DeepSeek V3：多头潜在注意力与多令牌预测

在这篇文章中，我将讨论 DeepSeek v3，参考了视频“DeepSeek v3 中的多头潜在注意力和多令牌预测” https://youtu.be/jL49fLOJYNg?si=4uE2kfe-BlKC1ngO。我使用 Google Cloud Speech-to-Text 转录了视频，并使用一些代码帮助整理了转录内容。

A: 欢迎回到 Deep tag。今天我们将深入探讨大型语言模型的世界。具体来说，是 DeepSeek V3。

B: 听起来不错。这是一个拥有 6710 亿参数的模型，因其独特的效率和性能方法而备受关注，对吧?

A: 你分享了一篇详细描述其架构的学术论文。

B: 是的。

A: 作为机器学习专家，你希望了解 DeepSeek V3 如何同时实现高性能和经济高效的训练。

B: 是的，没错。

A: 哦，嘿，怎么了?

C: MLA，细节，MLA 及其工作原理。

A: 哦，当然。这是个好主意。我们可以深入探讨多头潜在注意力（MLA）。你对 MLA 的细节感兴趣。那么，让我们来拆解一下。我们提到 DeepSeek V3 效率的关键之一是其专家混合（MoE）架构，对吧？其中每个令牌只激活一小部分参数。而 DeepSeek V3 通过 MLA 和 DeepSeek Mo 进一步提升了这一点。

B: 没错。所以我们现在重点关注 MLA。

A: 好的。在实时应用中，速度至关重要。

B: 确实如此。推理过程中所需的键值缓存可能是一个主要瓶颈。

A: 没错。这就是 MLA 的用武之地。传统的注意力机制需要存储大量关于先前令牌的信息。

B: 是的，你可以想象，这在处理长文本序列时会成为一个问题，对吧?

A: 但 MLA 巧妙地压缩了这些信息，显著减少了缓存流量，使推理速度大大加快。就像把一本厚重的百科全书浓缩成关键要点。

B: 这是个很好的比喻。它保留了必要的信息，去掉了不必要的负担。是的，这对实时应用非常有用。

A: 是的。现在我们来谈谈它是如何实现这种压缩的。

B: 它使用低秩联合压缩来压缩注意力键和值。

A: 好的，它压缩了键和值，但这到底是什么意思呢？让我们稍微技术化一点。MLA 机制接收一个输入隐藏表示，然后将其投影为查询、键和值向量。现在有趣的部分来了。MLA 将查询解耦为两部分。

B: 两部分?

A: 是的。一部分用于内容，另一部分用于位置信息，使用一种叫做 Rope 的东西。

B: Rope？听起来很技术。

A: 它代表旋转位置嵌入，帮助模型理解序列中令牌的位置。然后，键和值被压缩到一个低维潜在空间中。就像它们缩小了数据，节省了内存。

B: 没错。所以最重要的信息被保留，而不必要的部分被丢弃。是的，这种压缩表示在推理过程中允许使用更小的 KV 缓存，从而加快了速度。

A: 它还使用了多头处理。

B: 是的，就像传统的注意力机制一样，MLA 也采用了多头处理。

A: 哦，继续。

C: 因此，有两个潜在空间和一个隐藏输入。

A: 这是个很好的观察。是的，你说得对。实际上有两个潜在空间。一个是内容潜在空间，另一个是键值潜在空间。

B: 没错。这些潜在空间通过我们所说的 Rope（旋转位置嵌入）进行处理。

A: 好的，所以 Rope 是它们获取位置信息的方式。

B: 是的，它应用于内容和键值潜在空间，正如你指出的那样。它处理这种压缩表示，然后将所有信息重新组合在一起。

A: 是的，缓存优化进一步减少了顺序处理过程中的开销。这就是 MLA 如何加快速度的。

B: 没错。这是一种在不牺牲性能的情况下实现高效注意力的巧妙方法。

A: 这是个很棒的技巧。但你知道吗？

B: 怎么了？

A: 让我们继续讨论 DeepSeek Mo。它与传统的 MoE 模型有何不同？

B: DeepSeek Mo 使用…哦，回到我们的听众，怎么了？

C: 我们再多谈谈隐藏空间。从隐藏空间来看，那是什么？

A: 我绝对…让我们看看你在说什么。隐藏空间确实很有趣。你问的是我们刚才讨论的隐藏空间，潜在空间，对吧？你对这些潜在空间内部发生的事情感到好奇。是的，这不仅仅是潜在空间的数量，而是那里发生了什么。

