intricate, nonlinear relationships from large-scale datasets.

Despite the promising achievements brought by deep learning approaches, assessing the reliability and robustness of model predictions remains a critical challenge. Specifically, comprehensive uncertainty quantification is required to ensure the scientific credibility of deep learning-based position reconstruction algorithms. Quantifying uncertainties, including aleatoric (data-driven) and epistemic (model-related) uncertainties, is essential not only for controlling the confidence of predictions to avoid overconfident or underconfident, but also for the subsequent physical analysis.

In this thesis, we present a novel deep learning-based event reconstruction algorithm tailored explicitly to the unique characteristics of the SNO+ experiment. The proposed

model leverages Transformer-based neural network architectures, which is well-known for their strong capability in handling sequential and attention-based data processing tasks. To train and validate our deep learning model, we utilized Monte Carlo simulation data based on the reactor analysis tool in SNO+, reflecting realistic detector conditions and event distributions. Moreover, we introduced an innovative validation method based on naturally occurring radioactive ^{214}Bi - ^{214}Po coincidence events within the scintillator. This method offers a robust, data-driven validation approach to intuitively evaluate the spatial and temporal reconstruction accuracy of our model, complementing traditional simulation-based evaluations.

Through systematic assessments, our results demonstrate that the developed Transformer-based deep learning model significantly outperforms traditional MLE-based methods in terms of reconstruction precision, computational efficiency, and robustness to detector noise and uncertainties. Importantly, the implementation of Monte Carlo Dropout techniques enabled the simultaneous quantification of both aleatoric and epistemic uncertainties, providing comprehensive insight into the confidence levels of reconstruction results.

Overall, this research not only proposes and validates an innovative deep learning-based reconstruction framework with integrated uncertainty estimation but also contributes to advancing data analysis methodologies in neutrino physics experiments. The methodologies developed here have broader implications and can be adapted to other neutrino or rare-event detection experiments later after modification, marking an essential step toward leveraging artificial intelligence technologies to advance frontier research in particle physics.

Key Words: Double-beta decay, Neutrino physics, Reconstruction, Deep Learning, Uncertainty estimation, Liquid scintillator detector

目 录

第一章 前言

1.1 研究背景

中微子自被发现以来，一直是粒子物理学研究的热点之一。中微子是标准模型中最轻的基本粒子之一，具有极小的质量，且与物质的相互作用非常微弱，因此它们在宇宙中的传播几乎不受阻碍。中微子在宇宙学、天体物理学和粒子物理学等领域都有着重要的应用。到了 21 世纪，随着实验技术的进步和理论研究的深入，中微子物理学取得了显著的进展。中微子振荡现象的发现，揭示了中微子具有非零质量，并且不同味道的中微子之间可以相互转化，这一发现对粒子物理学和宇宙学都产生了深远的影响，也是目前确凿的超出标准模型的物理现象。日本的超级神冈 (Super-Kamiokande) 和加拿大 SNO 实验等一系列实验，验证了中微子振荡现象^{[1][2]}，并测量了中微子的质量差异和混合角度。这些实验结果为我们理解中微子的性质提供了重要的实验依据。而中微子仍旧是存在未解之谜，许多问题仍然悬而未决。例如，中微子的质量机制是什么？中微子是否是其自身的反粒子，即中微子是否是马约拉纳粒子？这些问题的解决将有助于我们更深入地理解基本粒子的性质和宇宙的演化。

SNO 实验在完成了对中微子振荡的测量后，进行了一系列的升级和改进，以提高探测器的灵敏度和分辨率。SNO+ 实验^{[3][4]}是 SNO 实验的后续项目，旨在利用液体闪烁体探测器对中微子进行更精确的测量，以及测量双贝塔衰变 ($2\nu\beta\beta$) 和寻找一个极其稀有的事件——无中微子双贝塔衰变 ($0\nu\beta\beta$)^[4]。SNO+ 实验采用了液体闪烁体作为探测介质，具有较高的光产额和较低的背景噪声，可以有效地提高中微子的探测效率。液体闪烁体探测器相比传统的水切伦科夫探测器，具有更高的能量分辨率和更好的时间分辨率，可以用于探测低能中微子事件。但是，由于液体闪烁体退激发产生的闪烁光产额远大于切伦科夫光，导致在进行事件的重建时相比于传统的水切伦科夫探测器更加困难。因此，合适的事件重建算法在大型液体闪烁体探测器中显得尤为重要。

传统的事件重建算法主要基于物理模型和几何模型，通过对探测器响应的模拟和拟合来实现事件的重建。这些算法通常需要对探测器的响应进行详细的建模，并且对噪声和背景有较强的依赖性。随着机器学习和深度学习技术的发展，基于数据驱动的方法逐渐成为事件重建的主流。这些方法通过对大量实验数据的学习，自动提取特征并进行事

件重建，具有更好的适应性和鲁棒性。深度学习方法在事件重建中的应用，主要包括卷积神经网络 (CNN)、循环神经网络 (RNN) 和图神经网络 (GNN) 等。这些方法通过对探测器响应数据的学习，可以自动提取特征并进行事件重建，具有更好的适应性和鲁棒性。

但是，深度学习方法在事件重建中的应用仍然面临一些挑战。一般来讲，深度学习训练出的模型在进行预测的时候，给出的结果是一个单值，而不是一个分布，这使得模型在进行预测的时候缺乏不确定性评估。同时，模型本身进行预测时也会带来一定的不确定性。模型的不确定性以及数据本身的统计不确定性都会影响到模型的预测结果。因此，如何对深度学习模型进行不确定性评估，并将其应用于事件重建中，是一个重要的研究方向。

1.2 研究目的

本论文旨在研究基于深度学习的事件重建方法，并对其进行不确定性评估。我们将采用深度学习模型对液体闪烁体探测器中的中微子事件进行重建，并对模型的预测结果进行不确定性评估。通过对不同不确定性的评估方法，分析其在事件重建中的应用效果，为后续的实验数据分析提供参考。

1.3 论文内容安排

本文的第一部分首先介绍了中微子物理学的背景和研究现状，重点介绍了中微子的发现历程和双贝塔衰变现象；第二部分介绍了深度学习以及其带来的不确定性分析；第三部分介绍了 SNO+ 实验，重点介绍在该实验中传统的重建方法；第四部分介绍了基于深度学习的事件重建方法，并对其进行不确定性评估；最后一部分总结了本文的研究工作，并展望了未来的研究方向。

第二章 中微子物理

中微子物理一直是粒子物理学中一个非常活跃的研究领域。中微子是轻子家族中的一员，具有极小的质量和非常弱的相互作用，这使得它们在宇宙中的传播和相互作用非常难以观测。然而，中微子物理的研究对于理解宇宙的起源、演化以及基本粒子之间的相互作用具有重要意义。中微子在宇宙中扮演着重要的角色，它们不仅参与了核反应和宇宙射线的产生，还可能参与宇宙的膨胀和冷却过程。

本章节主要介绍中微子物理，??节回顾了中微子的发现历程，??节简要介绍了中微子振荡现象，??节介绍了中微子质量以及 ??节介绍了双贝塔衰变的相关问题。

2.1 中微子的发现历程

1914 年，Chadwick J. 在研究 β 衰变过程中发现了一个令人困惑的问题： β 衰变中电子的能量呈现出连续分布的谱线，而非单一常数值。这一现象与当时理论预期的单一能量释放过程不符。为了解释这一现象，1930 年，Pauli W. 提出了一个假设：在 β 衰变过程中，除了电子外，还存在一个未被观测到的粒子参与其中，这个粒子后来被称为“中微子”。1934 年，Fermi E. 提出了中微子与电子的相互作用理论，并正式将其命名为“中微子”。1942 年王淦昌首先提出使用电子俘获的方法来探测中微子，而直到 1956 年，Reines F. 和 Cowans C. 才首次在实验中成功探测到了中微子，证实了它们的存在。^[5] 到 1962 年，Lederman L. 等人发现中微子不止一种，除了之前发现的电子中微子外，他们还发现了 μ 子中微子。当第三种轻子 τ 子于 1975 年被发现后，科学家们又在 2000 年证实了第三种中微子—— τ 中微子的存在。这三种中微子具有不同的味道，分别对应三种轻子：电子、 μ 子和 τ 子。

2.2 中微子振荡

由超级神冈对于大气中微子中穿过地球的 μ 子中微子缺失测量的结果和由 SNO 实验通过对太阳中微子的带电流和中性流的测量对“太阳中微子消失之谜”的解答可以看出，中微子在传播过程中会发生振荡现象，即中微子的三种味道会在其传播过程相互转化。这意味着中微子是具有内禀时钟，从而得出了与标准模型的预测完全不同的结果

——中微子具有静止质量。中微子振荡的现象可以用一个简单的模型来描述。现在有三种我们可以探测到的中微子，分别是电子中微子 ν_e 、 μ 子中微子 ν_μ 和 τ 子中微子 ν_τ 。那么根据薛定谔方程，考虑平面波，在真空中，从一种味道的中微子 $|\nu_\alpha\rangle$ 到另一种味道 $|\nu_\beta\rangle$ 的转化的概率是：¹

$$P_{\nu_\alpha \rightarrow \nu_\beta} = \delta_{\alpha\beta} - 4 \sum_{i>j=1} \mathbf{Re}(K_{\alpha\beta,ij}) \sin^2 \left(\frac{\Delta m_{ij}^2 L}{4E} \right) - 4 \sum_{i>j=1} \mathbf{Im}(K_{\alpha\beta,ij}) \sin \left(\frac{\Delta m_{ij}^2 L}{2E} \right) \quad (2.1)$$

其中

$$K_{\alpha\beta,ij} = U_{\alpha i} U_{\beta i}^* U_{\alpha j} U_{\beta j}^* \quad (2.2)$$

Δm_{ij}^2 为质量本征态 $|\nu_i\rangle$ 与质量本征态 $|\nu_j\rangle$ 的平方差值， $L = x = ct$ 为中微子源到探测器的距离。 U 为 PMNS 矩阵：

$$|\nu_\alpha\rangle = U_{PMNS} |\nu_i\rangle \quad \alpha = e, \mu, \tau; i = 1, 2, 3 \quad (2.3)$$

$$U = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_{23} & s_{23} \\ 0 & -s_{23} & c_{23} \end{pmatrix} \begin{pmatrix} c_{13} & 0 & s_{13}e^{-i\delta} \\ 0 & 1 & 0 \\ -s_{13}e^{i\delta} & 0 & c_{13} \end{pmatrix} \begin{pmatrix} c_{12} & s_{12} & 0 \\ -s_{12} & c_{12} & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (2.4)$$

其中 $s_{ij} = \sin \theta_{ij}$, $c_{ij} = \cos \theta_{ij}$ ($i, j = 1, 2, 3$), θ_{ij} 为混合角, δ 为 CP 相位。这是对于 Dirac 中微子的 PMNS (Pontecorvo–Maki–Nakagawa–Sakata) 矩阵, 而对于 Majorana 中微子, 会引入额外的两个相位参数:

$$U = U_{PMNS} \cdot \text{diag}(1, e^{i\alpha_1}, e^{i\alpha_2}) \quad (2.5)$$

但是不影响整体的中微子振荡, 因为取模之后这些因子都被消去了。

2.3 中微子质量

中微子振荡的发现打开了中微子研究的新领域, 其中最重要的问题是, 既然中微子具有质量, 那么它们的质量是多少? 以及它们的质量是如何产生的呢? 一般认为, 中微子

¹这里考虑了相对论效应, 且由于中微子能量 E 远大于中微子静止质量 m , 故可以在自然单位制下近似认为 $E = \bar{p}$, \bar{p} 为中微子的平均动量。

质量有两种来源：一是通过希格斯机制，类似于其他费米子。另外一种猜测是中微子可能是一直没有被发现的 Majorana 费米子，即中微子的反粒子是它本身。对于 Majorana 中微子，需要超出标准模型的理论进行解释。一个最吸引人的模型是所谓的“翘翘板”机制，其引入了具有超大 Majorana 质量项的右手中微子，从而可以获得一个超大的质量本征值和一个很小的质量本征值。

考虑左手中微子和超大质量的右手中微子以及对应的反中微子通过 Dirac 质量项和 Majorana 质量项的耦合，其拉格朗日量为^[6]¹：

$$2\mathcal{L} = m_D(\bar{\nu}_L N_R + \bar{N}_L^c \nu_R^c) + m_L \bar{\nu}_L \nu_R^c + m_R \bar{N}_L^c N_R + h.c. \quad (2.6)$$

$$= (\bar{\nu}_L, \bar{N}_L^c) \begin{pmatrix} m_L & m_D \\ m_D & m_R \end{pmatrix} \begin{pmatrix} \nu_R^c \\ N_R \end{pmatrix} + h.c. \quad (2.7)$$

其中 ν_L 为左手中微子， ν_R^c 为其对应的反中微子， m_L 为其质量 N_R 为引入的超大质量中微子， N_L^c 为其对应的反中微子。 m_R 为其质量。 m_D 为 Dirac 质量项， m_R 为 Majorana 质量项。在标准模型中，Majorana 质量项不能出现，因为它是非规范不变的。对于刚才提到的最简单的“翘翘板”机制，我们只需要令 $m_L = 0$, $m_R \gg m_D$ ，则我们得到了两个质量本征态：

$$m_\nu = m_1 = \frac{m_D^2}{m_R} \quad m_N = m_2 = m_R \left(1 + \frac{m_D^2}{m_R^2} \right) \approx m_R \quad (2.8)$$

图??是一个费米场，轻子以及中微子通过 Dirac 质量和 Majorana 质量耦合的示意图。(a) 为费米场的质量产生机制，(b) 为电子的质量产生机制，(c) 为“翘翘板”机制下中微子的质量产生机制，即上面的分析。

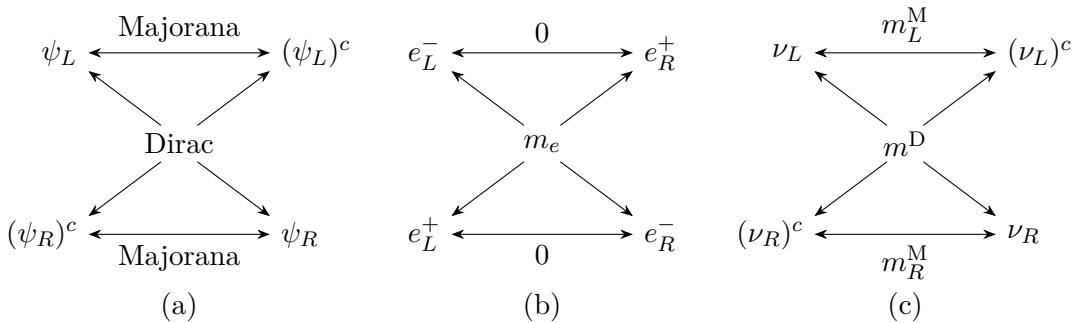


图 2.1 费米场，电子，中微子通过 Dirac 质量和 Majorana 质量耦合的示意图^[6]

¹h.c. 代表厄米共轭

2.4 双贝塔衰变

$\beta\beta$ (Double Beta Decay, 双贝塔衰变) 是一种非常罕见的放射性衰变过程。在这种衰变中，原子核内的两个中子 (或质子) 同时转变为两个质子 (或中子)，并释放出两个电子 (或正电子) 以及可能的中微子。

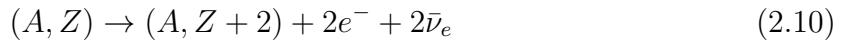
这个过程之所以受到关注，是因为它只发生在那些单贝塔衰变 (一个中子转变为一个质子，释放一个电子和一个反中微子) 在能量上被禁止或者因为角动量选择定则而被高度压抑的原子核中。换句话说，如果一个原子核满足

$$m(Z, A) > m(Z + 2, A) \quad m(Z, A) < m(Z + 1, A) \quad (2.9)$$

那么它就可能经历 $\beta\beta$ 。 $\beta\beta$ 主要有两种模式： $2\nu\beta\beta$ (Two-neutrino double beta decay, 双中微子双贝塔衰变) 和 $0\nu\beta\beta$ (Neutrinoless double beta decay, 无中微子双贝塔衰变)。

2.4.1 $2\nu\beta\beta$

$2\nu\beta\beta$ 是标准模型框架中允许的过程。原子核内的两个中子转变为两个质子，同时释放出两个电子和两个电子反中微子 ($\bar{\nu}_e$)。反应式可以写为：



这个过程遵守轻子数守恒定律 (初始轻子数为 0，末态轻子数为 $2 \times (+1) + 2 \times (-1) = 0$)。这一理论首先被 Goeppert M. 提出^[7]， $2\nu\beta\beta$ 衰变已经被实验明确观测到，其半衰期极长，通常在 10^{18} 到 10^{21} 年的量级或更长^[8]。

$\beta\beta$ 的半衰期 ($T_{1/2}$) 可以通过以下公式描述：

对于 $2\nu\beta\beta$:

$$[T_{1/2}^{2\nu}]^{-1} = G^{2\nu}(Q, Z)|M^{2\nu}|^2 \quad (2.11)$$

其中 $T_{1/2}^{2\nu}$ 是 $2\nu\beta\beta$ 衰变的半衰期。 $G^{2\nu}(Q, Z)$ 是可精确计算的相空间因子 (Phase Space Factor)，它依赖于衰变能量 Q 和原子核的电荷数 Z 。 $M^{2\nu}$ 是核矩阵元 (Nuclear Matrix Element, NME)，它的计算涉及复杂的核结构理论，是理论计算中的主要不确定性来源。

2.4.2 $0\nu\beta\beta$

$0\nu\beta\beta$ 是一个理论预言的过程，尚未被实验证实。如果发生，原子核内的两个中子转变为两个质子，只释放出两个电子，没有中微子被释放出来。反应式可以写为：



这个过程违反了轻子数守恒定律 ($\Delta L = +2$)。它的发生需要满足两个关键条件：

1. 中微子是 Majorana 粒子，即中微子和它的反粒子是同一种粒子 ($\nu = \bar{\nu}$)。
2. 中微子具有非零的质量。

在这个模型中，一个中子衰变放出一个电子和一个“反中微子”，这个“反中微子”因为是 Majorana 粒子，可以被第二个中子当作“中微子”吸收，并促使第二个中子衰变放出一个电子。Furry M. 在 1939 年首次提出可以通过寻找 $0\nu\beta\beta$ 来对中微子的本质做出判断。^[9]

对于 $0\nu\beta\beta$:

$$[T_{1/2}^{0\nu}]^{-1} = G^{0\nu}(Q, Z)|M^{0\nu}|^2|\langle m_{\beta\beta} \rangle|^2 \quad (2.13)$$

其中： $T_{1/2}^{0\nu}$ 是 $0\nu\beta\beta$ 衰变的半衰期。 $G^{0\nu}(Q, Z)$ 是相应的相空间因子。 $M^{0\nu}$ 是 $0\nu\beta\beta$ 过程的核矩阵元。 $\langle m_{\beta\beta} \rangle$ 是有效 Majorana 中微子质量，它是粒子物理学非常关注的参数。它与中微子质量本征值 (m_1, m_2, m_3) 和混合矩阵 (PMNS 矩阵) 的元素 (U_{ei}) 相关：

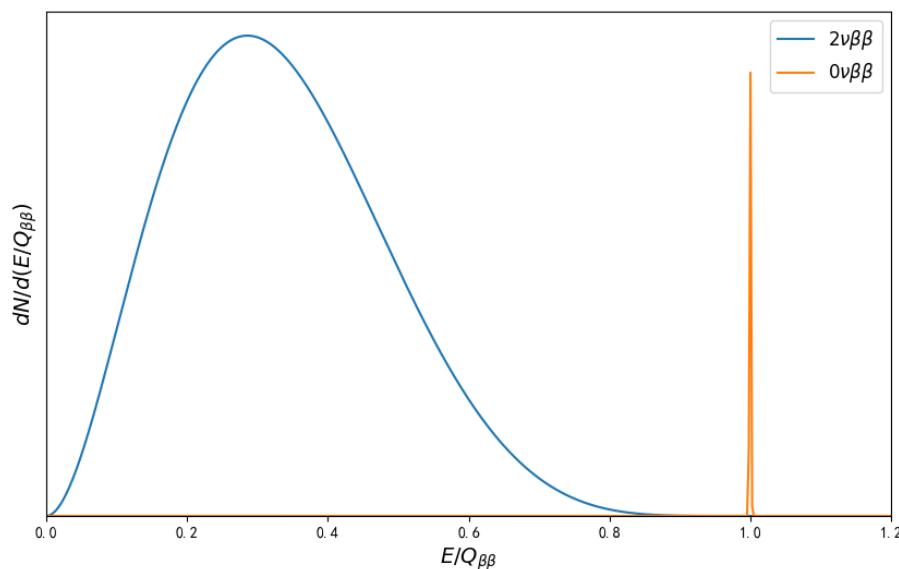
$$\langle m_{\beta\beta} \rangle = \left| \sum_{i=1}^3 U_{ei}^2 m_i \right| \quad (2.14)$$

这里求和遍历所有三个中微子质量本征态 $i = 1, 2, 3$ 。 U_{ei} 是 PMNS 矩阵中连接电子味态和质量本征态 i 的元素。

2.5 $0\nu\beta\beta$ 实验

由于 $0\nu\beta\beta$ 衰变的半衰期预计比 $2\nu\beta\beta$ 更长（如果存在的话，至少 $> 10^{26}$ 年），实验探测极其困难。实验需要在极低的放射性本底下运行，使用大量的同位素源，并具备极高的能量分辨率来区分 $2\nu\beta\beta$ 衰变（电子能量是连续谱）和 $0\nu\beta\beta$ 衰变（两个电子的总能量是一个固定值，等于衰变能 Q ），如图??所示。世界各地有许多实验正在进行或计划中，例如 GERDA/LEGEND，CUORE，EXO-200/nEXO，KamLAND-Zen，MAJORANA Demonstrator，NEXT，SNO+ 等。

