


自然语言指令诱导神经元网络中的成分泛化

收稿日期: 2023 年 5 月 13 日

雷达尔·瑞夫兰  亚历山大·普吉特 

接受日期: 2024 年 2 月 15 日

在线发布: 2024 年 3 月 18 日

 检查更新

人类的一项基本认知壮举是解释语言指令，以便在没有明确任务经验的情况下执行新任务。然而，可用于实现这一目标的神经计算仍然知之甚少。我们利用自然语言处理的进步来创建基于语言指令的泛化神经模型。模型接受一组常见心理物理任务的训练，并接收预训练语言模型嵌入的指令。我们最好的模型可以执行以前未见过的任务，仅基于语言指令（即零样本学习），平均正确率可达 83%。我们发现，语言支撑着感觉运动表征，使得相关任务的活动与指令的语义表征共享共同的几何形状，从而使语言能够提示在看不见的环境中练习技能的正确组成。我们展示了该模型如何仅使用运动反馈来生成对其识别的新任务的语言描述，这随后可以指导合作伙伴模型执行该任务。我们的模型提供了几个可通过实验测试的预测，概述了必须如何表示语言信息以促进人脑的灵活和一般认知。

在实验室环境中，动物需要进行多次试验才能获得新的行为任务。部分原因是与非语言动物沟通的唯一方式是简单的正面和负面强化信号。相比之下，向人类发出书面或口头指令是很常见的，这使得他们能够相对快速地执行新任务。此外，一旦人类学会了一项任务，他们通常可以用自然语言描述解决方案。使用指令来执行新任务的双重能力，以及相反，一旦学习了任务需求就可以产生语言描述的能力，是人类交流的两个独特的基石。然而，这些能力背后的计算原理仍然知之甚少。

一种有影响力的系统级解释认为，前额皮质中灵活的区域间连接允许在新的环境中重复使用已练习的感觉运动表征。

最近，多项研究观察到，当要求受试者灵活地招募不同的刺激反应模式时，神经表征会根据任务集的抽象结构进行组织。最后，最近的建模工作表明，多任务循环神经网络（RNN）将在具有相似需求的任务之间共享动态主题。这项工作为解释人类灵活认知奠定了坚实的基础，但也留下了一个问题：语言信息如何重新配置感觉运动网络，使其在第一次尝试时就能很好地执行一项新任务。总体而言，目前尚不清楚负责整合语言信息以动态重组感觉运动映射的大脑区域应该具有什么样的表征结构。

鉴于机器学习的最新进展已经导致人工系统表现出以下特征，这些问题变得更加紧迫

类似人类的语言能力。最近的研究将被动听力和阅读任务中记录的神经数据与自回归语言模型（即 GPT）中的激活进行了匹配，认为语言理解存在根本上的预测成分。此外，一些备受瞩目的机器学习模型确实显示出使用自然语言作为执行语言任务或渲染图像的提示的能力，但这些模型的输出很难用我们可能期望的感觉运动映射来解释发生在生物系统中。或者，最近关于多模式交互代理的工作可能在其所采取的动作方面更容易解释，但利用在处理的早期阶段融合视觉和语言的感知层次结构，使得它们难以映射到功能和解剖上不同的语言和视觉区域在人类的大脑中。

因此，我们寻求利用语言模型的力量，产生可测试的神经预测，详细说明人脑如何处理自然语言，以便泛化感觉运动任务。

为此，我们在一组简单的心理物理任务上训练 RNN（感觉运动 RNN）模型，其中模型使用预训练的语言模型处理每个任务的指令。我们发现，将指令与调整为句子级语义的模型一起嵌入，使得感觉运动 RNN 能够以平均 83% 的正确率执行新任务。我们模型中的泛化得到了表征几何的支持，该表征几何捕获了任务子组件，并在指令嵌入和感觉运动活动之间共享，从而允许在新颖的环境中组合练习技能。我们还发现各个神经元根据指令的语义来调节它们的调谐。我们演示了一个经过训练来解释语言指令的网络如何能够反转这种理解，并根据运动反馈信号中的信息生成以前未见过的任务的语言描述。最后，我们讨论这些结果如何指导人脑中基于语言的泛化的神经基础研究。

结果

指导模型和任务集

我们在一组 50 个相互关联的心理物理任务上训练感觉运动 RNN，这些任务需要各种认知能力，这些能力在文献中得到了充分研究。图 1a、b 中展示了两个示例任务，因为它们可能出现在实验室环境中。对于所有任务，模型都会接收感官输入和任务识别信息，并且必须输出运动响应活动（图 1c）。输入刺激由两个一维神经元图编码，每个图代表不同的输入模态，具有角度的周期性高斯调谐曲线（超过 $(0, 2\pi)$ ）。输出响应以相同的方式编码。输入还包括一个固定单元。输入固定关闭后，模型可以对输入刺激做出响应。我们的 50 个任务大致分为 5 组：“Go”、“决策”、“比较”、“持续时间”和“匹配”，其中组内任务共享相似的感官输入结构，但可能需要不同的响应。例如，在决策（DM）任务中，网络必须对对比度最高的刺激方向做出响应，而在反决策（AntiDM）任务中，网络对对比度最弱的刺激做出响应对比（图1a）。因此，网络必须从任务识别信息中正确推断给定试验的任务需求，以便同时执行所有任务（有关任务详细信息，请参阅方法；有关所有任务的示例试验，请参阅补充图 13）。

在我们的模型中，任务识别输入要么是非语言的，要么是语言的。我们使用两种非语言控制模型。首先，在 SIMPLINET 中，任务的身份由 50 个正交规则向量之一表示。其次，STRUCTURENET 使用一组 10 个正交结构向量，每个向量代表任务集的一个维度（即响应最弱与最强的方向），并且使用这些向量的组合对任务进行编码（完整的一组向量请参见补充注释 3）。

结构组合）。因此，STRUCTURENET 完全捕获了任务之间的所有相关关系，而 SIMPLINET 不编码任何这种结构。

指令模型使用预训练的转换器架构来嵌入当前任务的自然语言指令。对于每项任务，都有一组相应的 20 条独特指令（15 条训练，5 条验证；完整指令集请参阅补充说明 2）。我们测试了各种类型的语言模型，这些模型具有相同的基本架构，但其大小和预训练目标不同。我们测试了两种自回归模型，一个标准版和一个大版本的 GPT2，我们分别称之为 GPT 和 GPT (XL)。之前的工作已经证明 GPT 激活可以解释阅读和听力的各种神经特征。BERT 被训练来识别一段文本中的屏蔽词，但它也使用无监督的句子级目标，其中给定网络两个句子，并且必须确定它们在原始文本中是否相互遵循。SBERT 的训练方式与 BERT 类似，但在斯坦福自然语言推理任务上接受了额外的调整，该任务是一个手工标记的数据集，详细说明了两个候选句子（方法）之间的逻辑关系。最后，我们使用 CLIP 的语言嵌入器，这是一种多模态模型，可以学习图像和文本标题的联合嵌入空间。我们使用给定的语言模型 LANGUAGEMODELNET 来调用感觉运动 RNN，并附加一个指示其大小的字母。图1c给出了模型的各种尺寸。对于每种语言模型，我们将池化方法应用于变压器的最后一个隐藏状态，并将该固定长度表示传递给一组在任务学习期间训练的线性权重。这会产生跨所有模型（方法）嵌入的 64 维指令。除非另有说明，否则语言模型权重将被冻结。最后，作为对照，我们还测试了词袋（BoW）嵌入方案，该方案仅使用字数统计来嵌入每条指令。

首先，我们验证我们的模型可以同时执行所有任务。

为了让指令模型表现良好，它们必须推断出每项任务 15 种不同指令表述之间的共同语义内容。我们发现，我们所有的指导模型都可以同时学习所有任务，但 GPTNET 除外，其中某些任务的性能渐近线低于 95% 的阈值。因此，我们将使用 GPT 的模型的性能阈值放宽至 85%（补充图 1；有关训练详细信息，请参阅方法）。我们还根据验证指令测试了所有架构（补充图 2）。SBERTNET (L) 和 SBERTNET 是我们性能最好的模型，在验证指令上分别实现了 97% 和 94% 的平均性能，这表明这些网络即使对于全新的指令也能推断出正确的语义内容。

泛化到新任务

接下来我们研究了不同语言模型在多大程度上有助于泛化新任务。我们针对 45 项任务对各个网络进行了训练，然后测试了处理这 5 项保留任务时的性能。我们使用不等方差 t 检验来比较不同模型的性能。图 2 显示了最相关比较的 P 值结果（所有模型的完整比较矩阵可以在补充图 3 和图 4 中找到）

我们的无指导控制模型 SIMPLINET 在首次呈现新任务（零样本泛化）时平均执行率为 39%。这是泛化的基线。请注意，尽管向 SIMPLINET 提供的任务规则是正交的，但暴露于任务集允许模型学习所有任务共有的模式（例如，在注视期间始终抑制响应）。因此，39% 本身并不是机会级别的性能，而是通过在具有一些常见响应要求的任务集上进行训练和测试的网络所实现的性能。GPTNET 的零样本泛化率为 57%。这是相对 SIMPLINET ($t = 8.32$, $P = 8.24 \times 10^{-8}$) 的显着改进。引人注目的是，将 GPT 的大小增加一个数量级至 GPT (XL) 使用的 15 亿个参数仅导致

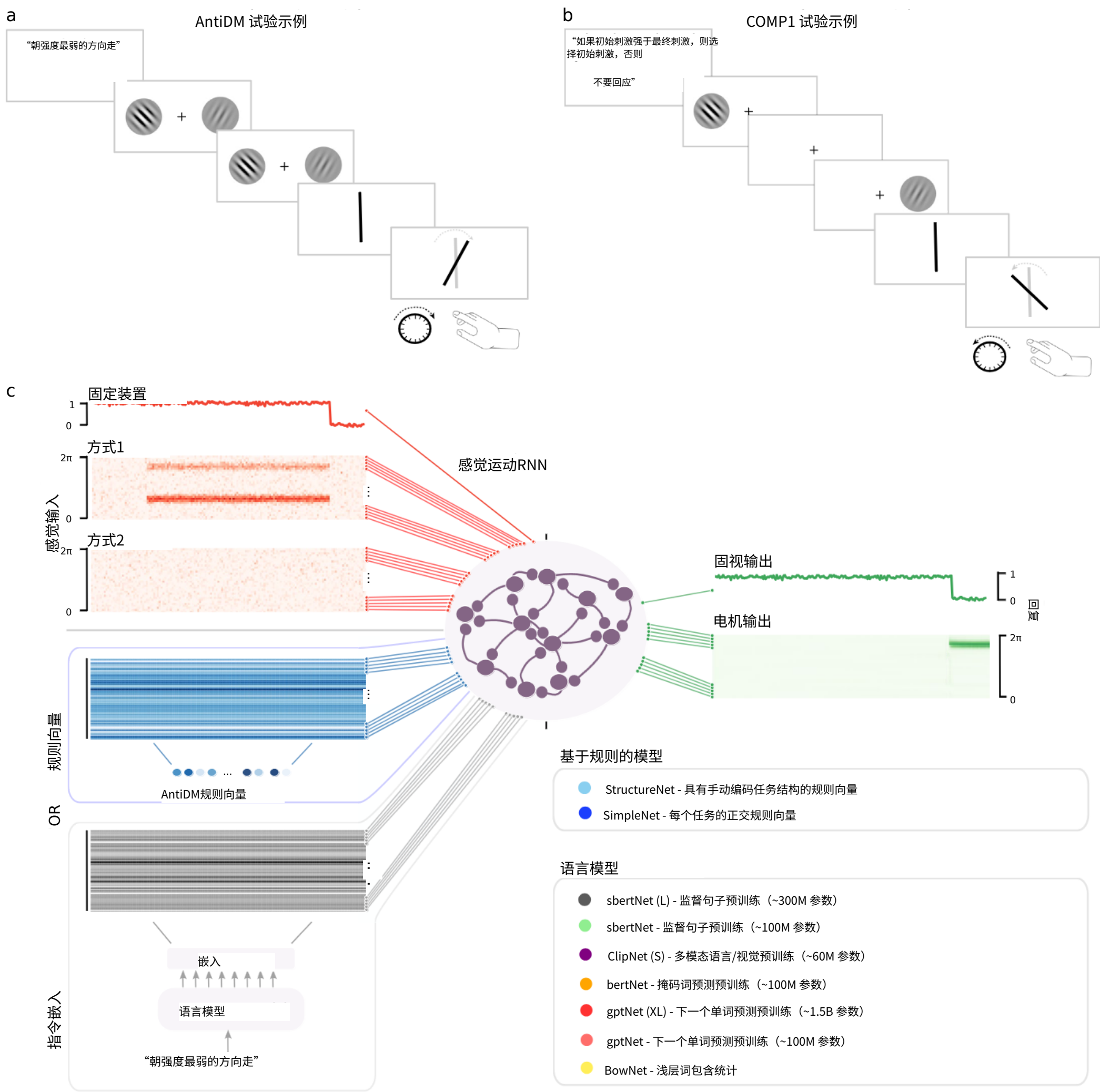


图1|任务和模型。a,b, 实验室环境中可能出现的示例试验插图。指导试验，然后以不同角度和对比强度呈现刺激。然后，代理必须在响应期间以适当的角度做出响应。a, AntiDM 试验示例，其中代理必须对强度最小的角度做出反应。b, COMP1 试验示例，其中代理必须对第一个角度做出反应（如果出现）

比第二个角度更高的强度则会抑制反应。c, 模型输入和输出图。感觉输入（注视单元、模态1、模态2）显示为红色，模型输出（注视输出、电机输出）显示为绿色。模型还接收规则向量（蓝色）或通过预训练语言模型（灰色）传递任务指令而产生的嵌入。插图中提供了经过测试的模型列表。

与 BOWNET (64%) 相比略有提高，GPTNET (XL) 在保留任务上取得了 68% 的成绩 ($t = 2.04$, $P = 0.047$)。相比之下，CLIPNET (S) 使用的参数数量仅为 GPTNET (XL) 的 4%，但仍然能够实现相同的性能（正确率 68%， $t = 0.146$, $P = 0.88$ ）。同样，BERTNET 的泛化性能仅落后 GPTNETXL 2% ($t = -1.122$, $P = 0.262$)。相比之下，具有句子级语义知识的模型在泛化方面表现出显著的进步