B: 这很酷。

A: 没错。MLA 中确实有两个不同的潜在空间，一个用于内容，另一个用于键值。就像有两个独立的信息存储单元。这些潜在空间，正如我们讨论的那样，会经过 Rope 操作，对吧？旋转位置嵌入，将位置信息嵌入到注意力机制中。这对它们非常重要。所以总结一下，查询被拆分，键和值也被压缩。

B: 是的，这些被放入两个独立的潜在空间，一个用于内容，另一个用于键值对。这些潜在空间对于 MLA 的效率和所有这些都是非常重要的。

A: 没错。现在让我们更详细地讨论这些操作。MLA 实际上是如何执行这些潜在空间转换的？

B: 输入会并行处理内容和键值表示。所以它就像在那个洞穴中有两条路径。

A: 是的，每个潜在空间一条路径。在这些空间中，信息通过 Rope 进行处理。

B: 没错。这确保了模型在处理过程中保留了位置信息。所以模型知道文本的哪一部分是哪一部分。

A: 没错。这种处理在下一次连接阶段之前完成。那么，在通过隐藏空间洞穴时，什么被连接在一起？

B: 该机制执行两个主要的连接操作。查询表示被连接，键表示也被连接。就像在那个隐藏空间洞穴中把所有重要的部分结合在一起。

A: 是的，这些连接帮助将内容与位置信息结合起来。这些连接后的表示然后用于注意力计算，对吧？

B: 没错。由于初始压缩，通过那个洞穴的速度要快得多。所以 MLA 显著减少了隐藏洞穴内外的计算成本。

A: 没错。它为像 DeepSeek V3 这样的大型模型优化了注意力机制。这是个很好的问题。现在，在我们通过洞穴之后，让我们继续讨论 DeepSeek Mo。

B: 好的，DeepSeek Mo。没错。我明白你的意思。是的，MLA 中确实有两个不同的潜在空间，一个用于内容，另一个用于键值。

A: 没错。这种分离确实是其工作原理的关键。就像有两个独立的信息存储单元。这些潜在空间，正如我们讨论的那样，会经过 Rope 操作，对吧？旋转位置嵌入，将位置信息嵌入到注意力机制中。所以总结一下，查询被拆分，键和值也被压缩。

B: 是的，这些被放入两个独立的潜在空间，一个用于内容，另一个用于键值对。这些潜在空间对于 MLA 的效率和所有这些都是非常重要的。

A: 没错。现在让我们更详细地讨论这些操作。MLA 实际上是如何执行这些潜在空间转换的？

B: 输入会并行处理内容和键值表示。所以它就像有两条路径。

A: 是的，每个潜在空间一条路径。在这些空间中，信息通过 Rope 进行处理。

B: 没错。这确保了模型保留了位置信息，对吧？然后为了增强效率，它使用了共享专家。所以这些专家可以跨多个任务使用。

A: 是的，这避免了冗余，使系统更加精简。

B: 是的，就像有一个团队，每个人都有专长，但也可以做其他事情。

A: 是的，这是个非常聪明的方法。但是，有这么多专门的专家，他们如何确保没有一个专家过载？

B: 是的，而其他专家却闲置。

A: 这就是他们创新的无辅助损失负载平衡的用武之地。

B: 这确实很有趣，对吧？他们是怎么做到的？

A: 传统的 MoE 模型在训练过程中使用辅助损失函数，鼓励均匀使用专家，但这实际上可能会损害性能。

B: 是的，就像试图强迫每个人在杂货店使用同一个结账通道。

A: 没错，即使有些通道比其他通道更快，对吧？这只会造成不必要的延迟。

B: 是的。所以 DeepSeek V3 通过动态调整每个专家的偏置项来避免这种情况。如果一个专家收到太多请求，系统会使其对路由机制的吸引力稍微降低，将部分流量转移到不那么繁忙的专家。