衰变	Q-value (keV)	自然界丰度 (%)	$G^{0\nu}$	$G^{2\nu}$
$^{48}_{20}\text{Ca} \rightarrow ^{48}_{22}\text{Ti}$	4262.96 ± 0.84	0.187	24.65	15536
$^{76}_{32}\text{Ge} \rightarrow ^{76}_{34}\text{Se}$	2039.006 ± 0.050	7.8	2.372	46.47
$^{82}_{34}\text{Se} \rightarrow ^{82}_{36}\text{Kr}$	2997.9 ± 0.3	9.2	10.14	1573
$^{96}_{40}\text{Zr} \rightarrow ^{96}_{42}\text{Mo}$	3356.097 ± 0.086	2.8	20.48	6744
$^{100}_{42}\text{Mo} \rightarrow ^{100}_{44}\text{Ru}$	3034.40 ± 0.17	9.6	15.84	3231
$^{110}_{46}\text{Pd} \rightarrow ^{110}_{48}\text{Cd}$	2017.85 ± 0.64	11.8	4.915	132.5
$^{116}_{48}\text{Cd} \rightarrow ^{116}_{50}\text{Sn}$	2813.50 ± 0.13	7.5	16.62	2688
$^{124}_{50}\text{Sn} \rightarrow ^{124}_{52}\text{Te}$	2292.64 ± 0.39	5.64	9.047	551.4
$^{130}_{52}\text{Te} \rightarrow ^{130}_{54}\text{Xe}$	2527.518 ± 0.013	34.5	14.25	1442
$^{136}_{54}\text{Xe} \rightarrow ^{136}_{56}\text{Ba}$	2457.83 ± 0.37	8.9	14.54	1332
$^{150}_{60}\text{Nd} \rightarrow ^{150}_{62}\text{Sm}$	3371.38 ± 0.20	5.6	61.94	35397

表 2.1: 不同核素 $\beta\beta$ 的 Q 值和相空间因子^[8]图 2.2: $0\nu\beta\beta$ 和 $2\nu\beta\beta$ 衰变的能谱对比 (非真实比例)

第三章 深度学习以及其不确定性量化简介

深度学习通过构建多层非线性变换网络实现对数据特征的层次化抽象，已成为现代人工智能的核心范式。本章在??节中介绍了神经网络的基础架构，阐述深度学习的基础理论框架，重点解析神经网络的结构特性与训练机制，在??节深入探讨了 Transformer 模型的自注意力机制，并在??节全面论述了不确定性量化的前沿方法与技术挑战。

3.1 神经网络基础架构简介

神经网络是一种模拟人脑神经元结构和功能的计算模型。它由大量的神经元通过连接权重相互连接而成。神经网络的基本单元是神经元，每个神经元接收来自其他神经元的输入信号，并通过激活函数将其转换为输出信号。神经网络的学习过程就是通过调整连接权重来最小化预测值与真实值之间的误差。神经网络的结构通常分为输入层、隐藏层和输出层。输入层接收输入数据，隐藏层通过激活函数对输入数据进行非线性变换，输出层将隐藏层的输出转换为最终的预测结果。神经元的输出信号可以表示为：

$$a_{i,j} = f \left(\sum_{k=1}^n w_{i,j,k} x_k + b_{i,j} \right) \quad (3.1)$$

其中， $a_{i,j}$ 为第 i 个神经元在第 j 层的输出信号， $w_{i,j,k}$ 为第 i 个神经元在第 j 层与第 k 个神经元在第 $j+1$ 层的连接权重， $b_{i,j}$ 为第 i 个神经元在第 j 层的偏置项， $f(\cdot)$ 为激活函数，用于引入非线性变换。常用的激活函数有 sigmoid 函数、反正切 (tanh) 函数和 ReLU(rectified linear unit, 整流线型单元) 函数等。其中 sigmoid 和 tanh 函数呈 S 型曲线，特别适合二分类问题；而 ReLU 函数则是一种分段线性函数，在多分类问题中表现优异。图??展示了一个简单的神经网络结构，其中包含一个输入层、两个隐藏层和一个输出层。

神经网络的训练过程通常采用反向传播算法，该算法通过计算损失函数的梯度来更新连接权重。损失函数用于量化预测值与真实值之间的差距，常见的损失函数包括均方误差和交叉熵等。训练过程中，通常使用梯度下降算法不断迭代优化这些权重，以最小化损失函数并找到最优参数，其数学本质为参数空间的最速下降过程：

$$\theta = \arg \min_{\theta} \mathcal{L}(f_{\theta}(x), y) \quad (3.2)$$

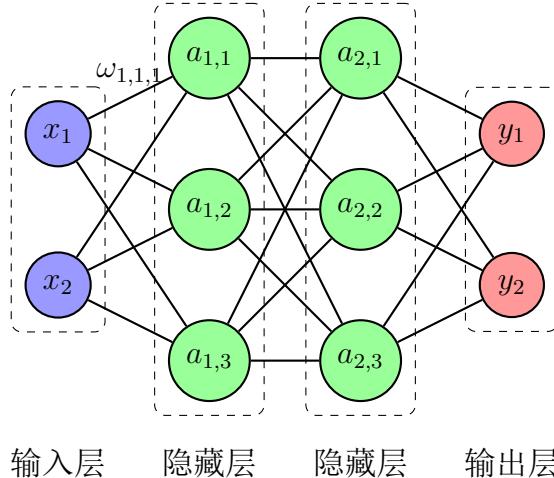


图 3.1: 神经网络基本结构的示意图, 这里包含输入层、两个隐藏层和输出层, 每个隐藏层的神经元都被连接到前一层的所有神经元。最后的输出层将隐藏层的输出转换为最终的预测结果, 在训练过程中, 最终的输出结果会与真实值进行比较, 从而计算损失函数。

其中, θ 为神经网络的参数, \mathcal{L} 为损失函数, $f_\theta(x)$ 为神经网络的预测函数, y 为真实值。

上文介绍的神经网络属于前馈神经网络, 基于前馈神经网络的深度学习模型有很多种, 最常见的有卷积神经网络 (CNN, convolutional neural network) 和循环神经网络 (RNN, recurrent neural network)。CNN 是一种特殊的前馈神经网络, 主要用于处理图像数据。它通过卷积层和池化层提取图像特征, 并通过全连接层进行分类。卷积层通过卷积操作提取局部特征, 池化层通过下采样操作减少特征图的尺寸, 从而降低计算复杂度和防止过拟合。RNN 是一种特殊的前馈神经网络, 主要用于处理序列数据。它通过循环结构捕捉序列数据中的时序关系, 并通过 LSTM(long short-term memory, 长短时记忆)^[10] 或 GRU(gated recurrent unit, 门控循环单元)^[11] 等结构解决长依赖问题。卷积神经网络和循环神经网络都是深度学习的重要组成部分, 不过对于传统的 RNN, 由于其在长序列数据上的训练效率较低, 且容易出现梯度消失或梯度爆炸等问题, 因此在处理长序列数据时, 除了 LSTM 和 GRU 等变种 RNN 外, 接下来要介绍的 Transformer 模型也被广泛应用。

3.2 Transformer

Transformer 是一种基于自注意力机制的神经网络模型, 最早由 Google 在 2017 年提出^[12]。Transformer 模型的主要特点是使用自注意力机制来捕捉输入序列中不同位置之间的依赖关系, 从而实现对序列数据的建模。Transformer 模型的基本结构包括编码

器和解码器两个部分。编码器将输入序列转换为一个上下文向量，解码器根据上下文向量生成输出序列。编码器和解码器都由多个相同的层堆叠而成，每一层都包含多头注意力机制和前馈神经网络。编码器和解码器之间通过一个交叉注意力机制进行连接，从而实现对输入序列和输出序列之间的依赖关系建模。编码器的输入是一个序列，输出是一个上下文向量。解码器的输入是上下文向量和前一个时刻的输出，输出是当前时刻的预测结果。编码器和解码器之间通过一个交叉注意力机制进行连接，从而实现对输入序列和输出序列之间的依赖关系建模。编码器的输入序列经过多个多头注意力机制和前馈神经网络的处理后，输出一个上下文向量。解码器的输入是上下文向量和前一个时刻的输出，经过多个自注意力机制和前馈神经网络的处理后，输出当前时刻的预测结果。解码器的输出序列经过一个线性变换和 softmax 函数的处理后，得到每个位置的预测概率分布。解码器的输出序列可以通过采样或贪心搜索等方法生成最终的输出序列。图??是 Transformer 结构的示意图。

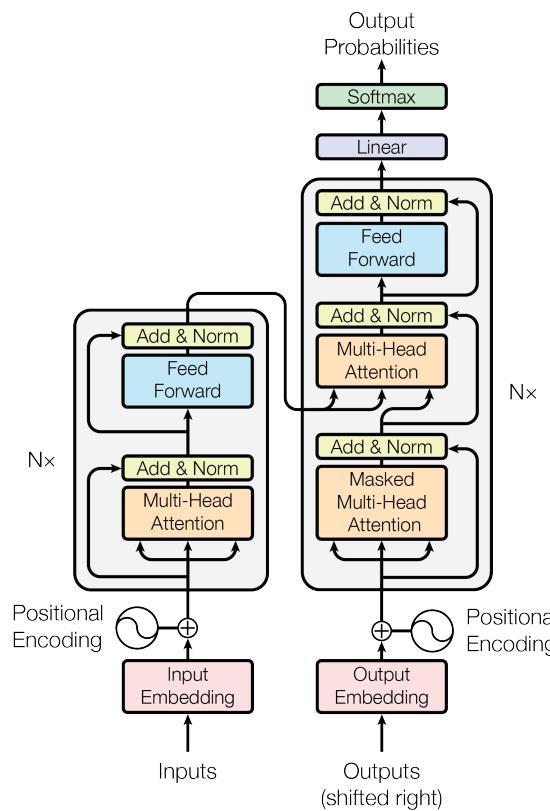


图 3.2: Transformer 模型的基本结构示意图^[12]

3.2.1 注意力机制

注意力机制是一种用于捕捉输入序列中不同位置之间的依赖关系的机制。注意力机制的基本思想是通过计算输入序列中每个位置与其他位置之间的相似度来生成一个上下文向量，从而实现对序列数据的建模。注意力机制的计算过程包括三个步骤：计算注意力权重、计算上下文向量和计算输出。

计算注意力权重是通过计算输入序列中每个位置与其他位置之间的相似度来生成一个注意力权重矩阵。注意力权重矩阵的每一行表示当前时刻的输入位置与其他位置之间的相似度，每一列表示其他位置对当前时刻的输入位置的影响程度。注意力权重矩阵的计算公式为^[12]：

$$\text{Attention} = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \quad (3.3)$$

其中， Q 为查询 (Query) 矩阵， K 为键 (Key) 矩阵， d_k 为键矩阵的维度。注意力权重矩阵的每一行表示当前时刻的输入位置与其他位置之间的相似度，每一列表示其他位置对当前时刻的输入位置的影响程度。上面分析的注意力机制被称为“缩放点积注意力 (Scaled dot-product attention)”，不过我们从图??可以看出，进行应用的是“多头注意力 (Multi-head attention)”。这是由多个缩放点积注意力经过投影 h 次然后经过 d_k ， d_k 和 d_v 并行运算最后叠加而成的。这一机制使得模型可以结合多个不同投影子空间的表现，从而获得更高精确度的预测。这两种注意力机制的示意图由图??给出，其中缩放点积注意力由公式??，而多头注意力用公式表征如下^[12]：

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O \quad (3.4)$$

$$\text{其中 } \text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \quad (3.5)$$

$$(3.6)$$

其中这些投影为参数矩阵 $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{model}}$.

总而言之，Transformer 模型通过其核心的自注意力机制，特别是多头注意力变体，有效地捕捉了序列数据中的长距离依赖关系。其编码器-解码器架构，结合位置编码和前馈网络，使其在序列到序列任务中取得了显著成功，并成为现代深度学习模型的重要基石。

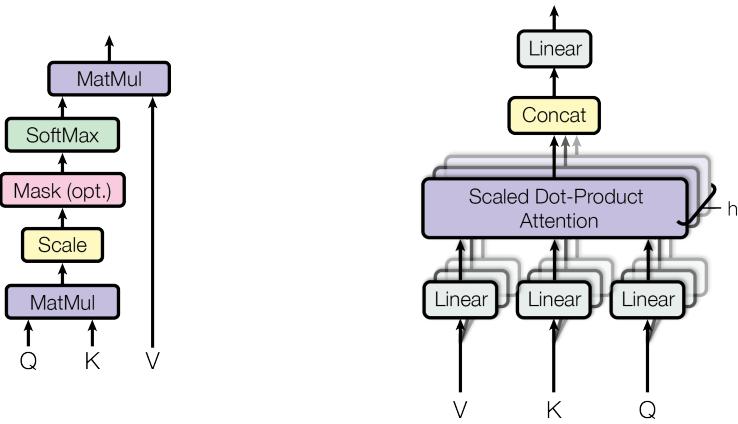


图 3.3: 左图为缩放点积注意力，右图为多头注意力由多个并行运行的缩放点积注意力层组成。^[12]

3.3 深度学习中的不确定性分析

深度学习模型在各种领域展现出了强大的预测能力，然而对这些模型预测结果的不确定性进行量化是一个重要且具有挑战性的问题。不确定性量化 (Uncertainty Quantification, UQ) 旨在评估和表征深度学习模型预测的可靠性和置信度。在实际应用中，了解模型预测的不确定性对于决策制定、风险评估和模型解释至关重要。

在深度学习领域，不确定性可分为认知不确定性（模型不确定性）和随机不确定性（数据不确定性）。^[13] 前者源于模型参数的不确定性，可通过贝叶斯方法量化；后者源于数据本身的噪声或随机性，通常通过概率分布建模。深度学习中的不确定性量化方法主要包括 BNN (Bayesian neural network, 贝叶斯神经网络)、深度集成方法^[14]和 MC(Monte Carlo, 蒙特卡洛) dropout 等技术。这些方法使我们能够对模型预测结果提供可靠的不确定性估计，从而增强模型在实际应用中的可靠性和可解释性。

3.3.1 BNN

BNN 的核心思想是将神经网络的权重 (ω) 和偏置视为概率分布，而不是固定的点估计值，如图??所示。其目标是根据观测数据 D 推断权重的后验分布 $p(\omega|D)$ 。对于新的输入 x^* ，BNN 的预测是通过对所有可能的权重配置进行边缘化（或积分）得到的，即进行贝叶斯模型平均 (Bayesian model averaging):^[13]

$$p(y^*|x^*, D) = \int p(y^*|x^*, w)p(w|D)dw \quad (3.7)$$

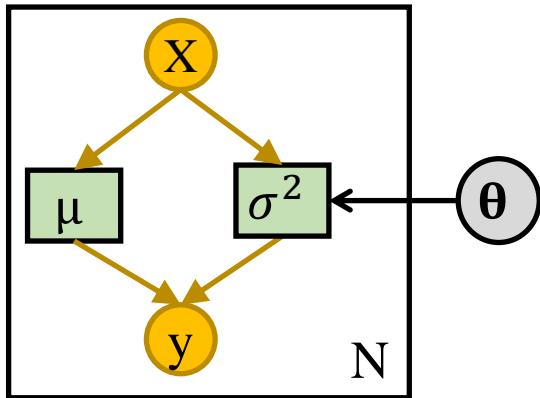


图 3.4: 贝叶斯神经网络的基本结构示意图^[15]

由于精确计算后验分布通常是不可行的，我们一般需要采用一些近似推断方法，例如 VI(Variational Inference, 变分推断)，MCMC(Markov Chain Monte Carlo, 马尔可夫链蒙特卡洛)，拉普拉斯近似等。

对于 VI，我们使用一个简单的、可处理的概率分布 $q(\theta)$ (例如，均值场高斯分布) 来近似真实的后验分布 $p(w|D)$ ，通过最小化它们之间的 Kullback-Leibler 散度^[16]来优化近似分布的参数。VI 在计算上通常比 MCMC 更高效，但其结果依赖于近似分布的选择。

对于 MCMC，我们通过构建马尔可夫链来从后验分布中采样。MCMC 一般被认为能够提供更加精确的估计，但计算成本非常高，尤其对于参数量巨大的深度神经网络。

对于拉普拉斯近似，我们将后验分布近似为以最大后验概率 (MAP) 估计为中心的高斯分布，其协方差矩阵由损失函数的 Hessian 矩阵的逆给出。

BNN 能够自然地捕捉认知不确定性 (通过权重分布的方差体现)。如果模型被设计为预测输出的分布参数 (例如，预测高斯分布的均值和方差)，则也可以捕捉偶然不确定性。一些研究工作尝试利用 BNN 来分离统计不确定性和系统不确定性。一方面，BNN 提供了严谨的贝叶斯框架；有潜力产生良好校准的不确定性估计；能够捕捉模型参数之间的复杂相关性。不过另一方面，BNN 计算成本高昂 (尤其是 MCMC)；结果可能依赖于先验分布的选择；而 VI 的近似质量可能受限；训练过程可能不稳定。

3.3.2 MC Dropout

Dropout 是一种常用的正则化技术，在训练过程中以一定概率随机“丢弃”（即设置为零）神经元的输出。MC dropout 将这一过程扩展到测试（推断）阶段：对同一个输入样本，通过多次随机前向传播，每次使用不同的随机 Dropout 掩码（mask），从而得到一组不同的预测结果，如图??所示。

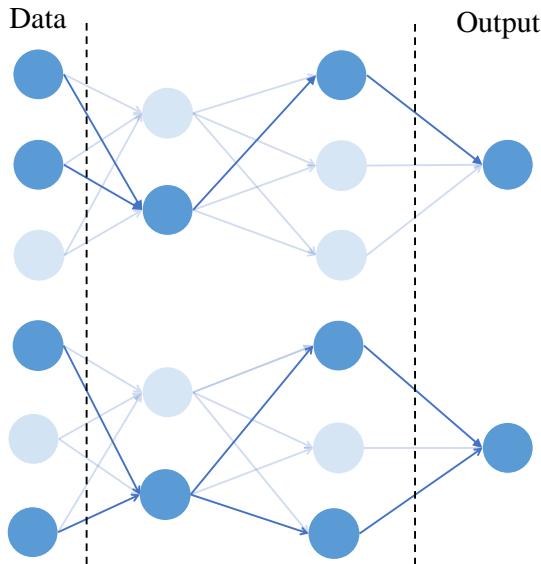


图 3.5: MC dropout 的基本结构示意图^[15]

MC dropout 主要捕捉认知不确定性（模型不确定性），这种不确定性体现在不同 dropout 掩码下预测结果的变化性上。它可以与预测偶然不确定性的模型结合使用。不过相比 BNN，MC dropout 实现更简单，可以方便地应用于已经使用 Dropout 进行正则化的现有网络架构；计算成本相比 BNN 更低。然而，其贝叶斯近似的质量可能有限；对于大型复杂模型，其不确定性估计可能校准不佳或不足；而且其性能（包括 UQ 质量）依赖于 dropout 率的选择和网络结构；不确定性的解释可能不够直接。

3.3.3 深度集成

深度集成是训练多个（N 个）结构相同（或相似）但参数独立的神经网络模型。^[14] 这些模型通常使用不同的随机权重初始化，有时也使用不同的训练数据子集（例如通过 bootstrapping，如图??所示）。进行训练。最终预测结果通过组合 N 个模型的输出得到（例如，对回归任务取均值，对分类任务取平均概率），而不确定性则由模型预测结果之间的差异（例如方差）来估计。

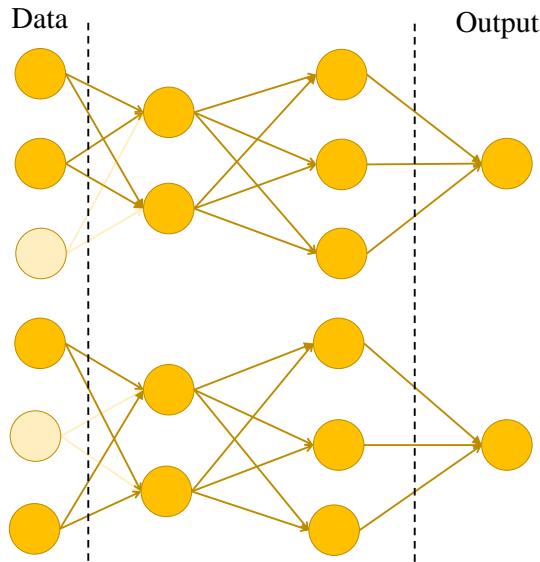


图 3.6: 深度集成训练中 Bootstrapping 的基本结构示意图^[15]

该方法主要通过模型多样性来捕捉认知不确定性。可以与能够预测偶然不确定性的模型（例如，预测输出分布的均值和方差）结合，使用集成预测的方差来估计认知不确定性部分。

深度集成概念简单直观；在许多基准测试和实际应用中，常常能达到顶尖的预测精度和良好的校准性能；模型训练过程可以并行化；但是训练和推断的计算成本很高（需要训练和运行 N 个模型），需要大量的存储空间；同时我们还需要确保集成成员具有足够的多样性；仅仅通过集成预测的差异来估计不确定性，缺乏严格的数学基础来保证其能可靠地同时涵盖偶然和认知不确定性。