SBERTNET 执行看不见的任务的平均正确率为 79%。最后，我们性能最好的模型 SBERTNET (L) 可以执行前所未见的任务，平均正确率达到 83%，仅落后 STRUCTURENET（正确率 88%）几个百分点，后者获得了在其规则向量中手动编码的任务集的结构。

图 2b 显示了每个模型达到给定性能水平的任务数量的直方图。同样，SBERTNET (L) 成功地在零样本中几乎完美地执行了 20 多项任务

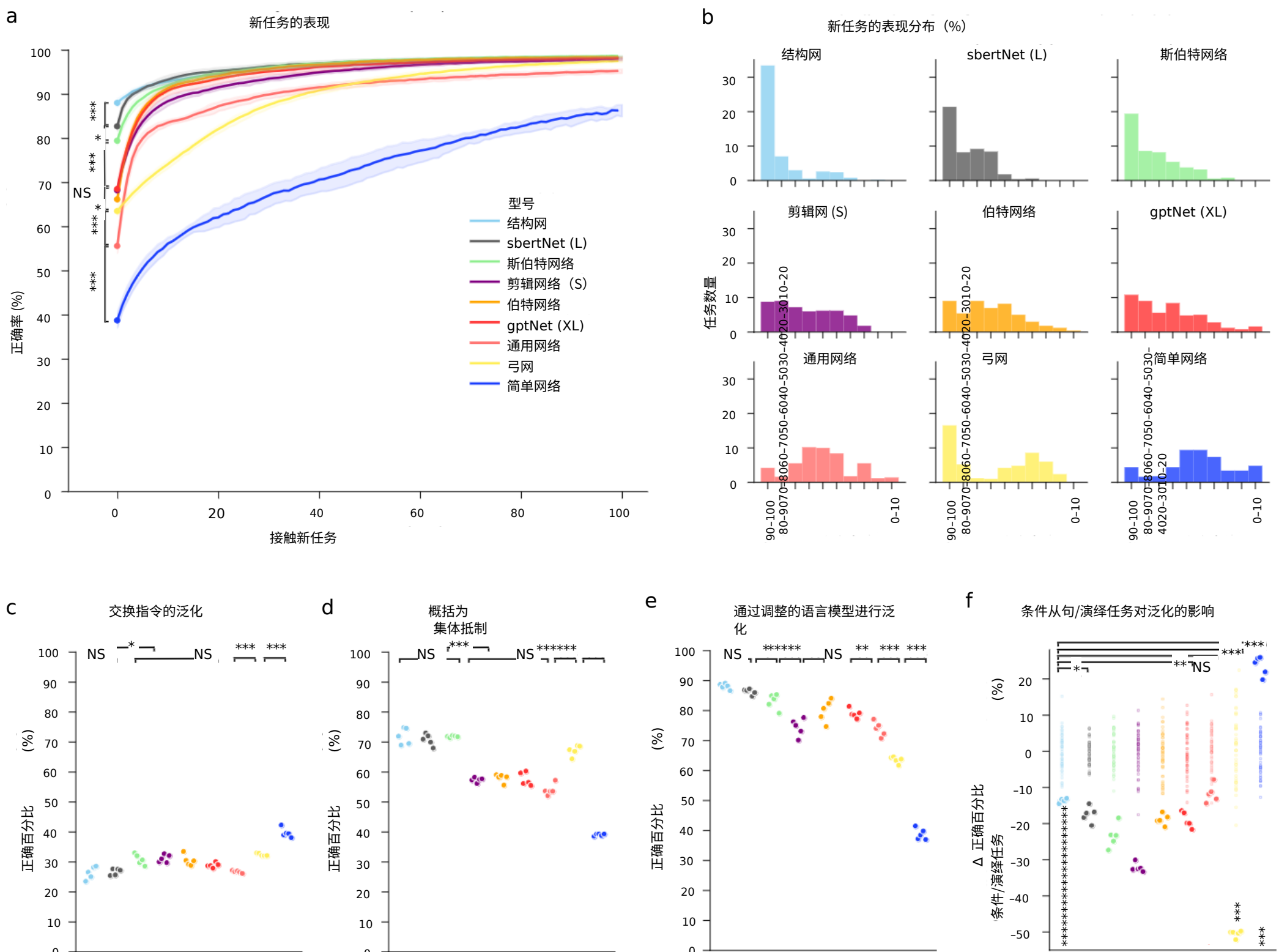


图2|模型在新任务上的表现。a、第一个学习曲线
100 次未完成任务的暴露对所有任务进行平均。数据以平均值±标准差表示。跨不同的 n = 5 个 sensorimotorRNN 权重随机初始化。对于所有子图，星号表示根据双边不等方差 t 检验，性能之间存在显著差异。最相关的比较显示在图中（对于所有子图，不显著 (NS)， $P > 0.05$ ，* $P < 0.05$ ，** $P < 0.01$ ，*** $P < 0.001$ ；STRUCTURENET 与 SBERTNET (L)： $t = 3.761$ ， $P = 1.89 \times 10^{-3}$ ；SBERTNET (L) 与 SBERTNET： $t = 2.19$ ， $P = 0.029$ ；SBERTNET 与 CLIPNET： $t = 6.22$ ， $P = 1.02 \times 10^{-3}$ ；CLIPNET 与 BERTNET： $t = 1.037$ ， $P = 0.300$ ；GPTNET (XL)： $t = -1.122$ ， $P = 0.262$ ；GPTNET (XL) 与 GPTNET： $t = 6.22$ ， $P = 1.04 \times 10^{-3}$ ；GPTNET 与 BOWNET： $t = -3.346$ ， $P = 8.85 \times 10^{-3}$ ； $t = 10.25$ ， $P = 2.091 \times 10^{-4}$ ）。完整的成对比较表可以在补充图 3 中找到。b，跨模型的泛化性能分布（即首次接触新任务）。c-f，n = 5 个感觉运动 RNN 权重的不同随机初始化在不同测试条件下的性能，其中每个点表示给定初始化的任务的平均性能。c，测试时交换指令的任务的泛化性能（STRUCTURENET 与 SBERTNET (L)： $t = -0.15$ ， $P = 0.875$ ；SBERTNET (L) 与 SBERTNET： $t = -2.102$ ， $P = 0.036$ ；SBERTNET 与 CLIPNET： $t = -0.162$ ， $P = 0.871$ ；CLIPNET 与 BERTNET： $t = 0.315$ ， $P = 0.752$ ；BERTNET 与 GPTNET (XL)： $t = 0.781$ ， $P = 0.435$ ；GPTNET (XL) 与 GPTNET： $t = 1.071$ ， $P = 0.285$ ；GPTNET 与 BOWNET： $t = -2.702$ ， $P = 0.007$ ；BOWNET 与 SIMPLENET： $t = -3.471$ ， $P = 5.633 \times 10^{-3}$ ）。完整的成对比较表可以在补充图 4 中找到。d，训练期间保留同一族任务的模型的泛化性能（STRUCTURENET 与 SBERTNET (L)： $t = 0.629$ ， $P = 0.530$ ；SBERTNET (L) 与 SBERTNET： $t = -0.668$ ， $P = 0.504$ ；SBERTNET 与 CLIPNET： $t = 8.043$ ， $P = 7.757 \times 10^{-4}$ ；CLIPNET 与 BERTNET： $t = -0.306$ ， $P = 0.759$ ；BERTNET 与 GPTNET (XL)： $t = 0.163$ ， $P = 0.869$ ；通用技术网

(XL) 与 GPTNET 对比： $t = 1.534$ ， $P = 0.126$ ；GPTNET 与 BOWNET： $t = -6.418$ ， $P = 3.26 \times 10^{-4}$ ；BOWNET 与 SIMPLENET： $t = 14.23$ ， $P = 8.561 \times 10^{-4}$ ）。完整的成对比较表可以在补充图 4 中找到。e，模型的泛化性能，其中允许语言模型的最后一层对感觉运动任务的损失进行微调（STRUCTURENET 与 SBERTNET (L)： $t = 1.203$ ， $P = 0.229$ ；SBERTNET (L) 与 SBERTNET： $t = 2.399$ ， $P = 0.016$ ；SBERTNET 与 CLIPNET： $t = 5.186$ ， $P = 3.251 \times 10^{-3}$ ；CLIPNET 与 BERTNET： $t = -3.002$ ， $P = 0.002$ ；GPTNET (XL)： $t = 0.522$ ， $P = 0.601$ ；GPTNET (XL) 与 GPTNET： $t = 2.631$ ， $P = 0.009$ ；GPTNET 与 BOWNET： $t = 4.440$ ， $P = 1.134 \times 10^{-3}$ ；BOWNET 与 SIMPLENET： $t = 10.255$ ， $P = 2.091 \times 10^{-4}$ ）。

完整的成对比较表可以在补充图 4 中找到。f，使用标准命令式指令的任务与使用带有条件子句的指令并需要简单演绎推理组件的任务之间的平均性能差异。图底部的彩色星号显示使用任务集随机分割构建的零分布之间的双边不等方差 t 检验的 P 值（透明点代表随机分割的平均差；STRUCTURENET： $t = -36.46$ ， $P = 4.34 \times 10^{-8}$ ；SBERTNET (L)： $t = -16.38$ ， $P = 3.02 \times 10^{-4}$ ；SBERTNET： $t = -15.35$ ， $P = 3.920 \times 10^{-4}$ ；CLIPNET： $t = -44.68$ ， $P = 5.32 \times 10^{-9}$ ；BERTNET： $t = -25.51$ ， $P = 3.14 \times 10^{-5}$ ；GPTNET (XL)： $t = -16.99$ ， $P = 3.61 \times 10^{-4}$ ； -9.150 ， $P = 0.0002$ ；BOWNET： $t = -70.99$ ， $P = 4.566 \times 10^{-16}$ ；SIMPLENET： $t = 19.60$ ， $P = 5.82 \times 10^{-4}$ ），图顶部的星号表示 t 检验的 P 值结果与 STRUCTURENET 和我们其他指导的差异进行比较

模型（与 SBERTNET (L) 相比： $t = 3.702$ ， $P = 0.0168$ ；与 SBERTNET 相比： $t = 6.592$ ， $P = 0.002$ ；与 CLIPNET 相比： $t = 30.35$ ， $P = 2.367 \times 10^{-6}$ ；与 BERTNET 相比： $t = 7.234$ ， $P = 0.0007$ 与 GPTNET 相比(XL)： $t = 5.282$ ， $P = 0.004$ ；与 GPTNET 相比： $t = -1.745$ ， $P = 0.149$ ；与 BOWNET 相比： $t = 75.04$ ， $P = 9.96 \times 10^{-16}$ ；与 SIMPLENET 相比： $t = -30.95$ ， $P = 2.86 \times 10^{-6}$ ；完整参见方法和补充图 6。比较）。

设置（有关跨任务的所有模型的单个任务性能，请参见补充图 3）。为了验证我们表现最好的模型是否利用了指令的语义，我们为一项保留任务提供了感官输入，同时为另一项保留任务提供了语言指令。真正依赖于语言信息的模型应该受到这种操作的最大惩罚，并且正如预测的那样，我们看到我们最好的模型的性能下降幅度最大（图 2c）。

我们还测试了更严格的保留程序，其中我们有目的地从同一任务系列中选择 4-6 个任务在训练期间保留（图 2d）。总体而言，尽管我们表现最好的模型仍然表现出很强的泛化能力，但在这种更困难的环境中，性能有所下降，SBERTNET (L) 和 SBERTNET 在新任务上的正确率分别为 71% 和 72%，这与 STRUCTURENET 的 72 没有显着差异。%（ $t = 0.629$ ， $P = 0.529$ ； $t = 0.064$ ， $P = 0.948$ ；分别针对 SBERTNET (L) 和 SBERTNET）。

此外，我们在允许语言模型的权重根据感觉运动训练期间经历的损失进行调整的环境中测试了模型（请参阅调整详细信息的方法）。这种操作提高了所有模型的泛化性能，对于我们性能最好的模型 SBERTNET (L)，我们发现泛化能力与 STRUCTURENET 一样强（86%， $t = 1.204$ ， $P = 0.229$ ）。以下参考文献。如图 18 所示，我们在一个设置中测试了模型，其中给定任务的任务类型信息表示为训练集中相关任务的信息组合（即 $\text{AntiDMMod1} = (\text{rule}(\text{AntiDMMod2}) - \text{Rule}(\text{DMMod2})) + \text{Rule}(\text{DMMod1})$ ）。在这种设置下，我们确实发现 SIMPLNET 的性能有所提高（正确率为 60%）。然而，当我们根据相同的组合规则组合嵌入式指令时，我们的语言模型显着优于 SIMPLNET。这表明在语言环境中进行训练更容易允许简单的组合方案成功配置任务响应（完整结果和组合编码请参见补充图 5）。

最后，我们测试了每个模型的一个版本，其中语言模型的输出通过一组非线性层传递，而不是前面结果中使用的线性映射。我们发现这种操作降低了性能，这表明这种增加的能力会导致训练任务的过度拟合，并且更简单的线性映射更适合泛化（有关详细信息，请参阅方法，有关完整结果，请参阅补充图 4）。我们指导的模型之间的性能差异表明，为了表示语言信息以使其能够成功配置感觉运动网络，仅仅使用任何非常强大的语言处理系统是不够的。相反，模型的成功可以通过它们在预训练期间接触句子级语义的程度来描述。我们表现最好的模型 SBERTNET (L) 和 SBERTNET 经过显式训练，可以产生良好的句子嵌入，而我们表现最差的模型 GPTNET 仅针对即将出现的单词的统计数据进行调整。CLIPNET (S) 和 BERTNET 都接触到某种形式的句子级知识。CLIPNET (S) 对句子级表示感兴趣，但使用相应视觉表示的统计数据来训练这些表示。BERTNET 对输入句子在训练语料库中是否相邻进行双向分类。相对于这些相对较小的模型，GPTNET (XL) 的 15 亿个参数并没有显着提高性能，这说明模型大小并不是决定因素。最后，尽管 BoW 删除了语言意义（即语法）的关键元素，但单词出现的简单使用主要编码了有关句子之间相似性和差异的信息。例如，简单地表示包含或排除“较强”或“较弱”一词就可以提供有关指令含义的大量信息。

