AugARC: Augmented Abstraction and Reasoning Benchmark for Large Language Models

Kiril Bikov[1], **Mikel Bober-Irizar**[1], **Soumya Banerjee**[1,*]

[1]University of Cambridge
Cambridge
United Kingdom
[*]Corresponding author: sb2333@cam.ac.uk

Abstract

The Abstraction and Reasoning Corpus (ARC) benchmarks broad generalization, and poses a significant challenge to existing machine learning models. In this work, we introduce augmented ARC datasets and a new benchmark (AugARC) for large-language models (LLMs), which measures abstraction and reasoning. We evaluate the accuracy of base LLMs on AugARC and show a consistent improvement in performance compared to the normal ARC benchmark. Using augmented ARC data, we fine-tune LLMs and observe a significant gain in ARC accuracy after training. Due to the limited size of the ARC training dataset (400 tasks), previous studies have not attempted to train LLMs on ARC. Our augmentation of ARC allows us to overcome this limitation. Using a reflection approach, we combine LLMs and a previous domain specific language (DSL) solver. Our work introduces an augmented version of ARC - AugARC, and motivates further research into enhancing data quality for better reasoning in AI systems.

Introduction

Despite significant progress in machine learning, today's AI systems still lack human-level abstract reasoning Korteling et al. (2021); Boden et al. (2017); Shneiderman (2020). To address the gap between human intelligence and AI models, François Chollet created the Abstraction and Reasoning Corpus (ARC) Chollet (2019). ARC consists of 1000 visual tasks, that capture essential aspects of abstraction and analogy. The ARC tasks are split into 400 for training, 400 for evaluation and hidden 200 tasks for testing. A Program Synthesis approach from 2020 solved 40% of the complete evaluation set Icecuber (2023), and a voting ensemble from 2024 solved 40.25% of the tasks in the evaluation set Bober-Irizar and Banerjee (2024).

We aim to fully explore the abilities of base large-language models (LLMs) on ARC and how those can be combined in multi-model systems. We introduce a new augmented ARC (AugARC) benchmark tailored towards LLMs, which shows consistently improved performance across all tested LLMs. We show the benefit of fine-tuning LLMs on augmented ARC data. Finally, we built a reflection system based on multiple solvers Bober-Irizar and Banerjee (2024).

AugARC: Augmented ARC for LLMs

The ARC training data can be utilized for fine-tuning LLMs and improving their performance on the evaluation and test sets. One potential issue with this approach is the size of the training set - it contains only 400 samples. Since LLMs have billions of parameters, they usually cannot be effectively trained on smaller datasets and instead require more samples. Therefore, due to its small size, the ARC training dataset limits the ability to fine-tune LLMs for improved broad generalization and reasoning.

Augmented Training Data

To overcome the limited number of ARC training tasks, we propose an augmentation procedure that can significantly extend the training dataset. Our approach expands the ARC training set by applying the following transformations:

- **Rotation:** clockwise rotation of each ARC grid for a given task by 90° or 270°.
- **Flipping:** flips each ARC grid of a task horizontally (along the y-axis) and vertically (along the x-axis).
- **Permutations:** rearranges the sequence of demonstration input-output pairs before the test input grid. We set a threshold for the maximum number of permutations per task to produce datasets of various sizes.

Depending on the transformations applied and the maximum number of permutations applied, the augmented ARC training datasets vary from 2000 up to over 18 million tasks. The AugARC data is available from the following repository: https://github.com/kiril-bikov/AugARC

3-Shot AugARC Benchmark

A key reason for the relatively scarce ARC research on LLMs is the lack of a textual version of the benchmark. The only benchmark suitable for LLMs that resembles Chollet's visual ARC Chollet (2019) is the AI2 Reasoning Challenge Clark et al. (2018); Pătras et al. (2022). AI2 is a multi-choice question answering benchmark that focuses on assessing reasoning. Although AI2 is a more popular and well-established reasoning benchmark for LLMs compared

Dataset Size	Max Permutations
2 000 tasks	-
4 000 tasks	2
5 715 tasks	3
7 430 tasks	4
9 145 tasks	5
18 668 610 tasks	All

Table 1: Size of the augmented ARC training datasets according to the maximum number of permutations. All datasets include 90° and 270° rotations, horizontal and vertical flipping. The augmented datasets range from 2000 to 18 million tasks.

to Chollet's ARC Chollet (2019), the latter is more effective at evaluating broad generalization abilities due to its handcrafted abstract logic.

Identifying that the lack of a textual ARC benchmark is a significant barrier for evaluating LLMs, we create the AugARC benchmark. The AugARC benchmark provides an easy and unified way to evaluate LLMs on 3-shot accuracy on reasoning tasks. In AugARC, each ARC task starts with a textual description explaining the format of the problem. Each ARC grid is represented as a 2D matrix of numbers.

AugARC Input to LLMs The first prediction is based on a normal ARC task, whereas the second and the third ones are 90° and 270° clockwise rotated versions of the same task. The AugARC benchmark is tailored towards LLMs' architecture, as those models process inputs in an auto-regressive, sequential manner. By rotating the ARC tasks, LLMs are presented with a different sequence of numbers (2D matrices) which contain the same abstract logic.

Reproducing ARC Solutions from AugARC Outputs Although the second and third shot in AugARC are based on rotated ARC tasks, the output of the LLMs can easily be transformed back to a solution to the original ARC problem. Once an output is generated by the LLM, it is simply rotated back in an anticlockwise direction. In this way, AugARC only changes the input representation of the ARC problems, but the outputs by the models are then rotated to valid ARC solutions. This process ensures that the results with the proposed AugARC approach are directly comparable with previous ARC attempts.

Method

Fine-tuning LLMs on augmented ARC tasks

Although LLMs have shown impressive capabilities, they can sometimes hallucinate. One potential way to reduce such hallucinations and improve performance on abstract logical tasks is to fine-tune LLMs. Due to the limited size of the ARC training dataset (400 tasks), previous studies have not attempted to train LLMs on ARC. Our augmentation of ARC allows us to overcome this limitation and have sufficient ARC data to fine-tune LLMs.

For efficient training of LLMs, we use Quantized Low-Rank Adaptation (QLoRA) with 4-bit NormalFloat (NF4) quantization (Dettmers et al. 2024). Low-Rank Adaptation constrains the update of a pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$ with a low-rank decomposition $W_0 + \Delta W = W_0 + BA$, where $B \in \mathbb{R}^{d\times r}$, $A \in \mathbb{R}^{r\times k}$, and the rank $r \ll \min(d, k)$ (Hu et al. 2021). During training, W_0 is frozen and does not receive gradient updates, while A and B contain trainable parameters. Both W_0 and $\Delta W = BA$ are multiplied with the same input, and their respective output vectors are summed coordinate-wise (Hu et al. 2021).

Using QLoRA, we fine-tune LLMs on an augmented ARC training dataset consisting of 2000 tasks 1. Due to a significant increase in computational complexity, we avoid fine-tuning the models on some of the bigger augmented ARC training sets from Table 1. For the same reason, we only train LLMs with parameters ranging from 7 to 13 billion.

Reflection System for ARC

A previous promising approach which solves 40.25% of the ARC evaluation tasks combines solutions from different ARC solvers Bober-Irizar and Banerjee (2024). The voting ensemble lacks any "intelligent" analysis of the potential solutions and instead uses a weighting algorithm Bober-Irizar and Banerjee (2024). Therefore, we propose a Reflection System for solving ARC.

The Reflection System relies on models that could have various architectures - LLMs and Program Synthesis solvers. It executes in two main stages, as visualised in Figure 1. In the first stage, each model makes a prediction on the given ARC task. The models work independently and cannot access the outputs of other models. Once the model produces ARC predictions, those are passed in the second stage to the reflection model Lee et al. (2024); Renze and Guven (2024). Conditioned on the given ARC task, the reflection model chooses the prediction from the models that is most likely to be correct.

Experiments

We perform all experiments on the ARC evaluation set which consists of 400 tasks. By design, the ARC evaluation set is significantly more challenging than the training set Chollet (2019). The creator of ARC, François Chollet, emphasised that the performance of intelligent systems should be measured by the fraction of solved tasks on the evaluation set Chollet (2019). Therefore, we perform our experiments on the evaluation set and use 3 shots per task, as set out in the ARC design Chollet (2019).

To present fully reproducible results, all experiments are executed on the complete evaluation set. Some previous solvers have been evaluated on a subset of the ARC evaluation data, making it difficult to understand the true performance of the solver Xu, Khalil, and Sanner (2023); Lei, Lipovetzky, and Ehinger (2024). Our testing approach ensures that future studies could easily use our results for direct comparison with new ARC solvers.

Performance on base ARC and AugARC

We start our experiments with LLMs on the base ARC benchmark, shown in Table 2. The ARC accuracy across 7-

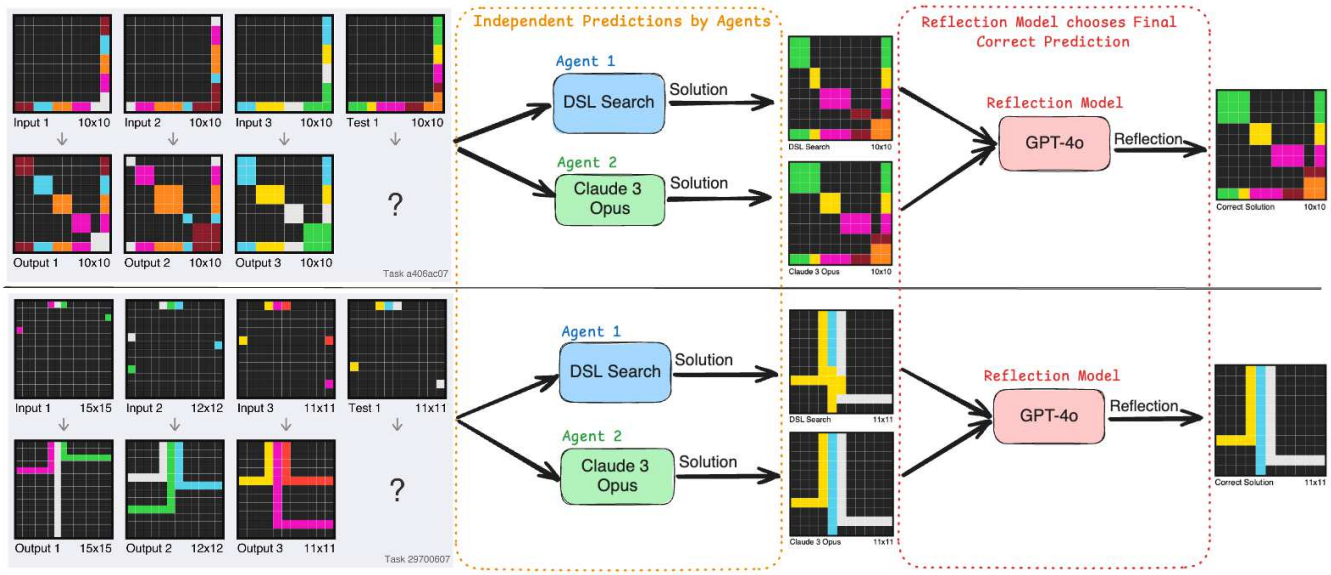

Figure 1: Reflection Systems - execution on two ARC evaluation tasks. Initially, multiple models(LLMs and DSL Search) make independent predictions on the task. Then, the task and the prediction are presented to the reflection model, which chooses the correct final prediction. In the example, model 1 is based on program synthesis (DSL Search) and model 2 is an LLM (Claude 3 Opus). The reflection model is an LLM (GPT-4o). Both task flows are actual demonstration of how our Reflection System configurations perform on ARC evaluation tasks. In both cases, the Reflection System produces correct final solution.

13 billion models ranges from 5 to 9 solved tasks. Bigger LLMs solve slightly more ARC tasks, from 7 to 20, with Gemini Pro achieving the highest accuracy (20).

Model	ARC	AugARC	Increase
Llama-2 7B	5/400	7/400	29%
Mistral 7B	9/400	15/400	67%
Llama-2 13B	5/400	8/400	100%
Llama-2 70B	7/400	14/400	100%
Mixtral 8x7B	9/400	18/400	**125%**
Gemini Pro	20/400	**33/400**	65%

Table 2: Performance of LLMs on ARC and AugARC (on the evaluation set). There is a consistent increase of the accuracy of LLMs when using the AugARC inputs compared to using the base ARC ones (29-125%).

Using the same LLMs, we evaluate the performance on AugARC. For all LLMs, there is a clear accuracy improvement on AugARC compared to the base ARC. The increase varies from 29% for Llama-2 7B up to 125% for Mixtral 8x7B, with the majority of models achieving at least 60%.

The significant improvement in all LLMs on AugARC compared to ARC suggests that changing the grid structure of the tasks for the second and third shot leads to enhanced accuracy. LLMs process the ARC tasks sequentially, and thus are directly influenced by the exact order of the grids. Based on the results, we conclude that the proposed AugARC benchmark is well suited for testing LLMs.

Since AugARC results are directly comparable to ARC, we proceed to use AugARC for the remainder of our experiments.

ARC accuracy across LLMs

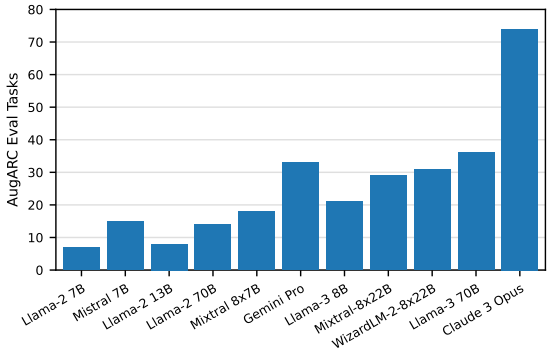

Figure 2: ARC evaluation tasks solved by LLMs. Claude 3 Opus solves the most ARC tasks (74).

The ARC accuracy of LLMs ranges between 7 to 74 solved tasks, as visualised in Figure 2. The best performance by a smaller 7B model is achieved by Llama-3 8B (21). Some bigger open-source LLMs can solve more than 30 ARC tasks, with Llama-3 70B achieving 36. The highest number of solved ARC tasks, 74, is by Claude 3 Opus.

The ARC results demonstrated some variability in performance across LLMs. Bigger models appear to be more accurate on ARC compared to smaller ones. Most LLMs achieve an accuracy in the range of 10-35 tasks, with the only excep-

tion being Claude 3 Opus with 74 out of 400 ARC tasks.

Performance of Fine-tuned LLMs on ARC

To observe whether we can reduce the performance gap between smaller and bigger LLMs on ARC, we fine-tune the 7 and 13B models. All training flows are executed on a single Nvidia A100 80GB GPU.

The results in Table 3 show that the fine-tuned LLMs solve between 18 and 34 ARC tasks. Training benefited all the models substantially - the small fine-tuned Llama-2 7B and 13B models achieved a performance on par with the base versions of significantly bigger models such as Llama-2 70B. After fine-tuning, Mistral 7B outperforms the standard Mixtral 8x7B by 5 correct tasks. The highest result of 34 correct solutions after fine-tuning by Llama-3 8B is impressive, as it outperforms Gemini Pro.