总而言之，不确定性量化在深度学习模型的应用中具有重要意义，有助于提升模型的可靠性和决策质量。选择合适的不确定性量化方法需要在计算成本和精度之间进行权衡。

3.4 本章小结

本章首先介绍了深度学习的基础，从神经网络的基本结构、神经元、激活函数和训练过程入手，阐述了前馈神经网络的基本原理，并简要提及了 CNN 和 RNN 等常见变种。随后，重点介绍了在序列数据处理中表现优异的 Transformer 模型，详细解释了其核心的自注意力机制，特别是缩放点积注意力和多头注意力。最后，本章深入探讨了深度学习中的不确定性量化问题，区分了认知不确定性和随机不确定性，并详细介绍了三种主

流的不确定性量化方法：贝叶斯神经网络 (BNN)、蒙特卡洛 Dropout(MC Dropout) 和深度集成。对每种方法的原理、优缺点和适用场景进行了分析，强调了不确定性量化对于提升模型可靠性和可解释性的重要性。

第四章 SNO+ 实验

SNO+ 实验是一个位于加拿大安大略省的 SNOLAB (Sudbury Neutrino Observatory Laboratory, 萨德伯里实验室) 的实验。该实验的低背景, 高分辨率, 很适合开展多方面的中微子研究, 例如反应堆中微子, 太阳中微子, 超新星中微子等。SNO+ 实验的主要科学目标是探测极为罕见的无中微子双贝塔衰变 ($0\nu\beta\beta$)。为实现这一雄心勃勃的目标, SNO+ 分阶段在一个直径 12 米的丙烯酸容器中先后装载了纯水 (2017-2019 年), 780 吨液体闪烁体 (2019-2024 年), 并计划向其中加入约 1.3 吨的同位素 ^{130}Te (碲-130)(预计 2025 年开始)^[3]。

这一章主要介绍 SNO+ 实验, ??节简要介绍了 SNO+ 实验的物理目标, ??节简要介绍了 SNO+ 实验的实验装置, ??节简要介绍了 SNO+ 实验中的部分背景事件。

4.1 物理目标

SNO+ 实验的主要科学目标是使用 ^{130}Te 探测一种极为罕见的物理现象——无中微子双贝塔衰变 ($0\nu\beta\beta$)。这一现象在??节中已经介绍过了。如果观测到这一现象, 将对中微子物理和基本粒子理论产生深远的影响。目前根据 COURE 实验的结果, 对于 ^{130}Te 的 $2\nu\beta\beta$ 衰变的半衰期为:^[17]

$$T_{1/2}^{2\nu} = 7.71_{-0.06}^{+0.08}(\text{stat.})_{-0.15}^{+0.12}(\text{syst.}) \times 10^{20} \text{ yr} \quad (4.1)$$

End point 能量为:^[18]

$$Q_{\beta\beta} = 2527.518 \pm 0.013 \text{ keV} \quad (4.2)$$

^{130}Te 相对其他同位素具有很多优势, 包括其较高的自然丰度 (34.1%), 以及相对更便宜的价格。这使得我们可以在不进行富集的情况下大批量地投入使用。同时, ^{130}Te 的 $2\nu\beta\beta$ 衰变的半衰期也相对较长, 这使得不可减少的 $2\nu\beta\beta$ 背景事件被压低了。

除了 $0\nu\beta\beta$ 之外, SNO+ 实验还可以用于其他的中微子研究, 例如低能量的太阳中微子, 地球中微子, 反应堆中微子等。SNO+ 探测器同时还可以作为一个超新星中微子监测器。

4.2 SNO+ 实验装置

SNO+ 实验的选址位于地下 2.1km 的 SNOLAB。等效水深约为 6010m，可以有效地屏蔽宇宙射线。SNO+ 复用了 SNO 实验的探测器，并进行了升级。SNO+ 探测器的主要结构由一个厚度为 55 mm, 直径为 12 m 的球形透明的 AV(acrylic vessel, 丙烯酸容器) 组成，用来装载 780 吨液体闪烁体，包括 LAB(linear alkyl benzene, 线性烷基苯), PPO(2,5-diphenyloxazole, 2,5-二苯基恶唑) 以及 bis-MSB(1,4-bis(2-methylstyryl)-benzene, 1,4-双(2-甲基苯乙烯基) 苯)。AV 由直径 17.8 m 的 PSUP(geodesic steel structure, 测地钢结构) 环绕，PSUP 内设有 9362 个带有低活性玻璃和聚光器的内向型 PMT(photomultiplier tubes, 光电倍增管)。91 个没有聚光器的 PMT 也被安装在朝向探测器外的地方，以探测来自 子和 PSUP 外部区域的其他光源的光。AV 和 PMT 之间以及包含探测器的其余腔体的体积由 7000 吨超纯水 (UPW) 填充。这个外部水量为来自 PSUP 和空腔墙的 AV 提供了数米的屏蔽，这两个空腔墙都是外部辐射的来源。如图??所示。

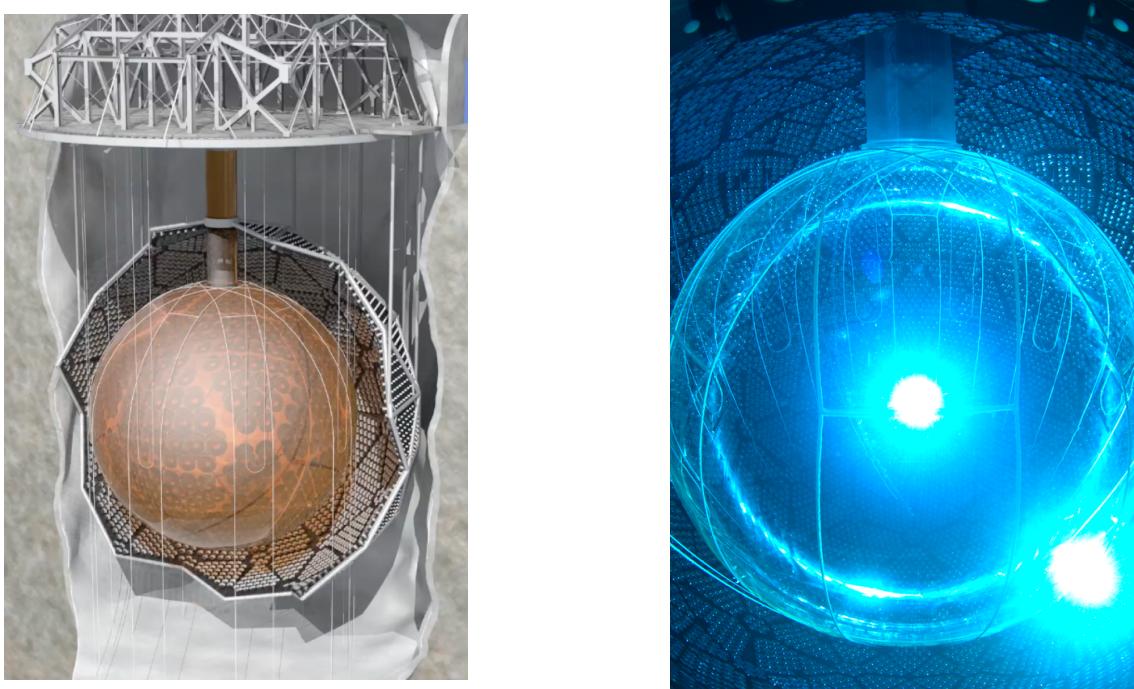


图 4.1: 左边是 SNO+ 实验装置的示意图^[3], 右边是 SNO+ 实验装载纯水时内部的照片^[3]

4.3 SNO+ 实验中的背景事件

如果我们要分辨出 $0\nu\beta\beta$ 的微弱信号，那么准确的背景分析至关重要。从图??中，我们可以看出，最终我们要探测的是后面 $0\nu\beta\beta$ 的峰，也即 $0\nu\beta\beta - ^{130}\text{Te}$ 的 Q 值。

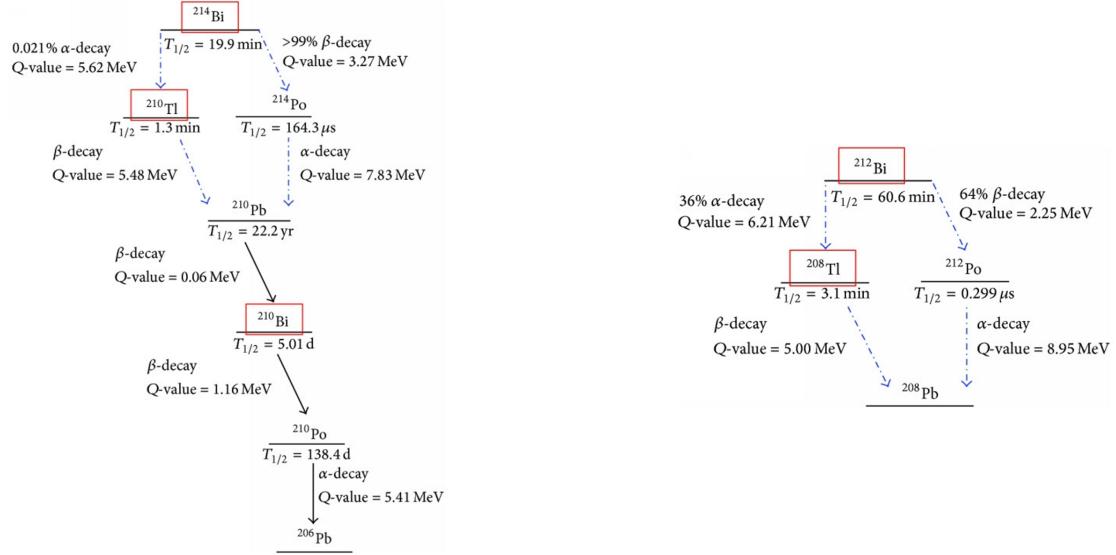


图 4.2: SNO+ 实验中 ^{214}Bi - ^{214}Po 和 ^{212}Bi - ^{212}Po 衰变的 Q 值^[3]

对于这一同位素，从表??中我们知道，其 Q 值约为 2.5MeV。故 SNO+ 中的背景分析需重点关注其 $(-0.5\sigma, 1.5\sigma)$ 内，即大约 2.47-2.69MeV 的事件。重点关注的背景事件有 ^{214}Bi - ^{214}Po , ^{212}Bi - ^{212}Po , ^8B 太阳中微子等。

从 NNDC(National Nuclear Data Center, 国家核数据中心)，以及图??中 ^{212}Bi - ^{212}Po 和 ^{214}Bi - ^{214}Po 衰变的 Q 值，我们可以看到，图中 Q 值并不位于 2.5MeV 附近，然而由于量子淬灭效应，使得实际在 SNO+ 探测器中探测到的 Q 值会偏低。^[19] 这样，这个衰变 ($\beta - \alpha$ 衰变中的 α 衰变) 的 Q 值会落在这一范围内。

而对于 ^8B 太阳中微子，我们从图??中可以看到，根据标准太阳模型，太阳光谱中 ^8B 中微子能谱的分布覆盖了 0.1-15MeV 的范围，当然也包括了 2.5MeV 附近的范围。我们可以看到， ^8B 中微子这一能量附近的流强约为 $10^4 \text{ cm}^{-2} \text{ s}^{-1} (100 \text{ kev}^{-1})$ 量级。

在所有 ^8B 太阳中微子的衰变道中，这一能量附近的 ^8B 中微子的衰变道中，只有弹性散射 (ES, elastic scattering) 会发生，即

$$\nu_e + e^- \rightarrow \nu_e + e^- \quad (4.3)$$

故探测到的是一个单信号 (即电子的信号)。因此 ^8B 太阳中微子也是一个重要的背景事

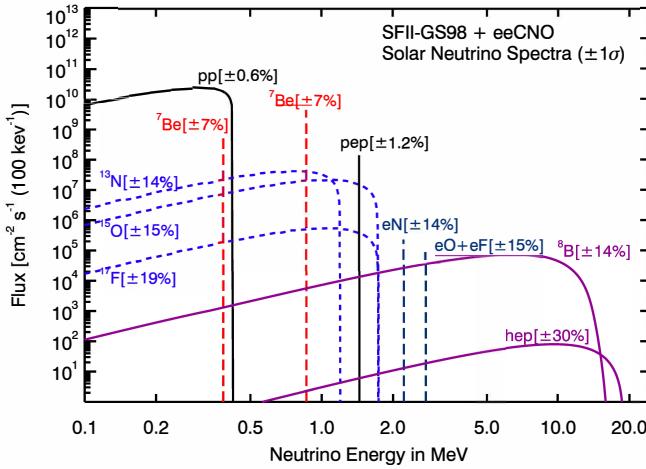


图 4.3: 太阳中微子能谱[8]

件。

4.4 小结

本章首先介绍了 SNO+ 实验的基本情况，包括其地理位置、发展阶段以及主要科学目标——探测 ^{130}Te 的无中微子双贝塔衰变。接着，详细阐述了 SNO+ 的物理目标，不仅包括核心的 $0\nu\beta\beta$ 探测，还涵盖了太阳中微子、地球中微子、反应堆中微子以及超新星中微子等其他研究方向，并解释了选择 ^{130}Te 作为目标同位素的优势。随后，本章描述了 SNO+ 实验的装置，包括其深地实验室的位置、探测器的主要组成部分（如丙烯酸容器、液体闪烁体、光电倍增管阵列和外部超纯水屏蔽层）。最后，讨论了 SNO+ 实验中对 $0\nu\beta\beta$ 探测构成挑战的主要背景事件，重点分析了 ^{214}Bi - ^{214}Po 、 ^{212}Bi - ^{212}Po 衰变链以及 ^8B 太阳中微子弹性散射事件，这些事件的能量可能落在感兴趣的区域内，需要精确认识和扣除。这些背景事件在下一章都会成为我们在机器学习中需要用到的数据集。

第五章 SNO+ 实验的位置重建以及 UQ

探测器中收集到的原始数据无法直接拿来做物理分析，这些数据需要经过一系列的计算，才能拿到后续所需要的物理量。例如经过计算拿到了衰变顶点的位置，方向，能量，粒子类型等，并进行分析。这一计算过程我们称之为重建。

这一章首先??节介绍 SNO+ 探测器中传统的重建方法，即最大似然法。接着在??节介绍深度学习在 SNO+ 探测器中的重建方法，并给出两者结果的比较。

同时也会给出两者重建结果不确定性的估计。

5.1 SNO+ 探测器中传统的位置重建方法

在 SNO+ 探测器中，传统的重建方法主要是基于 MLE (Maximum likelihood estimation, 最大似然估计) 的。一般我们通过 PMT 的时间和位置信息来进行重建。可能后期还会加入其他的信息，例如 PMT 的积分电荷等。SNO+ 实验中应用的 MLE 方法是基于一个假设，即 PMT 接收到了一个内部电子的光子信号。^[20]虽然这个假设可能不是一直成立，不过我们还会用其他的方法来分辨粒子类型。

更严谨地说，如果我们想确定一个事件的位置，我们会把一个基于时间残差的似然函数在每个 PMT 上最大化。时间残差 $T_{res,i}$ 的定义为：

$$T_{res,i} = T_{PMT,i} - T_{fit} - T_{transit}(\vec{x}_{PMT,i}, \vec{x}_{fit}) \quad (5.1)$$

其中 $T_{PMT,i}$ 为 PMT 被击中的时间， T_{fit} 为事件的拟合时间， $T_{transit}$ 为光子在 PMT 和事件之间传播的时间。一般我们认为光子沿直线传播，所以 $T_{transit}$ 可以表示为：

$$T_{transit}(\vec{x}_{PMT,i}, \vec{x}_{fit}) = \frac{||\vec{x}_{PMT,i} - \vec{x}_{fit}||}{c_{avg}} \quad (5.2)$$

其中 c_{avg} 为光子在液体闪烁体中的平均速度。不过在 SNO+ 中，光子需要穿过液体闪烁体，AV，UPW 和 PMT 的玻璃等介质，才能到达 PMT 的收集装置。

$T_{res,i}$ 的分布一般服从正态分布，其均值为 0，方差为 σ_i^2 。所以似然函数可以表示为：

完整的 PDF(Probability density function, 概率密度函数) 由于考虑了反射，PMT 本身的噪声以及各种其他的效应，更加的复杂。结果为一个具有长尾巴并且有多个峰的分布。这个结果是从模拟和刻度中得到的。^[20]

T_{fit} 和 x_{fit} 是我们需要拟合的参数。我们可以通过最大化似然函数来得到它们的值。

$$\mathcal{L}_{vertex} = \prod_{i=1}^{Nhits} P(T_{res,i}) \quad (5.3)$$

其中 $Nhits$ 是 PMT 被击中的数目。不过一般来说，我们不会直接最大化似然函数，而是最小化负对数似然函数：

$$-\log(\mathcal{L}_{vertex}) = -\sum_{i=1}^{Nhits} \log P(T_{res,i}) \quad (5.4)$$

更具体一点，SNO+ 使用的是 Powell 算法^[21]来最小化负对数似然函数。Powell 算法是一种无导数的优化算法，它通过迭代地调整参数来找到函数的最小值。该算法使用了一个线性组合的搜索方向，并在每次迭代中更新参数。Powell 算法的优点是它不需要计算梯度，因此适用于那些难以计算导数的函数。例如 SNO+ 目前重建所用的 PDF。当然，不止传统算法，神经网络在训练时的梯度下降算法的思想与 Powell 算法比较相似。

目前 SNO+ 实验所使用的重建方法涉及各种算法的先后结合。主要算法如??一样最小化负对数似然函数来确定事件位置。然而，尽管最大化似然函数是该过程中的第一步，由于需要拟合的参数必须初始化为某个值，有几个算法是事先运行的。这个被初始化的值通常称为种子。与神经网络反向传播的情况一样，初始值对是否能准确重建有很大的影响。

我们使用的初始化算法基于只需要 4 个 PMT 击中就可以解析地计算事件的位置和时间的原理。根据??， $T_{res,i}$ 设置为零时有四个参数需要拟合，所以只需要四次命中 PMT 的信息。当然，在实际应用中，计算位置和时间的准确性取决于选择哪四个 PMT，光子从事件到击中 PMT 的路径，以及与击中时间相关的噪声。

5.2 深度学习在 SNO+ 探测器中的位置重建方法

上面我们简要介绍了传统的位置重建方法，我们可以看到，传统的方法需要对 PMT 的响应进行建模，并且对噪声和背景有较强的依赖性。而深度学习方法则是基于数据驱动的方法，通过对大量实验数据的学习，可以捕捉到一些传统方法捕捉不到的信息并进行事件重建，具有更好的适应性和鲁棒性。

接下来我们介绍深度学习在 SNO+ 探测器中的位置重建方法。前面??节中介绍了神经网络的基础架构，以及 CNN 和 RNN，这两种神经网络在粒子物理实验的重建工作中

都有使用。我们使用的是在??节中介绍的 **Transformer** 模型。得益于 Transformer 模型的自注意力机制，我们可以在输入数据中捕捉到长距离的依赖关系，从而更好地建模复杂的事件特征。

5.3 数据集

我们在??节中提到，我们使用了 SNO+ 实验的模拟数据集来训练和测试模型。主要为了保证模型的泛化能力，我们使用了不同的能量范围和位置的数据集。主要使用了⁸B 衰变道的太阳中微子数据。这里与之前 SNO+ 组内使用的模拟数据不同的是，我们额外使用了不同探测器状态的模拟，以此来保证模型的泛化能力。¹ 而 SNO+ 合作组使用的 RAT(Reactor analysis tool, 反应堆分析工具) 软件提供的 MC 框架中提供了完整的 run-by-run 的模拟数据设置。也就是说，模拟可以完整包含对应 run number 的所有的 PMT 状态信息。这大大提升了训练出来的模型的鲁棒性，使得我们在预测时，即使在不同的探测器状态下也能有较好的效果。

我们的模拟使用的是 SNO+ 中液体闪烁体阶段的探测器。其液体闪烁体配方为 LAB+PPO+bis-MSB。在模拟完成后，我们提取对于位置重建有用的信息，如 PMT 的时间和位置，以及真实的事件位置，能量等。我们使用了大约 10^6 个事件来训练模型，分为训练集和测试集。训练集和测试集的比例为 70% 和 20%。

5.4 位置重建模型架构

我们采用基于 Transformer 的模型进行位置重建。该模型主要包括特征嵌入、Transformer 编码器和输出层。

5.4.1 输入与嵌入

模型的输入是每次事件中被触发的光电信增管 (PMT) 的命中序列。每个命中包含时间、PMT ID 及其三维坐标 (x, y, z)。这些原始特征首先被转换 (嵌入) 为固定维度的向量。连续特征 (时间、坐标) 通过线性层处理，分类特征 (PMT ID) 通过查找表转换。所有特征的嵌入向量被拼接起来，形成每个命中的初始表示。

¹ 在 SNO+ 实验中，探测器在不同的时间有不同的状态，例如在某些时候，探测器的某些 PMT 会关闭，或者会进行维护，这些状态会影响 PMT 的响应。

5.4.2 Transformer 编码器

嵌入后的命中序列被送入 Transformer 编码器。在进入编码器之前，会添加位置编码以保留命中顺序信息。编码器由多个堆叠的注意力模块 (Attention Block) 组成。每个模块包含：

- **多头自注意力 (Multi-Head Self-Attention)**：捕捉序列中不同 PMT 命中之间的复杂关联。
- **前馈神经网络 (Feed-Forward Network)**：对注意力层的输出进行进一步处理。