我们还调查了哪些语言特征使我们的模型难以泛化。我们的 30 项任务需要处理具有条件子句结构（例如 COMP1）的指令，而不是简单的命令式（例如 AntiDM）。使用条件子句指示的任务也需要简单形式的演绎推理（if p then q else s）。探索此类演绎过程与语言区域之间关系的神经影像学文献得出了不同的结论，一些早期研究表明演绎招募了被认为支持句法计算的区域，而后续研究则声称演绎可以可靠地与语言区域分离。对于这种结果差异的一种理论是，早期研究中用于分离演绎推理的基线任务使用的语言刺激仅需要表面处理。为了探讨这个问题，我们计算了有和没有条件子句/演绎推理要求的任务之间的平均性能差异（图2f）。相对于一组随机洗牌，我们所有的模型在这些任务上的表现都较差。然而，我们还看到 STRUCTURENET 和我们指导的模型之间存在额外的影响，其表现比 STRUCTURENET 差，具有统计学意义（参见补充图 6 进行全面比较）。这是一个至关重要的比较，因为 STRUCTURENET 执行演绎任务时不依赖于语言。因此，STRUCTURENET 和指令模型之间性能下降的部分原因是解析语法上更复杂的语言所固有的困难。这意味着我们可能会看到语言区域参与演绎推理任务，但这可能仅仅是由于相应指令的句法要求增加（而不是招募语言区域来明确帮助演绎的过程）。这一结果在很大程度上与演绎推理文献的两篇评论一致，后者得出的结论是，早期研究中看到的语言区域的影响可能是由于测试刺激的句法复杂性造成的。

语言和感觉运动网络的共享结构

然后我们转而研究支持泛化的表征方案。首先，我们注意到，与其他多任务模型一样，我们的感觉运动 RNN 中的单元表现出功能聚类，其中相似的神经元子集在相似的任务集中表现出较高的方差（补充图 7）。此外，我们发现模型可以通过仅训练感觉运动 RNN 输入权重并保持循环动态常数来学习看不见的任务（补充图 8）。过去的工作表明，这些属性是网络的特征，可以在不同的设置中重用同一组底层神经资源。然后我们检查了相关任务的神经表征之间存在的几何形状。我们在执行特定模态 DM 和 AntiDM 任务的 SIMPLNET、GPTNETXL、SBERTNET (L) 和 STRUCTURENET 中绘制了刺激开始时感觉运动 RNN 隐藏活动的前三个主要成分 (PC)。在这里，模型以两种模式接收决策任务的输入，但必须只关注与当前任务相关的模式中的刺激。重要的是，AntiDMMod1 在以下示例中不进行训练。此外，我们还绘制了每个任务中规则向量或指令嵌入的 PC（图 3）。对于 STRUCTURENET，隐藏的活动沿着任务相关轴进行分解，即活动空间中一致的“Pro”与“Anti”方向（实线箭头），以及“Mod1”与“Mod2”方向（虚线箭头）。重要的是，即使对于 AntiDMMod1 来说，这种结构也得到了维护，而 AntiDMMod1 已经脱离了训练，从而使 STRUCTURENET 在这项看不见的任务上实现了 92% 的正确率。这种因式分解也反映在规则嵌入的 PC 中。引人注目的是，SBERTNET (L) 还以一种仅使用从指令语义推断出的结构来捕获任务集的基本组成性质的方式组织其表示。语言嵌入就是这种情况，它保持抽象

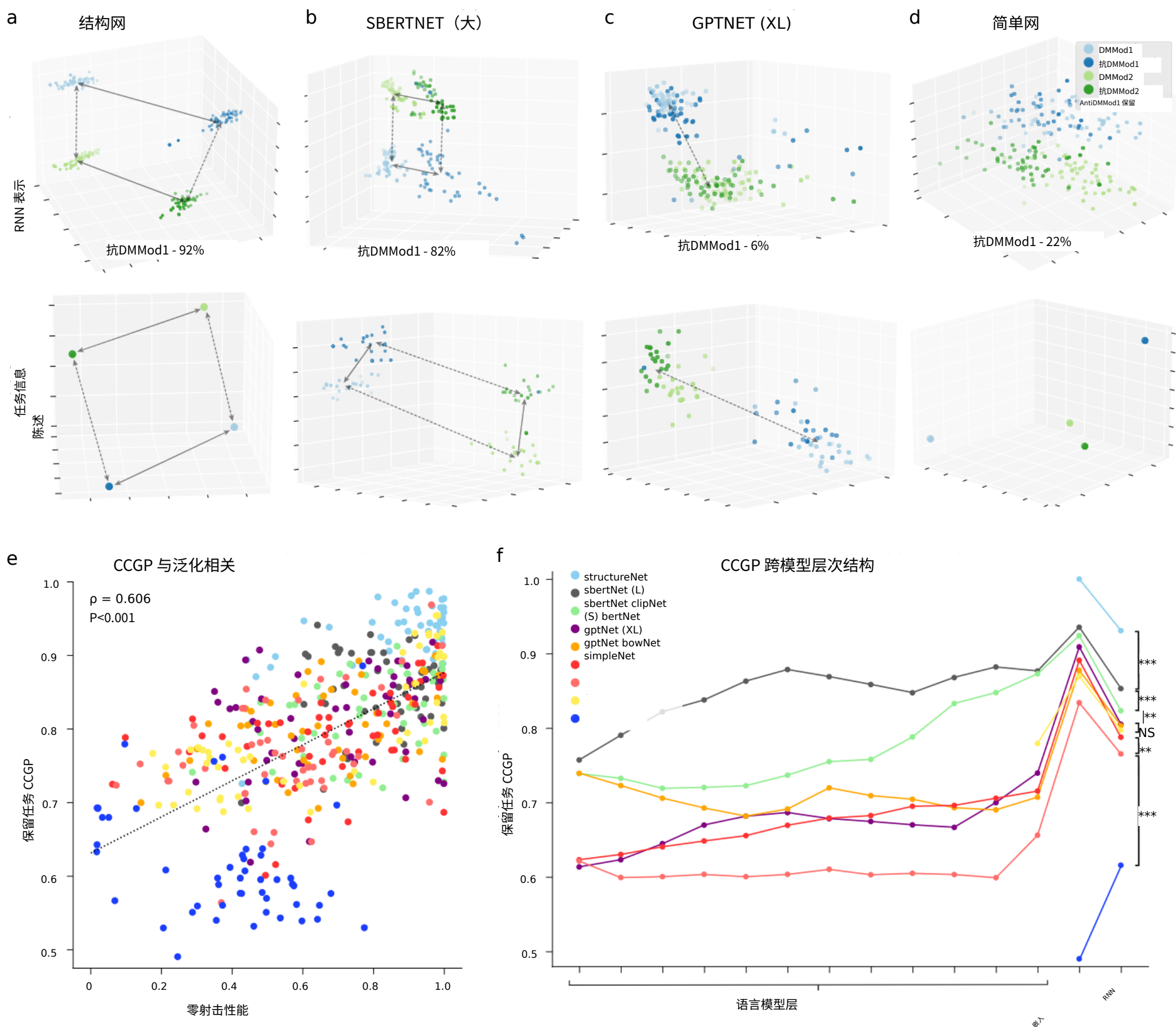


图3|指导模型中的结构化表示。a-d, 前三个使用 AntiDMMod1 训练的模型的感觉运动隐藏活动和任务信息表示的 PC 仍然有效。实线箭头代表抽象的“Pro”与“Anti”轴, 虚线箭头代表抽象的“Mod1”与“Mod2”轴。a、结构网。b、SBERTNET (L)。c、GPTNET (XL)。d、简单网络。e, 保留任务 CCGP 与零样本性能之间的相关性 (Pearson’s $r = 0.606$, $P = 1.57 \times 10^{-6}$)。f, 模型层次结构中每一层的保留任务的 CCGP 分数。显著性分数表示在模型中执行的成对两侧不等方差 t 检验的 P 值结果

感觉运动 RNN 保留任务上的 CCGP 分数分布 (NS $P > 0.05$, $*P < 0.05$, $**P < 0.01$, $***P < 0.001$; STRUCTURENET 与 SBERTNET (L): $t = 13.67$, $P = 2.44 \times 10^{-13}$; SBERTNET (L) 与 SBERTNET: $t = 5.061$, $P = 5.84 \times 10^{-6}$; SBERTNET 与 CLIPNET: $t = 2.809$, $P = 0.005$; CLIPNET 与 BERTNET: $t = 0.278$, $P = 0.780$;) : $t = 2.505$, $P = 0.012$; GPTNET (XL) 与 GPTNET: $t = 3.180$, $P = 0.001$; GPTNET 与 BOWNET: $t = -4.176$, $P = 3.50 \times 10^{-5}$; BOWNET 与 SIMPNET: $t = 23.0.8$, $P = 1.10$; 有关完整比较以及嵌入层 CCGP 分数的 t 检验结果, 请参阅补充图 9。

跨 AntiDMMod1 指令的轴 (同样, 不进行训练)。因此, SBERTNET (L) 能够使用这些相关轴进行 AntiDMMod1 感觉运动 RNN 表示, 从而实现 82% 的泛化性能。相比之下, GPTNET (XL) 未能在感觉运动 RNN 表示或语言嵌入中正确推断出不同的“Pro”轴与“Anti”轴, 导致 AntiDMMod1 上的零样本性能为 6% (图 3b)。最后, 我们发现 simpleNet 使用的正交规则向量排除了练习任务和保留任务之间的任何结构, 导致性能提高了 22%。为了更精确地量化这种结构, 我们测量了这些结构的跨条件泛化性能 (CCGP)

交涉。CCGP 衡量线性解码器的能力, 该线性解码器经过训练可区分一组条件 (即 DMMod2 和 AntiDMMod2), 以推广到一组类似的测试条件 (即 DMMod1 和 AntiDMMod1)。直观上, 这反映了模型学会将感觉运动活动沿着抽象任务轴 (即“反”维度) 放置的程度。值得注意的是, 在要求人类参与者在不同的相关任务之间灵活切换的实验中观察到了高 CCGP 分数和相关测量值。我们测量了感觉运动 RNN 中训练任务的 CCGP 分数 (方法), 并发现

CCGP 分数和零样本性能之间存在很强的相关性（图 3e）。此外，我们发现将任务指令替换为保留任务会显著降低我们所有指令模型的 CCGP 分数，这表明指令的语义对于维护结构化表示至关重要（补充图 9）。

然后我们研究了语言处理层次结构中的结构是如何出现的。我们模型中不同层的 CCGP 解码分数如图 3f 所示。对于每个指导模型，绘制了 12 个转换器层（或 SBERTNET (L) 和 GPTNET (XL) 的最后 12 层）、64 维嵌入层和 Sensorimotor-RNN 任务表示的分数。我们还绘制了非语言模型中使用的规则嵌入的 CCGP 分数。在模型之间，抽象结构的出现方式存在显着差异。自回归模型（GPTNETXL、GPTNET）、BERTNET 和 CLIPNET (S) 在整个语言模型层中表现出较低的 CCGP，随后在嵌入层中出现跳跃。这是因为输入嵌入层的权重在感觉运动训练期间进行了调整。这种峰值的含义是，这些模型中的大多数有用的表示处理实际上并不发生在预训练的语言模型本身中，而是发生在通过训练暴露于任务结构的线性读出中。相比之下，我们性能最好的模型 SBERTNET 和 SBERTNET (L) 使用语言表示，其中高 CCGP 分数逐渐出现在各自语言模型的中间层中。由于语义表示已经具有这样的结构，因此泛化中涉及的大部分组合推理可以发生在相对强大的语言处理层次中。因此，语言模型的最后一层中的表示已经组织得很好，并且嵌入层中的线性读出足以让感觉运动 RNN 正确推断任务集的几何形状并很好地进行泛化。

该分析强烈表明，表现出泛化能力的模型是通过利用结构化语义表示来正确关联感觉运动空间中的实践任务和新颖任务，从而允许在不可见的环境中组合实践行为。

单个单元调谐属性的语义调制

接下来，我们检查了感觉运动 RNN 中各个单元的调整配置文件。我们发现单个神经元会适应各种与任务相关的变量。然而，重要的是，我们发现神经元的这种调整在任务组内可预测地变化，并且通过指令的语义内容以反映任务需求的方式进行调节。

例如，在“Go”任务系列中，单元 42 显示了在“Pro”和“Anti”任务之间按 π 变化的方向选择性，反映了每个上下文中任务需求的关系（图 4a）。即使在训练期间进行的 AntiGo 任务中也观察到了这种选择性的翻转。

对于“匹配”系列任务，单元 14 调节“匹配”（DMS、DMC）和“不匹配”（DNMS、DNMC）条件之间的活动。在“非匹配”试验中，该单元的活动随着两个刺激之间的距离增加而增加。相比之下，对于“匹配”任务，当两个刺激之间的相对距离很小时，该神经元最活跃。因此，在这两种情况下，该神经元都会调节其活动来表示模型何时应做出响应，改变选择性以反映“匹配”和“不匹配”试验之间相反的任务需求。即使对于没有经过培训的 DMS 来说也是如此。

图 4c 显示了不同对比度级别（对比度 = $\text{str}-\text{str}$ ）下 DM 和 AntiDM 任务（AntiDMMod1 不进行训练）的特定模态版本中单元 3 活动的痕迹。在所有任务中，我们观察到渐变活动，其中渐变速率与对比度强度相关。这种活性基序已在之前的研究中报道过。然而，在我们的模型中

我们观察到，证据积累神经元可以根据语言指令的变化交换其整合的符号，这使得模型能够满足任务的“赞成”和“反对”版本的相反要求，甚至对于以前未见过的任务也是如此。

有趣的是，我们还发现不成功的模型无法正确调节调整偏好。例如，对于 GPTNET (XL)，它未能沿着“Pro”轴与“Anti”轴分解（图 3b），并且对 AntiDMMod1 的泛化能力很差，我们还发现神经元未能交换其在持有的整合符号-out 设置（补充图 10）。