Model	Base	Fine-tuned	Increase
Llama-2 7B	7/400	21/400	**200%**
Mistral 7B	15/400	23/400	53%
Llama-2 13B	8/400	18/400	125%
Llama-3 8B	21/400	**34/400**	62%

Table 3: ARC evaluation results of base and fine-tuned LLMs. The increase column shows the improvement in accuracy from a base LLM compared to its fine-tuned version. All LLMs consistently show improved ARC performance after fine-tuning, ranging from 62% to 200%.

The results in Table 3 demonstrate a significant increase in ARC performance across all fine-tuned LLMs compared to their base versions. The improvement in accuracy after training varies between 53% in Mistral 7B up to 200% in Llama-2 7B. While Llama-2 7B and 13B both achieve more than 100% improvement - 125% and 200% respectively, Mistral 7B and Llama-3 8B improved in the range of 50% to 65%.

Based on our results, we conclude that training small LLMs on an AugARC dataset consistently improves their performance. Notably, fine-tuning smaller LLMs (7-13B parameters) is so effective that it can lead to better ARC performance than significantly bigger base LLMs.

Performance of the Reflection System

We experiment with Reflection System configurations based on two or three models and with different reflection models. We always include the program synthesis solver (DSL Search (Icecuber 2023)) as a solver in all of our reflection system experiments. We also always include the LLM with highest ARC accuracy as a model (Claude 3 Opus). We experiment with base and fine-tuned LLMs for the reflection models (and a potential third model) to find the Reflection System configurations which achieve the highest ARC accuracy.

Table 4 shows that the ARC performance by different reflection system configurations varies between 133 and 166 solved evaluation tasks. In a 2-model setting, with DSL Search and Claude 3 Opus, Llama-3 70B struggles as a reflection model, solving only 133 tasks. GPT-4-turbo and GPT-4o perform significantly better as reflection models, solving 165 and 166 ARC tasks. When adding a fine-tuned Llama-3 8B as a third model, the reflection system solves 163 ARC tasks.

model 1	model 2	model 3	Reflection Model	ARC Correct
DSL Search	Claude 3 Opus	-	Llama-3 70B	133/400
DSL Search	Claude 3 Opus	-	GPT-4-turbo	165/400
DSL Search	Claude 3 Opus	-	GPT-4o	**166/400**
DSL Search	Claude 3 Opus	Fine-Tuned Llama-3 8B	Claude 3.5 Sonnet	163/400

Table 4: Correctly solved ARC evaluation tasks in a 3-shot setting by different Reflection System configurations. The best 2-model performance is with DSL Search and Claude 3 Opus as models and GPT-4o as a reflection model (166). The highest 3-model accuracy adds a fine-tuned Llama-3 8B model (163).

Our best 2-model and 3-model reflection system configurations both outperform the best single LLM, Claude 3 Opus (74), and the best program synthesis approach, which has been tested on the complete ARC evaluation set - the DSL Search (160). Based on the results, we argue that our reflection system is an effective approach for combining LLMs and Program Synthesis solvers into systems for enhanced ARC performance.

Limitations

Since we did not have access to the data used for pre-training the LLMs, we cannot exclude the possibility that some models might have been pre-trained either on ARC tasks or on other very similar abstract problems. It can be argued that the significant improvement after fine-tuning demonstrates that most of the tested LLMs have not been pre-trained on ARC. Nevertheless, the substantially higher ARC results by Claude 3 Opus compared to all other LLMs raise some concerns that this model might have been pre-trained on ARC.

Conclusion

We propose an augmentation procedure for ARC that rotates the tasks 90- and 270-degree clockwise. With the augmented ARC data, we fine-tune LLMs and produce improved results on the reasoning tasks. We also introduce a new AugARC benchmark, which leads to better results for LLMs compared to the normal ARC. Finally, we create a new Reflection System for solving ARC. In future work, AugARC can be extended using more complex data augmentation techniques such as geometric transformations instead of rotations and flipping. Additionally, future studies can attempt to fine-tune LLMs on larger augmented datasets.

References

Bober-Irizar, M.; and Banerjee, S. 2024. Neural networks for abstraction and reasoning: Towards broad generalization in machines. *arXiv preprint arXiv:2402.03507*.

Boden, M.; Bryson, J.; Caldwell, D.; Dautenhahn, K.; Edwards, L.; Kember, S.; Newman, P.; Parry, V.; Pegman, G.; Rodden, T.; et al. 2017. Principles of robotics: regulating robots in the real world. *Connection Science*, 29(2): 124–129.

Chollet, F. 2019. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*.

Clark, P.; Cowhey, I.; Etzioni, O.; Khot, T.; Sabharwal, A.; Schoenick, C.; and Tafjord, O. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Dettmers, T.; Pagnoni, A.; Holtzman, A.; and Zettlemoyer, L. 2024. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Icecuber. 2023. ARC: 1st Place Solution. Available at: https://www.kaggle.com/code/icecuber/arc-1st-place-solution/execution. Accessed: 01 August 2024.

Korteling, J. H.; van de Boer-Visschedijk, G. C.; Blankendaal, R. A.; Boonekamp, R. C.; and Eikelboom, A. R. 2021. Human-versus artificial intelligence. *Frontiers in artificial intelligence*, 4: 622364.

Lee, K.; Hwang, D.; Park, S.; Jang, Y.; and Lee, M. 2024. Reinforcement Learning from Reflective Feedback (RLRF): Aligning and Improving LLMs via Fine-Grained Self-Reflection. *arXiv preprint arXiv:2403.14238*.

Lei, C.; Lipovetzky, N.; and Ehinger, K. A. 2024. Generalized planning for the abstraction and reasoning corpus. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 20168–20175.

Pătras, C.-B.; Pîrtoacă, G.-S.; Rebedea, T.; Ruşeţi, Ş.; et al. 2022. More with Less: ZeroQA and Relevant Subset Selection for AI2 Reasoning Challenge. *Procedia Computer Science*, 207: 2757–2766.

Renze, M.; and Guven, E. 2024. Self-Reflection in LLM Agents: Effects on Problem-Solving Performance. *arXiv preprint arXiv:2405.06682*.

Shneiderman, B. 2020. Human-centered artificial intelligence: Three fresh ideas. *AIS Transactions on Human-Computer Interaction*, 12(3): 109–124.

Xu, Y.; Khalil, E. B.; and Sanner, S. 2023. Graphs, constraints, and search for the abstraction and reasoning corpus. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 4115–4122.

INTERFACE

rsif.royalsocietypublishing.org

Research

Cite this article: Banerjee S, Perelson AS, Moses M. 2017 Modelling the effects of phylogeny and body size on within-host pathogen replication and immune response. *J. R. Soc. Interface* **14**: 20170479.
http://dx.doi.org/10.1098/rsif.2017.0479

Received: 1 July 2017
Accepted: 18 October 2017

Subject Category:
Life Sciences – Mathematics interface

Subject Areas:
biomathematics, computational biology, systems biology

Keywords:
modelling diseases, emerging diseases, West Nile Virus, mathematical modelling; zoonotic diseases, disease modelling, Flavivirus, hierarchical Bayesian models

Author for correspondence:
Soumya Banerjee
e-mail: soumya.banerjee@maths.ox.ac.uk

Electronic supplementary material is available online at https://dx.doi.org/10.6084/m9.figshare.c.3918655.

THE ROYAL SOCIETY PUBLISHING

Modelling the effects of phylogeny and body size on within-host pathogen replication and immune response

Soumya Banerjee[1], Alan S. Perelson[2,3] and Melanie Moses[3,4]

[1]Mathematical Institute, University of Oxford, Oxford, Oxfordshire, UK
[2]Los Alamos National Laboratory, Los Alamos, NM, USA
[3]Santa Fe Institute, Santa Fe, NM, USA
[4]Department of Computer Science, University of New Mexico, Albuquerque, NM, USA

SB, 0000-0001-7748-9885; ASP, 0000-0002-2455-0002; MM, 0000-0002-8848-9554

Understanding how quickly pathogens replicate and how quickly the immune system responds is important for predicting the epidemic spread of emerging pathogens. Host body size, through its correlation with metabolic rates, is theoretically predicted to impact pathogen replication rates and immune system response rates. Here, we use mathematical models of viral time courses from multiple species of birds infected by a generalist pathogen (West Nile Virus; WNV) to test more thoroughly how disease progression and immune response depend on mass and host phylogeny. We use hierarchical Bayesian models coupled with nonlinear dynamical models of disease dynamics to incorporate the hierarchical nature of host phylogeny. Our analysis suggests an important role for both host phylogeny and species mass in determining factors important for viral spread such as the basic reproductive number, WNV production rate, peak viraemia in blood and competency of a host to infect mosquitoes. Our model is based on a principled analysis and gives a quantitative prediction for key epidemiological determinants and how they vary with species mass and phylogeny. This leads to new hypotheses about the mechanisms that cause certain taxonomic groups to have higher viraemia. For example, our models suggest that higher viral burst sizes cause corvids to have higher levels of viraemia and that the cellular rate of virus production is lower in larger species. We derive a metric of competency of a host to infect disease vectors and thereby sustain the disease between hosts. This suggests that smaller passerine species are highly competent at spreading the disease compared with larger non-passerine species. Our models lend mechanistic insight into why some species (smaller passerine species) are pathogen reservoirs and some (larger non-passerine species) are potentially dead-end hosts for WNV. Our techniques give insights into the role of body mass and host phylogeny in the spread of WNV and potentially other zoonotic diseases. The major contribution of this work is a computational framework for infectious disease modelling at the within-host level that leverages data from multiple species. This is likely to be of interest to modellers of infectious diseases that jump species barriers and infect multiple species. Our method can be used to computationally determine the competency of a host to infect mosquitoes that will sustain WNV and other zoonotic diseases. We find that smaller passerine species are more competent in spreading the disease than larger non-passerine species. This suggests the role of host phylogeny as an important determinant of within-host pathogen replication. Ultimately, we view our work as an important step in linking within-host viral dynamics models to between-host models that determine spread of infectious disease between different hosts.

1. Introduction

Zoonotic diseases that jump the species barrier from animals to humans cause 2.5 billion cases of human illness and 2.7 million human deaths per year [1]. Many emerging diseases are zoonotic in origin and infect multiple host species [2]. Multi-host pathogens may have very different dynamics in different species [3].

Understanding how quickly pathogens replicate and how quickly the immune system responds is important for predicting the epidemic spread of emerging pathogens. While host–pathogen interactions have been studied qualitatively using mathematical models, it is not known how the parameters characterizing the immune response and pathogen replication rates change from species to species. Emerging zoonotic diseases originate from species that differ in body size, e.g. birds in avian influenza and cattle in bovine spongiform encephalopathy. Thus, it is important to understand how body size affects immune response and pathogenesis.

Body size can affect pathogenesis in two ways.

(1) Host metabolism constrains energy delivery to cells [4–8] and could influence rates of pathogen replication and immune response rates [9]. The metabolic rate of each cell is constrained by the rate at which nutrients and oxygen are supplied by the cardiovascular network. The rate at which this network supplies nutrients to each cell (R_{cell}) scales as the body mass (M) raised to an exponent of $-\frac{1}{4}$: $R_{cell} \propto M^{-1/4}$ such that individual cellular metabolic rates decrease as the body mass increases [4–8]. This relationship holds over an incredible diversity of body sizes, from 10^{-13} g (microbes) to 10^{8} g (whales). Many biological rates, such as heart rates and reproductive rates, also scale as $M^{-1/4}$, while many characteristic times, such as blood circulation times and lifetimes, scale as $M^{1/4}$ [7].

    Cellular metabolic rate dictates the pace of many biological processes [8]. Cellular metabolism could affect immune system search times by reducing the movement and proliferation of immune cells [9]. Rates of DNA and protein synthesis also depend on the cellular metabolic rate and could influence the rate at which pathogens replicate inside infected cells [3]. Pathogens are expected to replicate slower in larger species [3,10]. The possibilities that cells of the immune system and pathogens may move and proliferate at speeds independent of host body mass M ($\propto M^0$) or proportional to cellular metabolic rate ($\propto M^{-1/4}$) lead to four hypotheses, as originally proposed by Wiegel & Perelson [9]. Here, we use mathematical models and experimental data to test whether viral production rates and immune response rates and times vary with species body mass. We use West Nile virus (WNV) as a model pathogen for our studies because it is a generalist pathogen that infects many species in different taxonomic groups and with a range of body sizes.

(2) An effective immune response requires efficient detection of pathogens that may be initially localized. The detection of small amounts of pathogen may be harder in larger animals due to larger physical spaces. We hypothesize that the immune system is capable of nearly scale-invariant detection and response, i.e. rates of immune response and time taken by the immune system to detect and respond to pathogens do not scale appreciably with host body size [11]. Previous work has suggested how (i) the physical architecture of the immune system, comprising anatomical structures called lymph nodes (that facilitate recognition of pathogen by immune system cells), and (ii) chemical signalling within the immune system, guiding immune system cells to sites of infection, enable efficient and nearly scale-invariant detection and response [11,12].

WNV pathogenesis also depends on host phylogeny, e.g. passerine species (like sparrows) sustain more viraemia than non-passerine species (like geese) [13]. Corvid species (like crows) are particularly susceptible to WNV infection [13]. Hence, host phylogeny is expected to affect immune response and pathogen replication rates. Viral dynamics may be similar in related species. Modelling techniques that take advantage of relatedness of infected species may produce more accurate results. This work incorporates host phylogenetic hierarchy to estimate biologically relevant quantities for WNV infection.

Hierarchical Bayesian models enable modellers to encapsulate knowledge about the underlying biology as priors. Hierarchical models also pool information across disparate individuals from different groups and are well suited for cases where there are a limited number of observations from several individuals. Suitable priors in a hierarchical Bayesian framework can help reduce the variance of parameter estimates [14,15]. A model that incorporates the hierarchal nature of host phylogeny and encodes this information as priors in a hierarchical Bayesian model may enable more accurate estimates of parameters characterizing WNV infection.

Hierarchical Bayesian models have been used in image processing [16], ecological modelling [17] and climate modelling [18]. Bayesian nonlinear mixed effects models with a single level of hierarchy have been applied to modelling of the within-host response to HIV [19–21] and influenza [22]. Multilevel data fitting approaches and Bayesian frameworks are expected to be helpful in modelling within-host viral dynamics. However, to the best of our knowledge, Bayesian nonlinear mixed effects models with multiple levels of hierarchy have not been applied to within-host modelling.

Here, we use mathematical models of viral time courses from multiple species of birds infected by WNV to test more thoroughly how disease progression and immune response depend on mass and host phylogeny. Mathematical models that combine within-host experimental data from multiple species, such as the ones presented here, may also be useful in studying other zoonotic diseases and help increase our understanding of these diseases.

The role of host phylogeny in determining WNV infection outcome is known qualitatively [13]. We provide a quantitative prediction for viral competency for passerines and corvids in particular. Furthermore, the fits to experimental data using our mathematical models enable us to form hypotheses as to why WNV viraemia is higher in these groups.

We also calculate a reservoir competence index that indicates the relative number of infectious mosquitoes that would be derived from feeding on these hosts. This could be helpful in coupling within-host dynamics of WNV or other zoonotic diseases to dynamics of spread between different hosts enabling multi-scale models of disease spread.