每个模块都使用了层归一化 (Layer Normalization) 和残差连接 (Residual Connection) 来稳定训练和提升性能。

5.4.3 输出层与损失函数

编码器处理完整个序列后，输出每个命中位置的上下文感知表示。为了得到代表整个事件的单一向量，我们对序列输出进行平均池化 (考虑了序列填充的掩码)。最后，这个池化后的向量经过一个线性层，输出预测的事件顶点三维坐标 (x, y, z)。

模型训练采用加权的均方误差 (MSE) 损失函数。该损失函数会根据预测位置与真实位置的距离误差来调整每个样本的权重，目的是降低特别大的误差 (异常值) 对模型训练的影响，或根据需要增加其影响。

5.4.4 模型训练

模型训练采用流式处理和分块方法处理大型数据集。主要步骤如下：

1. **初始化**: 配置训练参数、日志、计算设备 (CPU/GPU)，加载 PMT 位置信息，并构建特征嵌入层和 GPT 回归器模型。定义加权均方误差损失函数、Adam 优化器和余弦退火学习率调度器。
2. **数据处理**: 训练数据 (ROOT 文件) 被分成多个块 (chunks) 以适应内存。对每个块：
 - 加载事件数据，进行筛选 (如命中数、能量、位置范围)、排序和坐标转换。

- 将块内事件随机划分为训练集、验证集和测试集。测试集事件被累积起来用于最终评估。

3. 训练与验证循环:

- 在每个训练轮次 (epoch) 中，模型依次处理所有数据块。
- 对每个块的训练集部分，模型通过小批量梯度下降进行训练，计算损失并更新参数。
- 对每个块的验证集部分，模型在评估模式下计算损失，以监控性能。
- 显存会定期清理以处理大数据。

4. 模型选择与早停:

- 每个轮次结束后，计算平均训练和验证损失。
- 根据验证损失更新学习率。
- 如果当前验证损失优于历史最佳，则保存模型状态。
- 若验证损失连续多个轮次没有改善 (达到预设耐心值)，则触发早停机制，提前结束训练。

5. 最终评估与输出:

- 训练结束后，加载性能最佳的模型。
- 在累积的测试集上进行最终评估，计算预测位置与真实位置的距离误差等指标，并与传统方法¹进行比较。
- 生成损失曲线图、预测性能对比图 (如误差分布、分辨率随能量/命中数变化)，并将测试集的预测结果、真实值等信息保存到新的 ROOT 文件中。

这种方法允许在有限的内存资源下训练大型数据集，并通过验证集监控、早停和学习率调度来优化训练过程和模型性能。

5.5 深度学习位置重建结果

从??中我们可以看到，我们利用 ^8B 太阳中微子数据的验证集进行位置重建，并选择了三维空间中预测的坐标与真实坐标之间的距离作为评价指标。从图中我们可以看到，

¹这里使用的是 RAT 中集成的闪烁体拟合 “scintFitter” 中内嵌的重建方法

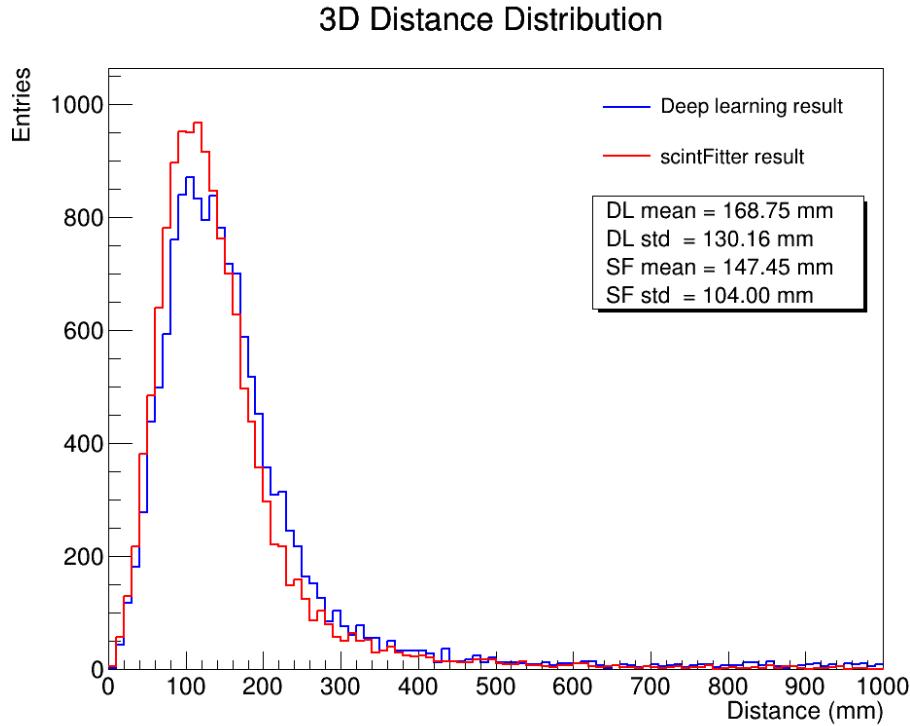


图 5.1: 深度学习位置重建结果

深度学习模型的重建结果明显优于传统的重建方法，同时在不同能量范围内，深度学习模型的重建结果也比较稳定。而传统的重建方法在低能量范围内的重建结果较差，在能量较高时，重建结果较好。这说明深度学习模型在不同能量范围内的重建结果都比较稳定。（如果最后训练出的结果满足，则使用这一表述，如不能则修改）

5.6 位置重建不确定性的量化

上一节中我们介绍了 SNO+ 探测器中位置重建的深度学习方法，并给出了模型的架构和训练方法。在本节中，我们将之前提到的理论投入实际应用，介绍如何具体量化这一深度学习位置重建模型的不确定性。

5.6.1 粒子物理实验中不确定性的估计

在正式开始量化深度学习模型的不确定性之前，我们首先要明确粒子物理实验领域和深度学习领域 UQ 的类型以及两者的区别和联系。^[8]

在??节中，我们了解到，深度学习领域中，不确定性常被区分为偶然不确定性 (aleatoric

uncertainty) 和认知不确定性 (epistemic uncertainty)。而它们与物理学中使用的概念并非完全相同，存在视角上的差异。

首先，在数据收集与不确定性缩减方面，物理学通常关注在固定的实验设计下，通过增加数据量来减少不确定性。在这种情况下，统计不确定性 (类似于偶然不确定性) 会随着数据量的增加而减小，而系统不确定性 (类似于认知不确定性) 通常保持不变，除非改进实验设计或进行更精确的校准。相比之下，深度学习领域更侧重于通过收集不同类型的数据来改进模型本身，从而减少认知不确定性。他们所认为的“不可约减”的偶然不确定性，是指在给定模型下的固有随机性。这种差异并非根本矛盾，而是反映了不同领域关注点的侧重不同。

其次，“模型”的含义也有所不同。在物理学中，“模型”通常指代底层的物理过程或通过模拟体现的探测器响应模型。系统不确定性或认知不确定性与对这些物理模型的理解程度密切相关。而在机器学习领域，“模型”通常指训练得到的函数，例如神经网络。认知不确定性常与训练后模型参数 θ 的不确定性相关联，这种不确定性可以通过收集更多的训练数据来减少。这部分是因为机器学习任务通常对数据生成过程 (如图像分类、自然语言处理) 的先验物理知识较少。

在与物理学 (特别是计算机模拟) 联系更紧密的 UQ 领域，术语的划分更为细致，有助于减少歧义。UQ 领域通常区分以下几种不确定性来源：参数不确定性，例如模型中扰动参数 (nuisance parameters) 的不确定性；结构不确定性，指模型本身与现实不符 (mismodelling) 导致的不确定性，例如在 SNO+ 中 MC 模拟与现实中探测器的实际情况的差异引起的不确定性；算法不确定性，源于数值计算方法引入的误差，例如??节中提到的 SNO+ 实际使用的 Powell 算法引入的不确定性；实验不确定性，包括实验分辨率限制和统计涨落导致的不确定性；以及插值不确定性，这是由于计算资源限制，在不同参数点之间进行插值而引入的不确定性。

5.6.2 在深度学习中量化不确定性的方法

深度学习中的不确定性分析方法在??节中已经介绍过了，我们这里使用的是 MC Dropout 方法和深度集成的方法来量化深度学习模型的不确定性。量化的模型是在??节中介绍的基于 Transformer 的模型。

对于 MC Dropout 方法，我们在训练时使用了 Dropout 层，并在测试时使用了多次不同 Dropout 并进行前向传播来获得模型的预测分布。

对于深度集成方法，我们使用了多个模型，每个模型都使用了不同的随机种子进行训练，然后将这些模型的预测结果进行平均，得到最终的预测分布。

5.6.3 不确定性量化结果

我们分别使用 MC Dropout 和深度集成方法对 ?? 节中描述的 Transformer 模型进行了不确定性量化。对于 MC Dropout，我们在训练期间使用的 Dropout 层在推理阶段保持启用状态。通过对同一输入事件进行 N 次前向传播（每次使用不同的 Dropout 掩码），我们得到了 N 个不同的位置预测 $\{\hat{x}_1, \dots, \hat{x}_N\}$ 。最终的预测位置 \hat{x} 是这些预测的平均值，而预测的不确定性则由这些预测的标准差或方差来估计。这种方法估计的不确定性通常被认为是认知不确定性和偶然不确定性的混合。对于深度集成方法，我们独立训练了 M 个具有相同架构但使用不同随机初始化种子和/或不同训练数据子集（例如通过 bootstrap）的模型。对于一个输入事件，每个模型 m 产生一个预测 \hat{x}_m 。最终的预测 \hat{x} 是 M 个模型预测的平均值。集成预测的方差 $\text{Var}(\{\hat{x}_m\})$ 主要量化了认知不确定性（模型不确定性）。如果每个模型还被训练来预测其自身的不确定性（例如，预测位置分布的方差 σ_m^2 ），则可以通过平均预测方差 $\frac{1}{M} \sum_m \sigma_m^2$ 来估计偶然不确定性（数据不确定性）。

为了评估不确定性量化的质量，我们主要关注几个方面。首先，一个好的不确定性估计应该与实际的预测误差相关，即当模型预测的不确定性较高时，其预测误差也倾向于较大。我们通过比较预测位置 \hat{x} 与真实位置 \vec{x}_{true} 的距离误差 $\|\hat{x} - \vec{x}_{true}\|$ 和模型预测的标准差 σ_{pred} 来评估这一点。其次，我们关注不确定性校准（Calibration），理想情况下，预测的不确定性应准确反映真实的误差水平，例如，对于预测标准差为 σ 的事件集合，其预测位置的均方根误差（RMSE）应接近 σ 。我们通过绘制校准曲线（比较不同预测不确定性分箱内的平均 RMSE）来评估校准程度。此外，利用深度集成方法，我们可以尝试进行不确定性分解，区分认知不确定性（来自模型本身，对应物理实验中的部分系统误差，可通过更多数据或更好的模型减少）和偶然不确定性（来自数据固有噪声，对应物理实验中的统计误差和部分系统误差，通常难以减少）。认知不确定性由集成成员预测之间的差异反映，而偶然不确定性可由每个模型预测的平均方差反映（如果模型输出方差）。最后，我们研究不确定性随物理量的变化，例如能量、真实位置半径、PMT 命中数等，这有助于理解模型在不同物理场景下的可靠性。

初步结果显示（如图 ?? 所示），通过 MC Dropout 和深度集成方法得到的预测不确定性与实际的预测误差表现出正相关性，表明模型在某种程度上能够识别其预测的置信

度。校准曲线显示，模型的不确定性估计在一定程度上是可靠的，但可能存在过度自信或自信不足的区域，需要进一步的校准工作。

如图??所示，我们观察到预测的总不确定性通常在低能量和靠近探测器边缘的区域较高。这符合预期，因为这些区域的可用信息较少或几何效应更复杂，导致重建更具挑战性。深度集成方法的结果表明，认知不确定性在训练数据覆盖不足或模型拟合能力有限的区域相对较高，而偶然不确定性则更多地反映了光子统计和探测器响应的固有波动。

这些量化的不确定性估计对于后续的物理分析至关重要。例如，可以将事件按照其预测不确定性进行加权，或者在进行信号和背景区分时，将具有高不确定性的事件排除或单独处理。通过区分认知不确定性和偶然不确定性，我们可以更好地理解误差的来源：高认知不确定性可能提示需要改进模型或收集更多样化的数据（对应减少模型相关的系统误差），而高偶然不确定性则反映了实验本身的固有局限性（对应统计误差和探测器相关的系统误差）。将这些基于深度学习的不确定性与传统方法（如 MLE 拟合中的参数误差）进行比较，也是未来工作的一个重要方向。

第六章 总结与展望

6.1 总结

本论文围绕大型液体闪烁体探测器 SNO+ 实验中的事件顶点重建问题，深入研究了基于深度学习的方法，并着重探讨了其预测结果的不确定性量化 (UQ)。

首先，论文系统回顾了中微子物理学的基本知识，包括中微子的发现、振荡现象、质量问题以及双贝塔衰变（特别是无中微子双贝塔衰变 $0\nu\beta\beta$ ），为理解 SNO+ 实验的物理目标奠定了基础。接着，介绍了深度学习的基本概念，从神经网络基础架构到卷积神经网络 (CNN)、循环神经网络 (RNN)，并重点阐述了在序列数据处理中表现优异的 Transformer 模型及其核心的自注意力机制。同时，论文详细讨论了深度学习中的不确定性量化问题，区分了认知不确定性和偶然不确定性，并介绍了贝叶斯神经网络 (BNN)、蒙特卡洛 Dropout (MC Dropout) 和深度集成等主流 UQ 方法。

随后，论文详细介绍了 SNO+ 实验的概况，包括其科学目标（探测 ^{130}Te 的 $0\nu\beta\beta$ ）、实验装置以及主要的背景来源（如 ^{214}Bi - ^{214}Po 、 ^{212}Bi - ^{212}Po 和 ^8B 太阳中微子）。在此基础上，论文对比了 SNO+ 中传统的基于最大似然估计 (MLE) 的位置重建方法和基于深度学习的方法。

本文的核心工作在于：

- 设计并实现了一个基于 Transformer 架构的深度学习模型，用于 SNO+ 实验中 ^8B 太阳中微子事件的顶点位置重建。该模型利用 PMT 的时间、位置等信息作为输入，通过自注意力机制捕捉 PMT 命中之间的复杂关联。
- 使用包含不同探测器运行状态的 SNO+ 模拟数据对模型进行了训练和评估。结果表明（如图??所示），该深度学习模型在位置重建精度上优于传统的 MLE 方法，尤其在低能量区域表现更稳定。
- 应用了 MC Dropout 和深度集成两种方法对所开发的 Transformer 重建模型进行了不确定性量化。分析了预测不确定性与实际重建误差的关系、不确定性的校准情况以及其随能量、径向位置等物理量的变化趋势。

- 探讨了粒子物理实验领域和深度学习领域中不确定性概念的异同，并尝试利用深度集成方法分解认知不确定性和偶然不确定性，为理解模型预测的可靠性来源提供了更深入的视角。

总之，本研究成功将 Transformer 深度学习模型应用于 SNO+ 事件重建，并对其进行了不确定性量化，验证了该方法在提升重建精度和提供可靠性评估方面的潜力。

6.2 展望

尽管本研究取得了一定的进展，但仍有许多值得进一步探索的方向：

- **模型优化与扩展：**可以尝试更先进的神经网络架构（如图神经网络 GNN），或将 PMT 电荷信息等更多特征融入模型输入，以期进一步提升重建精度。同时，可以将该方法扩展到能量重建、粒子鉴别等其他重建任务。
- **不确定性量化深化：**需要对不确定性估计进行更严格的校准研究。可以比较更多 UQ 方法（如 BNN 的变分推断）的效果。更重要的是，需要研究如何将量化的不确定性有效地应用于后续的物理分析中，例如在背景抑制、信号提取或系统误差评估中利用 UQ 信息。
- **真实数据验证：**目前的研究基于模拟数据，未来需要将训练好的模型和 UQ 方法应用于 SNO+ 的真实实验数据，检验其在实际应用中的表现，并解决可能存在的模拟与真实数据之间的差异问题。
- **计算效率与部署：**深度学习模型（尤其是 Transformer 和集成方法）的训练和推理计算成本较高，需要研究模型压缩、知识蒸馏或硬件加速等技术，以提高计算效率，便于在 SNO+ 的数据处理流程中进行部署。
- **推广应用：**本研究中开发的基于 Transformer 的重建和 UQ 方法具有一定的通用性，未来可以探索将其应用于其他类似的大型粒子物理实验中。

通过对上述方向的深入研究，有望进一步发挥深度学习和不确定性量化在粒子物理实验数据分析中的作用，为揭示中微子的奥秘和探索新物理提供更强大的工具。

参考文献

- [1] AHMAD Q R, ALLEN R C, ANDERSEN T C, et al. Measurement of the Rate of $\nu_e + d \rightarrow p + p + e^-$ Interactions Produced by 8B Solar Neutrinos at the Sudbury Neutrino Observatory [J/OL]. Phys. Rev. Lett., 2001, 87: 071301. <https://link.aps.org/doi/10.1103/PhysRevLett.87.071301>.
- [2] FUKUDA Y, HAYAKAWA T, ICHIHARA E, et al. Evidence for Oscillation of Atmospheric Neutrinos[J/OL]. Phys. Rev. Lett., 1998, 81: 1562-1567. <https://link.aps.org/doi/10.1103/PhysRevLett.81.1562>.
- [3] ANDRINGA S, ARUSHANOVA E, ASAHI S, et al. Current status and future prospects of the SNO+ experiment[J]. Advances in High Energy Physics, 2016, 2016(1): 6194250.
- [4] ALBANESE V, ALVES R, ANDERSON M, et al. The SNO+ experiment[J]. Journal of Instrumentation, 2021, 16(08): P08059.
- [5] COWAN C L, REINES F, HARRISON F B, et al. Detection of the Free Neutrino: a Confirmation[J/OL]. Science, 1956, 124(3212): 103-104. eprint: <https://www.science.org/doi/pdf/10.1126/science.124.3212.103>. <https://www.science.org/doi/abs/10.1126/science.124.3212.103>.
- [6] ZUBER K. Neutrino Physics[M/OL]. CRC Press, 2020. <https://books.google.ca/books?id=6-XkDwAAQBAJ>.
- [7] GOEPPERT-MAYER M. Double Beta-Disintegration[J/OL]. Phys. Rev., 1935, 48: 512-516. <https://link.aps.org/doi/10.1103/PhysRev.48.512>.
- [8] NAVAS S, et al. Review of particle physics[J]. Phys. Rev. D, 2024, 110(3): 030001.
- [9] FURRY W H. On Transition Probabilities in Double Beta-Disintegration[J/OL]. Phys. Rev., 1939, 56: 1184-1193. <https://link.aps.org/doi/10.1103/PhysRev.56.1184>.
- [10] HOCHREITER S, SCHMIDHUBER J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.
- [11] CHO K, van MERRIENBOER B, GÜLÇEHRE Ç, et al. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation[J/OL]. CoRR, 2014, abs/1406.1078. arXiv: 1406.1078. <http://arxiv.org/abs/1406.1078>.
- [12] VASWANI A, SHAZER N, PARMAR N, et al. Attention is all you need[J]. Advances in neural information processing systems, 2017, 30.
- [13] ABDAR M, POURPANAH F, HUSSAIN S, et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges[J/OL]. Information Fusion, 2021, 76: 243-297. <https://www.sciencedirect.com/science/article/pii/S1566253521001081>.
- [14] LAKSHMINARAYANAN B, PRITZEL A, BLUNDELL C. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles[C/OL]//GUYON I, LUXBURG U V, BEN-GIO S, et al. Advances in Neural Information Processing Systems: vol. 30. Curran Associates, Inc., 2017. https://proceedings.neurips.cc/paper_files/paper/2017/file/9ef2ed4b7fd2c810847ffa5fa85bce38-Paper.pdf.
- [15] ABDAR M, POURPANAH F, HUSSAIN S, et al. A Review of Uncertainty Quantification in Deep Learning: Techniques, Applications and Challenges[J/OL]. CoRR, 2020, abs/2011.06225. arXiv: 2011.06225. <https://arxiv.org/abs/2011.06225>.
- [16] KULLBACK S, LEIBLER R A. On Information and Sufficiency[J/OL]. The Annals of Mathematical Statistics, 1951, 22(1): 79-86 [2025-04-13]. <http://www.jstor.org/stable/2236703>.

- [17] ADAMS D Q, ALDUINO C, ALFONSO K, et al. Measurement of the $2\nu\beta\beta$ Decay Half-Life of ^{130}Te with CUORE[J/OL]. Phys. Rev. Lett., 2021, 126: 171801. <https://link.aps.org/doi/10.1103/PhysRevLett.126.171801>.
- [18] SCIELZO N D, CALDWELL S, SAVARD G, et al. Double- β -decay Q values of ^{130}Te , ^{128}Te , and ^{120}Te [J/OL]. Phys. Rev. C, 2009, 80: 025501. <https://link.aps.org/doi/10.1103/PhysRevC.80.025501>.
- [19] Von KROSIGK B. Measurement of proton and alpha-particle quenching in LAB based scintillators and determination of spectral sensitivities to supernova neutrinos in the SNO+ detector [D]. Dresden: Technical University of Dresden, 2015.
- [20] ANDERSON M. Studies of machine learning to improve sensitivities for the SNO+ detector and a local p-type point contact high purity germanium detector[D]. Kingston: Queen's University, 2024.
- [21] POWELL M J. An efficient method for finding the minimum of a function of several variables without calculating derivatives[J]. The computer journal, 1964, 7(2): 155-162.