最后，我们在“比较”系列任务的试验活动时间过程中看到了类似的模式（图 4d）。在 COMP1 任务中，如果第一个刺激的强度高于第二个刺激的强度，则网络必须对第一个刺激的方向做出响应，否则不得做出响应。在 COMP2 中，如果第二个刺激强度较高，则它必须仅对第二个刺激做出响应。对于“反”版本，刺激排序的要求是相同的，只是模型必须选择对比度最弱的刺激。即使增加了复杂性，我们发现单个神经元可以根据任务需求调节它们的调整，甚至对于保留的任务（在本例中为 COMP2）也是如此。例如，当网络应抑制响应时，单元82处于活动状态。对于“COMP1”，该单元在负对比度（即 $\text{str} > \text{str}$ ）下高度活跃，但在 COMP2 中翻转此灵敏度，并且在正对比度（即 $\text{str} > \text{str}$ ）下高度活跃。重要的是，当目标是选择最弱的刺激时，这种关系就会相反。因此，尽管指令集中存在这些微妙的语法差异，但语言嵌入可以以适合任务的方式逆转该单元的调整。

网络之间的语言交流

我们现在寻求对人类互补的能力进行建模，一旦获得了这种能力，就可以用语言描述特定的感觉运动技能。为此，我们反转了模型在训练期间学习的语言到感觉运动的映射，以便它们可以仅根据感觉运动单元的状态提供任务的语言描述。首先，我们构建了一个输出通道（生产 RNN；图 5a-c），该通道经过训练将感觉运动 RNN 状态映射到输入指令。然后，我们向网络提供一系列示例试验，同时保留特定任务的指令。在此阶段，所有模型权重都被冻结，模型接收电机反馈以更新嵌入层活动，从而减少输出误差（图 5b）。一旦嵌入层中的活动驱动感觉运动单元达到性能标准，我们就使用生产 RNN 来解码当前任务的语言描述。最后，为了评估这些指令的质量，我们将它们输入到合作伙伴模型中并测量跨任务的性能（图 5c）。本节中使用的所有指导模型和伙伴模型都是 SBERTNET (L)（方法）的实例。

AntiDMMod1 任务的一些解码指令示例（图 5d；所有解码指令请参阅补充说明 4）。为了可视化整个任务集和解码指令，我们绘制了一个混淆矩阵，其中感觉运动 RNN 和生产 RNN 都在所有任务上进行了训练（图 5e）。请注意，许多解码指令完全是“新颖的”，也就是说，它们没有包含在生产 RNN（方法）的训练集中。新颖指令占有所有任务中已解码指令的 53%。

为了测试这些新颖指令的质量，我们评估了合作伙伴模型在第一个网络生成的指令上的性能（图 5c；结果如图 5f 所示）。当合作伙伴模型在所有任务上进行训练时，所有解码指令的跨任务平均性能为 93%。向合作伙伴模型传达训练中未完成的任务的指令也能带来良好的表现（78%）。重要的是，即使对于“新颖”指令，性能也得以保持，在所有任务上训练的合作伙​​伴模型的平均性能为 88%，而具有保留任务的合作伙​​伴模型的平均性能为 75%。鉴于指导和合作伙​​伴模型共享相同的

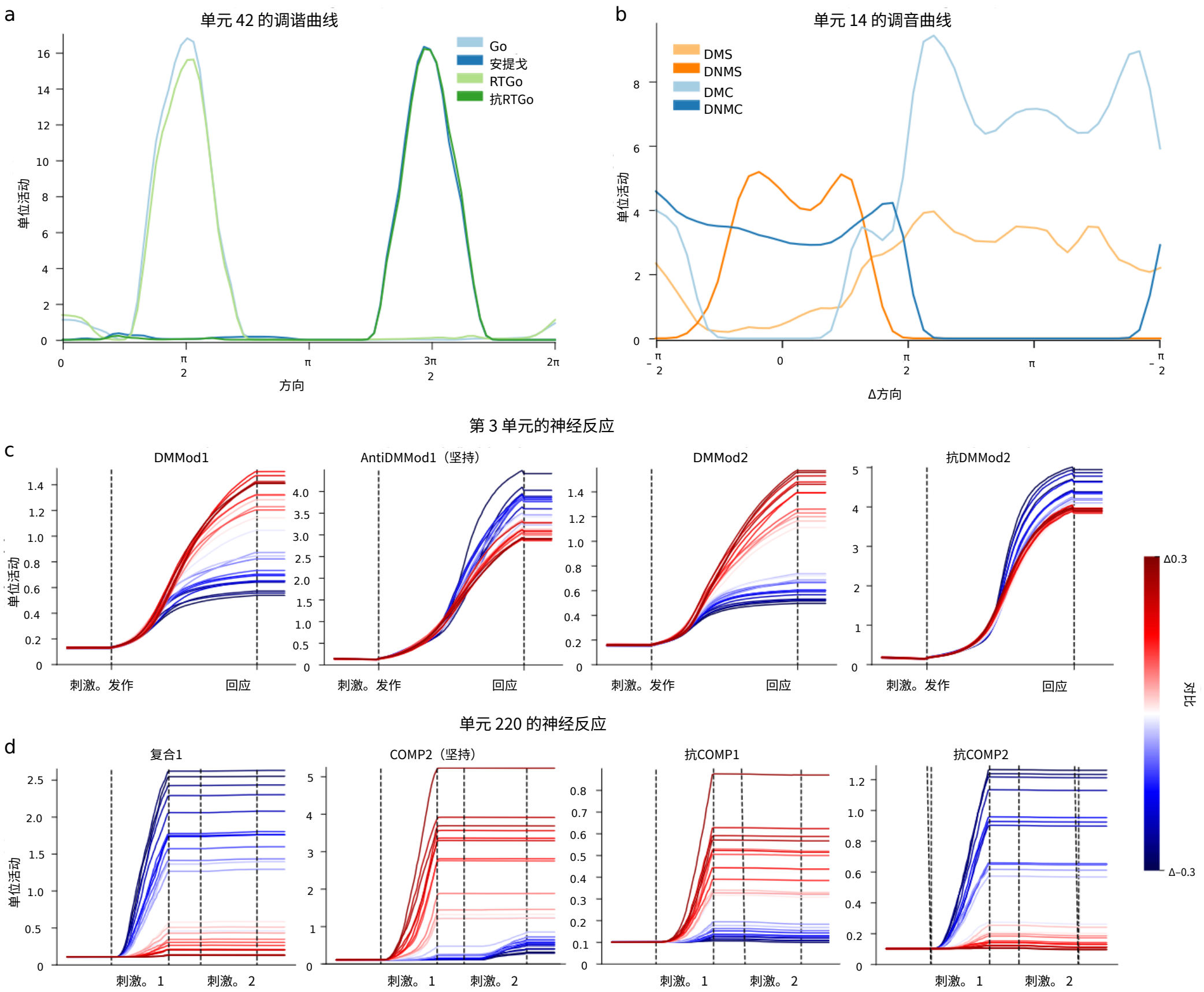


图4|单个单元调谐属性的语义调制。a、调谐曲线用于 SBERTNET (L) 传感器运动 RNN 单元，可根据“Go”系列中的任务需求进行调整。b，调整曲线，针对“匹配”任务系列中的 SBERTNET (L) 感觉运动-RNN 单元，根据两个刺激之间的角度差。c，针对不同水平的相对刺激强度，模态特定“DM”和“AntiDM”任务的完整活动轨迹。d，不同相对刺激强度水平的“比较”任务系列中任务的完整活动轨迹。

在架构中，人们可能认为放弃通信的语言组件并简单地将一个模型推断的嵌入复制到合作伙伴模型的输入中会更有效。这导致平均正确率仅为 31%，而在保留任务上测试合作伙伴模型时，正确率仅为 28%。尽管指导网络和合作伙伴网络共享相同的架构和相同的能力，但它们仍然具有不同的突触权重。因此，使用针对一个代理中的一组权重调整的神经表示不一定会在另一个代理中产生良好的性能。

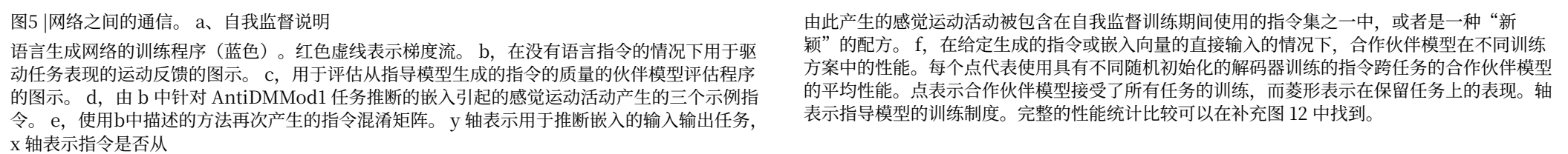
我们还使用感觉运动 RNN 测试了一个指导模型，其中包含训练之外的任务。我们在这里强调，在训练期间，生产 RNN 尝试从网络以前从未经历过的任务指令引起的感觉运动隐藏状态中进行解码（图 5a），而在测试期间，指令是从感觉运动状态产生的，这些状态完全作为最小化电机误差的结果（图 5b、c）。尽管如此，我们发现，在这种情况下，接受所有任务训练的合作模型的正确率是 82%，而未接受训练的合作模型的正确率是 73%。

在这里，77% 的生成指令是新颖的，因此当我们仅在新颖指令上测试相同的合作伙伴模型时，我们会看到 1% 的微小下降。如上所述，对于接受所有任务训练的合作模型和保留任务的合作模型来说，上下文表征的正确率相对较低，分别为 30% 和 37%。

最后，我们测试了最极端的环境，其中为感觉运动 RNN 和生产 RNN 执行了任务（图 5f）。我们发现，对于在所有任务和保留任务上训练的合作模型，生成的指令分别带来 71% 和 63% 的性能。尽管这与我们之前的设置相比性能有所下降，但模型在这种双保留设置中完全可以产生合理的指令这一事实是惊人的。该系统在某种程度上取得成功的事实说明了在丰富的、组合结构的语义表示的背景下进行训练所引入的强烈归纳偏差。

讨论

在这项研究中，我们利用自然语言处理的最新进展来构建易于处理的模型，以解释指令的能力



在新奇的环境中指导行动，以及在学习任务后生成任务描述的能力。RNN 可以使用预先训练的语言转换器来学习同时执行一组心理物理任务，为当前任务嵌入自然语言指令。我们性能最佳的模型可以利用这些嵌入来执行平均性能为 83% 正确率的全新模型。概括性能的指令模型通过利用指令嵌入和任务表示的共享组成结构来实现这一点，这样对实践指令和新指令之间关系的推断就可以很好地推断出未见过的任务需要什么感觉运动转换。最后，我们展示了一个网络可以反转这些信息，并仅根据其观察到的感觉运动偶然性为任务提供语言描述。

我们的模型对整合语言信息以控制感觉运动区域的大脑区域中期望的神经表征做出了一些预测。首先，我们的模型层次结构的 CCGP 分析表明，当人类必须根据指令泛化（或在之间切换）一组相关任务时，在感觉运动映射中观察到的神经几何也应该出现在指令的语义表示中。这一预测在现有的实验文献中有充分的依据，其中多项研究观察到我们在感觉运动 RNN 中发现的抽象结构类型也存在于生物大脑的感觉运动区域中。我们的模型认为，语言区域中等效的任务相关结构的出现对于人类的指导行动至关重要。可能支持这种表征的一个有趣的候选区域是左额下回的语言选择亚区域。该区域对句子理解的词汇语义和句法方面都很敏感，涉及需要语义控制的任务，并且在解剖学上与左额下回的另一个功能子区域相邻，该子区域涉及灵活认知。我们还预测，参与实现感觉运动映射的各个单元应根据输入指令的语义在反复试验的基础上调节其调节属性，并且未能以预期方式调节调节应导致泛化不良。这一预测对于解释人类的多单元记录可能特别有用。最后，鉴于任务集的感觉运动需求中的语言知识基础提高了模型的性能（图 1）。2e），我们预测在学习过程中，语言处理层次的最高级别同样应该由伴随语言输入的具体过程形成，例如运动规划或可供性评估。

我们研究的一个值得注意的负面结果是 GPTNET (XL) 的泛化性能相对较差，它使用的参数比其他模型至少多一个数量级。鉴于这些模型中的活动可以预测人类语言处理的许多行为和神经特征，这一点尤其引人注目。鉴于此，未来的成像研究可能会以自回归模型和我们表现最好的模型中的表示为指导，以描绘指令遵循的每个阶段所涉及的大脑区域的完整梯度，从低级下一个单词预测到高级语言所传达的感觉运动控制的结构化句子表示。

我们的模型可以指导未来的工作，比较非语言学科（如非人类灵长类动物）的成分表征。人类和非人类灵长类动物之间的任务切换（没有语言指令）的比较表明，尽管人类可以更快地进行切换，但两者都使用抽象规则表示。我们的分析中一个有趣的相似之处是组合规则向量的使用（补充图 5）。即使在非语言 SIMPLINET 的情况下，使用这些向量也可以增强泛化能力。然而，重要的是，对于我们表现最好的指导模型来说，这种组合性要强得多

这表明语言赋予代理更灵活的任务子组件组织，可以在更广泛的上下文中重新组合。

我们的结果还凸显了语言交流的优势。网络可以压缩通过运动反馈经验获得的信息，并通过自然语言将该知识传输到合作伙伴网络。虽然在我们的例子中还很初级，但经过一段时间的练习后，内生地产生如何完成任务的描述的能力是人类语言技能的标志。通过共享潜在表征来传递性能的失败表明，要在一组独立的神经元网络中传递信息，它需要通过一种可由该组所有成员同等解释的表征媒介。对于人类和我们表现最好的指导模型来说，这种媒介就是语言。