2. Background on West Nile Virus

We use WNV as a model pathogen for our studies because it is a generalist pathogen that infects many species in different taxonomic groups and with a range of body sizes. WNV has emerged globally as a major cause of viral encephalitis and is maintained in an enzootic cycle between mosquitoes and birds [23], but can also infect and cause disease in many other vertebrates including humans. Following its

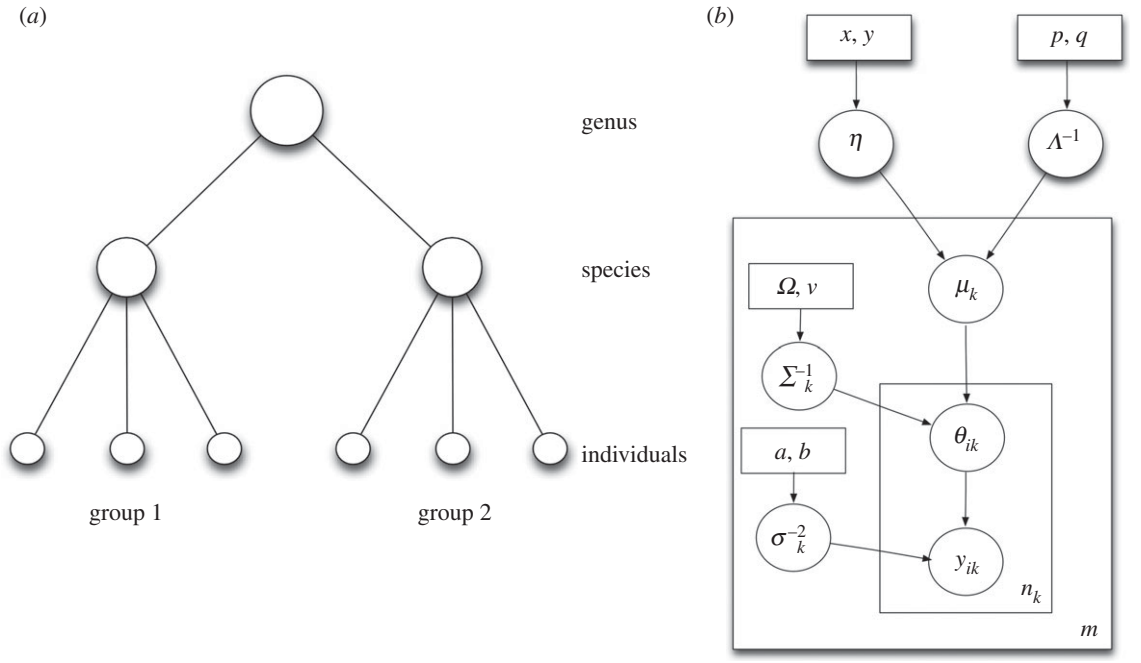

Figure 1. (*a*) Multi-level hierarchical model with two groups. Each group has three individuals. Also shown are the genus, species and individual levels. (*b*) Plate diagram for the multi-level hierarchical model. The plate denotes iteration of parameters and the number enclosed in the plate shows the number of iterations.

introduction into the United States in 1999, WNV spread rapidly across the North American continent in only four years and more recently has been reported in Mexico, South America and the Caribbean [24–26]. Although vaccines are available for animal use, no vaccines or specific therapies for WNV are currently approved for humans [27].

WNV is an enveloped flavivirus with a single-stranded, positive-sense, 11 kb RNA genome [27]. It is cytopathic and initially infects epidermal Langerhans cells, which then migrate to the draining lymph node and infect macrophages [27]. From the draining lymph node, WNV spreads to the spleen, kidneys and spinal cord and ultimately breaches the blood–brain barrier to infect neurons [27]. The standard pattern of WNV infection in birds is characterized by an initial exponential growth of virus that peaks around 3–4 days post infection (DPI), followed by an exponential decline until that leads to undetectable levels of virus by 6–8 DPI.

WNV infects a large number of species across different taxonomic groups. This allows us to test the effects of animal body size on pathogen replication and immune response. We focus on data from a study in which birds were experimentally infected with the same strain of WNV (WNV NY99-6480). Komar *et al.* [13] experimentally infected 25 species of birds (ranging from 3 g sparrows to 3 kg geese) with WNV NY99-6480 and took daily measurements of the concentration of infectious virions in blood (viraemia) over the course of infection. The initial inoculum size was not monitored as the infection was initiated by bites from infected mosquitoes.

3. Material and methods

3.1. Overview of methods

We use a Bayesian inference approach to estimate model parameters. Assume that a model (in our case, a differential equation model describing how plasma virus concentration changes over time) contains parameters Θ. The Bayesian approach allows us to include prior knowledge about model parameters in a systematic fashion. If we have information about Θ (e.g. from experimental evidence) which needs to be incorporated in our analysis, this is represented as a prior probability distribution $P(\Theta)$. Bayes rule allows us to incorporate the prior knowledge about parameters, $P(\Theta)$, and experimental data, D, to derive a posterior distribution of parameters:

$$P(\Theta \mid D) = \frac{P(D \mid \Theta) \cdot P(\Theta)}{P(D)}. \tag{3.1}$$

The multi-level hierarchical Bayesian model mimics the hierarchical nature of host phylogeny. There are three levels in the hierarchical tree representation: individual, species and genus (figure 1). Each level of the tree has an equation describing the distribution of model parameters (differential equation models, equations (3.2)–(3.5) and equations (3.2)–(3.4), (3.6)–(3.7)). The differential equation and hierarchical Bayesian models are explained in greater detail in the next section.

We compare two different hierarchical Bayesian models: a multi-level model that has three levels of hierarchy (individual, species and genus) (figure 1) and an aggregated model with two levels of hierarchy (all individuals of all species pooled together under a single genus) (figure 2).

3.2. Differential equation models of viral dynamics and immune response

We use two different viral dynamic models to account for the observed plasma viraemia. The first model assumes that infection is target cell limited in birds, i.e. the concentration of virus reaches a peak and then declines when few susceptible target cells remain. Models of target cell limited acute infection have been developed for HIV [28], influenza A virus [29], hepatitis C virus [30], simian immunodeficiency virus [31], hepatitis B virus [32] and Zika virus [33,34]. Here, we use a target cell limited model with an eclipse phase, given by the following

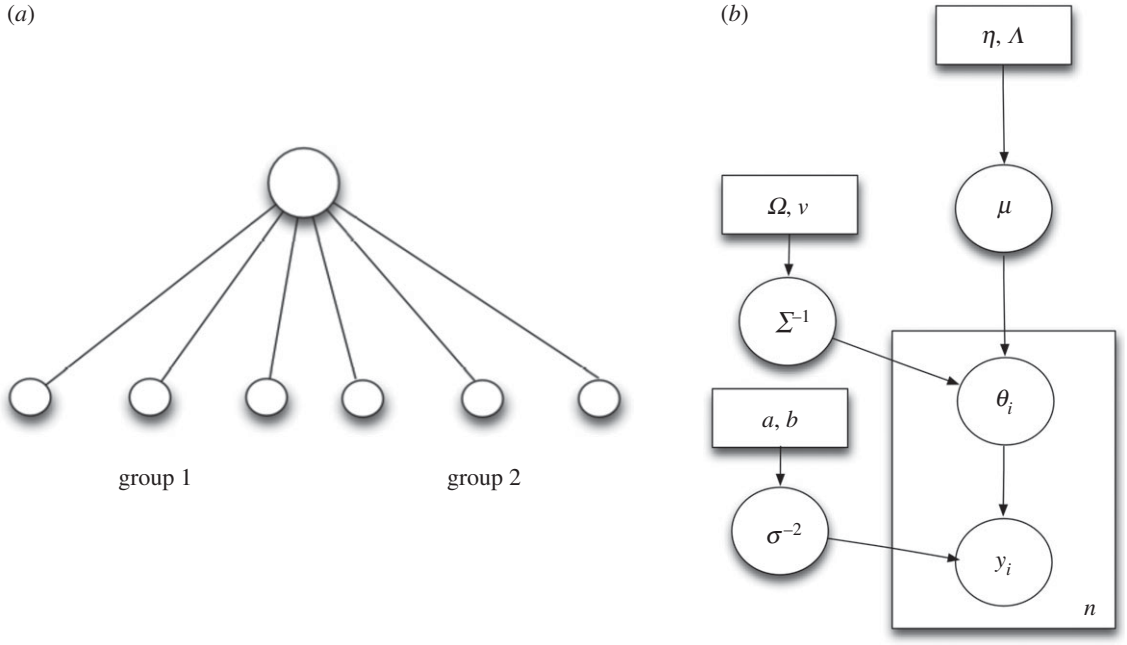

Figure 2. (*a*) Aggregated model with two groups combined. Each group has three individuals. Also shown are the genus, species and individual levels. (*b*) Plate diagram for the aggregated model. The plate denotes iteration of parameters and the number in the plate shows the number of iterations.

differential equations:

$$\frac{dT}{dt} = -\beta TV, \tag{3.2}$$

$$\frac{dI_1}{dt} = \beta TV - kI_1, \tag{3.3}$$

$$\frac{dI_2}{dt} = kI_1 - \delta I_2 \tag{3.4}$$

and

$$\frac{dV}{dt} = pI_2 - \gamma V - \beta TV, \tag{3.5}$$

where T is the density of uninfected target cells, and V is the viral titre measured in serum. Target cells become infected by virus at rate βTV , where β is the rate constant characterizing infection. The initial viral titre and the initial number of target cells are denoted V_0 and T_0, respectively. The initial density of infected cells is assumed to be zero. The separation of infected cells into two classes, I_1 cells that are infected but not yet producing virus and I_2 cells that produce virus, is similar to that in a model proposed earlier for influenza infection [29]. This separation increases the realism of the model, since delays in the production of virus after the time of initial infection are part of the viral life cycle (the eclipse phase). The parameter $1/k$ is the average transition time from I_1 to I_2. Productively infected cells (I_2) release virus at an average rate p per cell and die at rate δ per cell, where $1/\delta$ is the average lifespan of a productively infected cell. Free infectious virus is cleared at rate γ per infectious unit per day, for example by phagocytosis or loss of infectivity and is lost by entering cells during the infection process at rate βTV.

The Bayesian model infers (V_0, β, p, δ) from the viral titre data.

We also use a more sophisticated model that assumes viral decline is due to an adaptive antibody (induced IgM) response. Humoral immunity is an essential component of the immune response to WNV, as neutralizing antibodies limit dissemination of infection [35,36]. Diamond *et al.* [36] infected wild-type mice subcutaneously with WNV and measured titres of neutralizing antibody (analysed in [37]). The data can be described by the following piecewise linear function:

$$A(t) = \begin{cases} 0, & t < t_i \\ \eta(t - t_i), & t \geq t_i. \end{cases} \tag{3.6}$$

The level of neutralizing antibody at time t, $A(t)$, measured by the plaque reduction neutralization test (PRNT) is 0 before time t_i and increases linearly with time after that with rate η. We assume that neutralizing antibody, A, binds virus, V and neutralizes it with rate constant ρ, so that infectious virus is lost at rate $\rho A(t)V$. The model including neutralizing antibody consists of equations (3.2)–(3.4) and equation (3.6) with equation (3.5) replaced by

$$\frac{dV}{dt} = pI_2 - \gamma V - \beta TV - \rho A(t)V. \tag{3.7}$$

The Bayesian model infers $(V_0, \beta, p, \delta, \rho, t_i)$ from the viral titre data.

The ordinary differential equations describing our viral kinetic models were solved numerically in Matlab [38]. The Runge–Kutta 4 method of integration was employed. All model parameters and virus concentration are logged (base 10) in order to stabilize variance and ensure positive estimates from the Bayesian inference. A sample plot of the target limited model prediction of virus concentration over time (compared with observed virus concentration data) using one representative set of parameters inferred from the Bayesian model is shown in figure 3. Additional fits are shown in figures 8–10.

3.3. Constraints on model parameters

The parameter constraints are summarized in table 1. The initial density of uninfected target cells, T_0, is fixed to $2.3 \times 10^5\ \mathrm{ml}^{-1}$ (mean of the estimated range in mice [37]). The rate of clearance of free infectious virus, γ, is set to $44.4\ \mathrm{d}^{-1}$ (upper bound of estimate in mice [37]) for all species except three corvids (fish crows, blue jays and American crows) for which we could get good fits only with a lower value ($1\ \mathrm{d}^{-1}$).

The eclipse phase length, $1/k$, is fixed to the lower bound of a range from wild-type mice (6 h [37]). Similarly, based on calculations in wild-type mice [37], the lifetime of productively infected cells, $1/\delta$, is constrained to be greater than 1 h while obeying the relationship $1/\delta + 1/k = 24$ h. The rate at which

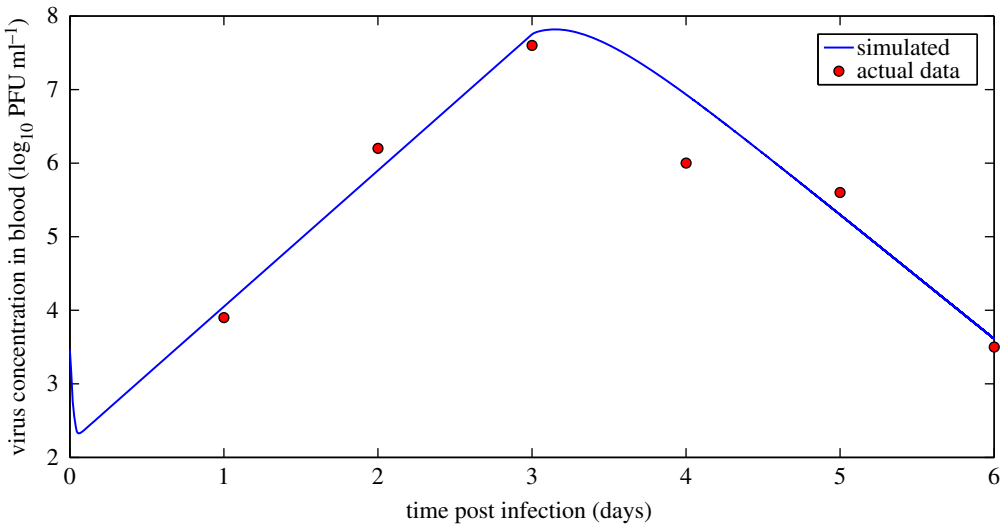

Figure 3. A sample ODE prediction for virus concentration (in $\log_{10}$ PFU ml^{-1}) over time post infection (blue) and experimental data on virus concentration (red). Data show viraemia of great-horned owls from [13]. (Online version in colour.)

Table 1. Parameter constraints for the target cell limited and adaptive immune response models in birds.

parameters	description	estimated ranges	source
γ	WNV clearance rate	44.4 d^{-1} for non-corvids and 1 d^{-1} for corvids	value for non-corvids estimated from fit to viral decay study in wild-type mice [39] ([37])
η	rate of IgM production	54.7 d^{-1}	fit to antibody titre study in wild-type mice [36] ([37])
T_0	initial target cell density	2.3×10^5 ml^{-1}	average of estimated range in mice from [40] ([37])
k	rate of transition from I_1 to I_2	4 d^{-1}	upper bound of estimate in mice from [41,42] ([37])
$1/\delta$	lifetime of productively infected cells	$\leq 24 - 1/k$ days	calculations in mice ([37])
t_i	time of initiation of IgM response (days post infection)	2–4 days	estimates in mice [36] ([37])
V_0	initial viral titre	0.1–10^{12} PFU ml^{-1}	model constraint in birds
p	rate of WNV production	0.5–10^{12} PFU d^{-1}	model constraint in birds
p/δ	infectious virion burst size	$\leq 10^{12}$ PFU	model constraint in birds
ρ	efficacy of antibody neutralization	0.05–1000 PRNT$_{50}^{-1}$ d^{-1}	model constraint in birds

neutralizing antibody, $A(t)$, increases linearly after time t_i (η) is fixed to 54.7 d^{-1} based on fit to antibody titre data in wild-type mice [36,37]. Finally, the time of initiation of antibody, t_i, is bounded between 2 and 4 days [37] based on experimental data and model fits to experimental data in wild-type mice.