致 谢

感谢部分。

附录 A 池化

池化 (Pooling)，也称为子采样 (Subsampling) 或下采样 (Downsampling)，是深度学习中常用的一种操作，尤其是在卷积神经网络 (CNN) 和处理序列数据的模型中。其主要目的包括：

1. 降低维度 (**Dimensionality Reduction**)：池化层通过聚合输入特征图 (Feature Map) 或序列中的信息，显著减少后续层的参数数量和计算量。这有助于控制模型的复杂度并防止过拟合。
2. 特征不变性 (**Feature Invariance**)：池化操作可以使模型对输入中的微小变化 (如平移、旋转) 更加鲁棒。例如，最大池化 (Max Pooling) 关注区域内最显著的特征，即使特征的位置发生轻微移动，池化后的输出也可能保持不变。
3. 增大感受野 (**Receptive Field**)：在 CNN 中，池化层可以有效地增大后续卷积层的感受野，使得网络能够捕捉到更大范围的上下文信息。

常见的池化操作类型有：

- 最大池化 (**Max Pooling**)：在一个局部区域 (池化窗口) 内，选择最大的特征值作为输出。它倾向于保留最显著的特征。
- 平均池化 (**Average Pooling**)：计算一个局部区域内所有特征值的平均值作为输出。它保留了区域内的整体信息。
- 全局池化 (**Global Pooling**)：将整个特征图或序列缩减为一个单一的值 (或一个固定大小的向量)。例如，全局平均池化 (Global Average Pooling, GAP) 计算整个特征图的平均值。全局池化常用于将卷积层的输出连接到全连接层，或者在处理序列数据时聚合整个序列的信息。

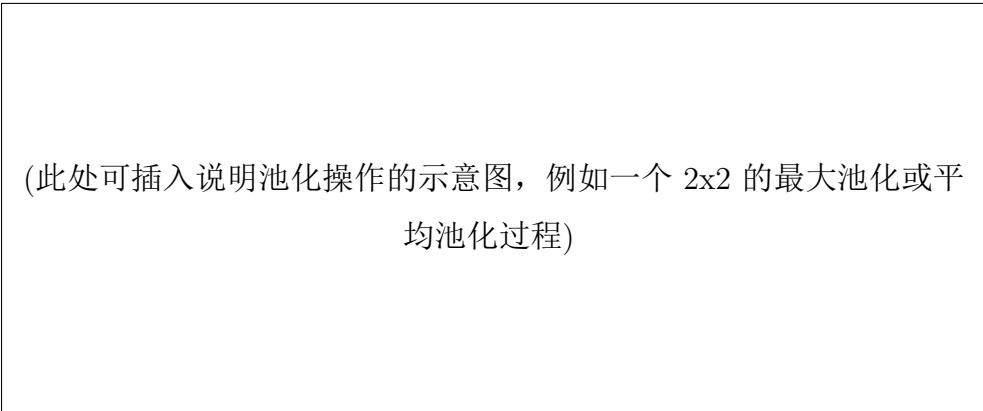


图 A.1: 池化操作示意图 (例如: 最大池化)

A.1 模型中的池化应用：带掩码的平均池化

在这个特定的模型中，输入是可变长度的 PMT 命中序列。经过特征嵌入和 GPT 编码器处理后，我们得到一个形状为 $(T \times d_{in})$ 的输出序列 X_{enc} ，其中 T 是序列长度 (可能包含填充)， d_{in} 是模型的内部嵌入维度。

由于输入序列的长度 T 是可变的，并且为了进行最终的顶点位置回归 (需要一个固定大小的输入)，我们需要将这个可变长度的序列 X_{enc} 聚合 (或“池化”) 成一个单一的、固定大小的向量。

该模型采用的是带掩码的平均池化 (Masked Average Pooling)。选择这种方法的原因如下：

- 处理变长序列：需要一种能将不同长度的序列映射到相同维度输出的方法。
- 全局信息聚合：目标是基于整个事件 (所有有效命中) 的信息来预测顶点，因此需要聚合整个序列的信息，而不是局部信息。全局池化是实现这一目标的自然选择。
- 忽略填充：由于较短的序列会被填充 (padding) 到最大长度 ‘max_seq_len’，在聚合信息时不应考虑这些填充部分。掩码 (Mask) 机制用于识别并忽略这些填充位置。
- 平均池化的选择：平均池化考虑了所有有效命中的贡献，将它们的特征表示进行平均，得到一个能代表整个事件 “平均” 特征的向量。相比之下，最大池化可能只关注最 “突出”的少数命中。对于顶点重建任务，综合所有命中的信息通常更合理。

A.1.1 具体实现

令 $X_{enc} = [x_1, x_2, \dots, x_T]^T \in \mathbb{R}^{T \times d_{in}}$ 为 GPT 编码器的输出序列，其中 $x_i \in \mathbb{R}^{d_{in}}$ 。令 $M = [m_1, m_2, \dots, m_T]$ 为对应的二进制掩码，其中 $m_i = 1$ 表示第 i 个位置是有效命中， $m_i = 0$ 表示该位置是填充。

带掩码的平均池化计算如下：

$$\bar{x}_{pool} = \frac{\sum_{i=1}^T x_i \cdot m_i}{\sum_{i=1}^T m_i + \epsilon}$$

这个公式计算了所有有效命中 ($m_i = 1$) 对应的输出向量 x_i 的和，然后除以有效命中的数量 ($\sum_{i=1}^T m_i$)，得到平均向量 $\bar{x}_{pool} \in \mathbb{R}^{d_{in}}$ 。 ϵ 是一个小的常数以防止除零。

这个池化后的向量 \bar{x}_{pool} 捕获了整个事件序列的全局信息，并具有固定的维度 d_{in} 。它随后被送入层归一化 (Layer Normalization) 和最终的线性输出层，以预测事件的 (x, y, z) 坐标。

附录 B 层归一化

层归一化 (Layer Normalization, LN) 是一种在深度学习中广泛使用的归一化技术，特别是在处理序列数据（如 RNN 和 Transformer）时表现出色。与批量归一化 (Batch Normalization, BN) 不同，BN 是在批次维度上对特征进行归一化，而 LN 是在单个样本的特征维度上进行归一化。

B.0.1 层归一化原理

对于神经网络某一层的一个样本的输出（或隐藏状态）向量 $h \in \mathbb{R}^d$ （其中 d 是该层的神经元数量或特征维度），层归一化首先计算该向量内所有元素的均值 μ 和标准差 σ ：

$$\mu = \frac{1}{d} \sum_{i=1}^d h_i$$

$$\sigma = \sqrt{\frac{1}{d} \sum_{i=1}^d (h_i - \mu)^2 + \epsilon}$$

其中 h_i 是向量 h 的第 i 个元素， ϵ 是一个很小的常数以防止除零。

然后，使用计算出的均值和标准差对该样本的输出向量 h 进行归一化：

$$\hat{h}_i = \frac{h_i - \mu}{\sigma}$$

最后，为了保持模型的表达能力，LN 引入了两个可学习的参数：增益 (gain) $\gamma \in \mathbb{R}^d$ 和偏置 (bias) $\beta \in \mathbb{R}^d$ （维度与 h 相同）。最终的输出 $LN(h)$ 计算如下：

$$LN(h)_i = \gamma_i \hat{h}_i + \beta_i$$

γ 和 β 在训练过程中与其他模型参数一起学习，允许网络自适应地缩放和平移归一化后的特征。

B.0.2 层归一化的优势

1. 独立于批次大小 (**Batch Size Independent**): LN 的计算完全在单个样本内部进行, 不依赖于批次中的其他样本, 因此它在批次大小很小 (甚至为 1) 或变化时也能稳定工作, 这对于 RNN 或处理变长序列的模型尤其有利。
2. 适用于序列数据: 对于变长的序列数据, 不同时间步的统计特性可能不同, BN 在这种情况下可能效果不佳, 而 LN 对每个时间步独立进行归一化, 更具鲁棒性。
3. 稳定训练动态: 与 BN 类似, LN 有助于平滑损失曲面, 稳定梯度, 加速模型收敛。

B.0.3 在模型中的应用

根据前文池化章节的描述, 在模型中, 层归一化的应用紧随在带掩码的平均池化 (**Masked Average Pooling**) 步骤之后:

1. 输入: 带掩码的平均池化层输出了一个固定维度的向量 $\bar{x}_{pool} \in \mathbb{R}^{d_{in}}$ 。这个向量聚合了来自 GPT 编码器输出的整个事件序列的全局信息。
2. 处理: 这个池化后的向量 \bar{x}_{pool} 被直接送入一个层归一化层。LN 层会计算 \bar{x}_{pool} 向量内部 d_{in} 个特征元素的均值和标准差, 并对其进行归一化和仿射变换 (使用可学习的 γ 和 β 参数)。
3. 目的: 在将聚合后的特征向量 \bar{x}_{pool} 输入到最终的线性输出层之前应用 LN, 主要目的是稳定这一层输入的分布, 减少内部协变量偏移 (Internal Covariate Shift), 有助于后续线性层更好地学习从全局特征到最终 (x, y, z) 坐标的映射, 可能加速训练收敛并提高模型的泛化能力。
4. 输出: 经过层归一化处理后的向量, 再被送入最终的线性层进行坐标预测。

附录 C 源代码

本章列出了用于模型训练、验证、测试以及与传统方法对比分析的核心 Python 脚本。

Listing C.1: 模型训练、验证与测试对比脚本 (train_val_test_compare_v1.py)

```
1 #!/usr/bin/env python
2
3 import os
4
5 import argparse
6
7 import numpy as np
8
9 from glob import glob
10
11 import uproot
12
13 import torch
14
15 import torch.nn as nn
16
17 import torch.nn.functional as F
18
19 from torch.utils.data import Dataset, DataLoader
20
21 import logging # 用于写 training.log
22
23 import matplotlib
24
25 matplotlib.use('Agg')
26
27 import matplotlib.pyplot as plt
28
29
30 #####
31
32 # 1. 参数解析
33 #####
34
35 def get_args():
36
37     parser = argparse.ArgumentParser(description="Stream-
38         based\u00d7training\u00d7val\u00d7test\u00d7SF\u00d7vs\u00d7DL\u00d7compare.")
39
40     parser.add_argument("--root_dir", type=str, required=
41
42         True,
43
44             help="包含所有\u00d7.root\u00d7文件的目录")
45
46     parser.add_argument("--tree_name", type=str, default="
```

```

    event_ntuple")
24 parser.add_argument("--pmt_position_file", type=str,
25     required=True,
26             help="PMT坐标文件，与训练脚本一致"
27 )
28
29 # 数据cut + mm->cm
30 parser.add_argument("--min_hits", type=int, default=1)
31 parser.add_argument("--max_hits", type=int, default
32 =99999)
33 parser.add_argument("--min_energy", type=float,
34     default=0.0)
35 parser.add_argument("--max_energy", type=float,
36     default=99999.0)
37 parser.add_argument("--pos_cut", nargs=6, type=float,
38             default=[-9999,9999, -9999,9999,
39             -9999,9999],
40             help="[x_min|x_max|y_min|y_max|
41             z_min|z_max],|mm")
42 parser.add_argument("--time_min", type=float, default
43 =0.0)
44 parser.add_argument("--time_max", type=float, default
45 =9999.0)
46 parser.add_argument("--max_seq_len", type=int, default
47 =1000)
48
49 parser.add_argument("--hit_features", nargs="+",
50     default=["time", "x", "y", "z"])
51 parser.add_argument("--embed_dims", nargs="+", type=
52     int, default=[8,8,8,8])
53 parser.add_argument("--sort_mode", type=str, choices=[
54     "time", "pmt_id"],
55             default="time")

```

```

43
44     # 流式: chunk_size => 一次处理多少 root 文件, 避免OOM
45     parser.add_argument("--chunk_size", type=int, default=
46         =5,
47             help="一次读取 chunk_size 个 root
48             文件 => parse_events => random
49             split_train/val/test => train
50             => accumulate test?")
51
52     # 事件级随机拆分比例 (train/val/test)
53     parser.add_argument("--train_ratio", type=float,
54         default=0.7)
55     parser.add_argument("--val_ratio", type=float, default=
56         =0.2)
57
58     parser.add_argument("--train_epochs", type=int,
59         default=10)
60     parser.add_argument("--batch_size", type=int, default=
61         =32)
62     parser.add_argument("--lr", type=float, default=1e-3)
63     parser.add_argument("--eta_min", type=float, default=1
64         e-5)
65     parser.add_argument("--early_stop_patience", type=int,
66         default=5)
67
68     parser.add_argument("--nhead", type=int, default=4)
69     parser.add_argument("--num_layers", type=int, default=
70         =4)
71     parser.add_argument("--dim_ff", type=int, default=256)
72
73     parser.add_argument("--num_gpus", type=int, default=1)
74     parser.add_argument("--save_model", type=str, default=
75         "best_model.pth")

```

```

64     parser.add_argument("--save_loss_fig", type=str,
65                         default="loss_curve.png")
66
67     # WeightedMSELoss
68     parser.add_argument("--outlier_threshold", type=float,
69                         default=10.0)
70
71     # 指定存放训练日志
72     parser.add_argument("--log_file", type=str, default="
73                             training.log",
74                             help="日志文件名，用于记录训练信息
75                             ")
76
77     # 测试集最终对比时：将新生成的 _DL.root 放在
78     # out_root_dir
79     parser.add_argument("--out_root_dir", type=str,
80                         default="dl_root_files",
81                         help="用户指定的新 ROOT 文件输出目
82                             录")
83
84     return parser.parse_args()
85
86
87     #####
88
89     # 2. 设置 logging
90     #####
91
92     def setup_logging(log_file):
93
94         logging.basicConfig(
95             filename= log_file,
96             filemode= "w",
97

```

```

88         level= logging.INFO,
89
90         format= "%(asctime)s - %(levelname)s - %(message)s"
91         "
92     )
93
94     # 同时在屏幕输出
95     console= logging.StreamHandler()
96     console.setLevel(logging.INFO)
97
98     formatter= logging.Formatter("%(asctime)s - %(levelname)s - %(message)s")
99
100    console.setFormatter(formatter)
101
102    logging.getLogger("").addHandler(console)
103
104
105    ######
106
107    # 3. load_pmt_positions
108
109    #####
110
111    def load_pmt_positions(pmt_file):
112
113        """
114            解析 pmt_position_file，返回 pmt_dict={pid:(px,py,pz)}
115            }, pmt_id_max
116
117        """
118
119        pmt_dict= {}
120
121        pmt_id_max= 0
122
123
124        with open(pmt_file,"r") as pf:
125
126            for line in pf:
127
128                arr= line.strip().split()
129
130                if len(arr)<4:
131
132                    continue
133
134                pid= int(arr[0])
135
136                px,py,pz= map(float, arr[1:4])
137
138                pmt_dict[pid]= (px,py,pz)
139
140                if pid> pmt_id_max:
141
142                    pmt_id_max= pid
143
144
145        return pmt_dict, pmt_id_max

```

```
118
119 ######
120 # 4. chunkify root file list
121 #####
122 def chunkify_files(file_list, chunk_size):
123     for i in range(0, len(file_list), chunk_size):
124         yield file_list[i : i+chunk_size]
125
126 #####
127 # 5. 读取 chunk files => parse events => shuffle => split
128     => train/val/test
129 #####
130
131 def load_chunk_events(file_chunk, tree_name, pmt_dict,
132 args):
133     """
134     一次加载这 chunk_size 个 root 文件 => parse => cut =>
135     事件列表。
136
137     返回 events = [ { 'time':..., 'pid':..., 'xx':..., 'yy'
138         :..., 'zz':..., 'target_cm':..., 'sf_x_mm':..., 'energy':..., ... }, ... ]
139
140     """
141
142     events= []
143
144     for rf in file_chunk:
145
146         data= uproot.concatenate(f"{rf}:{tree_name}",
147             library="np")
148
149         N= len(data["gtid"])
150
151         for i in range(N):
152
153             nh= data["nhits"][i]
154
155             if not(args.min_hits<= nh <= args.max_hits):
156
157                 continue
158
159             e_v= data["deposit_energy"][i]
160
161             if not(args.min_energy<= e_v <= args.
162
163                 max_energy):
```

```

144         continue
145
146         x_mm= data["position_x"][i]
147
148         y_mm= data["position_y"][i]
149
150         z_mm= data["position_z"][i]
151
152         if not(args.pos_cut[0]<= x_mm<=args.pos_cut[1]
153             and
154                 args.pos_cut[2]<= y_mm<=args.pos_cut[3]
155                     and
156                         args.pos_cut[4]<= z_mm<=args.pos_cut
157                             [5]):
158
159             continue
160
161
162             # mm->cm
163
164             x_cm= x_mm/10.0
165
166             y_cm= y_mm/10.0
167
168             z_cm= z_mm/10.0
169
170
171             ht= data["hit_time_v"][i]
172
173             pid= data["hit_PMT_id_v"][i]
174
175             xx_arr= np.array([pmt_dict.get(pp,(0,0,0))[0]
176                 for pp in pid], dtype=np.float32)
177
178             yy_arr= np.array([pmt_dict.get(pp,(0,0,0))[1]
179                 for pp in pid], dtype=np.float32)
180
181             zz_arr= np.array([pmt_dict.get(pp,(0,0,0))[2]
182                 for pp in pid], dtype=np.float32)
183
184
185             # time cut
186
187             mask_= (ht>=args.time_min)&(ht<=args.time_max)
188
189             ht= ht[mask_]
190
191             pid= pid[mask_]
192
193             xx_arr= xx_arr[mask_]
194
195             yy_arr= yy_arr[mask_]
196
197             zz_arr= zz_arr[mask_]

```

```
171         if len(ht)<1:
172             continue
173
174         if args.sort_mode=="time":
175             sidx= np.argsort(ht)
176
177         else:
178             sidx= np.argsort(pid)
179
180         ht= ht[sidx]
181
182         pid= pid[sidx]
183
184         xx_arr= xx_arr[sidx]
185
186         yy_arr= yy_arr[sidx]
187
188         zz_arr= zz_arr[sidx]
189
190         if len(ht)> args.max_seq_len:
191             ht= ht[:args.max_seq_len]
192             pid= pid[:args.max_seq_len]
193             xx_arr= xx_arr[:args.max_seq_len]
194             yy_arr= yy_arr[:args.max_seq_len]
195             zz_arr= zz_arr[:args.max_seq_len]
196
197
198         # SF
199
200         if "sf_position_x" in data:
201             sfx= data["sf_position_x"][i]
202             sfy= data["sf_position_y"][i]
203             sfz= data["sf_position_z"][i]
204
205         else:
206             sfx= sfy= sfz= 9999
207
208
209         events.append({
210             "time": ht,
211             "pid": pid,
212             "xx": xx_arr,
213             "yy": yy_arr,
214             "zz": zz_arr,
```

```

204             "target_cm": np.array([x_cm,y_cm,z_cm],  

205                         dtype=np.float32),  

206             "sf_x_mm": sfx,  

207             "sf_y_mm": sfy,  

208             "sf_z_mm": sfz,  

209             "energy": e_v,  

210             "nhits": nh  

211         })  

212  

213     return events  

214  

215 #####  

216 # 6. Dataset + DataLoader  

217 #####  

218  

219 class PositionDataset(Dataset):  

220  

221     def __init__(self, events):  

222         self.events= events  

223  

224     def __len__(self):  

225         return len(self.events)  

226  

227     def __getitem__(self, idx):  

228         ev= self.events[idx]  

229         # return hits + target  

230         return (ev["time"], ev["pid"], ev["xx"], ev["yy"],  

231                 ev["zz"], ev["target_cm"])  

232  

233     def collate_fn(batch):  

234         all_t, all_pid, all_x, all_y, all_z, all_tgt, all_mk =  

235             [], [], [], [], [], [], []  

236         max_len = 0  

237         for (ht, pid, xx, yy, zz, tgt) in batch:  

238             l = len(ht)  

239             mk_ = np.zeros((l,), dtype=np.float32)  

240             mk_[:l] = 1  

241             if l > max_len:

```

```
234         max_len = 1
235         all_t.append(ht)
236         all_pid.append(pid)
237         all_x.append(xx)
238         all_y.append(yy)
239         all_z.append(zz)
240         all_tgt.append(tgt)
241         all_mk.append(mk_)

242
243         bt, bp, bx, by, bz, bmk = [], [], [], [], [], []
244         for i in range(len(all_t)):
245             sl = len(all_t[i])
246             t_ = np.zeros((max_len,), dtype=np.float32)
247             p_ = np.zeros((max_len,), dtype=np.int64)
248             x_ = np.zeros((max_len,), dtype=np.float32)
249             y_ = np.zeros((max_len,), dtype=np.float32)
250             z_ = np.zeros((max_len,), dtype=np.float32)
251             m_ = np.zeros((max_len,), dtype=np.float32)

252
253             t_[:sl] = all_t[i]
254             p_[:sl] = all_pid[i]
255             x_[:sl] = all_x[i]
256             y_[:sl] = all_y[i]
257             z_[:sl] = all_z[i]
258             m_[:sl] = all_mk[i]

259
260             bt.append(t_)
261             bp.append(p_)
262             bx.append(x_)
263             by.append(y_)
264             bz.append(z_)
265             bmk.append(m_)
```

```

267     # 一次性将列表转换为 NumPy 数组后，再创建 tensor
268     bt = torch.from_numpy(np.array(bt, dtype=np.float32))
269     bp = torch.from_numpy(np.array(bp, dtype=np.int64))
270     bx = torch.from_numpy(np.array(bx, dtype=np.float32))
271     by = torch.from_numpy(np.array(by, dtype=np.float32))
272     bz = torch.from_numpy(np.array(bz, dtype=np.float32))
273     bmk = torch.from_numpy(np.array(bmk, dtype=np.float32))
274     )
275
276     tg = torch.from_numpy(np.array(all_tgt, dtype=np.
277         float32))
278
279     return (bt, bp, bx, by, bz, bmk, tg)
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297 ######
298
299 # 7. 网络定义
300 #####
301
302 def init_weights_gpt(m):
303
304     if isinstance(m,nn.Linear):
305
306         nn.init.normal_(m.weight, mean=0.0, std=0.02)
307
308         if m.bias is not None:
309
310             nn.init.zeros_(m.bias)
311
312     elif isinstance(m,nn.Embedding):
313
314         nn.init.normal_(m.weight, mean=0.0, std=0.02)
315
316
317
318 class FeatureEmbedding(nn.Module):
319
320     def __init__(self, max_pmt_id, feat_list,
321                  embed_dims_dict):
322
323         super().__init__()
324
325         self.feats= feat_list
326
327         self.embed_dict= nn.ModuleDict()
328
329         for f in self.feats:
330
331             dim= embed_dims_dict[f]
332
333             if f=="pmt_id":
334
335                 self.embed_dict[f]= nn.Embedding(

```

```

                max_pmt_id+1, dim)

297     else:
298         self.embed_dict[f]= nn.Linear(1, dim, bias
299                                     =True)
300
300     def forward(self,t,pid,x,y,z):
301         outs=[]
302         if "time" in self.feats:
303             outs.append(self.embed_dict["time"](t.
304                                         unsqueeze(-1)))
305         if "pmt_id" in self.feats:
306             outs.append(self.embed_dict["pmt_id"](pid))
307         if "x" in self.feats:
308             outs.append(self.embed_dict["x"](x.unsqueeze
309                                         (-1)))
310         if "y" in self.feats:
311             outs.append(self.embed_dict["y"](y.unsqueeze
312                                         (-1)))
313         if "z" in self.feats:
314             outs.append(self.embed_dict["z"](z.unsqueeze
315                                         (-1)))
316
317         return torch.cat(outs, dim=-1)

318
319     class SelfAttention(nn.Module):
320
321         def __init__(self, embed_dim, n_heads, dropout=0.1):
322             super().__init__()
323             self.embed_dim= embed_dim
324             self.n_heads= n_heads
325             self.head_dim= embed_dim//n_heads
326             self.c_attn= nn.Linear(embed_dim, 3*embed_dim)
327             self.c_proj= nn.Linear(embed_dim, embed_dim)
328             self.dropout= nn.Dropout(dropout)
329
330         def forward(self,x):

```

```

324         B,T,C= x.shape
325
326         qkv= self.c_attn(x)
327
328         q,k,v= qkv.split(C, dim=2)
329
330         q= q.reshape(B,T,self.n_heads, self.head_dim).
331             transpose(1,2)
332
333         k= k.reshape(B,T,self.n_heads, self.head_dim).
334             transpose(1,2)
335
336         v= v.reshape(B,T,self.n_heads, self.head_dim).
337             transpose(1,2)
338
339         y= F.scaled_dot_product_attention(q,k,v, None,
340             dropout_p=self.dropout.p)
341
342         y= y.transpose(1,2).reshape(B,T,C)
343
344         return self.c_proj(y)

345
346
347     class AttentionBlock(nn.Module):
348
349         def __init__(self, embed_dim, n_heads, dropout=0.1):
350
351             super().__init__()
352
353             self.ln1= nn.LayerNorm(embed_dim)
354
355             self.attn= SelfAttention(embed_dim,n_heads,dropout
356
357                 )
358
359             self.ln2= nn.LayerNorm(embed_dim)
360
361             self.mlp= nn.Sequential(
362
363                 nn.Linear(embed_dim,4*embed_dim),
364
365                 nn.GELU(),
366
367                 nn.Linear(4*embed_dim, embed_dim),
368
369                 nn.Dropout(dropout)
370
371             )
372
373             def forward(self,x):
374
375                 x= x + self.attn(self.ln1(x))
376
377                 x= x + self.mlp(self.ln2(x))
378
379                 return x

380
381     class GPTEncoder(nn.Module):

```

```

352     def __init__(self, embed_dim, n_heads, num_layers,
353                  dropout=0.1):
354         super().__init__()
355         self.blocks= nn.ModuleList([
356             AttentionBlock(embed_dim,n_heads,dropout) for
357             _ in range(num_layers)
358         ])
359
360     def forward(self,x):
361         for blk in self.blocks:
362             x= blk(x)
363
364         return x
365
366
367     class GPTRegressor(nn.Module):
368
369         def __init__(self,in_dim,n_heads,num_layers,out_dim=3,
370                      max_len=1000,dropout=0.1):
371             super().__init__()
372             self.inp_linear= nn.Linear(in_dim,in_dim)
373             self.pos_emb= nn.Embedding(max_len,in_dim)
374             self.encoder= GPEncoder(in_dim,n_heads,num_layers
375                                     ,dropout)
376             self.ln_f= nn.LayerNorm(in_dim)
377             self.fc_out= nn.Linear(in_dim,out_dim)
378             self.apply(init_weights_gpt)
379
380
381         def forward(self, feats, mask):
382             B,T,C= feats.shape
383             x= self.inp_linear(feats)
384             idx= torch.arange(T, device=x.device).unsqueeze(0)
385             x= x + self.pos_emb(idx)
386             x= self.encoder(x)
387             m_= mask.unsqueeze(-1)
388             x_pooled= (x*m_).sum(dim=1)/(m_.sum(dim=1)+1e-12)
389             x_pooled= self.ln_f(x_pooled)

```

```

381         return self.fc_out(x_pooled)

382

383 ######
384 # 8. WeightedMSELoss
385 #####
386 class WeightedMSELoss(nn.Module):
387     def __init__(self, threshold=10.0, scale_outlier=2.0):
388         super().__init__()
389         self.threshold = threshold
390         self.scale_outlier = scale_outlier
391     def forward(self, pred, tgt):
392         # pred,tgt: (B,3), unit= cm
393         err = pred - tgt
394         dr = torch.sqrt((err**2).sum(dim=1)+1e-12)
395         mask_big = (dr > self.threshold)
396         dr2 = dr**2
397         dr2[mask_big] = dr2[mask_big]*self.scale_outlier
398         return dr2.mean()

399

400 #####
401 # 9. 训练辅助函数
402 #####
403 def train_on_chunk(model, emb_module, event_subset,
404                     batch_size, criterion, optimizer, device):
405     """对给定一批 'event_subset' 事件做一次 mini-batch 训练（只返回平均 loss）."""
406
407     ds_chunk = PositionDataset(event_subset)
408     dl_chunk = DataLoader(ds_chunk, batch_size=batch_size,
409                           shuffle=False, collate_fn=collate_fn)
410     model.train()
411     emb_module.train()

```

```

411     total_loss=0.0
412
413
414     for batch in dl_chunk:
415         bt, bp, bx, by, bz, mk, tg= [x.to(device) for x in batch]
416
417         optimizer.zero_grad()
418
419         feats= emb_module(bt, bp, bx, by, bz)
420
421         out= model(feats, mk)
422
423         loss= criterion(out, tg)
424
425         loss.backward()
426
427         optimizer.step()
428
429
430         bs= bt.size(0)
431
432         total_loss+= loss.item()* bs
433
434         total_samples+= bs
435
436
437
438     if total_samples>0:
439
440         return total_loss/ total_samples
441
442     else:
443
444         return 9999.9
445
446
447
448 #####
449 # 10. 主函数
450 #####
451
452 def main():
453
454     args= get_args()
455
456
457     # 设置日志
458
459     if os.path.exists(args.log_file):
460
461         os.remove(args.log_file)
462
463         setup_logging(args.log_file)
464
465         logging.info("START SCRIPT with args=%s", str(args))

```

```

443
444     # 准备输出目录
445     os.makedirs(args.save_pred_fig_dir, exist_ok=True)
446     os.makedirs(args.out_root_dir, exist_ok=True) # 用户
447             指定存放 _DL.root 的目录
448
449     # 设备
450     if torch.cuda.is_available() and args.num_gpus>0:
451         device= torch.device("cuda:0")
452         logging.info(f"Using GPU:{device}")
453     else:
454         device= torch.device("cpu")
455         logging.info("Using CPU")
456
457     # 读 pmt
458     pmt_dict, pmt_id_max= load_pmt_positions(args.
459             pmt_position_file)
460
461     # 收集root文件
462     file_list= sorted(glob(os.path.join(args.root_dir,"*.
463             root")))
464
465     if not file_list:
466         logging.error("No root in %s", args.root_dir)
467         return
468
469     # 构建模型
470     feat_dict={}
471     for f,d_ in zip(args.hit_features, args.embed_dims):
472         feat_dict[f]= d_
473     emb_in_dim= sum(args.embed_dims)
474     emb_module= FeatureEmbedding(pmt_id_max, args.
475             hit_features, feat_dict).to(device)
476     model= GPTRegressor(

```

```

472         in_dim= emb_in_dim,
473
474         n_heads= args.nhead,
475
476         num_layers= args.num_layers,
477
478         out_dim= 3,
479
480         max_len= args.max_seq_len,
481
482         dropout= 0.1
483
484     ).to(device)
485
486
487     if device.type=="cuda" and args.num_gpus>1:
488
489         model= nn.DataParallel(model)
490
491         emb_module= nn.DataParallel(emb_module)
492
493
494         criterion= WeightedMSELoss(args.outlier_threshold,
495
496             args.scale_outlier)
497
498         optimizer= torch.optim.Adam(list(model.parameters())+
499
500             list(emb_module.parameters()), lr=args.lr)
501
502         scheduler= torch.optim.lr_scheduler.CosineAnnealingLR(
503
504             optimizer, T_max=args.train_epochs, eta_min=args.
505
506             eta_min)
507
508
509         # early stop
510
511         best_val_loss= float("inf")
512
513         wait_cnt= 0
514
515         train_losses= []
516
517         val_losses= []
518
519
520         # 准备用于 test compare
521
522         test_events_all= [] # 累加各chunk的 test events
523
524
525         # ====== 训练 EPOCH loop ======
526
527         for ep in range(args.train_epochs):
528
529             logging.info(f"==>EPOCH {ep+1}/{args.train_epochs}
530
531             }==>")

```

```

500
501     # chunkify files => for each chunk => load =>
502         split => train
503
504     chunk_files_list= list(chunkify_files(file_list,
505                               args.chunk_size))
506
507     chunk_sum_loss= 0.0
508
509     chunk_sum_count= 0
510
511     sum_val_loss= 0.0
512
513     sum_val_count= 0
514
515
516
517
518
519
520
521
522
523
524
525
526

```

```

# chunkify files => for each chunk => load =>
split => train

chunk_files_list= list(chunkify_files(file_list,
                                      args.chunk_size))

chunk_sum_loss= 0.0

chunk_sum_count= 0

sum_val_loss= 0.0

sum_val_count= 0

for ic, chunk_fs in enumerate(chunk_files_list,
                               start=1):
    events_chunk= load_chunk_events(chunk_fs, args
                                     .tree_name, pmt_dict, args)

    if not events_chunk:
        continue

    # shuffle chunk events
    np.random.shuffle(events_chunk)

    Nch= len(events_chunk)

    nch_train= int(Nch* args.train_ratio)

    nch_val= int(Nch* args.val_ratio)

    nch_test= Nch- nch_train- nch_val

    train_part= events_chunk[:nch_train]

    val_part= events_chunk[nch_train: nch_train+
                           nch_val]

    test_part= events_chunk[nch_train+nch_val:]  #

                    for final compare

# 累加 test

test_events_all.extend(test_part)

# 训练 (train_part)

```

```

527         if train_part:
528             train_loss= train_on_chunk(model,
529                                         emb_module, train_part, args.batch_size
530                                         , criterion, optimizer, device)
531             chunk_sum_loss+= train_loss* len(
532                                         train_part)
533             chunk_sum_count+= len(train_part)
534             logging.info(f"[Epoch_{ep+1}]_{chunk}_{ic}/{len(chunk_files_list)}]train_loss={train_loss:.4f},events={len(train_part)}")
535
536     # 验证 (val_part)
537     if val_part:
538         ds_val= PositionDataset(val_part)
539         dl_val= DataLoader(ds_val, batch_size=args
540                           .batch_size, shuffle=False, collate_fn=
541                           collate_fn)
542         model.eval()
543         emb_module.eval()
544         val_loss= 0.0
545         val_count=0
546         with torch.no_grad():
547             for batch in dl_val:
548                 bt, bp, bx, by, bz, mk, tg= [x.to(device)
549                                              for x in batch]
550                 feats= emb_module(bt, bp, bx, by, bz)
551                 out= model(feats, mk)
552                 loss= criterion(out, tg)
553                 bs= bt.size(0)
554                 val_loss+= loss.item()* bs
555                 val_count+= bs
556
557             if val_count>0:

```

```

551             val_loss/= val_count
552
553             sum_val_loss+= val_loss* val_count
554             sum_val_count+= val_count
555             logging.info(f"[Epoch_{ep+1}]_{chunk}_{ic}/{len(chunk_files_list)}] val_loss={val_loss:.4f}, events={val_count}")
556
557         del events_chunk, train_part, val_part,
558         test_part
559         torch.cuda.empty_cache()
560
561     # end chunk loop for epoch
562     if chunk_sum_count>0:
563         epoch_train_loss= chunk_sum_loss /
564         chunk_sum_count
565     else:
566         epoch_train_loss=9999.9
567     train_losses.append(epoch_train_loss)
568
569     if sum_val_count>0:
570         epoch_val_loss= sum_val_loss / sum_val_count
571     else:
572         epoch_val_loss=9999.9
573     val_losses.append(epoch_val_loss)
574
575     scheduler.step()
576
577     logging.info(f"===[EPOCH_{ep+1}] DONE: TrainLoss={epoch_train_loss:.4f}, ValLoss={epoch_val_loss:.4f}")
578
579     # early stop
580     if epoch_val_loss< best_val_loss:

```

```

578         best_val_loss= epoch_val_loss
579
580         torch.save(model.state_dict(), args.save_model
581             )
582
583         torch.save(emb_module.state_dict(), args.
584             save_model+"_emb")
585
586         logging.info("**[Info] Best model updated!")
587
588         wait_cnt=0
589
590     else:
591
592         wait_cnt+=1
593
594         if wait_cnt>= args.early_stop_patience:
595
596             logging.info("**[Info] Early stopping
597                         triggered!")
598
599             break
600
601
602     # 画 loss 曲线
603
604     plt.figure()
605
606     plt.plot(range(1,len(train_losses)+1), train_losses,
607             label="Train")
608
609     plt.plot(range(1,len(val_losses)+1), val_losses, label
610             ="Val")
611
612     plt.xlabel("Epoch")
613
614     plt.ylabel("WeightedMSELoss")
615
616     plt.legend()
617
618     plt.tight_layout()
619
620     loss_fig= os.path.join(args.save_pred_fig_dir, args.
621             save_loss_fig)
622
623     plt.savefig(loss_fig)
624
625     plt.close()
626
627     logging.info(f"Training done. Loss curve=>{loss_fig}
628
629             ")
630
631
632     # 加载 best
633
634     if isinstance(model, nn.DataParallel):

```

```

604         model.module.load_state_dict(torch.load(args.
605                                         save_model, map_location=device))
606         emb_module.module.load_state_dict(torch.load(args.
607                                         save_model+"_emb", map_location=device))
608     else:
609         model.load_state_dict(torch.load(args.save_model,
610                                         map_location=device))
611         emb_module.load_state_dict(torch.load(args.
612                                         save_model+"_emb", map_location=device))
613
614         # ===== 最终 test 对比 SF vs DL => 画对比图 & 输出 root =====
615         logging.info(f"Test_events_all={len(test_events_all)}")
616
617         if not test_events_all:
618             logging.info("No test events collected, skip final compare.")
619
620             return
621
622
623             dl_x,dl_y,dl_z= [],[],[]
624             sf_x,sf_y,sf_z= [],[],[]
625             rx,ry,rz= [],[],[]
626             e_list= []
627             nh_list= []
628
629             with torch.no_grad():
630                 for ev in test_events_all:
631                     ht= ev["time"]
632                     pid= ev["pid"]
633                     xx= ev["xx"]

```

```

630     yy= ev["yy"]
631     zz= ev["zz"]
632     if len(ht)<1:
633         dl_x.append(9999); dl_y.append(9999); dl_z
634             .append(9999)
635         sf_x.append(ev["sf_x_mm"]); sf_y.append(ev
636             ["sf_y_mm"]); sf_z.append(ev["sf_z_mm"])
637             ])
638         r_ = ev["target_cm"]*10.0
639         rx.append(r_[0]); ry.append(r_[1]); rz.
640             append(r_[2])
641         e_list.append(ev["energy"])
642         nh_list.append(ev["nhits"])
643         continue
644
645     mk= np.ones((len(ht)), dtype=np.float32)
646     if len(ht)> args.max_seq_len:
647         ht= ht[:args.max_seq_len]
648         pid= pid[:args.max_seq_len]
649         xx= xx[:args.max_seq_len]
650         yy= yy[:args.max_seq_len]
651         zz= zz[:args.max_seq_len]
652         mk= mk[:args.max_seq_len]
653
654         b_time= torch.tensor(ht, dtype=torch.float32).
655             unsqueeze(0).to(device)
656         b_pid= torch.tensor(pid, dtype=torch.long).
657             unsqueeze(0).to(device)
658         b_x= torch.tensor(xx, dtype=torch.float32).
659             unsqueeze(0).to(device)
660         b_y= torch.tensor(yy, dtype=torch.float32).
661             unsqueeze(0).to(device)
662         b_z= torch.tensor(zz, dtype=torch.float32).

```

```

        unsqueeze(0).to(device)

655      b_m= torch.tensor(mk, dtype=torch.float32).
        unsqueeze(0).to(device)

656      feats= emb_module(b_time,b_pid,b_x,b_y,b_z)

657      out_cm= model(feats,b_m) # (1,3)

658      out_mm= out_cm[0].cpu().numpy()*10.0 # => mm

659      dl_x.append(out_mm[0])

660      dl_y.append(out_mm[1])

661      dl_z.append(out_mm[2])

662

663      sf_x.append(ev["sf_x_mm"])

664      sf_y.append(ev["sf_y_mm"])

665      sf_z.append(ev["sf_z_mm"])

666

667      real_mm= ev["target_cm"]*10.0

668      rx.append(real_mm[0]); ry.append(real_mm[1]);
      rz.append(real_mm[2])

669

670      e_list.append(ev["energy"])

671      nh_list.append(ev["nhits"])

672

673      dl_x= np.array(dl_x); dl_y= np.array(dl_y); dl_z= np.
        array(dl_z)

674      sf_x= np.array(sf_x); sf_y= np.array(sf_y); sf_z= np.
        array(sf_z)

675      rx= np.array(rx); ry= np.array(ry); rz= np.array(rz)

676      e_list= np.array(e_list)

677      nh_list= np.array(nh_list)

678

679      dl_dist= np.sqrt((dl_x - rx)**2 + (dl_y - ry)**2 + (
        dl_z - rz)**2)

680      sf_dist= np.sqrt((sf_x - rx)**2 + (sf_y - ry)**2 + (
        sf_z - rz)**2)

```

```

681
682     mean_dl= dl_dist.mean()
683     std_dl= dl_dist.std()
684     mean_sf= sf_dist.mean()
685     std_sf= sf_dist.std()
686     logging.info(f"Test: Dist => DL mean={mean_dl:.2f}, std
687                   ={std_dl:.2f}, SF mean={mean_sf:.2f}, std={std_sf
688                   :.2f}")
689
690     # compare_3Ddistance_SF_DL
691     plt.figure()
692     label_dl= f"DL: mean={mean_dl:.2f}, std={std_dl:.2f}"
693     label_sf= f"SF: mean={mean_sf:.2f}, std={std_sf:.2f}"
694     bins=50
695     plt.hist(dl_dist, bins=bins, alpha=0.5, edgecolor='
696               black', color='red', label=label_dl)
697     plt.hist(sf_dist, bins=bins, alpha=0.5, edgecolor='
698               black', color='blue', label=label_sf)
699     plt.xlabel("3D Distance (mm)")
700     plt.ylabel("Count")
701     plt.title("Test: SF vs DL 3D distance")
702     plt.legend()
703     plt.tight_layout()
704     compare_fig= os.path.join(args.save_pred_fig_dir, "
705                               compare_3Ddistance_SF_DL_v1.png")
706     plt.savefig(compare_fig)
707     plt.close()
708     logging.info(f"Saved=>{compare_fig}")

# resolution vs energy
nbins=10
e_min,e_max= e_list.min(), e_list.max()
be= np.linspace(e_min,e_max,nbins+1)