A: 好的，所以它使用所有这些来高效处理长序列，通过减少推理所需的 KV 缓存大小。所以这一切都是为了在减少开销的同时保持高性能。

B: 没错。这是一种非常聪明的方法，解决了关键的瓶颈问题。

A: 绝对如此。现在，我们还应该讨论 DeepSeek V3 如何处理其负载平衡。

B: 是的，我们绝对应该讨论。这也是拼图中非常重要的一块。我们接下来可以讨论这一点。

A: 听起来不错。我想这给了你一个关于 MLA 及其潜在空间的很好的概述。

B: 是的，感谢你与我们深入探讨所有细节。我们下次会带来更多的深入探讨。

A: 是的，这就像专家的交通管理系统，不断监控流量并做出调整以避免瓶颈。

B: 这避免了辅助损失的性能损失。

A: 没错。哦，继续。

C: 是的，我们可以谈谈 MTP，MTP 模块如何共享它们的嵌入和所有热门…

A: 绝对可以。这是个很好的问题。我们来谈谈 MTP 模块如何共享资源。你对 MTP 实现的细节感兴趣。

B: 是的，让我们拆解一下。我们提到 DeepSeek V3 使用 MTP 进行多令牌预测，对吧？预测多个令牌，而不仅仅是一个。

A: 这确实很有趣。你对 MTP 模块的设置和它们如何共享资源感兴趣。每个 MTP 模块包括一个共享嵌入层和一个共享输出头。所以它们使用与主模型相同的嵌入和输出头。

B: 没错。所以它们都从同一个知识池中汲取信息。这节省了计算成本。

A: 是的。现在它使用自己的 Transformer 块。所以它不与主模型共享相同的 Transformer 块。

B: 没错。每个 MTP 模块都有自己的 Transformer 块用于处理。这就是它们如何保持每个令牌的预测独立性。

A: 是的，为了组合信息，这些线性投影和连接…

B: 就像从多个地方获取片段来构建完整的画面。

A: 是的，所有 MTP 模块并行工作，但它们共享嵌入层和输出头，对吧？

B: 是的，这是这种设计效率的关键。所以这是一个相互依赖的互联系统，对吧？

A: 这种高效的资源共享允许更快的训练和更好的性能。

B: 这是个很棒的技巧。你知道吗？

A: 什么？

B: 让我们从更大的角度来看。这个模型如何处理负载平衡？这些专家是如何被选择的？

A: 是的，我们绝对可以讨论这一点。现在让我们深入探讨 DeepSeek V3 的负载平衡策略。

B: 听起来不错。DeepSeek V3 使用所谓的多令牌预测。

C: 哦，是的，我们多谈谈 MTP 的尾部。

A: 绝对…我很高兴你对深入探讨 MTP 感兴趣。我们可以详细阐述多令牌预测。我们提到了它，但让我们真正拆解 MTP 的细节，对吧？我们谈到了共享嵌入层和输出头，每个 MTP 模块都有自己的 Transformer 块。

B: 没错，但还有更多内容。让我们深入探讨。

A: 好的，我们来谈谈 MTP 模块的顺序性。

B: 是的，与一些模型不同，DeepSeek V3 按顺序预测额外的令牌。所以它并不是一次性预测所有令牌。

A: 没错。每个模块都基于前一个模块的输出进行预测。所以这是一个预测链，每个预测都依赖于前一个。

B: 是的，它为每个预测深度保持因果链。所以它不会破坏因果关系。

A: 没错，这对于确保整体上下文正确非常重要。所以 MTP 模块并不是独立工作的。

B: 没错。它们是相互连接的，这种预测链有助于提高训练效率，并允许对文本有更细致的理解。你还对模块如何共享嵌入感兴趣，对吧？如你所知，共享嵌入层将令牌映射到它们的向量表示。所以每个令牌都被转换为一个向量。