The infectious virion burst size, p/δ, is constrained to not exceed 10^{12} plaque forming units (PFU) based on fits of the target cell limited model to birds. Similarly, the efficacy of antibody neutralization, ρ, is bounded between 0.05 and 1000 PRNT$_{50}^{-1}$ d^{-1} based on model fits to experimental infection data on birds (where the amount of antibody is measured in units of plaque reduction neutralization titre for 50% inhibition of plaques, PRNT$_{50}$). The initial viral titre, V_0, is constrained to be within 0.1 and 10^{12} PFU ml^{-1} and the rate of production of virus, p, is constrained to be within 0.5 and 10^{12} PFU d^{-1} based on fits in birds.

3.4. Experimental data

WNV infects bird species ranging from 3 g sparrows to 3 kg geese. This wide range of species mass allows us to test the effects of animal body size on pathogen replication and immune response. We focus on data from a study which experimentally infected animals with the same strain of WNV (WNV NY99-6480). Komar *et al.* [13] experimentally infected 25 species of birds with WNV NY99-6480 and took daily measurements of the concentration of infectious virions in blood (viraemia) over the course of infection (sample plot shown in figure 3, data points in red). Viraemia was reported in plaque forming units (PFU, a measure of the number of infectious virions) over a span of at most 7 days post infection.

The order Passeriforme includes more than half of all bird species. The experimental infection study infected 10 passerine species and 15 non-passerine species. The passerine species were house finches, house sparrows, red-winged blackbirds, blue jays, American robins, European starlings, common grackles, black-billed magpies, fish crows and American crows. WNV is exceptionally viraemic in corvid species (a family within the order Passeriforme). The corvid species in the study were blue jays, black-billed magpies, fish crows and American crows. The number of individuals infected per species ranged from 1 (American coots) to 8 (American crows).

The experimental infection study included data on 87 individual birds belonging to 25 distinct species. Of the 87 individuals infected, 26 succumbed to infection. With the exception of one individual (a non-passerine ring-billed gull), all mortality was in passerine species, primarily corvids.

From this experimental dataset, we excluded data on two species (Japanese quail and ring-necked pheasant). The virus kinetics in Japanese quail were erratic and viraemia levels in the ring-necked pheasant were approximately constant over time; as such the viral dynamics in these species could not be simulated by our model.

The experimental study also reported the level of detection (LOD) of their viral assay as $10^{1.7}$ PFU. The raw experimental data had reports of consecutive viral measurements at LOD in the declining phase of viraemia for some species. In these cases, we considered only the first data point below the LOD in the time series and discarded the remaining points. Considering all remaining LOD data points would have led our models to an underestimate of the slope of the declining phase due to the flat trajectory induced by multiple consecutive viraemia measurements at the same level. We note that there are alternative approaches for fitting measurements under the LOD as done for dengue virus [43].

3.5. Hierarchical Bayesian model

3.5.1. Aggregated model

Given a population of individuals, the aggregated model assumes that the dynamics of infection of each individual is characterized by parameters ($\theta_i, i = 1, \ldots, n$ for n individuals), where θ_i is a parameter vector that represents the differential equation model parameters (V_0, β, p, δ) for a particular individual. Each θ_i is drawn from a common top-level distribution (μ). In contrast with the multi-level Bayesian model (presented in the next section), the aggregated model has no species level and aggregates all individuals of all species together.

The aggregated model has 87 individuals. The individuals are aggregated to form a tree of height 2. The model is shown graphically in figure 2*a*. There are two levels in the hierarchical tree representation: individual and all individuals of all species combined. Each level of the tree has a distribution of ODE model parameters (equations (3.2)–(3.5) for the target cell limited model and equations (3.2)–(3.4), (3.6)–(3.7) for the model with an antibody response). The ith individual has a parameter vector θ_i that represents (V_0, β, p, δ) for a particular individual. This parameter θ_i is then drawn from a higher distribution with mean μ. The details of the model are given in the electronic supplementary material.

3.5.2. Multi-level hierarchical model

We also devised a multi-level hierarchical Bayesian model that mimics the hierarchical nature of host phylogeny. There are three levels in the hierarchical tree representation: individual, species and order (figure 1). There are 23 groups (species) with a total of 87 individuals. The groups are arranged hierarchically to form a tree of height 3. The model is shown graphically in figure 1*a*.

Each level of the tree has a distribution of ODE model parameters (equations (3.2)–(3.5) for the target cell limited model and equations (3.2)–(3.4), (3.6)–(3.7) for the model with an antibody response). The ith individual has a parameter vector θ_i that represents (V_0, β, p, δ) for a particular individual. This parameter vector θ_i is then drawn from a species-level distribution with mean μ. Finally, the species-level estimate is itself drawn from a order-level distribution centred around η. The details of the model are given in the electronic supplementary material.

We also compare the two different hierarchical Bayesian models: a multi-level model that has three levels of hierarchy (for individual, species and order) and an aggregated model with two levels of hierarchy (all individuals of all species pooled together under a single top-level entity; figure 2). Each of these Bayesian models is run on three different kinds of experimental data: all passerine and non-passerine species combined, passerine and non-passerine species separately and finally only corvid species (a subset of passerines).

3.5.3. Markov chain Monte Carlo implementation

We are interested in inferring the posterior distribution of the ODE parameters given the data ($P(\Theta \mid D)$, equation (3.1)). However, the distribution does not have an analytic form. Markov chain Monte Carlo (MCMC) techniques enable us to sample from a target distribution; the approximation gets better as more samples are drawn. We combine the Metropolis–Hastings algorithm and the Gibbs sampler. The Gibbs sampler is a type of MCMC algorithm that divides the parameters into a number of components and at each iteration sequentially updates each of them by conditioning on the others. The Metropolis–Hastings (M–H) algorithm updates Θ, and the Gibbs sampler updates all the remaining variables. We use a blockwise update scheme where proposed new values of parameters from the M–H algorithm are either all updated simultaneously or all reverted to their previous values. The M–H algorithm draws a subsequent sample from the target distribution centred around a proposal distribution. Our proposal distribution is a multi-variate normal distribution centred around the current value of θ_i.

The choice of dispersion of the proposal distribution is important. If the dispersion is high, many MCMC moves will be rejected and the procedure will take longer to converge. If the dispersion is low, the chain may not explore the parameter space thoroughly [44,45]. The standard deviation of the proposal distribution is chosen to be 0.01 on the log scale (base 10) for all the parameters. If a value returned by the proposal distribution is outside of the imposed bounds for ODE parameters, we reflect the value back into range. Convergence is checked informally based on graphical techniques [46].

All models were run for 10 000 iterations. This constitutes a single 'run' of the Bayesian model. The initial 1000 iterations were discarded (burn-in phase). Of the remaining 9000 samples, we retained every fifth sample.

3.5.4. Calculation of averages for ODE parameters at different levels of hierarchy

We use the posterior samples of ODE parameters to generate the averages in the following way: for each individual, species or order, we have 10 000 samples originally (as described in the previous section) at that level of the hierarchy. After burn-in, this is reduced to 9000 samples and thinning of the samples leads to a total of 1800 samples. The average of these samples (henceforth referred to as individual-, species- or genus-level averages) for each ODE parameter (V_0, β, p, δ) is used in 'Results' section.

3.6. Calculation of reservoir competencies

We calculate a reservoir competence index that indicates the relative number of infectious mosquitoes that would be derived from feeding on each host species and is calculated from the viraemia that develops in the hosts after infection. We extend a competence index developed by Komar *et al.* [13] to account for time-varying viraemia and species relatedness. Species that sustain WNV viraemia above 10^5 PFU ml^{-1} are considered infectious for mosquito vectors [13]. The reservoir competence index (C_i) is a function of susceptibility (s), the proportion of birds that become infected as a result of daily exposure; mean daily infectiousness (i), the proportion of exposed mosquito vectors that become infected per day; and the duration of infectious viraemia above a threshold (d) [13]. Komar *et al.* [13] calculate the competence as $C_i = s \times i \times d$, where $i = (\log_{10}(V_P) - 5)/10 + 0.02$ and V_P is the peak viraemia attained in a host; this assumes that hosts sustain a fixed viraemia (at the level of peak viraemia) throughout the duration of infectivity.

Our competency index for each species is calculated as follows:

$$C_i = s \times \int_0^{t_d} i(t)\,\mathrm{d}t \tag{3.8}$$

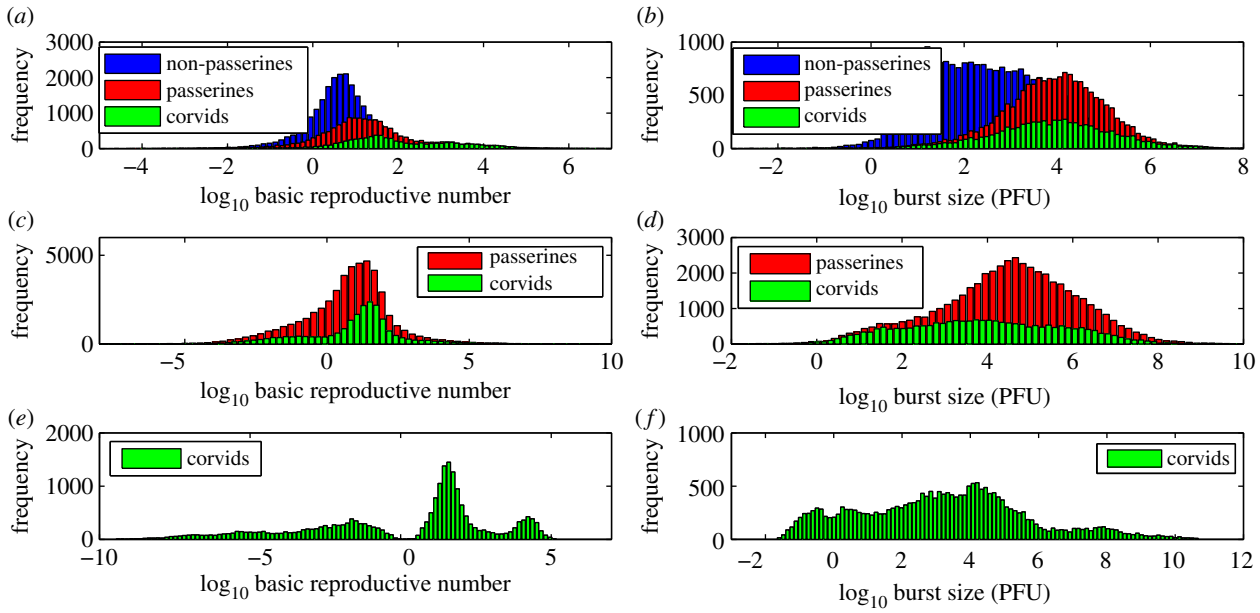

Figure 4. Posterior distribution of $\log_{10} R_0$ and burst size ($\log_{10} p/\delta$) for the multi-level model (target cell limited model). (*a*,*b*) Multi-level model with all passerines (red), corvids (green) and non-passerines (blue). (*c*,*d*) Multi-level model with only passerines (red) and corvids (green). (*e*,*f*) Multi-level model with only corvids (green).

and

$$i(t) = \begin{cases} \frac{\log_{10} V(t) - 5}{10} + 0.02, & \log_{10} V(t) \geq 5 \\ 0, & \log_{10} V(t) < 5, \end{cases} \quad (3.9)$$

where t_d is the time until viraemia is measured experimentally, $i(t)$ is the infectivity at time t and $V(t)$ is the ODE model predicted viraemia at time t. We note that our competency calculations also implicitly account for species relatedness via the hierarchical Bayesian model. Following bird mortality calculations made in Komar *et al.* [13], we set the susceptibility (s) to be 0.7 for budgerigars and 1 for all other species.

3.7. Estimate of accuracy in viraemia prediction

We estimated the accuracy of viraemia prediction by calculating three different estimates of a sum of squared residuals (SSR), between the data and the ODE model prediction (for both the multi-level and aggregated Bayesian models).

(1) Individual-level SSR. We calculated individual-level averages of ODE parameters for each species as described in §3.5.4. For each individual, we used these individual-level parameter estimates to generate a SSR between data and model prediction in the following way: for each individual in a species, we gave the ODE solver the individual-level parameter estimates and calculated the SSR between model prediction and individual-level data. The SSR was then summed up for all individuals of all species and then divided by the number of individuals. We call this the individual-level SSR.

(2) Species-level SSR. We calculated species-level averages of ODE parameters for each species as described in §3.5.4. For each species, we used these species-level parameter estimates to generate a SSR between data and model prediction in the following way: for each individual in a species, we gave the ODE solver the species-level parameter estimates and calculated the SSR between model prediction and individual-level data. The SSR was then summed up for all individuals within a species and then for all species and then divided by the number of individuals. We call this the species-level SSR.

(3) Order-level SSR. We calculated order-level averages of ODE parameters for each species as described in §3.5.4. For each individual, we used these order-level parameter estimates to generate a SSR between data and model prediction in the following way: for each individual in a species, we gave the ODE solver the genus-level parameter estimates and calculated the SSR between model prediction and individual-level data. The SSR was then summed up for all individuals of all species and then divided by the number of individuals. We call this the genus-level SSR.

4. Results

4.1. Effect of host phylogeny on parameter estimates

We investigated the effect of host phylogeny on two biologically relevant quantities—the basic reproductive number (R_0) and the infectious virion burst size (p/δ). Estimates of the mean values of R_0 from the multi-level model are higher for passerine species ($R_0 = 23.8$) compared with non-passerines ($R_0 = 4.7$) (figure 4*a*,*b*). Within passerines, corvid species (fish crows, blue jays, American crows and black-billed magpies) have the highest value of R_0 (93.2) (figure 4*a*,*b*). Infectious virion burst sizes in passerine species (8690 PFU) are much higher than those in non-passerines (340 PFU) (figure 4*a*,*b*).

Finally, estimates of R_0 in the aggregated model (figure 14) are orders of magnitude higher than in the multi-level model (figure 4). We hypothesize that as the aggregated model aggregates all species (including corvids) under a single hierarchy, non-corvid species are significantly influenced by parameter estimates from corvid species (which have high R_0). The multi-level model hierarchically separates corvid species from non-passerines and hence may moderate the influence of parameter estimates from corvid species on other non-corvid individuals.