```

```

709         bc= 0.5*(be[:-1]+ be[1:])
710
711         res_sf= []
712
713         res_dl= []
714
715         for i in range(nbins):
716
717             sel= (e_list>= be[i]) & (e_list< be[i+1])
718
719             if np.sum(sel)==0:
720
721                 res_sf.append(0)
722
723                 res_dl.append(0)
724
725             else:
726
727                 res_sf.append(sf_dist[sel].mean())
728
729                 res_dl.append(dl_dist[sel].mean())
730
731
732         plt.figure()
733
734         plt.plot(bc, res_sf, '-o', color='blue', label="SF")
735         plt.plot(bc, res_dl, '-o', color='red', label="DL")
736
737         plt.xlabel("Energy (MeV)")
738
739         plt.ylabel("Mean 3D distance (mm)")
740
741         plt.title("Resolution vs Energy (Test set)")
742
743         plt.legend()
744
745         plt.tight_layout()
746
747         fig_e= os.path.join(args.save_pred_fig_dir, ""
748                           "resolution_vs_energy_v1.png")
749
750         plt.savefig(fig_e)
751
752         plt.close()
753
754         logging.info(f"Saved=>{fig_e}")
755
756
757         # resolution vs nhits
758
759         nbins2=10
760
761         nh_min, nh_max= nh_list.min(), nh_list.max()
762
763         bh= np.linspace(nh_min, nh_max, nbins2+1)
764
765         bc2= 0.5*(bh[:-1]+ bh[1:])
766
767         sf2, dl2= [], []
768
769         for i in range(nbins2):
770
771             sel= (e_list>= bh[i]) & (e_list< bh[i+1])
772
773             if np.sum(sel)==0:
774
775                 sf2.append(0)
776
777                 dl2.append(0)
778
779             else:
780
781                 sf2.append(sf_dist[sel].mean())
782
783                 dl2.append(dl_dist[sel].mean())
784
785         plt.figure()
786
787         plt.plot(bc2, sf2, '-o', color='blue', label="SF")
788         plt.plot(bc2, dl2, '-o', color='red', label="DL")
789
790         plt.xlabel("Energy (MeV)")
791
792         plt.ylabel("Mean 3D distance (mm)")
793
794         plt.title("Resolution vs Energy (Test set)")
795
796         plt.legend()
797
798         plt.tight_layout()
799
800         fig_n= os.path.join(args.save_pred_fig_dir, ""
801                           "resolution_vs_nhits_v1.png")
802
803         plt.savefig(fig_n)
804
805         plt.close()
806
807         logging.info(f"Saved=>{fig_n}")
808
809
810         # resolution vs nhits
811
812         nbins3=10
813
814         nh_min, nh_max= nh_list.min(), nh_list.max()
815
816         bh= np.linspace(nh_min, nh_max, nbins3+1)
817
818         bc3= 0.5*(bh[:-1]+ bh[1:])
819
820         sf3, dl3= [], []
821
822         for i in range(nbins3):
823
824             sel= (e_list>= bh[i]) & (e_list< bh[i+1])
825
826             if np.sum(sel)==0:
827
828                 sf3.append(0)
829
830                 dl3.append(0)
831
832             else:
833
834                 sf3.append(sf_dist[sel].mean())
835
836                 dl3.append(dl_dist[sel].mean())
837
838         plt.figure()
839
840         plt.plot(bc3, sf3, '-o', color='blue', label="SF")
841         plt.plot(bc3, dl3, '-o', color='red', label="DL")
842
843         plt.xlabel("Energy (MeV)")
844
845         plt.ylabel("Mean 3D distance (mm)")
846
847         plt.title("Resolution vs Energy (Test set)")
848
849         plt.legend()
850
851         plt.tight_layout()
852
853         fig_n= os.path.join(args.save_pred_fig_dir, ""
854                           "resolution_vs_nhits_v2.png")
855
856         plt.savefig(fig_n)
857
858         plt.close()
859
860         logging.info(f"Saved=>{fig_n}")
861
862
863         # resolution vs nhits
864
865         nbins4=10
866
867         nh_min, nh_max= nh_list.min(), nh_list.max()
868
869         bh= np.linspace(nh_min, nh_max, nbins4+1)
870
871         bc4= 0.5*(bh[:-1]+ bh[1:])
872
873         sf4, dl4= [], []
874
875         for i in range(nbins4):
876
877             sel= (e_list>= bh[i]) & (e_list< bh[i+1])
878
879             if np.sum(sel)==0:
880
881                 sf4.append(0)
882
883                 dl4.append(0)
884
885             else:
886
887                 sf4.append(sf_dist[sel].mean())
888
889                 dl4.append(dl_dist[sel].mean())
890
891         plt.figure()
892
893         plt.plot(bc4, sf4, '-o', color='blue', label="SF")
894         plt.plot(bc4, dl4, '-o', color='red', label="DL")
895
896         plt.xlabel("Energy (MeV)")
897
898         plt.ylabel("Mean 3D distance (mm)")
899
900         plt.title("Resolution vs Energy (Test set)")
901
902         plt.legend()
903
904         plt.tight_layout()
905
906         fig_n= os.path.join(args.save_pred_fig_dir, ""
907                           "resolution_vs_nhits_v3.png")
908
909         plt.savefig(fig_n)
910
911         plt.close()
912
913         logging.info(f"Saved=>{fig_n}")
914
915
916         # resolution vs nhits
917
918         nbins5=10
919
920         nh_min, nh_max= nh_list.min(), nh_list.max()
921
922         bh= np.linspace(nh_min, nh_max, nbins5+1)
923
924         bc5= 0.5*(bh[:-1]+ bh[1:])
925
926         sf5, dl5= [], []
927
928         for i in range(nbins5):
929
930             sel= (e_list>= bh[i]) & (e_list< bh[i+1])
931
932             if np.sum(sel)==0:
933
934                 sf5.append(0)
935
936                 dl5.append(0)
937
938             else:
939
940                 sf5.append(sf_dist[sel].mean())
941
942                 dl5.append(dl_dist[sel].mean())
943
944         plt.figure()
945
946         plt.plot(bc5, sf5, '-o', color='blue', label="SF")
947         plt.plot(bc5, dl5, '-o', color='red', label="DL")
948
949         plt.xlabel("Energy (MeV)")
950
951         plt.ylabel("Mean 3D distance (mm)")
952
953         plt.title("Resolution vs Energy (Test set)")
954
955         plt.legend()
956
957         plt.tight_layout()
958
959         fig_n= os.path.join(args.save_pred_fig_dir, ""
960                           "resolution_vs_nhits_v4.png")
961
962         plt.savefig(fig_n)
963
964         plt.close()
965
966         logging.info(f"Saved=>{fig_n}")
967
968
969         # resolution vs nhits
970
971         nbins6=10
972
973         nh_min, nh_max= nh_list.min(), nh_list.max()
974
975         bh= np.linspace(nh_min, nh_max, nbins6+1)
976
977         bc6= 0.5*(bh[:-1]+ bh[1:])
978
979         sf6, dl6= [], []
980
981         for i in range(nbins6):
982
983             sel= (e_list>= bh[i]) & (e_list< bh[i+1])
984
985             if np.sum(sel)==0:
986
987                 sf6.append(0)
988
989                 dl6.append(0)
990
991             else:
992
993                 sf6.append(sf_dist[sel].mean())
994
995                 dl6.append(dl_dist[sel].mean())
996
997         plt.figure()
998
999         plt.plot(bc6, sf6, '-o', color='blue', label="SF")
1000        plt.plot(bc6, dl6, '-o', color='red', label="DL")
1001
1002        plt.xlabel("Energy (MeV)")
1003
1004        plt.ylabel("Mean 3D distance (mm)")
1005
1006        plt.title("Resolution vs Energy (Test set)")
1007
1008        plt.legend()
1009
1010        plt.tight_layout()
1011
1012        fig_n= os.path.join(args.save_pred_fig_dir, ""
1013                           "resolution_vs_nhits_v5.png")
1014
1015        plt.savefig(fig_n)
1016
1017        plt.close()
1018
1019        logging.info(f"Saved=>{fig_n}")
1020
1021
1022        # resolution vs nhits
1023
1024        nbins7=10
1025
1026        nh_min, nh_max= nh_list.min(), nh_list.max()
1027
1028        bh= np.linspace(nh_min, nh_max, nbins7+1)
1029
1030        bc7= 0.5*(bh[:-1]+ bh[1:])
1031
1032        sf7, dl7= [], []
1033
1034        for i in range(nbins7):
1035
1036            sel= (e_list>= bh[i]) & (e_list< bh[i+1])
1037
1038            if np.sum(sel)==0:
1039
1040                sf7.append(0)
1041
1042                dl7.append(0)
1043
1044            else:
1045
1046                sf7.append(sf_dist[sel].mean())
1047
1048                dl7.append(dl_dist[sel].mean())
1049
1050        plt.figure()
1051
1052        plt.plot(bc7, sf7, '-o', color='blue', label="SF")
1053        plt.plot(bc7, dl7, '-o', color='red', label="DL")
1054
1055        plt.xlabel("Energy (MeV)")
1056
1057        plt.ylabel("Mean 3D distance (mm)")
1058
1059        plt.title("Resolution vs Energy (Test set)")
1060
1061        plt.legend()
1062
1063        plt.tight_layout()
1064
1065        fig_n= os.path.join(args.save_pred_fig_dir, ""
1066                           "resolution_vs_nhits_v6.png")
1067
1068        plt.savefig(fig_n)
1069
1070        plt.close()
1071
1072        logging.info(f"Saved=>{fig_n}")
1073
1074
1075        # resolution vs nhits
1076
1077        nbins8=10
1078
1079        nh_min, nh_max= nh_list.min(), nh_list.max()
1080
1081        bh= np.linspace(nh_min, nh_max, nbins8+1)
1082
1083        bc8= 0.5*(bh[:-1]+ bh[1:])
1084
1085        sf8, dl8= [], []
1086
1087        for i in range(nbins8):
1088
1089            sel= (e_list>= bh[i]) & (e_list< bh[i+1])
1090
1091            if np.sum(sel)==0:
1092
1093                sf8.append(0)
1094
1095                dl8.append(0)
1096
1097            else:
1098
1099                sf8.append(sf_dist[sel].mean())
1100
1101                dl8.append(dl_dist[sel].mean())
1102
1103        plt.figure()
1104
1105        plt.plot(bc8, sf8, '-o', color='blue', label="SF")
1106        plt.plot(bc8, dl8, '-o', color='red', label="DL")
1107
1108        plt.xlabel("Energy (MeV)")
1109
1110        plt.ylabel("Mean 3D distance (mm)")
1111
1112        plt.title("Resolution vs Energy (Test set)")
1113
1114        plt.legend()
1115
1116        plt.tight_layout()
1117
1118        fig_n= os.path.join(args.save_pred_fig_dir, ""
1119                           "resolution_vs_nhits_v7.png")
1120
1121        plt.savefig(fig_n)
1122
1123        plt.close()
1124
1125        logging.info(f"Saved=>{fig_n}")
1126
1127
1128        # resolution vs nhits
1129
1130        nbins9=10
1131
1132        nh_min, nh_max= nh_list.min(), nh_list.max()
1133
1134        bh= np.linspace(nh_min, nh_max, nbins9+1)
1135
1136        bc9= 0.5*(bh[:-1]+ bh[1:])
1137
1138        sf9, dl9= [], []
1139
1140        for i in range(nbins9):
1141
1142            sel= (e_list>= bh[i]) & (e_list< bh[i+1])
1143
1144            if np.sum(sel)==0:
1145
1146                sf9.append(0)
1147
1148                dl9.append(0)
1149
1150            else:
1151
1152                sf9.append(sf_dist[sel].mean())
1153
1154                dl9.append(dl_dist[sel].mean())
1155
1156        plt.figure()
1157
1158        plt.plot(bc9, sf9, '-o', color='blue', label="SF")
1159        plt.plot(bc9, dl9, '-o', color='red', label="DL")
1160
1161        plt.xlabel("Energy (MeV)")
1162
1163        plt.ylabel("Mean 3D distance (mm)")
1164
1165        plt.title("Resolution vs Energy (Test set)")
1166
1167        plt.legend()
1168
1169        plt.tight_layout()
1170
1171        fig_n= os.path.join(args.save_pred_fig_dir, ""
1172                           "resolution_vs_nhits_v8.png")
1173
1174        plt.savefig(fig_n)
1175
1176        plt.close()
1177
1178        logging.info(f"Saved=>{fig_n}")
1179
1180
1181        # resolution vs nhits
1182
1183        nbins10=10
1184
1185        nh_min, nh_max= nh_list.min(), nh_list.max()
1186
1187        bh= np.linspace(nh_min, nh_max, nbins10+1)
1188
1189        bc10= 0.5*(bh[:-1]+ bh[1:])
1190
1191        sf10, dl10= [], []
1192
1193        for i in range(nbins10):
1194
1195            sel= (e_list>= bh[i]) & (e_list< bh[i+1])
1196
1197            if np.sum(sel)==0:
1198
1199                sf10.append(0)
1200
1201                dl10.append(0)
1202
1203            else:
1204
1205                sf10.append(sf_dist[sel].mean())
1206
1207                dl10.append(dl_dist[sel].mean())
1208
1209        plt.figure()
1210
1211        plt.plot(bc10, sf10, '-o', color='blue', label="SF")
1212        plt.plot(bc10, dl10, '-o', color='red', label="DL")
1213
1214        plt.xlabel("Energy (MeV)")
1215
1216        plt.ylabel("Mean 3D distance (mm)")
1217
1218        plt.title("Resolution vs Energy (Test set)")
1219
1220        plt.legend()
1221
1222        plt.tight_layout()
1223
1224        fig_n= os.path.join(args.save_pred_fig_dir, ""
1225                           "resolution_vs_nhits_v9.png")
1226
1227        plt.savefig(fig_n)
1228
1229        plt.close()
1230
1231        logging.info(f"Saved=>{fig_n}")
1232
1233
1234        # resolution vs nhits
1235
1236        nbins11=10
1237
1238        nh_min, nh_max= nh_list.min(), nh_list.max()
1239
1240        bh= np.linspace(nh_min, nh_max, nbins11+1)
1241
1242        bc11= 0.5*(bh[:-1]+ bh[1:])
1243
1244        sf11, dl11= [], []
1245
1246        for i in range(nbins11):
1247
1248            sel= (e_list>= bh[i]) & (e_list< bh[i+1])
1249
1250            if np.sum(sel)==0:
1251
1252                sf11.append(0)
1253
1254                dl11.append(0)
1255
1256            else:
1257
1258                sf11.append(sf_dist[sel].mean())
1259
1260                dl11.append(dl_dist[sel].mean())
1261
1262        plt.figure()
1263
1264        plt.plot(bc11, sf11, '-o', color='blue', label="SF")
1265        plt.plot(bc11, dl11, '-o', color='red', label="DL")
1266
1267        plt.xlabel("Energy (MeV)")
1268
1269        plt.ylabel("Mean 3D distance (mm)")
1270
1271        plt.title("Resolution vs Energy (Test set)")
1272
1273        plt.legend()
1274
1275        plt.tight_layout()
1276
1277        fig_n= os.path.join(args.save_pred_fig_dir, ""
1278                           "resolution_vs_nhits_v10.png")
1279
1280        plt.savefig(fig_n)
1281
1282        plt.close()
1283
1284        logging.info(f"Saved=>{fig_n}")
1285
1286
1287        # resolution vs nhits
1288
1289        nbins12=10
1290
1291        nh_min, nh_max= nh_list.min(), nh_list.max()
1292
1293        bh= np.linspace(nh_min, nh_max, nbins12+1)
1294
1295        bc12= 0.5*(bh[:-1]+ bh[1:])
1296
1297        sf12, dl12= [], []
1298
1299        for i in range(nbins12):
1300
1301            sel= (e_list>= bh[i]) & (e_list< bh[i+1])
1302
1303            if np.sum(sel)==0:
1304
1305                sf12.append(0)
1306
1307                dl12.append(0)
1308
1309            else:
1310
1311                sf12.append(sf_dist[sel].mean())
1312
1313                dl12.append(dl_dist[sel].mean())
1314
1315        plt.figure()
1316
1317        plt.plot(bc12, sf12, '-o', color='blue', label="SF")
1318        plt.plot(bc12, dl12, '-o', color='red', label="DL")
1319
1320        plt.xlabel("Energy (MeV)")
1321
1322        plt.ylabel("Mean 3D distance (mm)")
1323
1324        plt.title("Resolution vs Energy (Test set)")
1325
1326        plt.legend()
1327
1328        plt.tight_layout()
1329
1330        fig_n= os.path.join(args.save_pred_fig_dir, ""
1331                           "resolution_vs_nhits_v11.png")
1332
1333        plt.savefig(fig_n)
1334
1335        plt.close()
1336
1337        logging.info(f"Saved=>{fig_n}")
1338
1339
1340        # resolution vs nhits
1341
1342        nbins13=10
1343
1344        nh_min, nh_max= nh_list.min(), nh_list.max()
1345
1346        bh= np.linspace(nh_min, nh_max, nbins13+1)
1347
1348        bc13= 0.5*(bh[:-1]+ bh[1:])
1349
1350        sf13, dl13= [], []
1351
1352        for i in range(nbins13):
1353
1354            sel= (e_list>= bh[i]) & (e_list< bh[i+1])
1355
1356            if np.sum(sel)==0:
1357
1358                sf13.append(0)
1359
1360                dl13.append(0)
1361
1362            else:
1363
1364                sf13.append(sf_dist[sel].mean())
1365
1366                dl13.append(dl_dist[sel].mean())
1367
1368        plt.figure()
1369
1370        plt.plot(bc13, sf13, '-o', color='blue', label="SF")
1371        plt.plot(bc13, dl13, '-o', color='red', label="DL")
1372
1373        plt.xlabel("Energy (MeV)")
1374
1375        plt.ylabel("Mean 3D distance (mm)")
1376
1377        plt.title("Resolution vs Energy (Test set)")
1378
1379        plt.legend()
1380
1381        plt.tight_layout()
1382
1383        fig_n= os.path.join(args.save_pred_fig_dir, ""
1384                           "resolution_vs_nhits_v12.png")
1385
1386        plt.savefig(fig_n)
1387
1388        plt.close()
1389
1390        logging.info(f"Saved=>{fig_n}")
1391
1392
1393        # resolution vs nhits
1394
1395        nbins14=10
1396
1397        nh_min, nh_max= nh_list.min(), nh_list.max()
1398
1399        bh= np.linspace(nh_min, nh_max, nbins14+1)
1400
1401        bc14= 0.5*(bh[:-1]+ bh[1:])
1402
1403        sf14, dl14= [], []
1404
1405        for i in range(nbins14):
1406
1407            sel= (e_list>= bh[i]) & (e_list< bh[i+1])
1408
1409            if np.sum(sel)==0:
1410
1411                sf14.append(0)
1412
1413                dl14.append(0)
1414
1415            else:
1416
1417                sf14.append(sf_dist[sel].mean())
1418
1419                dl14.append(dl_dist[sel].mean())
1420
1421        plt.figure()
1422
1423        plt.plot(bc14, sf14, '-o', color='blue', label="SF")
1424        plt.plot(bc14, dl14, '-o', color='red', label="DL")
1425
1426        plt.xlabel("Energy (MeV)")
1427
1428        plt.ylabel("Mean 3D distance (mm)")
1429
1430        plt.title("Resolution vs Energy (Test set)")
1431
1432        plt.legend()
1433
1434        plt.tight_layout()
1435
1436        fig_n= os.path.join(args.save_pred_fig_dir, ""
1437                           "resolution_vs_nhits_v13.png")
1438
1439        plt.savefig(fig_n)
1440
1441        plt.close()
1442
1443        logging.info(f"Saved=>{fig_n}")
1444
1445
1446        # resolution vs nhits
1447
1448        nbins15=10
1449
1450        nh_min, nh_max= nh_list.min(), nh_list.max()
1451
1452        bh= np.linspace(nh_min, nh_max, nbins15+1)
1453
1454        bc15= 0.5*(bh[:-1]+ bh[1:])
1455
1456        sf15, dl15= [], []
1457
1458        for i in range(nbins15):
1459
1460            sel= (e_list>= bh[i]) & (e_list< bh[i+1])
1461
1462            if np.sum(sel)==0:
1463
1464                sf15.append(0)
1465
1466                dl15.append(0)
1467
1468            else:
1469
1470                sf15.append(sf_dist[sel].mean())
1471
1472                dl15.append(dl_dist[sel].mean())
1473
1474        plt.figure()
1475
1476        plt.plot(bc15, sf15, '-o', color='blue', label="SF")
1477        plt.plot(bc15, dl15, '-o', color='red', label="DL")
1478
1479        plt.xlabel("Energy (MeV)")
1480
1481        plt.ylabel("Mean 3D distance (mm)")
1482
1483        plt.title("Resolution vs Energy (Test set)")
1484
1485        plt.legend()
1486
1487        plt.tight_layout()
1488
1489        fig_n= os.path.join(args.save_pred_fig_dir, ""
1490                           "resolution_vs_nhits_v14.png")
1491
1492        plt.savefig(fig_n)
1493
1494        plt.close()
1495
1496        logging.info(f"Saved=>{fig_n}")
1497
1498
1499        # resolution vs nhits
1500
1501        nbins16=10
1502
1503        nh_min, nh_max= nh_list.min(), nh_list.max()
1504
1505        bh= np.linspace(nh_min, nh_max, nbins16+1)
1506
1507        bc16= 0.5*(bh[:-1]+ bh[1:])
1508
1509        sf16, dl16= [], []
1510
1511        for i in range(nbins16):
1512
1513            sel= (e_list>= bh[i]) & (e_list< bh[i+1])
1514
1515            if np.sum(sel)==0:
1516
1517                sf16.append(0)
1518
1519                dl16.append(0)
1520
1521            else:
1522
1523                sf16.append(sf_dist[sel].mean())
1524
1525                dl16.append(dl_dist[sel].mean())
1526
1527        plt.figure()
1528
1529        plt.plot(bc16, sf16, '-o', color='blue', label="SF")
1530        plt.plot(bc16, dl16, '-o', color='red', label="DL")
1531
1532        plt.xlabel("Energy (MeV)")
1533
1534        plt.ylabel("Mean 3D distance (mm)")
1535
1536        plt.title("Resolution vs Energy (Test set)")
1537
1538        plt.legend()
1539
1540        plt.tight_layout()
1541
1542        fig_n= os.path.join(args.save_pred_fig_dir, ""
1543                           "resolution_vs_nhits_v15.png")
1544
1545        plt.savefig(fig_n)
1546
1547        plt.close()
1548
1549        logging.info(f"Saved=>{fig_n}")
1550
1551
1552        # resolution vs nhits
1553
1554        nbins17=10
1555
1556        nh_min, nh_max= nh_list.min(), nh_list.max()
1557
1558        bh= np.linspace(nh_min, nh_max, nbins17+1)
1559
1560        bc17= 0.5*(bh[:-1]+ bh[1:])
1561
1562        sf17, dl17= [], []
1563
1564        for i in range(nbins17):
1565
1566            sel= (e_list>= bh[i]) & (e_list< bh[i+1])
1567
1568            if np.sum(sel)==0:
1569
1570                sf17.append(0)
1571
1572                dl17.append(0)
1573
1574            else:
1575
1576                sf17.append(sf_dist[sel].mean())
1577
1578                dl17.append(dl_dist[sel].mean())
1579
1580        plt.figure()
1581
1582        plt.plot(bc17, sf17, '-o', color='blue', label="SF")
1583        plt.plot(bc17, dl17, '-o', color='red', label="DL")
1584
1585        plt.xlabel("Energy (MeV)")
1586
1587        plt.ylabel("Mean 3D distance (mm)")
1588
1589        plt.title("Resolution vs Energy (Test set)")
1590
1591        plt.legend()
1592
1593        plt.tight_layout()
1594
1595        fig_n= os.path.join(args.save_pred_fig_dir, ""
1596                           "resolution_vs_nhits_v16.png")
1597
1598        plt.savefig(fig_n)
1599
1600        plt.close()
1601
1602        logging.info(f"Saved=>{fig_n}")
1603
1604
1605        # resolution vs nhits
1606
1607        nbins18=10
1608
1609        nh_min, nh_max= nh_list.min(), nh_list.max()
1610
1611        bh= np.linspace(nh_min, nh_max, nbins18+1)
1612
1613        bc18= 0.5*(bh[:-1]+ bh[1:])
1614
1615        sf18, dl18= [], []
1616
1617        for i in range(nbins18):
1618
1619            sel= (e_list>= bh[i]) & (e_list< bh[i+1])
1620
1621            if np.sum(sel)==0:
1622
1623                sf18.append(0)
1624
1625                dl18.append(0)
1626
1627            else:
1628
1629                sf18.append(sf_dist[sel].mean())
1630
1631                dl18.append(dl_dist[sel].mean())
1632
1633        plt.figure()
1634
1635        plt.plot(bc18, sf18, '-o', color='blue', label="SF")
1636        plt.plot(bc18, dl18, '-o', color='red', label="DL")
1637
1638        plt.xlabel("Energy (MeV)")
1639
1640        plt.ylabel("Mean 3D distance (mm)")
1641
1642        plt.title("Resolution vs Energy (Test set)")
1643
1644        plt.legend()
1645
1646        plt.tight_layout()
1647
1648        fig_n= os.path.join(args.save_pred_fig_dir, ""
1649                           "resolution_vs_nhits_v17.png")
1650
1651        plt.savefig(fig_n)
1652
1653        plt.close()
1654
1655        logging.info(f"Saved=>{fig_n}")
1656
1657
1658        # resolution vs nhits
1659
1660        nbins19=10
1661
1662        nh_min, nh_max= nh_list.min(), nh_list.max()
1663
1664        bh= np.linspace(nh_min, nh_max, nbins19+1)
1665
1666        bc19= 0.5*(bh[:-1]+ bh[1:])
1667
1668        sf19, dl19= [], []
1669
1670        for i in range(nbins19):
1671
1672            sel= (e_list>= bh[i]) & (e_list< bh[i+1])
1673
1674            if np.sum(sel)==0:
1675
1676                sf19.append(0)
1677
1678                dl19.append(0)
1679
1680            else:
1681
1682                sf19.append(sf_dist[sel].mean())
1683
1684                dl19.append(dl_dist[sel].mean())
1685
1686        plt.figure()
1687
1688        plt.plot(bc19, sf19, '-o', color='blue', label="SF")
1689        plt.plot(bc19, dl19, '-o', color='red', label="DL")
1690
1691        plt.xlabel("Energy (MeV)")
1692
1693        plt.ylabel("Mean 3D distance (mm)")
1694
1695        plt.title("Resolution vs Energy (Test set)")
1696
1697        plt.legend()
1698
1699        plt.tight_layout()
1700
1701        fig_n= os.path.join(args.save_pred_fig_dir, ""
1702                           "resolution_vs_nhits_v18.png")
1703
1704        plt.savefig(fig_n)
1705
1706        plt.close()
1707
1708        logging.info(f"Saved=>{fig_n}")
1709
1710
1711        # resolution vs nhits
1712
1713        nbins20=10
1714
1715        nh_min, nh_max= nh_list.min(), nh_list.max()
1716
1717        bh= np.linspace(nh_min, nh_max, nbins20+1)
1718
1719        bc20= 0.5*(bh[:-1]+ bh[1:])
1720
1721        sf20, dl20= [], []
1722
1723        for i in range(nbins20):
1724
1725            sel= (e_list>= bh[i]) & (e_list< bh[i+1])
1726
1727            if np.sum(sel)==0:
1728
1729                sf20.append(0)
1730
1731                dl20.append(0)
1732
1733            else:
1734
1735                sf20.append(sf_dist[sel].mean())
1736
1737                dl20.append(dl_dist[sel].mean())
1738
1739        plt.figure()
1740
1741        plt.plot(bc20, sf20, '-o', color='blue', label="SF")
1742        plt.plot(bc20, dl20, '-o', color='red', label="DL")
1743
1744        plt.xlabel("Energy (MeV)")
1745
1746        plt.ylabel("Mean 3D distance (mm)")
1747
1748        plt.title("Resolution vs Energy (Test set)")
1749
1750        plt.legend()
1751
1752        plt.tight_layout()
1753
1754        fig_n= os.path.join(args.save_pred_fig_dir, ""
1755                           "resolution_vs_nhits_v19.png")
1756
1757        plt.savefig(fig_n)
1758
1759        plt.close()
1760
1761        logging.info(f"Saved=>{fig_n}")
1762
1763
1764        # resolution vs nhits
1765
1766        nbins21=10
1767
1768        nh_min, nh_max= nh_list.min(), nh_list.max()
1769
1770        bh= np.linspace(nh_min, nh_max, nbins21+1)
1771
1772        bc21= 0.5*(bh[:-1]+ bh[1:])
1773
1774        sf21, dl21= [], []
1775
1776        for i in range(nbins21):
1777
1778            sel= (e_list>= bh[i]) & (e_list< bh[i+1])
1779
1780            if np.sum(sel)==0:
1781
1782                sf21.append(0)
1783
1784                dl21.append(0)
1785
1786            else:
1787
1788                sf21.append(sf_dist[sel].mean())
1789
1790                dl21.append(dl_dist[sel].mean())
1791
1792        plt.figure()
1793
1794        plt.plot(bc21, sf21, '-o', color='blue', label="SF")
1795        plt.plot(bc21, dl21, '-o', color='red', label="DL")
1796
1797        plt.xlabel("Energy (MeV)")
1798
1799        plt.ylabel("Mean 3D distance (mm)")
1800
1801        plt.title("Resolution vs Energy (Test set)")
1802
1803        plt.legend()
1804
1805        plt.tight_layout()
1806
1807        fig_n= os.path.join(args.save_pred_fig_dir, ""
1808                           "resolution_vs_nhits_v20.png")
1809
1810        plt.savefig(fig_n)
1811
1812        plt.close()
1813
1814        logging.info(f"Saved=>{fig_n}")
1815
1816
1817        # resolution vs nhits
1818
1819        nbins22=10
1820
1821        nh_min, nh_max= nh_list.min(), nh_list.max()
1822
1823        bh= np.linspace(nh_min, nh_max, nbins22+1)
1824
1825        bc22= 0.5*(bh[:-1]+ bh[1:])
1826
1827        sf22, dl22= [], []
1828
1829        for i in range(nbins22):
1830
1831            sel= (e_list>= bh[i]) & (e_list< bh[i+1])
1832
1833            if np.sum(sel)==0:
1834
1835                sf22.append(0)
1836
1837                dl22.append(0)
1838
1839            else:
1840
1841                sf22.append(sf_dist[sel].mean())
1842
1843                dl22
```



```
772         "dl_position_x": dl_x,
773         "dl_position_y": dl_y,
774         "dl_position_z": dl_z,
775         "real_position_x": rx,
776         "real_position_y": ry,
777         "real_position_z": rz,
778         "sf_position_x": sf_x.astype(np.float32),
779         "sf_position_y": sf_y.astype(np.float32),
780         "sf_position_z": sf_z.astype(np.float32),
781         "energy": e_list.astype(np.float32),
782         "nhits": nh_list.astype(np.float32)
783     }
784
785     outroot= os.path.join(args.out_root_dir, "test_set_DL_v1.root")
786     with uproot.recreate(outroot) as fout:
787         fout[args.tree_name]= outdict
788         logging.info(f"Wrote=>{outroot}")
789
790         logging.info("All done. Check logs in %s", args.log_file)
791
792 if __name__=="__main__":
793     main()
```