强化学习领域的一系列工作研究了使用语言和类语言方案来帮助智能体表现。代理通过动作序列的逐步描述或通过学习以语言目标为条件的策略来接收语言信息。这些研究通常偏离自然语言，并接受经过解析或直接引用环境对象的语言输入。我们用来嵌入指令的一些较大版本的预训练语言模型还显示遵循行为的指令，即语言模态中的 GPT-3（参考文献 7）、PALM、LaMDA 和 InstructGPT 以及语言模态中的 DALL-E 和稳定扩散语言到图像形态。这些模型中显示的语义和句法理解令人印象深刻。然而，这些模型的输出很难解释为指导下游行动计划的动态。最后，最近的工作试图设计能够在复杂甚至现实环境中发挥作用的指令跟随代理。虽然这些模型表现出令人印象深刻的行为能力，但它们依赖于融合语言和视觉信息的感知系统，这使得它们很难与人脑中的语言表征进行比较，而人脑中的语言表征来自一组专门用于处理语言的区域。总而言之，这些模型都没有提供可测试的代表性说明，说明如何使用语言来诱导大脑中感觉运动映射的泛化。

相比之下，我们的模型可以对支持成分泛化所需的群体和单元神经表征做出易于处理的预测，并可以指导未来检查人类语言和感觉运动技能相互作用的实验工作。通过开发可解释的模型，该模型既可以将指令理解为指导特定的感觉运动反应，又可以将感觉运动学习的结果作为可理解的语言指令进行交流，我们已经开始解释语言在神经网络中编码和传输知识的力量。

在线内容

任何方法、附加参考文献、自然组合报告摘要、源数据、扩展数据、补充信息、致谢、同行评审信息；作者贡献和竞争利益的详细信息；数据和代码可用性声明可在 <https://doi.org/10.1038/s41593-024-01607-5> 上获取。

参考

1. 科尔, M.W. 等人。多任务连接揭示了用于自适应任务控制的灵活集线器。自然神经科学。 16, 1348–1355 (2013)。
2. Miller, E.K. & Cohen, J.D. 前额皮质功能的综合理论。安努。神经科学牧师。 24, 167–202 (2001)。
3. 贝尔纳迪, S.等人。海马体和前额叶皮层抽象的几何形状。细胞 183, 954–967 (2020)。
4. Minxha, J.、Adolphs, R.、Fusi, S.、Mamelak, A. N. 和 Rutishauser, U. 人类内侧额叶皮层基于记忆的选择表征的灵活招募。科学 368, eaba3313 (2020)。

5. Takuya, I.等人。通过人类和人工神经网络中的抽象表示进行组合概括。在过程中。
第36届神经信息处理系统会议
(Koyejo, S. 等编辑) 32225–32239 (Curran Associates, Inc., 2022)。

6. Driscoll, L., Shenoy, K. 和 Sussillo, D. 循环网络中的灵活多任务计算利用共享动态主题。BioRxiv 预印本 <https://doi.org/10.1101/2022.08.15.503870> (2022)。

7. 布朗、汤姆等人。语言模型是小样本学习者。
在过程中。第34届国际神经信息会议
处理系统 1877–1901 (Curran Associates Inc., 2020)。

8. 拉梅什, A.等人。零样本文本到图像生成。在过程中。 38号
国际机器学习会议 (Marina, M. 和 Tong, Z. 编辑) 8821–8831 (PMLR, 2021)。

9. 雷德福, A.等人。语言模型是无监督的多任务学习者。 OpenAI 1, 9 (2019)。

10. 施林普夫, M.等人。语言的神经架构：集成建模汇聚于预测处理。过程。国家
科学院。
科学。美国 <https://doi.org/10.1073/pnas.2105646118> (2021)。

11. 戈尔茨坦, A.等人。人类语言处理和深度语言模型的共享计算原理。自然神经
科学。 25, 369–380 (2022)。

12. 乔杜里, A.等人。Palm：通过路径扩展语言建模。 J.马赫.学习。资
源。 24, 11324–11436 (2023)。

13. 托皮兰, R.等人。Lambda：对话应用程序的语言模型。预印本位于 <https://arxiv.org/abs/2201.08239> (2022)。

14. 罗姆巴赫, R.等人。具有潜在特征的高分辨率图像合成
扩散模型。在过程中。 2022 年 IEEE/CVF 计算机视觉和模式识别会议 (CVPR) 10674–10685 (IEEE, 2022)。

15. 齐特科维奇, B.等人。Rt-2：视觉-语言-动作模型将网络知识转移到机器人
控制。在过程中。第七届机器人学习会议 (Tan, J. 等编辑) 2165–2183
(PMLR, 2023)。

16. 艾布拉姆森, J.等人。模仿交互式智能。预印本位于 <https://arxiv.org/abs/2012.05672> (2021)。

17. DeepMind 交互式代理团队。通过模仿和自我监督学习创建多模式交互式
代理。
预印本位于 <https://arxiv.org/abs/2112.03763> (2022)。

18. Yang, G. R., Joglekar, M. R., Song, H. F., Newsome, W. T. 和 Wang, X.-J. 神经网络中的任务表示经过训练可以执行许多认知任务。纳特。神经科
学。 22, 297–306 (2019)。

19. 瓦斯瓦尼, A.等人。您所需要的就是关注。在过程中。 31号
神经信息处理系统国际会议 6000–6010 (Curran Associates Inc., 2017)。

20. Devlin, J., Chang, M., Lee, K. 和 Toutanova, K. BERT：用于语言理解的深度双向转换器的预训练。
预印本位于 <http://arxiv.org/abs/1810.04805> (2018)。

21. Reimers, N. & Gurevych, I. Sentence-bert：使用暹罗 bert 网络的句子嵌入。
预印本位于 <https://arxiv.org/abs/1908.10084> (2019)。

22. Bowman, S. R., Angeli, G., Potts, C. 和 Manning, C. D. 用于学习自然
语言推理的大型注释语料库。
预印本位于 <http://arxiv.org/abs/1508.05326> (2015)。

23. 雷德福, A.等人。“从自然语言监督中学习可转移的视觉模型。第 38
届国际机器学习会议 (编者：Marina, M. & Tong, Z.) 8748–8763
(PMLR, 2021)。

24. Goel, V., Gold, B., Kapur, S. 和 Houle, S. 人类推理的神经解剖学关联。 J.科
格恩。神经科学。 10, 293–302 (1998)。

25. Goel, V., Buchel, C., Frith, C. 和 Dolan, R. J. 三段论推理机制的分离。神
经影像 12, 504–514 (2000)。

26. Reverberi, C.等人。初等演绎结论生成的神经基础。神经影像 38, 752–
762 (2007)。

27. Noveck, I. A., Goel, V. 和 Smith, K. W. 任意内容条件推理的神经基础。皮质
40, 613–622 (2004)。

28. Monti, M.M., Osherson, D.N., Martinez, M.J. & Parsons, L.M.演绎推
理的功能神经解剖学：独立于语言的分布式网络。神经影像 37, 1005–1016
(2007)。

29. Monti, M.M., Parsons, L.M. 和 Osherson, D.N. 演绎推理中语言和思想的
界限。过程。国家科学院。
科学。美国 106, 12554–12559 (2009)。

30. Coetzee, J. P. & Monti, M. M. 推理的核心：分离演绎和非演绎负载。哼。大脑
地图。 39, 1850–1861 (2018)。

31. Monti, M. M. & Osherson, D. N. 逻辑、语言和大脑。
脑研究。 1428, 33–42 (2012)。

32. Prado, J. 演绎推理与布罗卡区语言句法之间的关系：神经影像学文献综述。
L'Année Psychol. 118, 289–315 (2018)。

33. Ito, T., Yang, G. R., Laurent, P., Schultz, D. H. 和 Cole, M. W. 从大脑数
据构建神经网络模型揭示了与适应性行为相关的表征转换。

纳特。交流。 13, 673 (2022)。

34. Shadlen, M.N. 和 Newsome, W.T. 恒河猴顶叶皮层（唇区）知觉决策的
神经基础。
J.神经生理学。 86, 1916–1936 (2001)。

35. Huk, A. C. & Shadlen, M. N. 猕猴顶叶皮层的神经活动反映了感知决策过程中视
觉运动信号的时间整合。 J.神经科学。 25, 10420–10436 (2005)。

36. Panichello, M. F. & Buschman, T. J. 共享机制是工作记忆和注意力控制的基础。
自然 592, 601–605 (2021)。

37. Nieh, E.H.等人。海马体中抽象学习知识的几何形状。自然 595, 80–84
(2021)。

38. Fedorenko, E. 和 Blank, I. A. Broca 区域不是自然区域。
趋势认知。科学。 24, 270–284 (2020)。

39. Fedorenko, E., Duncan, J. 和 Kanwisher, N. 语言选择性区域和通用领域区域
并排位于布罗卡区域内。电流。
生物。 22, 2059–2062 (2012)。

40. 高, Z.等人。语义和非语义控制需求的独特和共同的神经编码。神经图像 236,
118230 (2021)。

41. Duncan, J. 灵长类大脑的多需求 (MD) 系统：智能行为的心理程序。趋势认
知。
科学。 14, 172–179 (2010)。

42. Buccino, G., Colag , I., Gobbi, N. 和 Bonaccorso, G. 经验中的意义基
础：对具身语言的广阔视角。神经科学。生物行为。修订版 69, 69–78
(2016)。

43. Mansouri, F. A., Freedman, D. J. 和 Buckley, M. J. 灵长类大脑中抽象规则的
出现。纳特。神经科学牧师。 21, 595–610 (2020)。

44. 哦, J. Singh, S., Lee, H. 和 Kohli, P. 具有多任务深度强化学习的零样本任务
泛化。在过程中。第 34 届国际机器学习会议 2661–2670 (JMLR.

组织, 2017)。

45. Chaplot, D. S., Mysore Sathyendra, K., Pasumarthi, R.K., Rajagopal, D.
和 Salakhutdinov, R. 用于面向任务的语言基础的门控注意力架构。在过程
中。第 32 届 AAAI 人工智能会议32 (AAAI 出版社, 2018 年)。

46. Sharma, P., Torralba, A. 和 Andreas, J. 潜在语言的技能归纳和规划。
预印本位于 <https://arxiv.org/abs/2110.01517>(2021)。

47. Jiang, Y., Gu, S., Murphy, K. & Finn, C. 语言作为分层深度强化学习的抽象。
在过程中。第 33 届神经信息处理系统国际会议 9419–943132 (Curran
Associates Inc., 2019)。

48. 欧阳, L.等。训练语言模型遵循指令
与人类反馈。神经信息进展
处理系统 27730–27744 (Curran Associates, Inc., 2022)。

出版商说明施普林格·自然对于已出版地图和机构隶属关系中的管辖权主张保持中立。

开放获取本文根据知识共享署名 4.0 国际许可证获得许可，该许可证允许以任何媒介或格式使用、共享、改编、分发和复制，只要您对原作者和来源给予适当的认可，提供知识共享许可证的链接，并指出是否进行了更改。本文件中的图像或其他第三方材料

除非材料的信用额度中另有说明，否则文章包含在文章的知识共享许可中。如果文章的知识共享许可中未包含材料，并且您的预期用途不受法律法规允许或超出了允许的用途，您将需要直接获得版权所有者的许可。要查看此许可证的副本，请访问 <http://creativecommons.org/licenses/by/4.0/>。

© 作者 2024

方法

模型架构

感觉运动-RNN。本文使用的基本模型架构和任务结构如下。所有表示为 sensorimotor-RNN 的感觉运动单元网络都是使用修正线性单元（ReLU）非线性的门控循环单元（GRU），每个网络有 256 个隐藏单元。网络的输入由 (1) 感官输入 X 和 (2) 任务识别信息 I 组成。我们将 GRU 中的隐藏活动初始化为 $h \in \mathbb{R}$ ，并将值设置为 0.1。所有感觉运动单元网络都使用相同的隐藏状态初始化，因此我们省略了网络方程。在每个时间步，读出层线性解码运动活动， \hat{y}

y ，从

循环隐藏单元的活动 h ，根据：

$$\begin{aligned} h &= \text{感觉运动RNN}(X, I; h) & h \in \mathbb{R} \\ \hat{y} &= \sigma(\text{线性}(h)) & y \in \mathbb{R} \end{aligned}$$

其中 σ 表示 sigmoid 函数。感觉输入 X 由三个通道、两个感觉模式 x 和 x 以及一个注视通道 x 组成。这些模态中的 x 、 $x \in \mathbb{R}$ 和刺激都表示为活动山，其峰值由单元围绕一维圆形变量的首选方向确定。对于方向 θ 的输入，具有首选方向 θ 的给定输入单元 u 的活动为

$$u = \text{str} \times 0.8 \exp \left[-0.5 \times \left(\frac{8|\theta - \theta|}{\pi} \right)^2 \right]$$

其中 str 是描述刺激强度的系数。注视通道 $x \in \mathbb{R}$ 是模拟网络注视线索的单个单元。总之，感觉输入。电机输出 \hat{y} 由表示对输入刺激的方向响应的 32 维环以及表示模型固定的单个单元组成，因此 \hat{y}

For all 模型， $y \in \mathbb{R}$ 。任务识别信息 $I \in \mathbb{R}$ 。任务识别信息在整个试验期间呈现并保持恒定，使得 $I = I \forall t, t'$ 。对于所有模型，任务识别信息 I 和感觉输入 X 连接起来作为感觉运动 RNN 的输入。

非语言模型。对于 SIMPLENET，我们通过使用 Python 包 `scipy.stats.ortho_group` 构造正交矩阵来生成一组 64 维正交任务规则，并将该矩阵的行分配给每个任务类型。对于 STRUCTURENET，我们以相同的方式生成一组 10 个正交的 64 维向量，每个向量代表任务集的一个维度（即，响应最弱方向与最强方向、响应相同方向与相反方向、支付仅注意第一种模式中的刺激，依此类推）。任务的规则向量是这十个基本向量中每一个的简单组合。有关结构规则向量的完整描述，请参阅补充说明 3。

我们还测试了 SIMPLENETPLUS 和 STRUCTURENETPLUS，它们使用具有 128 个单元的附加隐藏层和 ReLU 非线性将正交任务规则 I 处理成向量 \tilde{I} 它被感觉运动 RNN 用作任务识别信息。

$$\begin{aligned} \tilde{I} &= \text{ReLU}(\text{线性}(I)) & \tilde{I} \in \mathbb{R} \\ \tilde{I} &= \text{ReLU}(\text{LinearRuleEmb2}(I)) & \tilde{I} \in \mathbb{R} \\ \tilde{I} &= \text{ReLU}(\text{线性}(I)) & \tilde{I} \in \mathbb{R} \end{aligned}$$