4.2. Estimates of reservoir competence

We calculated a reservoir competence index that indicates the relative number of infectious mosquitoes that would be derived from feeding on these hosts (§3.6). The reservoir competency indices we calculate are correlated with and approximately half

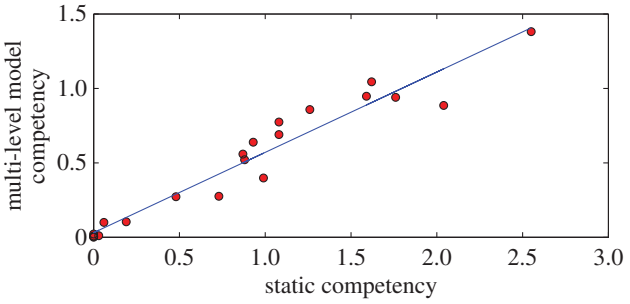

Figure 5. Correlation between multi-level model predicted competency and competency from Komar *et al.* [13] assuming static viraemia ($r^2 = 0.94$, slope = 0.54, p-value = 1×10^{-6}). (Online version in colour.)

of the estimates in Komar *et al.* [13] (figure 5, $r^2 = 0.94$, slope = 0.54, p-value = 1×10^{-6}). Passerine species have a higher mean reservoir competency (0.8) than non-passerine species (0.2),

Komar *et al.* [13] assume that virus is constant over time (at a level equal to peak viraemia) and hence they overestimate the total amount of virus produced over the duration of infectivity. In reality, viraemia changes over time and is actual highest at the time of peak viraemia. Our estimates of competency account for this and hence are lower than those of Komar *et al.* [13]. Area under the curve in a simple model where virus increases linearly (on a log scale) and then declines linearly after some time forms a triangle. The triangle has approximately half the total area of the rectangle formed by assuming viraemia is maintained at peak levels throughout the duration of infection. Hence, our model has approximately half the area (and half the value) compared to the estimates of Komar *et al.* [13].

4.3. Scaling of biologically relevant quantities

A scaling relationship between a parameter η in our model and host body mass, M, is written as $\eta \propto M^{\alpha}$ where α is the scaling exponent. Taking the logarithm of both sides of this relationship, we note

$$\log \eta \propto \alpha \log M. \tag{4.1}$$

Thus, the scaling exponent can be derived as the slope of a log–log plot of η versus M.

In order to derive scaling relations with host body size, we used the slope of the correlation between individual-level averages of ODE parameters and species mass, when the correlation was statistically significant (figure 6, multi-level model). The scaling relationships and statistics are summarized in table 2. We observed that the production rate of infectious virions (p, PFU per day) is correlated and declines with species body mass (all species combined: p-value = 0.002, $r^2 = 0.11$, slope = −1.1; only passerines: p-value = 0.04, $r^2 = 0.1$, slope = −1). We note that although the 95% confidence interval (CI) for the slope [−1.5, 0.03] includes the theoretically predicted exponent of −0.25, the confidence interval is too large to provide a meaningful estimate of the scaling exponent.

The basic reproductive number represents the average number of second-generation infections produced by a single infected cell placed in a population of susceptible cells. If R_0 is greater than 1, then an infection can be established, whereas an infection rapidly dies out if R_0 is less than 1. For the target cell limited model (equations (3.2)–(3.5)), R_0 is given by [47]

$$R_0 = \frac{p\beta T_0}{\delta(\gamma + \beta T_0)}. \tag{4.2}$$

The basic reproductive number, R_0, had a significant overall decreasing correlation with mass (all species combined: p-value = 0.006, $r^2 = 0.09$, slope = −1.3; only passerines: p-value = 7×10^{-5}, $r^2 = 0.33$, slope = −3.7). Peak viraemia (P_v, the maximum viraemia attained by an individual over the time course of infection) had a marginally significant correlation with species mass (all species combined: p-value = 0.06, r^2 = 0.04, slope = −0.82).

The burst size (p/δ, PFU), representing the number of infectious virions released by an infected cell over its productively infected lifespan, had a significant decreasing relationship with species body mass (all species combined: p-value = 0.001, $r^2 = 0.12$, slope = −1.1; only passerines: p-value = 0.03, $r^2 = 0.12$, slope = −1.1). Also the viraemia at day 1 (V (1)) was correlated with mass (all species combined, p-value = 0.003, $r^2 = 0.1$, slope = −1.1).

The density of inoculated virions (V_0) and the infectivity of WNV (β) had no significant relationship with species mass (all species combined: p-value = 0.35, $r^2 = 0.1$ and p-value = 0.64, $r^2 = 0.003$). Finally, the death rate of productively infected cells (δ, d^{-1}) had no relationship with mass (all species combined, $p = 0.6$).

We also used the adaptive immune response model (equations (3.2)–(3.4), (3.6)–(3.7)) to investigate how parameters related to the immune response scale with host body mass (multi-level model, figure 7 and table 2). The rate of adaptive immune system mediated neutralization of virus (ρ, $PRNT_{50}^{-1}\,d^{-1}$) did not have a significant relationship with body mass (all species combined: p-value = 0.11, $r^2 = 0.03$, slope = −0.19), although we observed that in passerines it declined significantly with mass (p-value = 0.04, $r^2 = 0.1$, slope = −0.51). The time of initiation of IgM response (t_i) also did not have a significant relationship with species mass (all species combined: slope = 0.002, p-value = 0.56, r^2 = 0.004; only passerines: slope = 0.01, p-value = 0.07, $r^2 = 0.08$).

The model parameter estimates and confidence intervals for all species are summarized in tables 3 and 4. The ODE model predictions for viraemia (for the parameters shown in tables 3 and 4) for all species are shown in figures 8–10.

We also show the posterior distributions of parameters for two selected species to highlight the differences in model parameters between these species in figure 11 (target cell limited model parameters V_0, β, p, δ for two species: American crows and Canadian geese). This underscores the variation in parameters between different species. Similar variation was seen between other species.

Finally, we show the trace plots of samples from the posterior distribution (the Monte Carlo Markov chain after burn-in and thinning) in figure 12. These show that the inference procedure converges around a particular value with variation around it. There is variation around a mean consistent with what would be observed when parameters have been identified. Similar behaviour was seen for posterior distributions of other species (data not shown).

4.4. Accuracy in viraemia prediction

We estimated the accuracy of viraemia prediction by calculating three different estimates of a SSR between the data and the ODE

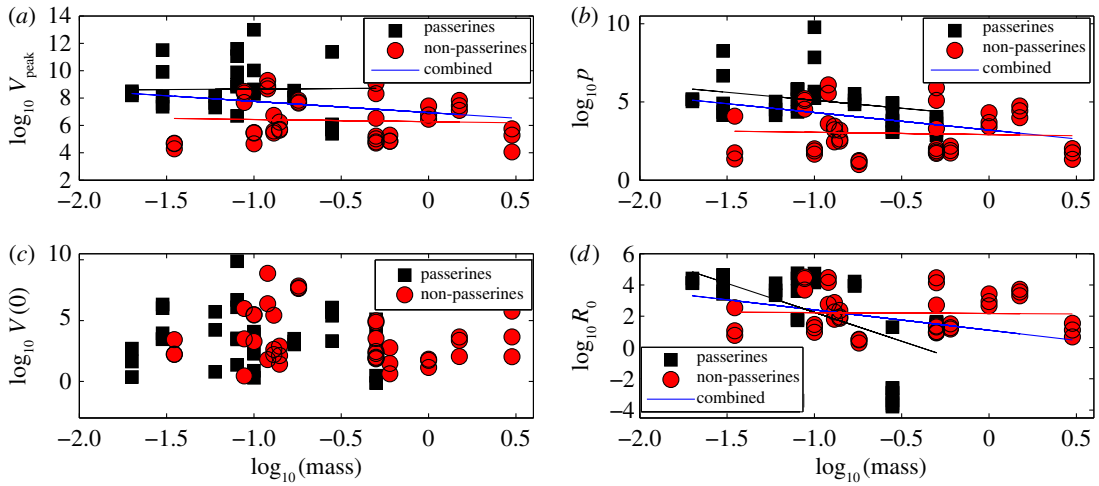

Figure 6. Scaling of biologically relevant quantities with host mass for the multi-level model: passerines (black square and black regression line, non-passerines (red circle and red regression line) and all combined (blue regression line). (*a*) Peak viraemia (V_p), slope = −0.82, *p*-value = 0.06, $r^2 = 0.04$. (*b*) WNV production rate (p), slope = −1.1, *p*-value = 0.002, $r^2 = 0.11$. (*c*) Inoculated density of virions (V_0), *p*-value = 0.35. (*d*) R_0, slope = −1.3, *p*-value = 0.006, $r^2 = 0.09$. (Online version in colour.)

Table 2. Statistics of scaling relationships with species mass (all species combined) from multi-level model with target cell limitation (TCL) or an adaptive immune response (AIR).

parameters	combined slope	*p*	r^2
p (TCL): WNV production rate	−1.1	0.002	0.11
p (AIR): WNV production rate	−0.84	0.04	0.05
R_0 (TCL): basic reproductive number	−1.3	0.006	0.09
R_0 (AIR): basic reproductive number	−0.57	0.08	0.04
p/δ (TCL): burst size	−1.1	0.001	0.12
p/δ (AIR): burst size	−0.86	0.03	0.06
V_p (TCL): peak viraemia	−0.82	0.06	0.04
V_p (AIR): peak viraemia	—	0.54	0.005
V_0 (TCL): inoculated WNV density	—	0.35	0.1
V_0 (AIR): inoculated WNV density	—	0.1	0.03
β (TCL): WNV infectivity	—	0.64	0.003
β (AIR): WNV infectivity	—	0.39	0.009
δ (TCL): death rate of productively infected cells	—	0.6	0.003
δ (AIR): death rate of productively infected cells	—	0.51	0.005
ρ (AIR): rate of adaptive immune system mediated virus neutralization	—	0.11	0.03
t_i (AIR): time of initiation of IgM response	—	0.56	0.004

model prediction (for both the multi-level and aggregated Bayesian models), as described in 'Estimate of accuracy in viraemia prediction' section.

We observed that the multi-level model produced more accurate estimates at the individual level and species level (figure 13*a*). A representative viraemia prediction for both the multi-level and aggregated model is also shown (figure 13*b*). The performance of both the models at the order level was similar.

We note that in some cases our hierarchical Bayesian models produce parameter estimates that appear to be biologically unreasonable, e.g. the production rate of WNV from infected cells (p) for some corvid species is predicted to be approximately 10^8 PFU d^{-1}, which is unreasonably high. This could be due to the fact that the empirically measured viraemia goes up to 10^{15} PFU ml^{-1} in some of these species and the initial susceptible cell population density (T_0 in our models) is much higher in corvids. This may suggest that other cell types are also available for infection in these species. Further experimental data on the susceptible cell population in corvid species may lead to more realistic estimates of the production rate (as these two parameters are correlated in our models).

5. Discussion

5.1. Summary

Understanding how quickly pathogens replicate and how quickly the immune system responds is important for

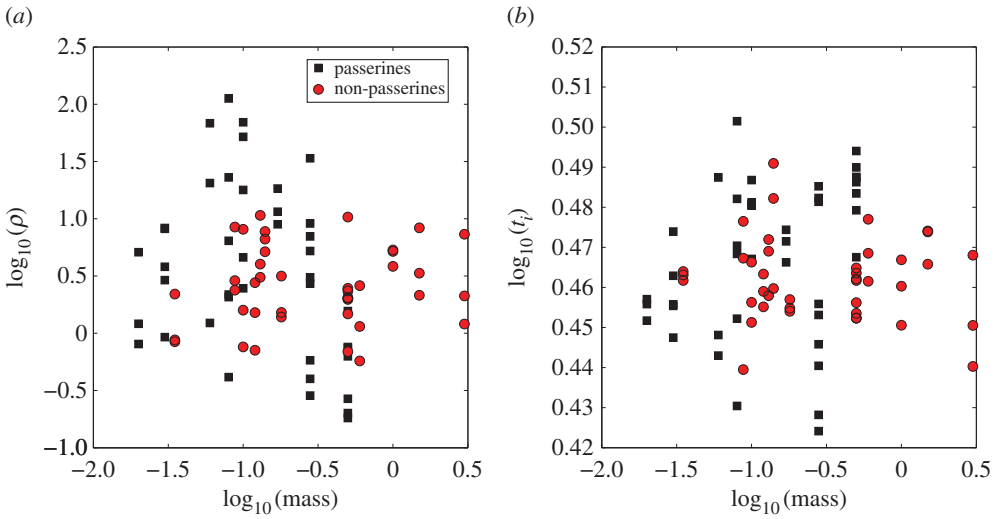

Figure 7. Scaling of immune response parameters with host mass for the multi-level model with immune response: passerines (black square) and non-passerines (red circle). (*a*) Rate of adaptive immune system mediated virus neutralization (ρ, $PRNT_{50}^{-1}$ d^{-1}). (*b*) Time of initiation of IgM response (t_i, days) (combined and each group separately are non-significant). (Online version in colour.)

Table 3. Estimated parameters (mean and 95% CI) for all species from the multi-level model with target cell limitation (TCL). V_0, inoculated virus density (PFU ml^{-1}); β, rate constant of infection (ml d^{-1}); p, infectious virus production rate (PFU d^{-1}); δ, death rate of productively infected cells (d^{-1}).

species	V_0	β	p	δ
fish crow	5×10^3 [4.3×10^3, 5.8×10^3]	3.7×10^{-9} [2.8×10^{-9}, 4.8×10^{-9}]	2.1×10^4 [1.8×10^4, 2.4×10^4]	4.8 [4.5, 5]
blue jay	2.9×10^3 [2.5×10^3, 3.3×10^3]	3.9×10^{-8} [3.3×10^{-8}, 4.6×10^{-8}]	4.9×10^4 [4.4×10^4, 5.5×10^4]	5.3 [4.9, 5.6]
American crow	740 [650.4, 842.2]	1.6×10^{-9} [1.4×10^{-9}, 1.9×10^{-9}]	1.4×10^4 [1.3×10^4, 1.6×10^4]	6.2 [5.9, 6.4]
common grackle	185.1 [163.1, 210]	4.4×10^{-8} [3.8×10^{-8}, 5×10^{-8}]	2.5×10^5 [2.2×10^5, 2.8×10^5]	6.8 [6.6, 7]
American robin	1.1×10^3 [0.9×10^3, 1.3×10^3]	4.8×10^{-8} [4.2×10^{-8}, 5.4×10^{-8}]	1.5×10^5 [1.3×10^5, 1.7×10^5]	7.4 [7.1, 7.6]
red-winged blackbird	731.4 [648.7, 824.6]	8×10^{-7} [7.2×10^{-7}, 8.7×10^{-7}]	1.5×10^4 [1.4×10^4, 1.6×10^4]	6.4 [6.2, 6.6]
black-billed magpie	1.4×10^3 [1.2×10^3, 1.5×10^3]	1×10^{-7} [0.9×10^{-7}, 1.1×10^{-7}]	5.2×10^4 [4.8×10^4, 5.7×10^4]	5.3 [5.1, 5.5]
house finch	305.9 [269.2, 347.8]	1.3×10^{-7} [1.1×10^{-7}, 1.5×10^{-7}]	3.5×10^4 [3.1×10^4, 3.8×10^4]	3.1 [2.9, 3.3]
house sparrow	3.4×10^3 [3×10^3, 3.9×10^3]	2.2×10^{-7} [1.9×10^{-7}, 2.5×10^{-7}]	7.4×10^4 [6.6×10^4, 8.2×10^4]	4.6 [4.5, 4.7]
mallard	164.2 [146.4, 184.1]	2.5×10^{-6} [2.3×10^{-6}, 2.9×10^{-6}]	3.3×10^3 [3×10^3, 3.7×10^3]	5.4 [5.2, 5.7]

predicting the epidemic spread of emerging pathogens. Host body size, through its correlation with metabolic rates, is theoretically predicted to impact pathogen replication and immune system response [9]. Prior work suggests that body mass affects pathogen replication [3,11,48], but immune response times are either independent of mass or increase only very slowly as mass increases [11,49]. Here, we use mathematical models of viral time courses from multiple species of birds infected by WNV to test more thoroughly how disease progression and immune response depend on host phylogeny and mass. The mathematical framework presented here is an approach to uncover systematic differences in disease progression and immune response in animals that differ in body mass and phylogeny.