附录 D 山东大学本科毕业论文（设计）撰写规范

山东大学文件

山大教字〔2023〕6号

关于印发《山东大学本科毕业论文（设计）管理办法》的通知

全校各单位：

《山东大学本科毕业论文（设计）管理办法》业经 2023 年第 1 次校长办公会议审议通过，现予以印发，请遵照执行。

山东大学
2023 年 4 月 11 日

为规范我校本科毕业论文（设计）管理，提升毕业论文（设计）质量，特制定本规范。本规范约定的书写格式主要适用于用中文撰写的毕业论文（设计）。涉外专业用英文或其他外国语撰写毕业论文（设计）的书写规范可参照本规范执行。各学院和培养单位可根据不同学科、专业特点制定符合本类别毕业论文（设计）的撰写规范。

D.1 毕业论文（设计）结构与装订

毕业论文（设计）一般由以下几部分组成，依次为：封面-成绩评定表-中文摘要（含关键词）-英文摘要（含关键词）-目录-正文-参考文献-致谢-附录-译文中文（可选）-译文原文（可选）-封底。

D.2 毕业论文（设计）主要内容及要求

D.2.1 题目

题目应简洁、精炼、准确且具概括性，能够准确概括论文（设计）的核心内容，一般不超过 20 字，必要时可增加副标题。

D.2.2 摘要

中文摘要应具有高度的概括性，语言精炼、明确，扼要叙述论文（设计）的主要内容，包括研究目的与意义、研究内容与方法以及研究结论等，同时需要突出论文（设计）的新论点、新见解或创造性成果。英文摘要内容应与中文摘要一致，语句通顺，语法正确，准确反映论文（设计）内容。

中文摘要一般约 300-800 个汉字，英文摘要约 200-600 个单词。

D.2.3 关键词

关键词是供检索用的主题词条，应采用能覆盖毕业论文（设计）主要内容的通用技术词条（参照相应的技术术语标准），可从标题或正文中选择 3-5 个最能表达主要内容的词语作为关键词，按词条的外延层次排列（外延大的排在前面）。关键词有中、英文对照，分别附于中、英文摘要后。

D.2.4 目录

目录一般按三级标题编写（如 1、1.1、1.1.1……），层次清晰，与正文中标题一致。

D.2.5 正文

正文包括前言、本论、结论三个部分。

1. 前言

前言应说明毕业论文（设计）的目的、意义、研究范围及要求达到的技术参数；国内外的发展概况及存在问题；指导思想和应解决的主要问题。

2. 本论

本论是毕业论文（设计）的主体，必须言之成理，论据可靠，严格遵循本学科国际通行的学术规范。写作上要注意结构合理、层次分明、重点突出，章节标题、公式图表符号必须规范统一。

根据不同学科毕业论文（设计）主体的内容及特点，本论一般包括毕业论文（设计）总体方案或选题的论证；各部分的设计实现，如实验数据的获取、数据可行性及有效性的处理与分析、各部分的设计计算等；对研究内容及成果的客观阐述，如理论依据、创新见解、创造性成果及其改进与实际应用价值等。

3. 结论

结论应集中反映作者的研究成果，要求语言精炼、表述准确且完整，着重阐述自己的创造性成果及其在本研究领域中的意义、作用，同时应包括所得结果与已有结果的比较和本课题尚存在的问题，以及进一步开展研究的见解与建议。

D.2.6 参考文献

参考文献是毕业论文（设计）所参考的专著、论文及其他资料（20篇及以上），所列参考文献应按参考或引证的先后顺序排列，一般应为最新正式发表的文献，其中理、工、医类外文参考文献占比一般不少于50%。

注明引用文献的方法通常有三种，即文中注：正文中在引用的地方用括号说明文献出处；脚注：正文中只在引用地方写一个脚注标号，在当页最下方以脚注方式按标号顺序说明文献出处；文末注：在正文引用的地方标号（一般以出现的先后顺序编号，编号以方括号括起，放右上角），然后在全文末“参考文献”一节，按标号顺序说明文献出处。不同学科可能要求不同，但都应遵循国际上通用习惯以及我国有关国家标准规定，且全文统一，不能混用。

D.2.7 致谢

致谢是对在毕业论文（设计）工作中给予各类资助、指导、协助以及提供各种有利条件的单位、指导教师或其他人员表示感谢，语言应实事求是，切忌浮夸之词。

D.2.8 附录

附录主要包括一些不宜放入正文中的支撑材料，如公式的推演过程、编写的算法、语言程序、各种篇幅较大的图纸等。

D.2.9 译文

中期检查前，学生应完成一篇与毕业论文（设计）课题紧密相关的外文译文（不少于2000个汉字）。外文资料由指导教为学生指定，应是一篇完整的选自近期外文书刊的文献，若原文较长，可为文献的核心章节。

D.3 毕业论文（设计）的书写和打印规范

D.3.1 文字和字数

除部分特殊专业外，毕业论文（设计）一律采用国家语言文字工作委员会正式公布的简化汉字书写，英文授课本科专业国际学生的毕业论文（设计）可以采用英文书写，但须附中文摘要。外语类专业毕业论文（设计）采用外语书写。

D.3.2 页面设置

论文（设计）应使用A4纸单面或双面纵向打印，上、下页边距2.5cm，左、右各页边距3.0cm。

页眉：从正文开始设置，以小五号宋体键入“山东大学本科毕业论文（设计）”，居中显示。

页码：目录页码使用罗马数字（、、）编排；正文页码从正文开始至附录（正文-参考文献-致谢-附录）使用阿拉伯数字编排，小五号Times New Roman居中。封面、封

底、成绩评定表、中文摘要（含关键词）、外文摘要（含关键词）、外文资料及译文不编入页码。

段前、段后：章标题段前 0.8 行，段后 0.5 行；节标题 0.5 行，段后 0.5 行。

D.3.3 字体与字号

1. 封面

毕业论文（设计）采用山东大学本科毕业论文（设计）统一封面。中文题目用黑体小二号加粗，外文题目用三号黑体字加粗，姓名、学号、学院、年级、指导教师为宋体四号。（英文均采用“Times New Roman”字体）。

2. 中文摘要

“摘要”二字为黑体小二号加粗居中，中间空 4 个空格；容为宋体小四号、1.5 倍行距、首行缩进两字符。

3. 中文关键词

“关键字”三字为黑体小四号加粗；内容为宋体小四号，各关键词用分号（;）隔开，无缩进。

4. 英文摘要

“ABSTRACT”为小二号加粗居中；内容为小四号、1.5 倍行距、首行缩进两字符。

5. 英文关键词

“Key Words”为小四号加粗，内容为小四号，各关键词之间用逗号（,）分开，无缩进。

6. 目录

“目录”二字为黑体小二号加粗居中，空 4 个空格；内容为宋体小四号。

7. 正文

中文一级标题为黑体三号加粗，二级标题为黑体四号加粗，三级及以下标题为黑体小四号加粗；英文一级标题为 15pt 加粗，二级标题为 14pt 加粗；三级及以下标题为 13pt 加粗。

正文内容为宋体小四号、1.5 倍行距、首行缩进 2 字符。

8. 参考文献

“参考文献”四字为黑体小二号加粗居中；内容为宋体五号、单倍行距、首行无缩进。

9. 致谢

“致谢”二字为黑体小二号加粗居中，中间空4个空格；内容为宋体小四号、1.5倍行距、首行缩进两字符。

10. 附录

“附录”二字为黑体小二号加粗居中，中间空4个空格；内容为宋体小四号、首行缩进两字符、1.5倍行距。

11. 译文

译文标题为黑体小二号加粗居中，内容为宋体小四号、1.5倍行距、首行缩进两字符。外文原文标题为小二号加粗居中，内容小四号、1.5倍行距、首行缩进两字符，或直接使用原文PDF版本。

D.3.4 标题层次

毕业论文（设计）的全部标题层次应有条不紊，整齐清晰，相同的层次应采用统一的表示体例，正文中各级标题下的内容应同各自的标题对应，不应有与标题无关的内容。

理学、工学、医学类学位论文（设计）的章节编排建议遵循《CY / T 35-2001 科技文献的章节编号方法》，以阿拉伯数字编号，如1, 1.2, 1.2.1逐级递推。人文社科类学位论文（设计）建议采用第一章第一节一、（一）形式编排。英文撰写的文（设计）建议采用Chapter1, 1.2, 1.2.1逐级递推。

D.3.5 图片、表格及公式

1. 图片

毕业论文（设计）的插图应与文字紧密配合，文图相符，技术内容正确。选图要力求精练，线条要匀称，图面要整洁美观，居中，每幅插图应有图序和图题（宋体五号加粗居中），全文插图可以统一编序，也可以每章单独编序（如图1.1 XXX, 图2.1XXX），不管采用哪种方式，图序必须连续，不得重复或跳缺。由若干分图组成的插图，分图用a、b、c……标序，分图的图名以及图中各种代号的意义，以图注形式写在图题下方，先写分图名，另起一行后写代号的意义。

图应在描纸或洁白纸上用墨线绘成，或用计算机绘图，电气图或机械图应符合相应的国家标准。坐标图：横纵坐标必须标注物理量、单位，坐标名置于图的下方居中，宋体五号加黑。

图应放在离正文首次出现处的近处，不应过分超前或拖后。

2. 表格

每个表格应有自己的表序和表题，位于表格上方正中，表序后不加标点，空一格后写表题，表题末尾不加标点。表题用宋体五号加粗居中，表格内中文用宋体五号，英文用 Times New Roman，字体五号，表格格式采用简明三线表。

全文的表格可以统一编序，也可以每章单独编序（如表 1-XXX，表 2-1 XXX），不管采用哪种方式，表序必须连续。表格允许下页接写，接写时表题省略，表头应重复书写，并在右上方写“续表 ××”。此外，表格应放在离正文首次出现处的近处，不应过分超前或拖后。

3. 公式

公式应另起一行居中对齐，一行写不完的长公式，最好在等号处转行，如做不到这点，应在数学符号（如“+”、“-”号）处转行，且数学符号应写在转行后的行首。公式的编号用圆括号括起放在公式右边行末，靠右对齐，公式和编号之间不加虚线，公式可按全文统一编序号，也可以每章单独编序（如第 1 章中第一个公式编号为 1-1，第 2 章中第二公式编号为 2-2），公式序号必须连续，不得重复或跳缺。重复引用的公式不得另编新序号。公式中分数的横分线要写清楚，特别是连分数（即分子和分母也出现分数时）更要注意分线的长短，并将主要分线和等号对齐。在叙述中也可将分数的分子和分母平列在一行，用斜线分开表述。

D.3.6 量和单位

毕业论文（设计）中的量和单位必须采用中华人民共和国国家标准 GB 3100~GB 3102-1993，它是以国际单位制（SI）为基础的。非物理量的单位，如件、台、人、元等，可用汉字与符号构成组合形式的单位，例如：件/台、元/km。

D.3.7 标点符号、数字

标点符号应按《标点符号用法》（中华人民共和国国家标准 GB/T15834-2011）使用。测量、统计数据一律用阿拉伯数字，如 5.25MeV 等。在叙述不是特别大的数目时，一般不宜用阿拉伯数字。

D.3.8 名词、名称

科学技术名词术语尽量采用全国科学技术名词审定委员会公布的规范词或国家标准、部标准中规定的名称。尚未统一规定或叫法有争议的名词术语，可采用惯用的名称。使用外文缩写代替某一名词术语时，首次出现时应在括号内注明其含义，如：OECD (Organization for Economic Cooperation and Development) 代替经济合作发展组织。

外国人名一般采用外文原名，可不译成中文，英文人名按姓前名后的原则书写，如：CRAY P.，不可将外国人姓名中的名部分漏写，例如：不能只写 CRAY，应写成 CRAY 的外国人名（如牛顿、爱因斯坦、达尔文、马克思等）可按通常标准译法写译名。

D.3.9 参考文献著录格式示例

书写格式应符合 GB/T 7714-2015《信息与文献参考文献著录规则》。

山东大学校长办公室

2023年4月11日印发