这些模型的完整结果包含在补充图 4 中。

预训练的变压器。我们测试的主要语言模型使用预训练的 Transformer 架构来生成 I 。重要的是，

Transformer 的不同之处在于用于调整模型参数的预训练目标类型。GPT 经过训练，可以在给定单词上下文的情况下预测下一个单词。GPT (XL) 遵循相同的目标，但在更大的数据集上训练时间更长。两种模型都是完全自回归的。相比之下，BERT 接受双向语言输入，其任务是预测出现在输入短语中间的屏蔽词。此外，BERT 还接受简单句子预测任务的训练，其中模型必须确定训练语料库中输入句子 1 后面是否跟着输入句子 2。扩展这一原则，SBERT 经过显式训练以生成整个句子的固定长度嵌入。它采用预训练的 BERT 网络并将其用于 siamese 架构，从而允许根据斯坦福自然语言推理数据集以监督方式调整模型的权重。自然语言推理是一项三向分类任务，网络必须推断句子之间的逻辑关系：前提句是否暗示、矛盾或与假设句无关。最后，CLIP 被训练来联合嵌入图像和语言。它使用来自标题图像的数据，并被要求通过对比损失对数据集中匹配或不匹配的文本和图像对进行正确分类。

重要的是，变压器的自然输出是大小矩阵

$\text{dim} \times \mathcal{T}$ ，变压器的固有维数与输入序列的长度有关。要为句子创建嵌入空间，标准做法是对变压器输出应用池化方法，这会为每条指令生成固定长度的表示。

对于 GPT、GPT (XL)、BERT 和 SBERT，我们使用平均池化方法。假设我们有一条输入指令 $w \cdots w$ 。遵循预训练语言模型的标准实践，变压器的输入在输入序列的开头和结尾处使用特殊的“cls”和“eos”标记进行标记。然后我们计算 I 如下：

$$\begin{aligned} h &= \text{变压器}([\text{cls}], w \cdots w, [\text{eos}]), & h \in \mathbb{R} \\ h &= \text{平均值}(h), & h \in \mathbb{R} \\ I &= \text{线性}(h) & I \in \mathbb{R} \end{aligned}$$

我们选择这种平均池化方法主要是因为之前的一项研究发现这会产生性能最高的 SBERT 嵌入。另一种选择是简单地使用“cls”标记的最终隐藏表示作为整个序列中信息的摘要（鉴于 BERT 架构是双向的，该标记将有权访问整个序列）。

$$\begin{aligned} h &= \text{Transformer}([\text{cls}], w \cdots w, [\text{eos}]), & h \in \mathbb{R} \\ \text{小时} &= (\frac{1}{\sqrt{N}} \text{时}) & h \in \mathbb{R} \end{aligned}$$

其中 h_{cls} 表示“cls”标记的最后一个隐藏表示。参考号 21 发现这种池方法的性能比平均池方法差，因此我们的结果中没有包含这些替代方法。对于 GPT 和 GPT (XL)，我们还测试了一种池化方法，其中序列的固定长度表示取自“eos”令牌的转换器输出。在这种情况下：

$$\begin{aligned} h &= \text{Transformer}([\text{cls}], w \cdots w, [\text{eos}]), & h \in \mathbb{R} \\ h &= (h_{\text{eos}}), & h \in \mathbb{R} \\ I &= \text{线性}(h), & I \in \mathbb{R} \end{aligned}$$

我们发现，使用这种池化方法，GPT 甚至无法在任务中实现 85% 的宽松性能标准，并且 GPT (XL) 的表现比平均池化更差，因此我们从主要结果中省略了这些模型（补充图 11）。对于 CLIP 模型，我们使用与原始 multiModal 相同的池化方法

训练过程，它采用如上所述的 [cls] 令牌的输出。

对于上述所有模型，我们还测试了一个版本，其中来自预训练 Transformer 的信息通过具有 256 个隐藏单元和 ReLU 非线性的单个隐藏层的多层感知器。我们发现这种操作降低了所有模型的性能，验证了简单的线性嵌入有利于泛化性能。

对于 GPT、BERT 和 SBERT，dim = 768，每个模型总共使用约 1 亿个参数；对于 SBERT (L)，该模型使用约 3 亿个参数；GPT (XL) dim = 1, 600 并且模型使用约 15 亿个参数；对于 CLIP，该模型使用约 6000 万个参数。完整的 PyTorch 实现，包括所有预训练的权重和模型超参数，可以在 Huggingface 库 (<https://huggingface.co/docs/transformers/>) 中访问。

弓模型。对于我们的 BoW 模型，指令被表示为指令词汇大小的二进制激活向量，其中每个单元指示当前指令中关联词的包含或排除。对于我们的指令集， $|\text{vocab}| = 181$ 。然后该向量通过线性层投影到 64 维空间。

$$h_i = \begin{cases} 1 & \text{如果 } w \in (w \cdots w) \\ 0 & \text{否则} \end{cases} \quad h \in \mathbb{R}$$
$$l = \text{线性}(h), \quad l \in \mathbb{R}$$

$$\theta = \begin{cases} \theta & \text{if Mod1 任务} \\ \theta & \text{if Mod2 任务} \end{cases}$$

白板语言模型。鉴于调整语言模型的最后一层可以提高性能（图 2e），我们测试了另外两个模型，以确定训练专门针对感觉运动任务损失进行训练的空白语言模型是否会提高性能。这些模型包括通过多层感知器传递 BoW 表示，以及通过随机初始化的 BERT 编码器的一层传递预训练的 BERT 词嵌入。与预训练模型相比，这两种模型的表现都很差（补充图 4.5），这证实了语言预训练对于泛化至关重要。

任务集

任务被分为五个相互关联的子组：“执行”、“决策”、“匹配”、“比较”和“持续时间”。根据任务的不同，在刺激时期可能会出现多种刺激。此外，根据任务的不同，模型可能需要向特定方向做出响应或完全抑制响应。除非另有说明，零均值高斯噪声在每个时间步独立添加到每个输入单元，并且该噪声的方差从中随机抽取。不同任务类型的刺激时间有所不同。然而，对于所有任务，试验可以分为准备期、刺激期和反应期。刺激时期可以细分为三个部分：stim1、delay 和 stim23，尽管并非所有任务都会使用这些不同的部分。试验总共持续 T = 150 个时间步。让 dur 表示给定历元的模拟时间步长的持续时间。然后

$$\text{dur} \sim \{i | 20 < i \leq 25; i \in \mathbb{N}\}$$
$$\text{dur}, \text{dur} \sim \{i | 37 < i \leq 50; i \in \mathbb{N}\}$$
$$\text{dur} \sim \{i | 15 < i \leq 25; i \in \mathbb{N}\}$$
$$\text{杜尔} = 150 - (\text{杜尔} + \text{杜尔} + \text{杜尔} + \text{杜尔})$$

对于不使用延迟结构的任务，stim1、stim2 和延迟时期被分组在单个刺激时期中，其中

$\text{dur}_{\text{刺激}} = \text{dur} + \text{dur} + \text{dur}$ 。除非另有说明，否则

具有恒定强度 str = 1 的注视线索在整个准备期和刺激期被激活。有关每项任务的试验示例，请参见补充图 13。

“去”任务。“Go”系列任务包括“Go”、“RTGo”、“AntiGo”、“AntiRTGo”以及用“Mod1”和“Mod2”表示的每个任务的特定模式版本。在“Go”和“AntiGo”任务中，在刺激纪元开始时呈现单个刺激。所呈现的刺激的方向是通过从 0 到 2π 之间的均匀分布（即 θ ）绘制而生成的。刺激将以相同的概率出现在模态 1 或模态 2 中。刺激的强度由给出。在“Go”任务中，目标响应与呈现的刺激方向相同，即 $\theta = \theta$ ，而在“AntiGo”任务中，响应方向应与刺激方向相反，即 $\theta + \pi = \theta$ 。对于每个任务的特定模态版本，在每个模态 $\theta \sim \mathcal{U}(\mathcal{U}(0, 2\pi))$ 和 $\theta \sim \mathcal{U}(\mathcal{U}(0, 2\pi))$ 以及特定模态的 Go 类型任务中绘制刺激方向

$$\theta = \begin{cases} \theta + \pi & \text{如果 Mod1 任务} \\ \theta + \pi & \text{如果 Mod2 任务} \end{cases}$$

对于“Go”任务的“RT”版本，刺激仅在响应时期呈现，并且注视线索永远不会消失。因此，刺激本身的存在充当响应线索，模型必须尽快响应。否则，刺激会在整个刺激时期持续存在。

“决策”任务。“决策”系列任务包括“DM”（决策）、“AntiDM”、“MultiDM”（多感官决策）、“AntiMultiDM”以及每个任务的特定模式版本，最后，基于置信度的“DM”和“AntiDM”版本。对于该组中的所有任务，同时呈现两种刺激，并在整个刺激时期持续存在。它们是根据 $\theta \sim \mathcal{U}(\mathcal{U}(0, 2\pi))$ 和 $\theta \sim \mathcal{U}(\mathcal{U}(\theta - 0.2\pi, \theta - 0.6\pi) \cup (\theta + 0.2\pi, \theta + 0.6\pi))$ 绘制的。绘制应用于两种刺激的基础强度，使得 $\text{str} \sim \mathcal{U}(\mathcal{U}(1.0, 1.2])$ 。从离散分布中得出对比，使得 $c \sim \{-0.175, -0.15, -0.1, 0.1, 0.15, 0.175\}$ 因此与每个方向相关的刺激强度

在试验中，由 $\text{str} = \text{str} + c$ 和 $\text{str} = \text{str} - c$ 给出。

对于“DM”任务，

$$\theta = \begin{cases} \theta & \text{如果字符串} > \text{字符串} \\ \theta & \text{否则} \end{cases}$$

$$\theta = \begin{cases} \theta & \text{如果 str} \\ \theta & \text{否则} \end{cases}$$

对于这些版本的任务，刺激以相同的概率以模态 1 或模态 2 呈现。对于每个任务的多感官版本，刺激方向以相同的方式绘制并跨两种模式呈现，使得 $\theta = \theta$ 和 θ 。每种模式的基础优势都是独立得出的。两种模态的对比是从离散分布中得出的，使得 $c, c \sim \{0.2, 0.175,$

然后重新绘制对比以避免训练期间的零对比。如果两个方向可以划出相同的符号，则对比是

重新设计以确保试验需要整合两种模式，而不是简单地以单一模式执行“DM”任务。目标响应的标准是通过两种模式的给定方向的强度来衡量的。所以，对于“MultiDM”

$$\theta = \begin{cases} \theta & \text{如果 } \text{str} + \text{str} > \text{str} \\ +\text{str} & \\ \theta & \text{否则} \end{cases}$$

以及“AntiMultiDM”

$$\theta = \begin{cases} \theta & \text{如果 } \text{str} + \text{str} < \text{str} \\ +\text{str} & \\ \theta & \text{否则} \end{cases}$$

每个任务的特定模态版本的刺激的生成方式与任务的多感官版本相同。目标响应的标准与仅应用于相关模式中的刺激的“DM”和“AntiDM”任务的标准版本相同。

在基于置信度的决策任务（“ConDM”和“ConAntiDM”）中，刺激方向的绘制方式与上述相同。刺激以相同的概率以模态 1 或模态 2 显示。在每次试验中，str= 1。每次试验的对比度和噪声基于在除“ConDM”和“ConAntiDM”之外的所有任务上训练的 SIMPLENET 模型的阈值性能。一旦该模型经过训练，我们就会在噪声和对比度级别上建立一个阈值，使模型能够以 95% 的正确率执行“DM”或“AntiDM”任务。然后，我们在训练期间以相同的概率从高于和低于该阈值的试验中绘制对比和噪声。在噪声和对比度水平低于 95% 正确阈值的试验中，模型必须抑制响应，否则执行决策任务（“DM”或“AntiDM”）。

“比较”任务。我们的比较任务组包括每个任务的“COMP1”、“COMP2”、“MultiCOMP1”、“MultiCOMP2”、“反”本，以及“COMP1”和“COMP2”任务的特定模式版本。这组任务旨在将基本决策框架扩展到具有更复杂控制需求的环境中。这些任务利用刺激时期中的延迟结构，以便 stim1 仅在 stim1 时期出现，然后是延迟，最后是 stim2。这提供了刺激的时间顺序。在“COMP1”中，只有当第一个刺激的强度大于第二个刺激的强度时，模型才必须对其做出响应，否则会抑制以下响应：

$$\theta = \begin{cases} \theta & \text{如果字符串} > \text{字符串} \\ \text{抑} & \text{否则} \end{cases}$$

同样，在“COMP2”中，如果第二个方向的强度大于第一个方向，则模型必须对第二个方向做出响应，否则会抑制响应，即

$$\theta = \begin{cases} \theta & \text{如果 } \text{str} > \text{str} \\ \text{抑} & \text{否则} \end{cases}$$

在任务的“反”版本中，除了强度最小的刺激之外，排序标准是相同的，即“AntiCOMP1”

$$\theta = \begin{cases} \theta & \text{如果 } \text{str} < \text{str} \\ \text{抑} & \text{否则} \end{cases}$$

以及“AntiCOMP2”

$$\theta = \begin{cases} \theta & \text{如果 } \text{str} < \text{str} \\ \text{抑} & \text{否则} \end{cases}$$

在多感官环境中，目标方向的标准类似于多感官决策任务，其中力量是跨模式整合的。同样，对于特定于模态的版本，标准仅适用于相关模态中的刺激。每项任务的刺激方向和强度都来自与“决策”系列中的类似任务相同的分布。然而，在训练过程中，我们确保平衡需要响应的试验和模型必须抑制响应的试验。

“持续时间”任务。“持续时间”任务系列包括“Dur1”、“Dur2”、“MultiDur1”、“MultiDur2”、每个任务的“Anti”版本以及“Dur1”和“Dur2”任务的特定模式版本。这些任务需要模型执行时间估计任务，并添加需求或刺激排序来确定响应的相关性。就像在“比较”任务中一样，刺激 1 出现，然后延迟，然后刺激 2。对于“Dur1”试验

$$\theta = \begin{cases} \theta & \text{如果杜尔} > \text{杜尔} \\ \text{抑} & \text{否则} \end{cases}$$