Host phylogeny is an important determinant of the pathogenesis of WNV. Passerine species sustain more

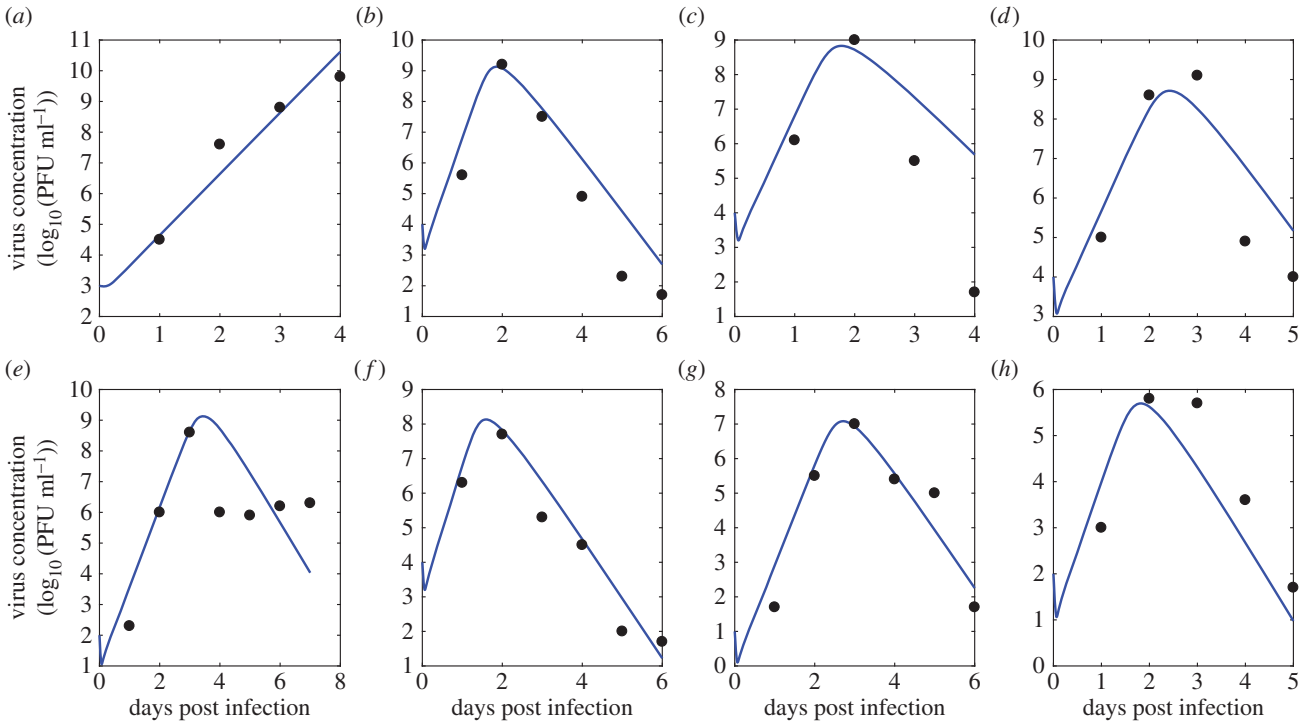

Figure 8. Predictions from the ODE model given by equations (3.2)–(3.5) for plasma virus concentration (in $\log_{10}$ PFU ml^{-1}) over time post infection (blue) and experimental data on virus concentration (black). Parameters used are the mean for each species from the multi-level model. Data show the viraemia of the following species from [13]: (*a*–*h*) American crow, American robin, red-winged blackbird, black-billed magpie, house finch, house sparrow, mallard and mourning dove, respectively. (Online version in colour.)

Table 4. Estimated parameters (mean and 95% CI) for all species from multi-level model with target cell limitation (TCL) (continued from table 3). V_0, inoculated virus density (PFU ml^{-1}); β, rate constant of infection (ml d^{-1}); p, infectious virus production rate (PFU d^{-1}); δ, death rate of productively infected cells (d^{-1}).

species	V_0	β	p	δ
mourning dove	867.9 [750, 1000.4]	5.3×10^{-6} [4.8×10^{-6}, 5.9×10^{-6}]	1.1×10^{3} [1×10^{3}, 1.2×10^{3}]	5.3 [5.1, 5.6]
ring-billed gull	3.2×10^{3} [2.8×10^{3}, 3.7×10^{3}]	1.8×10^{-7} [1.6×10^{-7}, 2.1×10^{-7}]	2.8×10^{4} [2.5×10^{4}, 3.1×10^{4}]	3.5 [3.4, 3.6]
great-horned owl	1×10^{3} [0.9×10^{3}, 1.1×10^{3}]	3.4×10^{-7} [3.1×10^{-7}, 3.7×10^{-7}]	1.5×10^{4} [1.4×10^{4}, 1.7×10^{4}]	4.6 [4.5, 4.9]
American kestrel	2.1×10^{3} [1.8×10^{3}, 2.4×10^{3}]	2.5×10^{-7} [2.2×10^{-7}, 2.9×10^{-7}]	3.3×10^{4} [3×10^{4}, 3.8×10^{4}]	5.5 [5.3, 5.7]
killdeer	871.7 [769.5, 987.3]	5.9×10^{-7} [5.3×10^{-7}, 6.7×10^{-7}]	1.9×10^{4} [1.8×10^{4}, 2.1×10^{4}]	5.2 [4.9, 5.4]
northern bobwhite	4.7×10^{3} [3.9×10^{3}, 5.6×10^{3}]	2.6×10^{-6} [2.1×10^{-6}, 3.1×10^{-6}]	1.1×10^{3} [0.9×10^{3}, 1.3×10^{3}]	2.7 [2.5, 2.9]
northern flicker	250.3 [223.6, 280.2]	7.2×10^{-6} [6.6×10^{-6}, 8×10^{-6}]	884.3 [811.6, 963.5]	5.9 [5.7, 6.3]
rock dove	228.4 [200.7, 259.9]	8.5×10^{-5} [7.9×10^{-5}, 9×10^{-5}]	95.7 [90.8, 100.9]	5.5 [5.3, 5.7]
American coot	208.7 [182.4, 238.7]	2.1×10^{-5} [1.8×10^{-5}, 2.3×10^{-5}]	207.4 [187.9, 228.8]	5.3 [5, 5.6]
monk parakeet	2.3×10^{3} [1.9×10^{3}, 2.7×10^{3}]	1.3×10^{-5} [1.1×10^{-5}, 1.5×10^{-5}]	296.3 [264.9, 331.5]	6.1 [5.7, 6.5]
budgerigar	674.7 [599.4, 759.4]	2.9×10^{-6} [2.5×10^{-6}, 3.4×10^{-6}]	860.8 [757.2, 978.6]	4.8 [4.5, 5.1]
Canada goose	647.3 [553.2, 757.3]	4.1×10^{-5} [3.7×10^{-5}, 4.6×10^{-5}]	144.5 [131, 159.4]	4.8 [4.5, 5.2]

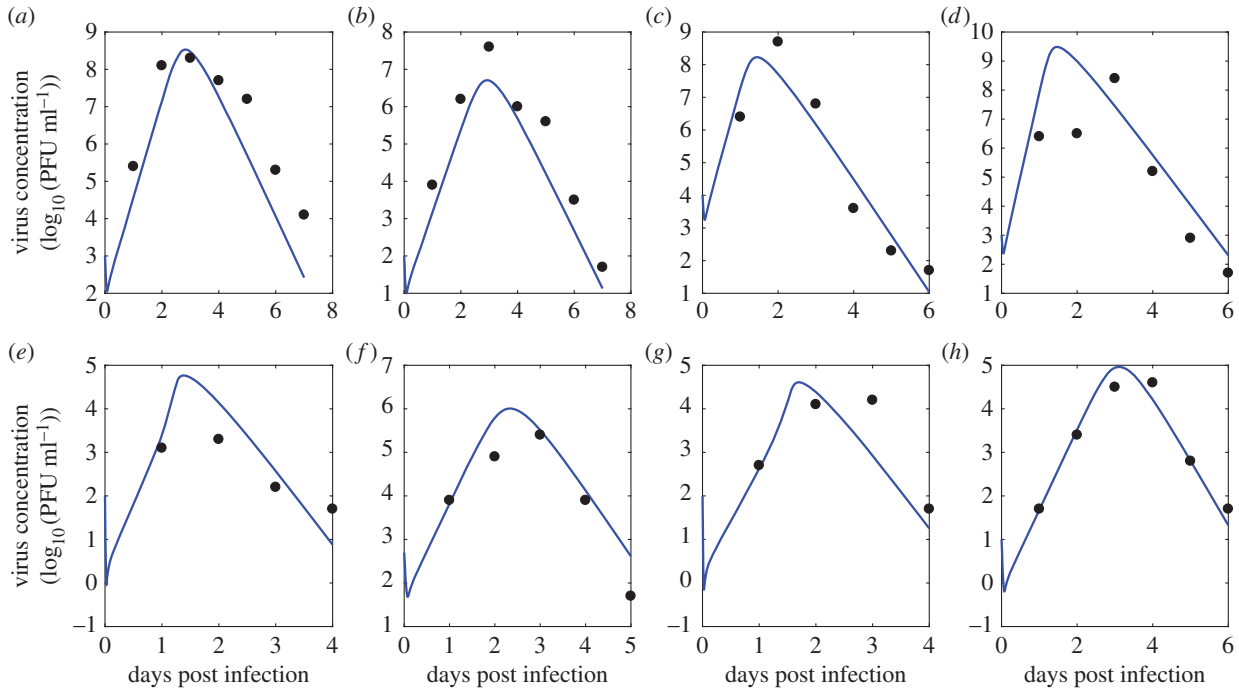

Figure 9. Predictions from the ODE model given by equations (3.2)–(3.5) for plasma virus concentration (in $\log_{10}$ PFU ml^{-1}) over time post infection (blue) and experimental data on virus concentration (black). Parameters used are the mean for each species from the multi-level model. Data show the viraemia of the following species from [13]: (*a*–*h*) ring-billed gull, great-horned owl, American kestrel, killdeer, northern bobwhite, northern flicker, rock dove and American coot, respectively. (Online version in colour.)

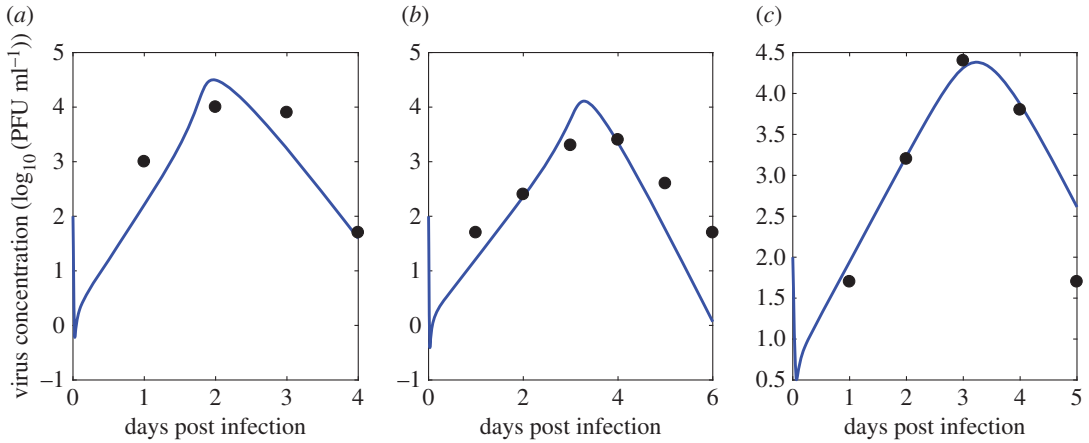

Figure 10. Predictions from the ODE model given by equations (3.2)–(3.5) for plasma virus concentration (in $\log_{10}$ PFU ml^{-1}) over time post infection (blue) and experimental data on virus concentration (black). Parameters used are the mean for each species from the multi-level model. Data show the viraemia of the following species from [13]: (*a*–*c*) Monk parakeet, Canadian geese and common grackle, respectively. (Online version in colour.)

viraemia than non-passerine species, and corvid species are particularly susceptible to WNV infection [13]. Therefore, any analysis of disease dynamics in multi-host pathogens like WNV should take host phylogeny into account. We accomplish this using hierarchical Bayesian models that incorporate the hierarchical nature of phylogeny. We observe that the multi-level model produces more accurate predictions of viraemia at the individual and species level when compared with hierarchical models that have fewer levels of hierarchy (figure 13). Additionally, multi-level models with passerines and non-passerines or just passerines (figure 4*a*–*d*) produced more realistic and more strongly peaked distributions of R_0 and burst size, than a multi-level model with only corvids (figure 4*e*,*f*) and the aggregated model (figure 14). This appears to be due to the ability of multi-level models to pool information at the species level from different related species.

A major contribution of this work is a computational framework for infectious within-host disease modelling that leverages data from multiple species. This is particularly important given recent analysis of many zoonotic diseases that can or have the potential to cross species boundaries and infect humans [50]. Our models estimate a competency for infected hosts to infect mosquito vectors that can then sustain the disease between hosts. This link between within- and between-host disease models is an important step towards understanding the threat from emerging zoonotic diseases and identifying likely candidate species for biosurveillance.

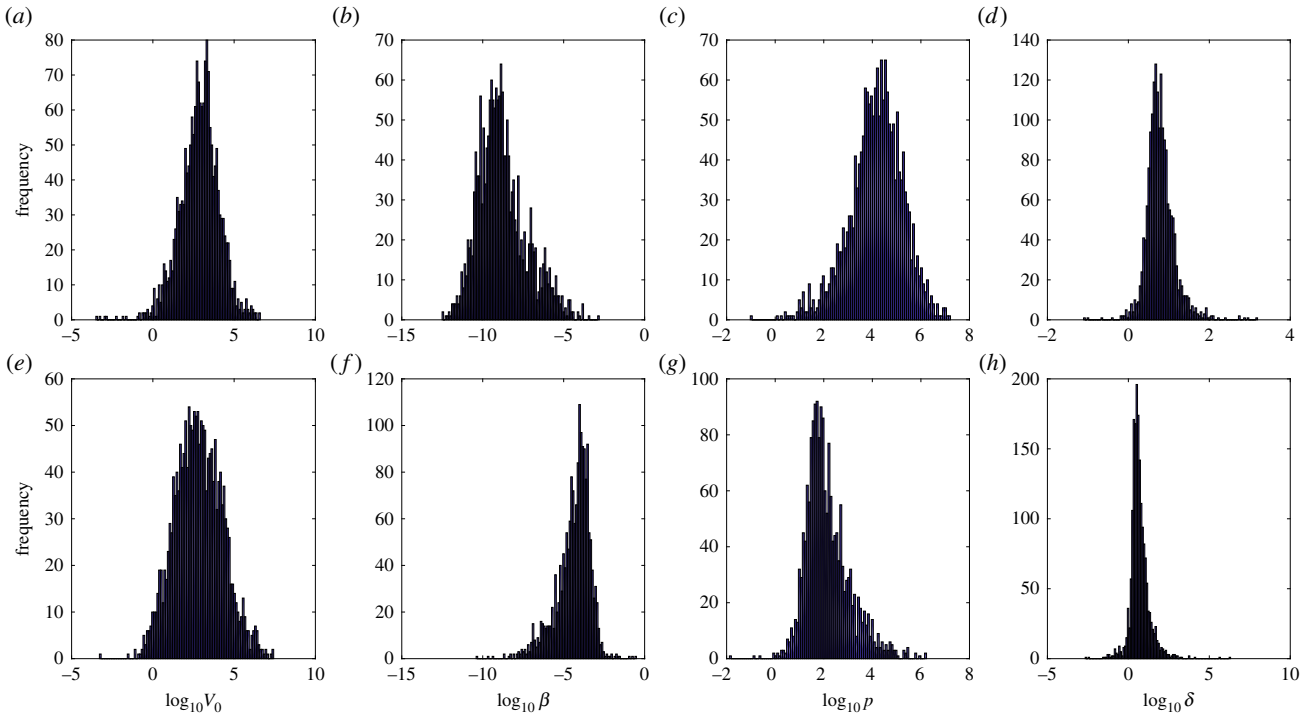

Figure 11. (*a–d*) Posterior distribution of target cell limited model parameters for American crows. (*e–h*) Posterior distribution of target cell limited model parameters for Canadian geese. Parameters shown are V_0, inoculated virus density (PFU ml^{-1}); β, rate constant of infection (ml d^{-1}); p, infectious virus production rate (PFU d^{-1}); δ, death rate of productively infected cells (d^{-1}).