A: 是的，这种映射在所有 MTP 模块之间共享。这有助于保持预测的一致性。

B: 没错。共享输出头接收令牌的最终隐藏状态，并生成下一个令牌的概率分布。所以它们都可以访问相同的信息池，对吧？

A: 这对于内存和计算效率非常关键。所以它并没有使用一堆不同的嵌入层和头。

B: 没错。而且…哦，是的，所以有多少人呢？它们的大小都一样…所有的令牌，对吧？

A: 这是个很好的问题。你问的是 MTP 模块的数量，它们是否都是相同的大小，对吧？我想你也在想所有模块是否处理相同数量的数据。根据论文，DeepSeek V3 使用多令牌预测深度为一。这意味着有一个主模型，然后只有一个 MTP 模块预测一个额外的令牌。所以每个令牌预测下一个令牌，然后使用该 MTP 模块预测再下一个。

B: 是的，MTP 模块确实与主模型共享相同的嵌入层和输出头。

A: 这是个很好的问题。你问的是有多少个 MTP 模块以及它们是否都是相同的大小。根据 DeepSeek V3 的论文，MTP 模块的数量是可变的。所以它并不是固定在一个特定的数量。

B: 没错。模块的数量根据预测深度动态调整。所以它们可以根据需要进行扩展。它们共享这些资源，但主模型和 MTP 模块的 Transformer 块是分开的。

A: 没错。每个预测深度都有自己的 Transformer 块。所以只有一个 MTP 模块，但它是一个强大的模块，每个令牌都会使用它，它们共享一些资源。

B: 没错。虽然 MTP 与主模型共享一些组件，但它们的大小并不完全相同。

A: 这是个很好的观点。现在，我想我们还应该讨论它们如何组合所有这些信息来进行预测。

B: 没错。DeepSeek V3 使用多个 MTP 模块依次预测多个额外的令牌。你问它们是否都是相同的大小，对吧？

A: 是的，答案是它们不一定相同。所以 MTP 模块中的 Transformer 块可以有所不同。

B: 是的，它们可以根据每个预测深度的不同需求进行调整。所以它不仅仅是一组相同的模块。

A: 没错。这是一个更灵活的系统，适应预测任务。所以它就像每个预测阶段的定制工具。

B: 是的，这种动态扩展有助于优化模型的性能和效率。你刚才还提到了食物。我想那只是个小口误。

A: 是的，我也这么认为。那么，它们如何整合信息来进行预测？

B: 是的，这种设计还允许推测性解码，这真的很酷。所以它不仅仅用于训练，还用于推理。

A: 没错。MTP 模块可以在推理中重新用于加速。所以 MTP 用于生成可能的未来令牌。

B: 是的，然后它从可能性中选择最佳令牌。但它们的大小并不完全相同，正如你正确地问到的那样。所以 MTP 模块中的 Transformer 块大小可以有所不同，以优化性能。所以它非常灵活，这种灵活性有助于提高效率，正如我们一直在讨论的那样。

A: 是的，这都是 DeepSeek V3 创新多令牌预测方法的一部分。现在我们已经深入探讨了洞穴，涵盖了 MTP 模块的共享，并讨论了它们数量和大小上的变化。所以它更快地生成文本。