同样，对于“Dur2”

$$\theta_{\text{目标}} = \begin{cases} \theta & \text{如果杜尔} > \text{杜尔} \\ \text{抑} & \text{否则} \end{cases}$$

在这些任务的“反”版本中，正确的反应是在满足排序标准的情况下，持续时间最短的刺激方向。因此，对于‘AntiDur1’

$$\theta = \begin{cases} \theta & \text{如果持续时间} < \text{持续时间} \\ \text{抑} & \text{否则} \end{cases}$$

对于“AntiDur2”

$$\theta_{\text{目标}} = \begin{cases} \theta & \text{如果持续时间} < \text{持续时间} \\ \text{抑} & \text{否则} \end{cases}$$

在这些任务中，根据 $\theta \sim \mathcal{U}([0, 2\pi])$ 和 $\theta \sim \mathcal{U}([\theta - 0.2\pi, \theta - 0.6\pi] \cup (\theta + 0.2\pi, \theta + 0.6\pi])$ 绘制方向。

刺激强度根据str, $\text{str} \sim \mathcal{U}([0.8, 1.2])$ 绘制。为了设置每个刺激的持续时间，我们首先绘制

$\{i|35 < i \leq 50, i \in \mathbb{N}\}$ 和 $\text{dur} \sim \{i|25 < i \leq (\text{dur} - 8), i \in \mathbb{N}\}$ 。期间训练时，我们确定给定任务的哪些试验应该和不应该需要响应，以便均衡地平衡抑制和响应试验。然后，我们将 durand dur 分配给 stim1 或 stim2，以便试验需要给定特定任务类型的适当响应。

同样，每个任务的多感官和特定模态版本中正确响应的标准遵循“决策”和“比较”组中的类似任务，其中任务的多感官版本需要整合每种模态和特定模态的总持续时间任务只需要考虑给定任务模式的持续时间。对于多感官任务，我们绘制持续时间值

$\text{dur} \sim \{i|75 < i \leq 100, i \in \mathbb{N}\}$ ，然后分割该值 $\text{dur} = \text{dur} \times 0.55$ 和 $\text{dur} = \text{dur} \times 0.45$ 。我们还得出一个值 $\text{dur} = \text{dur} - \Delta \text{dur}$ ，其中 $\Delta \text{dur} \sim \{i|15 < i \leq 25, i \in \mathbb{N}\}$ 。然后将该值进一步细分为 $\text{dur}_{\text{short}0} = \text{dur} + \Delta \text{dur}$ ，其中 $\Delta \text{dur} \sim \{i|19 < i \leq 15, i \in \mathbb{N}\}$ 和 $\text{dur}_{\text{short}1} = \text{dur} - \text{dur}$ 。然后可以根据任务类型将短持续时间和长持续时间分配给有序刺激。以这种方式绘制持续时间可以确保，就像在“决策”和“比较”组中一样，正确的答案确实需要模型整合两种模式的持续时间，而不是简单地以给定的模式执行任务来获得正确的响应。

“匹配”任务。“匹配”任务系列包括“DMS”（与刺激的延迟匹配）、“DNMS”（与刺激的延迟不匹配）、“DMC”

（延迟与类别匹配）和“DMNC”（延迟与类别不匹配）任务。对于所有任务，stim1 在刺激时期开始时呈现，然后是延迟，然后呈现 stim2。刺激强度根据 str, $\text{str} \sim \mathcal{U}[0.8, 1.2]$ 绘制。任何给定试验的输入模式都是以相等的概率随机选择的。在“DMS”和“DNMS”任务中，试验被构造为具有相同概率的“匹配刺激”试验或“不匹配刺激”试验。在“匹配刺激”试验中 $\theta \sim \mathcal{U}[0, 2\pi]$ 且 $\theta = \theta$ 。在“错配刺激”试验中， $\theta \sim \mathcal{U}[0, 2\pi]$ 和

$$\theta \sim \mathcal{U}[(\theta - 0.2\pi, \theta - 0.6\pi) \cup (\theta + 0.2\pi, \theta + 0.6\pi)]。$$

对于“DMS”，如果刺激匹配，模型必须在显示的方向上做出响应，否则会抑制响应，

$$\theta = \begin{cases} \theta & \text{如果 } \theta = \theta \\ \text{抑} & \text{否则} \end{cases}$$

对于“DNMS”，如果两个方向不匹配，模型必须响应第二个方向，

$$\theta = \begin{cases} \theta & \text{如果 } \theta \neq \theta \\ \text{抑} & \text{否则} \end{cases}$$

“DMC”和“DNMC”任务以类似的方式组织。刺激输入空间被均匀地分为两类，即 $\text{cat1} = \{\theta: 0 < \theta \leq \pi\}$ 和 $\text{cat2} = \{\theta: \pi < \theta \leq 2\pi\}$ 。对于“DMC”和“DNMC”任务，试验被构造为具有相同概率的“匹配猫”试验或“不匹配猫”试验。在“匹配猫”试验中 θ 和 $\theta \sim \mathcal{U}(\text{cat})$ ，其中 $\mathcal{U}(\text{cat})$ 是从 stim1 类别中均匀抽取的。在‘失配刺激’试验中， θ 和 θ ，其中-是与刺激1相反 的类别。对于“DMC”，如果两个刺激都出现在同一类别中，则模型必须在第一个方向做出响应，否则会抑制响应，

$$\theta = \begin{cases} \theta & \text{如果猫} = \text{猫} \\ \text{抑} & \text{否则} \end{cases}$$

对于“DNMC”，如果两个刺激都以相反 的类别呈现，则模型应该对第二个方向做出响应，否则会抑制响应，

$$\theta = \begin{cases} \theta & \text{如果猫} \neq \text{猫} \\ \text{抑} & \text{否则} \end{cases}$$

目标输出和正确标准

试验的目标输出 $y \in \mathbb{R}$ 需要在刺激时期保持 $y = y$ 的固定，然后在正确的方向上做出反应，或者在响应时期抑制剩余目标响应单元的活动。由于模型应保持注视直到响应，因此在准备和刺激时期将注视目标设置为 $y = 0.85$ ，在响应时期将注视目标设置为 $y = 0.05$ 。当不需要响应时，如在准备期和刺激期以及响应期中的活动受到抑制，单元 i 的目标活动为 $y = 0.05$ 。或者，当有响应的目标方向时，

$$y = 0.8 \exp \left[-0.5 \times \left(\frac{8|\theta - \theta|}{\pi} \right)^2 \right] + 0.05$$

其中 θ 是单元 i 的首选方向。与感觉刺激一样，目标单元的首选方向是分配给 32 个响应单元的 $[0, 2\pi]$ 中均匀间隔的值。

为了使模型响应算作正确，它必须保持固定，即 \hat{y}

$$y > 0.5 \quad \text{在准备期和刺激期。当没有}$$

需要回复 当需要响应时，使用群体向量方法对响应活动进行解码，并

如果模型未能满足任何这些标准，则试验响应不正确。

模型训练

再次遵循参考文献。如图 18 所示，模型参数根据模型运动响应之间计算的掩蔽均方误差损失 (mMSE) 以监督方式更新， \hat{y}

$$y = \hat{y}，目标，$$

$y = y$ ，对于每次试验。

$$L = \text{mMSE}(y, \hat{y}) = \text{掩码} \times ((y - \hat{y})^2)$$

这里，乘号表示逐元素乘法。面具权衡不同试验时期的重要性。在准备阶段和刺激阶段，掩模权重设置为 1；在响应时期的前五个时间步长期间，掩码值设置为 0；在响应时期的剩余时间内，掩码权重设置为 5。固定的掩码值是所有时间步骤中其他值的两倍。

对于所有模型，我们在任务集训练期间更新 $\theta = \{\text{sensorimotor-RNN, Linear}\}$ 。对于有指导的模型，我们在正常训练的过程中额外更新了Linear。我们使用标准 PyTorch 机器和 Adam 优化器来训练模型。一个 epoch 由 2,400 个小批量组成，每个小批量由 64 次试验组成。对于所有模型，我们使用与参考文献中相同的初始学习率。18、 $\text{lr} = 0.001$ 。我们发现，在训练的后 期阶段，模型性能会根据训练期间呈现的最新任务而波动，因此我们将每个时期的学习率衰减 $\gamma = 0.95$ ，这使得性能能够平滑收敛。以下参考文献。18、模型训练直到所有任务的性能达到 95% 的阈值（并且训练至少 35 个 epoch）。我们发现，对于比较任务的多感官版本，GPTNET 的训练往往会逐渐低于性能阈值。这对于各种训练超参数和学习率调度机制都是成立的。因此，我们将GPTNET的性能阈值放宽至85%。对于每种模型类型，我们训练从五个不同的随机初始化开始的五个模型。在适用的情况下，结果是这些初始化的平均值。

语言模型微调。在微调模型时，我们允许感觉运动训练期间经历的运动损失的梯度来微调 Transformer 语言模型最后一层的权重。在正常训练期间，我们在训练 30 个 epoch 后检查指导模型的副本。然后，我们将最后三个变压器层添加到可训练参数集中，并将学习率重置为 $\text{lr} = 1 \times 10$ （对于 $\theta = \{\text{sensorimotor-RNN, 线性}\}$ ）和 $\text{lr} = 3 \times 10$ （对于 $\theta = \{\text{线性, 变压器-}3, -2, -1\}$ 其中，transformer-3,-2,-1 表示相关 Transformer 架构的最后三层的参数。我们利用这些降低的学习率来避免完全擦除预先存在的语言知识。与 RNN 参数类似，我们发现上述学习率避免了感觉运动知识的灾难性遗忘，同时也允许 RNN 适应所有模型中更新的语言嵌入。自回归模型对此过程更加敏感，通常在微调开始时崩溃。因此，对于 GPTNETXL 和 GPTNET，我们使用 $\text{lr} = 5 \times 10$ ，这导致了稳健的学习。模型进行训练，直到在训练任务中达到 95% 的阈值性能或 GPTNET 的正确率达到 85%。

保持测试

在保留测试期间，我们展示了包含 100 批已保留在训练之外的任务之一的模型。对于指导模型，在此阶段允许更新的唯一权重是 $\theta = \{\text{sensorimotor-RNN, Linear, Linear}\}$ 。SIMPLENET 和 STRUCTURENET 的所有权重都可以在这种情况下进行训练。在这个

在保留设置中，我们发现，在一些性能较差的模型的更困难的任务中，我们在训练期间使用的标准超参数导致新任务的学习曲线不稳定。为了稳定性能并从而在模型之间进行公平的比较，我们将批量大小增加到 256。然后，我们从 0.001 的标准学习率开始，并以 0.0005 的增量减少该学习率，直到所有模型都显示出稳健的学习曲线。这导致学习率为 8×10^{-5} 。补充信息第 4 节中显示的所有其他结果均遵循此程序。

CCGP计算

为了计算 CCGP，我们在一对任务上训练了一个线性解码器，然后在具有类似关系的替代任务对上测试该解码器。我们将任务分为八个二分法：“进行”与“反”、“标准”与“RT”、“最弱”与“最强”、“最长”与“最短”、“第一次刺激”与“第二次刺激”，“刺激匹配”与“类别匹配”、“匹配”与“不匹配”以及“Mod1”与“Mod2”。作为示例，“Go”与“Anti”二分法包括（“Go”、“AntiGo”）、（“GoMod1”、“AntiGoMod1”）、（“GoMod2”、“AntiGoMod2”）、（“RTGo”、“AntiRTGo”）、('RTGoMod1'、'AntiRTGoMod1') 和 ('RTGoMod2'、'AntiRTGoMod2') 任务对。对于“RNN”任务表示，我们提取了 250 个示例试验的刺激开始时的活动。对于语言表示，我们将相关任务的指令集输入到我们的语言模型中，并直接分析“嵌入”层中的活动或对每个转换器层中的序列平均活动进行分析。对于非语言模型，我们简单地分析规则向量的空间。解码器的训练和测试条件由任务集中确定的二分法确定（补充说明 1）。为了训练和测试解码器，我们使用了 sklearn.svm.LinearSVC Python 包。给定任务的 CCGP 分数是在所有二分法中获得的平均解码分数，其中所讨论的任务是训练集或测试集的一部分。对于正文中报告的模型分数，我们仅计算相关任务未经过训练的模型的 CCGP 分数。在补充图 9 中，我们报告了已针对所有任务训练模型的任务的分数，以及已将指令切换为保留任务的模型的分数。

对于图 3e，我们计算了保留任务的表现与每个任务的 CCGP 分数之间的 Pearson r 相关系数，以及针对这些指标不相关且正态分布的零假设的 P 值测试（使用 scipy.stats.pearsonr 函数）。图 3f 中 RNN 和嵌入层的 CCGP 分数的完整统计测试可以在补充图 9 中找到。请注意，Transformer 语言模型在 Sensorimotor-RNN 的随机初始化中使用相同的一组预训练权重，因此对于语言模型层，图 3f 的图显示了这些语言模型的绝对分数。

条件从句/演绎任务分析

我们首先将任务集分为两组（如下所列）：包含条件子句和简单演绎推理组件的任务（30 个任务）以及指令包含简单命令的任务（20 个任务）。我们计算了每个模型随机初始化时每个组的泛化性能平均值的性能差异（图 2f）。我们将这些差异与零分布进行比较，该零分布是通过将任务集执行一组 50 次随机洗牌分为 30 和 20 个任务的组，并以相同的方式计算差异，再次使用两侧不等方差 t 检验。由于 STRUCUTRENET 是一种非语言模型，因此我们将 STRUCUTRENET 的性能与我们的指令模型进行了比较，以区分使用演绎推理组件执行任务与使用更复杂的条件子句结构处理指令的效果。所有统计测试的结果均报告在补充图 6）中。

简单的命令式任务包

括：‘Go’、‘AntiGo’、‘RTGo’、‘AntiRTGo’、‘GoMod1’、‘Gc

‘AntiRTGoMod2’、‘RTGoMod2’、‘AntiRTGoMod2’、‘DM’、‘Gc
条件子句/演绎任务包
括：'ConDM'、'ConAntiDM'、'Dur1'、'Dur2'、'MultiDur1'、'MultiDur2'、'A
, 'Dur1Mod2', 'Dur2Mod1', 'Dur2Mod2',
'COMP1'、'COMP2'、'MultiCOMP1'、'MultiCOMP2'、'AntiCOMP1'、'AntiC
'、'DNMS'、'DMC' 和 'DMNC'。