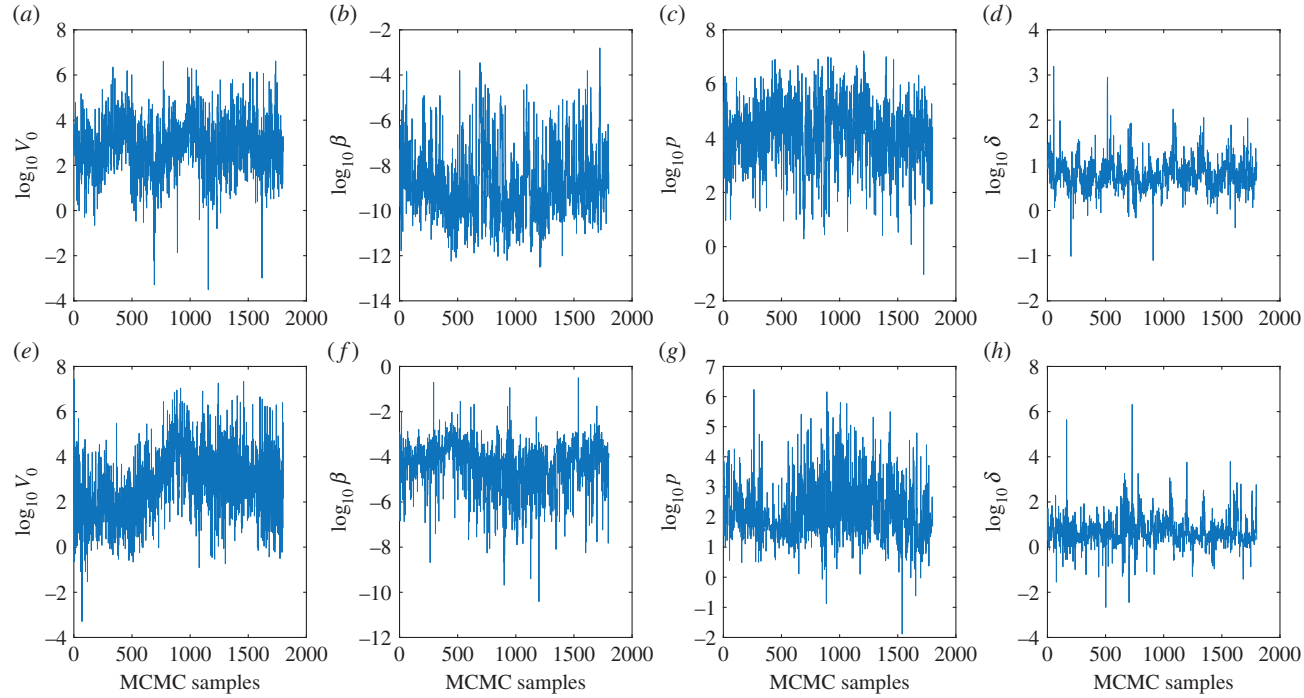

Figure 12. Trace of samples from the Monte Carlo Markov Chain after burn-in and thinning (taking every 5th sample). (*a–d*) Trace of target cell limited model parameters for American crows. (*e–h*) Trace of target cell limited model parameters for Canadian geese. Parameters shown are V_0, inoculated virus density (PFU ml^{-1}); β, rate constant of infection (ml d^{-1}); p, infectious virus production rate (PFU d^{-1}); δ, death rate of productively infected cells (d^{-1}). (Online version in colour.)

5.2. Effect of host phylogeny on within-host pathogen replication

We quantified the basic reproductive number (R_0), which represents the average number of infections produced by a single infected cell. The mean values of R_0 from the multi-level model are higher for passerine species ($R_0 = 23.8$) compared to non-passerines ($R_0 = 4.7$) (figure 4*a*,*b*). Within passerines, corvid species have the highest mean value of $R_0 = 93.2$.

Burst sizes of passerine species (8690 PFU) are also much higher than those in non-passerines (340 PFU) (figure 4*a*,*b*). This is consistent with experimental studies that found passerines and corvids have higher reservoir competency (ability to transmit infection to mosquitoes) than non-passerines [13]. Overall, host phylogeny emerges as a key determinant of biologically relevant quantities like the basic reproductive number and burst size. The mean passerine values for R_0 (23.8) and burst size (8690 PFU) are also

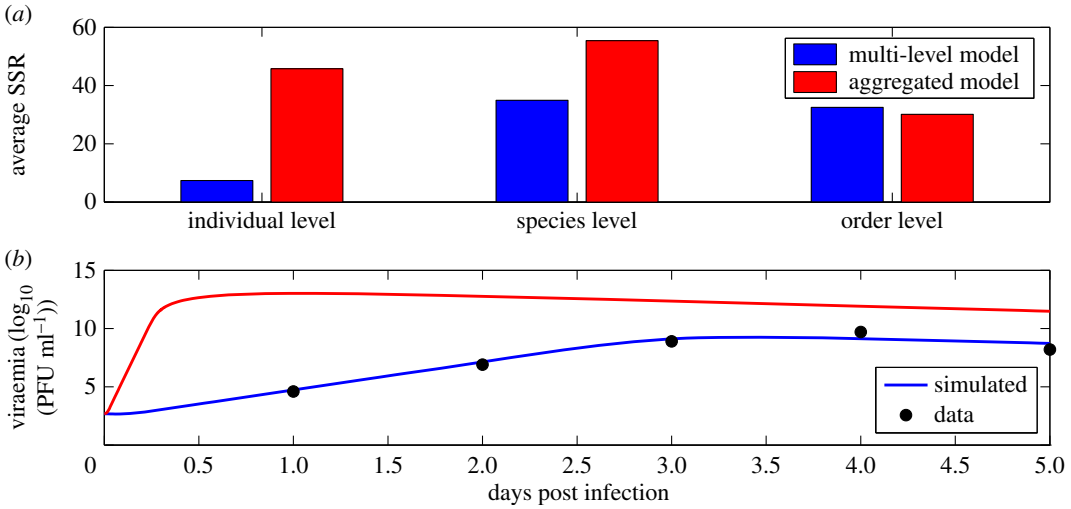

Figure 13. (*a*) Accuracy in viraemia prediction between multi-level model (blue) and aggregated model (red) for three different levels—individual, species and order. SSR—sum of squared residuals between model predicted viraemia and data. (*b*) A sample viraemia prediction from the multi-level (blue) and aggregated model (red).

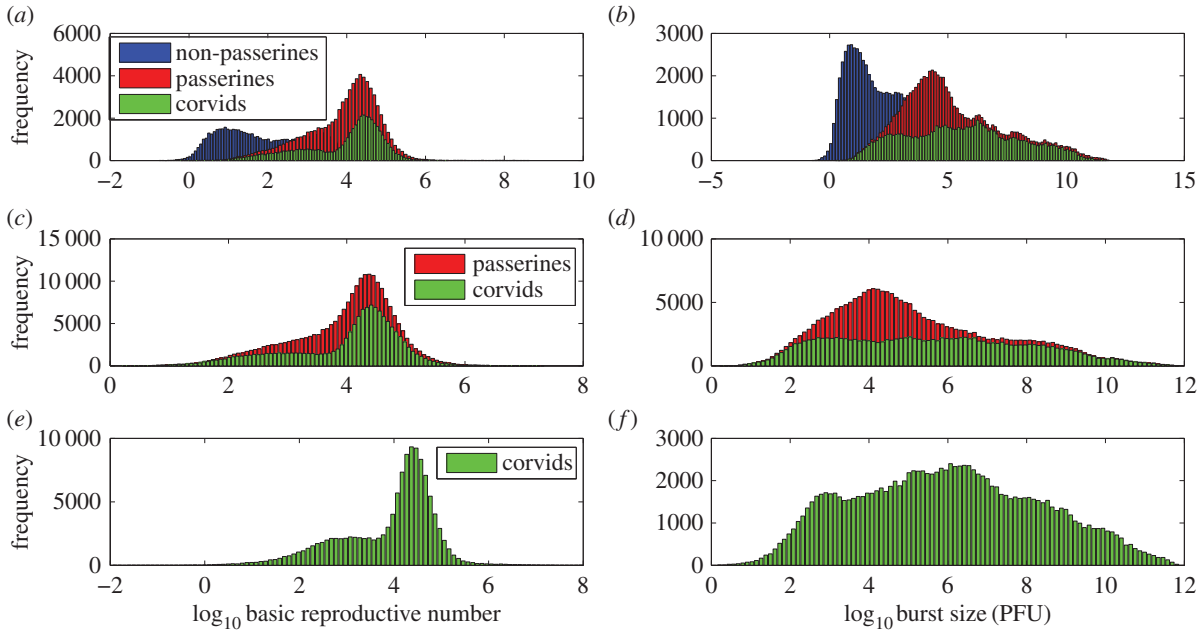

Figure 14. Posterior distribution of $\log_{10} R_0$ and burst size ($\log_{10} p/\delta$) for the aggregated model (target cell limited model). (*a,b*) Aggregated model with all passerines (red), corvids (green) and non-passerines (blue). (*c,d*) Aggregated model with only passerines (red) and corvids (green). (*e,f*) Aggregated model with only corvids (green).

higher than the corresponding model's predicted values in mice (mean $R_0 = 2.5$ and burst size = 3 PFU [37]).

We also calculated a reservoir competence index that indicates the relative number of infectious mosquitoes that would be derived from feeding on these hosts (§3.6) and accounts for time-varying viraemia and species relatedness (via the multi-level Bayesian model). The mean reservoir competence for passerine species (0.8) is higher than that of non-passerines (0.2). This lends mechanistic insight into why some species (smaller passerine species) are pathogen reservoirs and some (larger non-passerine species) are potentially dead-end hosts for WNV.

5.3. Effect of host body size on virion production rate and immune response

Individual cellular metabolic rates and biological times, if constrained by host metabolism, are expected to scale with host body mass as $M^{-1/4}$ and $M^{1/4}$, respectively [4–9]. We observe that the production rate of infectious virions (p) declined significantly with host body mass. While the 95% confidence interval for the slope (CI = [−1.5, 0.03]) includes the theoretically predicted exponent of −0.25 (figure 6), the confidence interval is too large to provide a meaningful estimate of the scaling exponent.

The death rate of productively infected cells (δ), also theoretically predicted to decrease with mass, had no relationship with species mass. The basic reproductive number (R_0) declined significantly with body mass (figure 6) and the relationship is even more significant within passerine species. This is most likely because the production rate of virions (p), a component of the R_0 (equation (4.2)), declined with mass.

We found that the mean of peak viral concentration in serum (peak viraemia) declines with species body mass (figure 6) although the correlation was only marginally significant. Additionally, most passerine species had higher peak viraemia than non-passerines. This is likely because

most of the passerine species had a higher value of the viral production rate (p) than the non-passerines. Overall, we observed modest trends in the direction predicted by scaling theory, but host phylogeny is a more important predictor of WNV dynamics.

The density of inoculated virions (V_0) has no significant relationship with species mass (figure 6). This is surprising because if mosquitoes inoculated a fixed dose directly into the host bloodstream, the density of virions (V_0) would be expected to decline with host mass because the dilution would be higher in the larger blood volume of a larger species. Mosquito-inoculated WNV, that is initially localized at the site of infection [51], is taken up by immune cells to nearby lymph nodes and then trafficks to different organs via the bloodstream. However, for computational efficiency, we use differential equation models that assume all components are well mixed. Our results point to the importance of accounting for spatial patterns of viral spread, especially in the initial stages of an infection when virus is localized, and this should be considered in future work.

The target cell limited and adaptive immune response models both produce similar scaling predictions when coupled to the multi-level hierarchical model (table 2).

We used the adaptive immune response model (equations (3.2)–(3.4), (3.6)–(3.7)) to investigate how parameters related to the immune response scale with host body mass (multi-level model, figure 7). The rate at which the adaptive (antibody) immune response neutralizes virus (ρ) did not have a significant relationship with body mass, although we observed that in passerine species it declined significantly with mass. The time of initiation of the adaptive (antibody) immune response (t_i) also did not have a significant relationship with species mass.

Hence, in sharp contrast to the vast majority of biological rates that slow systematically as the body size increases [7,8], the rate at which the adaptive (antibody) immune response clears WNV and the time at which it is initiated do not vary systematically with mass (figure 7). Previous work has suggested two plausible mechanisms that both enable this scale-invariant search and response. A sub-modular architecture of lymph nodes, which balances local pathogen detection and global antibody production, may lead to nearly scale-invariant search and response times [11,52]. Chemical signals released from infected sites also help guide immune system cells towards infected regions, reducing the time for the immune system to search for infection [12,53,54].

5.4. Implications for spread of West Nile Virus and other zoonotic diseases

Our modelling suggests that rates of WNV production decline with species mass, whereas rates and times related to the adaptive immune response do not vary systematically with mass. Taken together, these results provide an understanding of how epidemiological determinants vary with species body mass. The probability of infecting an uninfected mosquito increases with viraemia above a certain threshold for WNV [13]. This has also been observed for other flaviviruses such as dengue virus [55,56]). Above a threshold of viraemia of 10^5 PFU ml^{-1} of blood, a host is capable of infecting an uninfected mosquito and maintaining WNV in an enzootic cycle [13]. Hence, hosts that can sustain high viraemia above this threshold and for a longer duration are pathogen reservoirs [13]. The dependence of viraemia on host mass and phylogeny may give important insights into the role of mass and phylogeny on the spread of WNV. For example, larger non-passerine species, predicted to have lower viraemia, may be less competent as pathogen reservoirs.

Incorporating our within-host viral dynamics model into a between-host model that incorporates interactions between vectors and hosts, mosquito biting preferences [57–59] and bird abundance would be a fruitful avenue for future work. This analysis should consider the multiple ways in which host body size and phylogeny effect mosquito biting rates. Larger animals might have a greater probability of receiving an infective bite due to their larger surface area. However, the relationship between body size and the frequency of mosquito bites is complicated by several factors. For example, mosquitoes also have distinct biting preferences across species [57,58] and the ecological niche occupied by the species may influence biting rates. The effect of total body surface area may be limited given that the footpads, neck and the face are the only exposed regions in birds that do not have feathers and are most accessible for biting. Finally, body size affects population density which is approximately inversely proportional to host metabolism [60], and relative abundance of a species is another important factor in determining the spread of WNV. There are many unknowns and we speculate that models that combine species body size and other ecological factors with within-host modelling will be important in predicting outcomes of emerging diseases and formulating outbreak control strategies, as has been proposed for vector-borne diseases [61], dengue virus [55], Zika virus [62] and Ebola virus [63].