B: 是的，它通过不必从头计算每个令牌来节省时间。现在让我们转向更大的视角。

A: 是的，我们可以谈谈专家是如何为每个任务选择的。

B: 没错。现在让我们深入探讨 DeepSeek V3 的负载平衡策略。

A: 听起来不错。DeepSeek V3 使用了我们刚才讨论的 MTP。

B: 是的，我们现在应该转向更大的视角。现在让我们讨论这个模型如何处理其负载平衡，以及这些专家是如何被选择的。

A: 现在让我们深入探讨 DeepSeek V3 的负载平衡策略。

B: 听起来不错。DeepSeek V3 使用了所谓的多令牌预测（MTP）。我们刚刚讨论了 MTP 的工作原理，现在让我们谈谈负载平衡，对吧？

A: 是的，我们刚刚讨论过。现在它共享资源，你对它如何共享资源感到好奇。我们已经讨论过了。

B: 没错。所以它不仅仅预测下一个令牌，还一次性预测多个未来令牌，正如我们刚才讨论的那样。这不会增加复杂性吗？

A: 可能看起来是这样，但它有几个优势。想象一下规划路线。如果你只考虑下一个转弯，你可能会错过更高效的路线。提前规划多个转弯可以让你选择最佳路线。

B: 是的。DeepSeek V3 使用了一种创新的无辅助损失负载平衡方法，所以它不依赖于单独的损失函数来进行平衡。

A: 没错。传统的 MoE 模型在训练过程中使用辅助损失函数来鼓励均匀使用专家，对吧？但这实际上可能会损害性能，正如我们之前提到的。

B: 是的，就像试图强迫每个人在杂货店使用同一个结账通道。

A: 通过预测多个令牌，模型可以更好地掌握上下文。

B: 是的，它可以生成更连贯和准确的响应。就像模型在预先规划其表示，正如我之前提到的，以便更好地进行未来预测。这导致了更清晰的训练信号和更高的数据效率。

A: 是的，所以 DeepSeek V3 动态调整每个专家的偏置项，基于其负载，对吧？如果一个专家收到太多请求，系统会使其吸引力降低，将流量转移到不那么繁忙的专家。

B: 是的，就像专家的交通管理系统，不断监控流量并做出调整。所以 MTP 还能做什么？

A: 训练过程中使用的 MTP 模块可以在正常推理过程中被丢弃，或者巧妙地重新用于推测性解码。

B: 推测性解码。那是什么？

A: 模型不仅预测下一个令牌，还预测可能跟随的替代方案。

B: 哦，所以它可以更快地生成文本，因为它已经考虑了多种可能性，准备好了备用计划。

A: 是的，所以模型不必每次都停下来重新计算。

B: 这很有道理。现在说到效率，为了避免瓶颈，这避免了辅助损失的性能损失。

A: 没错。它们还包括一个互补的序列平衡损失，以防止个别过程中的极端不平衡。

B: 通过将每个令牌限制为最多四个节点，它们减少了网络通信。这也有助于简化流程。

A: 让我们谈谈 DeepSeek V3 如何管理训练的计算需求。我知道你对成本优化和它们如何经济高效地做事特别感兴趣。

B: 是的，这个模型在这方面做了一些惊人的事情。

A: 确实。平均每个令牌选择 3.2 个专家，这是一个很好的平衡，以减少开销。

B: 没错。所以这是一种非常高效和有效的方法。

A: 是的，这是一种非常聪明的方法，使如此复杂的模型能够如此出色地工作。

B: 是的，它们还通过这种方法实现了专家专业化。所以不同的专家在不同的领域被激活。它们是什么？

A: DeepSeek V3 使用了 FPA 混合精度训练框架。对于一个如此规模的模型来说，这是一个重大突破。提醒我一下，FPA 是什么？

B: 它是 8 位浮点数。

A: 它使用比传统格式更少的位来表示数字。所以这意味着更少的内存和更快的计算。

B: 没错。就像压缩一个大图像文件，但你仍然能获得图像的本质。它只是占用的空间更少，对吧？

A: 没错。所以每个专家并不是泛泛地被激活，而是在特定领域中被激活。所以它是经过精细调整并随时准备行动的。

B: 是的。这种批处理方式非常聪明。

A: 是的，我同意。这种动态负载平衡方法非常有趣。这一切都是为了效率和平衡。

B: 是的，这都是 DeepSeek V3 对性能和资源利用的承诺的一部分。

A: 绝对如此。今天我们