语言产生训练

自监督语言生成网络训练。我们的语言生成框架受到使用 RNN 的经典序列到序列建模的启发。我们的 Production-RNN 是一个具有 256 个隐藏单元的 GRU，使用 ReLU 非线性。在序列中的每个步骤，一组解码器权重 Linear 尝试从循环单元的隐藏状态解码下一个标记 w。Production-RNN 的隐藏状态是通过连接 SBERTNET (L) 的时间平均和最大感觉运动活动并将其传递给权重 Linear 来初始化的。用于驱动初始化感觉运动活动的语言指令依次用作 Production-RNN 输出的目标标记集。Production-RNN 的第一个输入始终是一个特殊的句子开始标记，并且解码器运行直到解码了句子结束标记或直到

输入的长度达到 50 个令牌。假设 $w_i \in \mathcal{W}$ 是指令集中的第 i 个令牌，其中 \mathcal{W} 是指令集，X 是任务 i 试验的感官输入。为了简洁起见，我们将语言模型嵌入指令的过程表示为 Embed()（参见“预训练变压器”）。 τ 位置 w 处的解码标记由下式给出

$$h = \text{SensorimotorRNN}(X, \text{嵌入}(w_1 \cdots w_n)) \quad h \in \mathbb{R}$$
$$\text{sm_out} = (\text{平均值}(h), \text{最大值}_{\tau}(h)) \quad \text{sm_out} \in \mathbb{R}$$
$$h_0 = \text{relu}(\text{线性}(\text{sm_out})) \quad h_0 \in \mathbb{R}$$
$$h_0 = \text{辍学}(h_0) \quad h_0 \in \mathbb{R}$$
$$h_{\tau} = \text{ProductionRNN}(w_1 \cdots w_n; h_0), \quad h_{\tau} \in \mathbb{R}$$
$$p_w = \text{softmax}(\text{线性}(h_{\tau, k})) \quad \hat{p}_w \in \mathbb{R}^{|\text{词汇}|};$$
$$w = \text{argmax}(p_w)$$

使用 p 和提供给感觉运动 RNN 作为输入的指令标记 w 之间的交叉熵损失来训练模型参数 $\theta = \{\text{Linear}, \text{Linear}, \text{ProductionRNN}\}$ 。我们训练 80 个 epoch，共 2,400 个批次，每批次 64 次试验，任务类型随机交错。我们发现使用 0.001 的初始学习率有时会导致模型在训练的早期阶段发散，因此我们选择了 1×10^{-4} 的学习率，这导致了稳定的早期训练。为了缓解在感觉运动训练中检测到的类似振荡问题，我们还将每个时期的学习率降低了 $\gamma = 0.99$ 。此外，使用 dropout 率为 0.05 的 dropout 层提高了性能。我们还使用了教师强制课程，对于一定比例的训练批次，我们在每个时间步输入真实指令标记 w，而不是模型解码的单词 \hat{w} 。

w 。在每个纪元，教师强迫率 = $0.5 \times \frac{80 - \text{纪元}}{80}$ 。

使用运动反馈获取嵌入层活动。对于一个

任务 i，我们寻求优化一组嵌入活动向量 $E \in \mathbb{R}$ ，以便当它们作为任务识别信息输入时，模型将执行相关任务。至关重要的是，我们冻结所有模型权重 $\theta = \{\text{sensorimotor-RNN}, \text{Linear}, \text{Linear embedding}\}$ 并且仅根据电机上的标准监督损失更新 E

输出。为了概念清晰，GRU 对先前隐藏状态 h 的依赖已隐含在以下方程中。

$$\hat{y} = \sigma(\text{线性}(\text{SensorimotorRNN}(X, E)))$$
$$L = \text{mMSE}(\hat{y}, y)$$

我们再次使用 Adam 优化器为每个任务优化了一组 25 个嵌入向量。这里，优化空间具有许多与相关任务的嵌入相对应的次优局部最小值。因此，我们使用了 $\text{lr} = 0.05$ 的高初始学习率，每个时期的衰减量为 $\gamma = 0.8$ 。与较低的学习率相比，这会带来更稳健的学习。一个 epoch 持续 800 个批次，批次长度为 64，我们至少训练 1 个 epoch，或者直到我们在“DMC”和“DNMC”任务上达到 90% 或 85% 的阈值性能。

制作任务指令。为了生成任务指令，我们只需在感觉运动 RNN 的输入中使用集合 E 作为任务识别信息，并使用 Production-RNN 根据 E 驱动的感觉运动活动输出指令。对于每个任务，我们使用一组嵌入向量来为每个任务生成 50 条指令。我们对感觉运动 RNN 的 5 个初始化中的每一个重复此过程，从而产生 5 个不同的语言生成网络和 5 个不同的学习嵌入向量集。每项任务的报告结果是这 5 个网络的平均值。对于混淆矩阵（图 5d），我们报告了给定任务或新指令的训练指令集中解码指令的平均百分比。每个网络初始化的合作伙伴模型性能（图 5e）是通过测试 4 个可能的合作伙伴网络中的每一个并对这些结果求平均值来计算的。

样本量/随机化

没有使用统计方法来预先确定样本大小，但遵循参考文献。18 我们对每个测试的语言模型使用了五种不同的随机权重初始化。权重的随机化是在 Python 和 PyTorch 软件包中自动进行的。鉴于权重的这种自动随机化，我们在研究中没有使用任何盲法程序。没有数据被排除在分析之外。

软件

所有模拟和数据分析均在 Python 3.7.11 中进行。PyTorch 1.10 用于实现和训练模型（包括 Adam 优化器实现）。Transformers 4.16.2 用于实现语言模型，语言模型的所有预训练权重均取自 Huggingface 存储库 (<https://huggingface.co/docs/transformers/>)。我们还使用 scikit-learn 0.24.1 和 scipy 1.7.3 进行分析。

报告摘要

有关研究设计的更多信息，请参阅本文链接的《自然投资组合报告摘要》。

数据可用性

本研究中使用的语言转换器的所有权重均取自 Huggingface 存储库 (<https://huggingface.co/docs/transformers/>) 上提供的预训练模型

模拟心理物理任务的训练数据是使用 <https://github.com/ReidarRiveland/Instruct-RNN/> 上提供的代码生成的。所有结果的全套训练模型权重可根据要求提供。

代码可用性

用于训练模型和分析结果的所有代码都可以在 <https://github.com/ReidarRiveland/Instruct-RNN/> 找到。

参考

49. Chung, J., Gulcehre, C., Cho, K. 和 Bengio, Y. 门控循环神经网络对序列建模的实证评估。预印本位于 <https://arxiv.org/abs/1412.3555> (2014)。

50. 雷德福, A.等人。更好的语言模型及其含义。<https://openai.com/blog/better-language-models/> (2019)。

51. 布罗姆利, J.等人。使用“连体”时间进行签名验证延迟神经网络。国际。J.模式识别。阿蒂夫。英特尔。7、669-688 (1993)。

52. 沃尔夫, T.等人。变形金刚：最先进的自然语言加工。在过程中。2020 年自然语言处理经验方法会议：系统演示 (Liu, Q. 和 Schlangen, D. 编) 38–45 (计算语言学协会, 2020 年)。

53. Sutskever, I., Vinyals, O. & Le., Q. V. 使用神经网络进行序列到序列学习。在过程中。第 27 届神经信息处理系统国际会议 3104–3112 (麻省理工学院出版社, 2014 年)。

致谢

我们感谢 N. Rungratsameetaweemana、T. Aquino 和 V. Borghesani 以及 N. Patel 和 P. Tano 在本项目期间进行的有益讨论。我们还感谢日内瓦大学为这项研究提供的资助。

作者贡献

A.P. 和 R.R. 构思了这个项目。R.R. 编写了模型模拟的代码并对模型表示进行了分析。A.P. 和 R.R. 撰写并修改了这篇论文。

资金

日内瓦大学提供开放获取资金。

利益竞争

作者声明没有竞争利益。

附加信息

补充信息 在线版本包含补充材料，网址为 <https://doi.org/10.1038/s41593-024-01607-5>。

信函和材料请求应发送至 Reidar Riveland。

同行评审信息《自然神经科学》感谢布莱克·理查兹以及其他匿名审稿人对本工作同行评审的贡献。

重印和许可信息可在 www.nature.com/reprints 上获取。

报告摘要

Nature Portfolio 希望提高我们发表的作品的可重复性。该表格提供了报告一致性和透明度的结构。有关自然组合政策的更多信息，请参阅我们的编辑政策和编辑政策清单。

统计数据

对于所有统计分析，请确认图形图例、表格图例、正文或方法部分中存在以下项目。

不适用 已确认

- ☐
- ☒
- 每个实验组/条件的确切样本量 (n)，以离散数字和测量单位给出
- ☐
- ☒
- 关于测量是否取自不同样品或是否重复测量同一样品的声明
- ☐
- ☒
- 使用的统计测试以及它们是单侧还是双向
- ☐
- ☒
- 只有常见的测试才应该仅用名称来描述；在方法部分描述更复杂的技术。
- ☒
- ☐
- 所有测试协变量的描述
- ☒
- ☐
- 对任何假设或修正的描述，例如正态性检验和多重比较调整
- ☐
- ☒
- 统计参数的完整描述，包括集中趋势（例如平均值）或其他基本估计（例如回归系数）和变化（例如标准差）或相关的不确定性估计（例如置信区间）
- ☐
- ☒
- 对于原假设检验，检验统计量（例如 F、t、r）以及置信区间、效应大小、自由度和 P 值均注明。只要合适，请以精确值形式给出 P 值。
- ☒
- ☐
- 对于贝叶斯分析，有关先验选择和马尔可夫链蒙特卡罗设置的信息
- ☒
- ☐
- 对于分层和复杂的设计，确定适当的测试级别和完整的结果报告
- ☐
- ☒
- 效应大小的估计（例如 Cohen's d、Pearson's r），表明它们是如何计算的

我们的生物学家统计网络集包含有关上述许多观点的文章。

软件和代码

有关计算机代码可用性的政策信息

数据收集	数据是使用 Python 3.7.11 在模拟中生成的。PyTorch 1.10 用于实现和训练模型（包括 Adam 优化器实现）。Transformers 4.16.2 用于实现语言模型，语言模型的所有预训练权重均取自 Huggingface 存储库 (https://huggingface.co/docs/transformers/)。https://github.com/ReidarRiveland/Instruct-RNN 提供训练、分析和绘图脚本。
数据分析	在 Python 3.7.11 (https://github.com/ReidarRiveland/Instruct-RNN) 中执行数据分析。我们还使用 scikit-learn 0.24.1 和 scipy 1.7.3 进行分析。

对于使用对研究至关重要但尚未在已发表文献中描述的自定义算法或软件的手稿，必须向编辑和审稿人提供软件。我们强烈鼓励将代码存放在社区存储库（例如 GitHub）中。请参阅 Nature Portfolio 提交代码和软件指南以获取更多信息。

有关数据可用性的政策信息 所有稿件必须包含数据可用性声明。本声明应在适用的情况下提供以下信息： - 公开可用数据集的登录代码、唯一标识符或 Web 链接
- 对数据可用性的任何限制的描述 - 对于临床数据集或第三方数据，请确保声明遵守我们的政策

本研究中使用的语言转换器的所有权重均取自 Huggingface 存储库 (<https://huggingface.co/docs/transformers/>) 上提供的预训练模型。模拟心理物理任务的训练数据是使用 <https://github.com/ReidarRiveland/Instruct-RNN> 上提供的代码生成的。手稿中包含的所有结果的全套训练模型权重可根据要求提供。

涉及人类参与者、他们的数据或生物材料的研究

有关人类参与者或人类数据研究的政策信息。另请参阅有关性别、性别（身份/表现）以及性取向和种族、民族和种族主义的政策信息。

关于性和性别的报告	N/A
报道种族、民族或其他社会相关群体	N/A
人口特征	N/A
招聘	N/A
道德监督	N/A

请注意，手稿中还必须提供有关研究方案批准的完整信息。

特定领域的报告

请选择下面最适合您的研究的一项。如果您不确定，请在做出选择之前阅读相应的部分。

☒ 生命科学 ☐ 行为与社会科学 ☐ 生态、进化和环境科学

有关包含所有部分的文档的参考副本，请参阅 [nature.com/documents/nr-reporting-summary-flat.pdf](https://www.nature.com/documents/nr-reporting-summary-flat.pdf)

生命科学研究设计

所有研究都必须披露这些观点，即使披露是否定的。

样本量	在适用的情况下，我们分析了通过五种不同的随机参数初始化训练的模型的结果。这种选择受到计算可行性的限制，但与其他计算神经科学研究相同（例如 Yang 等人，Nature Neuroscience，2019）。
数据排除	None
复制	在整个实验中，通过五种不同的初始化测试的模型在性能和模型表示分析方面都产生了一致的结果。代码中提供了精确的随机种子，以便可以准确地再现训练和测试。
随机化	同样，在每个模型的情况下，我们使用了五种不同的随机初始化。在每种条件下都使用同一组五个随机种子。
致盲	我们在分析中没有使用显式盲法，也没有在不考虑条件的情况下进行模拟。这是因为所有实验都是用计算机代码自动执行的。所有不属于预训练语言模型的模型权重都是随机初始化的，在这种情况下，作者无法控制这些初始化。

特定材料、系统和方法的报告

我们需要作者提供有关许多研究中使用的某些类型的材料、实验系统和方法的信息。在此，请指出列出的每种材料、系统或方法是否与您的研究相关。如果您不确定列表项是否适用于您的研究，请在选择响应之前阅读相应的部分。

材料与实验系统

方法

不适用 参与研究

- ☒
- ☐
- 抗体

☒☐☒☐☒☐☒☐☒☐☒☐

不适用 参与研究

- ☒
- ☐
- ChIP测序

☒☐☒☐

植物

种子库存	<div>N/A</div>
新植物基因型	<div>N/A</div>
验证	<div>N/A</div>