Our analysis suggests an important role for both species mass and host phylogeny in dictating epidemiological determinants such as the basic reproductive number, WNV production rate, peak viraemia in blood and host competency to infect mosquitoes. Our model is based on a principled analysis and gives a quantitative prediction for key epidemiological determinants and how they vary with species mass and phylogeny. The WNV production rate (p) and the basic reproductive number (R_0) decline with host body mass (figure 6). The relationship is even more significant within passerine species (figure 6).

We calculated a reservoir competence index that indicates the relative number of infectious mosquitoes that would be derived from feeding on these hosts (§3.6). Our reservoir competency index is a refinement on previously published indices [13] because they take time-varying viraemia and species relatedness (via the multi-level Bayesian model) into account. The reservoir competency indices we calculate are correlated with and approximately half of the estimates in Komar *et al.* [13] (figure 5). The mean reservoir competence for passerine species (0.8) is higher than that of non-passerines (0.2). Thus, our models provide insight into why some species (smaller passerine species) are pathogen reservoirs and some (larger non-passerine species) are potentially dead-end hosts for WNV.

WNV infection progresses along multiple scales: infection within hosts is related to dynamics of WNV spread between hosts by mosquito vectors. Coupling two different processes over multiple scales, from individual cells to epidemic spread in bird populations, is challenging and could yield valuable insights. Our predictions of host competency to infect mosquitoes (equation (3.8) and figure 5) can be coupled to models of WNV spread between multiple species. Such an

approach would help link within-host WNV dynamics to dynamics between hosts and may help produce accurate estimates of spread of WNV.

Hierarchical Bayesian models coupled with within-host viral dynamics models can leverage data from multiple species to infer how body size and host phylogeny affect viral dynamics. This approach can be applied to model other zoonotic diseases and multi-host pathogens [37], particularly other emerging viruses such as dengue [64,52], Ebola [63] and Zika virus [33,34,62].

Data accessibility. This article has no additional data.

Author's contributions. S.B. carried out the analysis and implementation, participated in the design of the study and drafted the manuscript; A.S.P. carried out the analysis, drafted the manuscript and coordinated the study. M.M. carried out the analysis, drafted the manuscript and coordinated the study. All authors gave final approval for publication.

Competing interests. We declare we have no competing interests.

Funding. M.M. acknowledges the support from a James S. McDonnell Foundation Complex Systems Scholar Award and National Science Foundation EF 1038682. Portions of this work were performed under the auspices of the US Department of Energy under contract DE-AC52-06NA25396 and supported by NIH grants R01-AI078881, R01-OD011095, R01-AI028433 and HHSN272201000055C to A.S.P. The authors acknowledge the UNM Center for Evolutionary and Theoretical Immunology (CETI) funded by the NIH IDeA Program of the National Center for Research Resources P20RR018754 for funding and support of this collaboration.

Acknowledgements. We thank Dr Nicholas Komar for sharing his experimental data with us and Drs Kimberly Kanigel Winner, Diane Oyen, John Cohen, Stephen Haben, Wenyun Zuo and Thomas Chittenden for fruitful discussions.

References

1. Grace D *et al.* 2012 Mapping of poverty and likely zoonoses hotspots. Report to the department for international development.
2. Woolhouse M, Taylor L, Haydon D. 2001 Population biology of multihost pathogens. *Science* **292**, 1109–1112. (doi:10.1126/science.1059026)
3. Cable J, Enquist B, Moses M. 2007 The allometry of host–pathogen interactions. *PLoS ONE* **2**, e1130. (doi:10.1371/journal.pone.0001130)
4. Kleiber M. 1947 Body size and metabolic rate. *Physiol. Rev.* **27**, 511–541.
5. Kleiber M. 1932 Body size and metabolism. *Hilgardia: J. Agric. Sci.* **6**, 315–353. (doi:10.3733/hilg.v06n11p315)
6. Peters RH. 1986 *The ecological implications of body size*, vol. 2. Cambridge, UK: Cambridge University Press.
7. West G, Brown J, Enquist B. 1997 A general model for the origin of allometric scaling laws in biology. *Science* **276**, 122–126. (doi:10.1126/science.276.5309.122)
8. Brown JH, Gillooly JF, Allen AP, Savage VM, West GB. 2004 Toward a metabolic theory of ecology. *Ecology* **85**, 1771–1789. (doi:10.1890/03-9000)
9. Wiegel FW, Perelson A. 2004 Some scaling principles for the immune system. *Immunol. Cell Biol.* **82**, 127–131. (doi:10.1046/j.0818-9641.2004.01229.x)
10. Althaus CL. 2015 Of mice, macaques and men: scaling of virus dynamics and immune responses. *Front. Microbiol.* **6**, 355. (doi:10.3389/fmicb.2015.00355)
11. Banerjee S, Moses M. 2010 Scale invariance of immune system response rates and times: perspectives on immune system architecture and implications for artificial immune systems. *Swarm Intell.* **4**, 301–318. (doi:10.1007/s11721-010-0048-2)
12. Banerjee S, Levin D, Koster F, Forrest S, Moses M. 2011 The value of inflammatory signals in adaptive immune responses. In *Artificial Immune Systems, 10th International Conference, ICARIS 2011* (eds Liò P, Nicosia G, Stibor T). Lecture Notes in Computer Science, vol. 6825, pp. 1–14. Berlin, Germany: Springer.
13. Komar N, Langevin S, Hinten S, Nemeth N, Edwards E, Hettler DL, Davis BS, Bowen RA, Bunning ML. 2003 Experimental infection of North American birds with the New York 1999 strain of West Nile virus. *Emerg. Infect. Dis.* **9**, 311–322. (doi:10.3201/eid0903.020628)
14. Rouder J, Lu J, Speckman P. 2005 A hierarchical model for estimating response time distributions. *Psychon. Bull. Rev.* **12**, 195–223. (doi:10.3758/BF03257252)
15. Pearl J. 2009 *Causality*. Cambridge, UK: Cambridge University Press.
16. Fei-Fei L, Perona P. 2005 A bayesian hierarchical model for learning natural scene categories. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (*CVPR'05*), vol. 2, pp. 524–531. Washington, DC: IEEE Computer Society.
17. Wikle C. 2003 Hierarchical bayesian models for predicting the spread of ecological processes. *Ecology* **84**, 1382–1394. (doi:10.1890/0012-9658(2003)084[1382:HBMFPT]2.0.CO;2)
18. Wikle C, Berliner L, Cressie N. 1998 Hierarchical bayesian space-time models. *Environ. Ecol. Stat.* **5**, 117–154. (doi:10.1023/A:1009662704779)
19. Huang Y, Wu H, Acosta EP. 2010 Hierarchical Bayesian inference for HIV dynamic differential equation models incorporating multiple treatment factors. *Biometrical J.* **52**, 470–486. (doi:10.1002/bimj.200900173)
20. Huang Y, Liu D, Wu H. 2006 Hierarchical Bayesian methods for estimation of parameters in a longitudinal HIV dynamic system. *Biometrics* **62**, 413–423. (doi:10.1111/j.1541-0420.2005.00447.x)
21. Han C, Chaloner K, Perelson A. 2002 *Bayesian analysis of a population HIV dynamic model*. *Case studies in Bayesian Statistics*, vol. 6 (eds C Gatsonis, RE Kass, A Carriquiry, A Gelman, D Higdon, DK Parker, I Vardinelli). Lecture Notes in Statistics, pp. 223–237. New York, NY: Springer.
22. Canini L, Carrat F. 2011 Population modeling of influenza A/H1N1 virus kinetics and symptom dynamics. *J. Virol.* **85**, 2764–2770. (doi:10.1128/JVI.01318-10)
23. Hayes EB, Komar N, Nasci RS, Montgomery SP, O'Leary DR, Campbell GL. 2005 Epidemiology and transmission dynamics of West Nile virus disease. *Emerg. Infect. Dis.* **11**, 1167–1173. (doi:10.3201/eid1108.050289a)
24. Deardorff E *et al.* 2006 Introductions of West Nile virus strains to Mexico. *Emerg. Infect. Dis.* **12**, 314–318. (doi:10.3201/eid1202.050871)
25. Komar N, Clark GG. 2006 West Nile virus activity in Latin America and the Caribbean. *Rev. Panam Salud Publica* **19**, 112–117. (doi:10.1590/S1020-49892006000200006)
26. Lanciotti RS *et al.* 1999 Origin of the West Nile virus responsible for an outbreak of encephalitis in the northeastern United States. *Science* **286**, 2333–2337. (doi:10.1126/science.286.5448.2333)
27. Samuel MA, Diamond MS. 2006 Pathogenesis of West Nile virus infection: a balance between virulence, innate and adaptive immunity, and viral evasion. *J. Virol.* **80**, 9349–9360. (doi:10.1128/JVI.01122-06)
28. Stafford M, Cao Y, Ho D, Corey L. 2000 Modeling plasma virus concentration and CD4+ T cell kinetics during primary HIV infection. *J. Theor. Biol.* **203**, 285–301. (doi:10.1006/jtbi.2000.1076)
29. Baccam P, Beauchemin C, Macken C, Hayden F, Perelson A. 2006 Kinetics of influenza a virus infection in humans. *J. Virol.* **80**, 7590–7599. (doi:10.1128/JVI.01623-05)
30. Rong L, Dahari H, Ribeiro RM, Perelson AS. 2010 Rapid emergence of protease inhibitor resistance in hepatitis C virus. *Sci. Transl. Med.* **2**, 30ra32. (doi:10.1126/scitranslmed.3000544)
31. Vaidya NK, Ribeiro RM, Miller CJ, Perelson AS. 2010 Viral dynamics during primary simian immunodeficiency virus infection: effect of time-dependent virus infectivity. *J. Virol.* **84**, 4302–4310. (doi:10.1128/JVI.02284-09)
32. Nowak MA, Bonhoeffer S, Hill AM, Boehme R, Thomas HC, McDade H. 1996 Viral dynamics in hepatitis B virus infection. *Proc. Natl Acad. Sci. USA* **93**, 4398–4402. (doi:10.1073/pnas.93.9.4398)
33. Osuna CE *et al.* 2016 Zika viral dynamics and shedding in rhesus and cynomolgus macaques. *Nat. Med.* **22**, 1448–1455. (doi:10.1038/nm.4206)

34. Best K, Guedj J, Madelain V, de Lamballerie X, Lim SY, Osuna CE, Whitney JB, Perelson AS. 2017 Zika plasma viral dynamics in nonhuman primates provides insights into early infection and antiviral strategies. *Proc. Natl Acad. Sci. USA* **114**, 8847–8852. (doi:10.1073/pnas.1704011114)
35. Diamond MS, Sitati EM, Friend LD, Higgs S, Shrestha B, Engle M. 2003 A critical role for induced IgM in the protection against West Nile virus infection. *J. Exp. Med.* **198**, 1853–1862. (doi:10.1084/jem.20031223)
36. Diamond MS, Shrestha B, Marri A, Mahan D, Engle M. 2003 B cells and antibody play critical roles in the immediate defense of disseminated infection by West Nile encephalitis virus. *J. Virol.* **77**, 2578–2586. (doi:10.1128/JVI.77.4.2578-2586.2003)
37. Banerjee S, Guedj J, Ribeiro RM, Moses M, Perelson AS. 2016 Estimating biologically relevant parameters under uncertainty for experimental within-host murine West Nile virus infection. *J. R. Soc. Interface* **13**, 20160130. (doi:10.1098/rsif.2016.0130)
38. The MathWorks Inc. 2011 MATLAB version 7.13.0.564.
39. Styer LM, Kent KA, Albright RG, Bennett CJ, Kramer LD, Bernard KA. 2007 Mosquitoes inoculate high doses of West Nile virus as they probe and feed on live hosts. *PLoS Pathog.* **3**, e132. (doi:10.1371/journal.ppat.0030132)
40. Martín P, Ruiz SR, Martínez del Hoyo G, Anjuère F, Vargas HH, López-Bravo M, Ardavín C. 2002 Dramatic increase in lymph node dendritic cell number during infection by the mouse mammary tumor virus occurs by a CD62L-dependent blood-borne DC recruitment. *Blood* **99**, 1282–1288. (doi:10.1182/blood.V99.4.1282)
41. Purtha W, Chachu K, Virgin H, Diamond MS. 2008 Early B-cell activation after West Nile virus infection requires alpha/beta interferon but not antigen receptor signaling. *J. Virol.* **82**, 10 964–10 974. (doi:10.1128/JVI.01646-08)
42. Cheeran MCJ, Hu S, Sheng WS, Rashid A, Peterson PK, Lokensgard JR. 2005 Differential responses of human brain cells to West Nile virus infection. *J. Neurovirol.* **11**, 512–524. (doi:10.1080/13550280500384982)
43. Clapham HE, Tricou V, Van Vinh Chau N, Simmons CP, Ferguson NM. 2014 Within-host viral dynamics of dengue serotype 1 infection. *J. R. Soc. Interface* **11**, 20140094. (doi:10.1098/rsif.2014.0094)
44. Carlin B, Louis T. 1997 Bayes and empirical bayes methods for data analysis. *Stat. Comput.* **7**, 153–154. (doi:10.1023/A:1018577817064)
45. Gilks W, Richardson S, Spiegelhalter D. 1995 *Markov Chain Monte Carlo in practice: interdisciplinary statistics*, vol. 2. London, UK: Chapman & Hall/CRC.
46. Gelfand A, Smith A. 1990 Sampling-based approaches to calculating marginal densities. *J. Am. Stat. Assoc.* **85**, 398–409. (doi:10.1080/01621459.1990.10476213)
47. Nowak M, May RM. 2000 *Virus dynamics: mathematical principles of immunology and virology*. Oxford, UK: Oxford University Press.
48. Banerjee S, Moses M. 2009 A hybrid agent based and differential equation model of body size effects on pathogen replication and immune system response. In *Artificial Immune Systems, 8th International Conference, ICARIS 2009* (eds Andrews PS, Timmis J, Owens NDL, Aickelin U, Hart E, Hone A, Tyrrell AM). Lecture Notes in Computer Science, vol. 5666, pp. 14–18. Berlin, Germany: Springer.
49. Moses M, Banerjee S. 2011 Biologically inspired design principles for scalable, robust, adaptive, decentralized search and automated response (RADAR). In *Proc. of the 2011 IEEE Conference on Artificial Life*, pp. 30–37. Washington, DC: IEEE Computer Society.
50. Olival KJ, Hosseini PR, Zambrana-Torrelio C, Ross N, Bogich TL, Daszak P. 2017 Host and viral traits predict zoonotic spillover from mammals. *Nature* **546**, 646–650. (doi:10.1038/nature22975)
51. Styer LM, Kent KA, Albright RG, Bennett CJ, Kramer LD, Bernard KA. 2007 Mosquitoes inoculate high doses of West Nile virus as they probe and feed on live hosts. *PLoS Pathog.* **3**, 1262–1270. (doi:10.1371/journal.ppat.0030132)
52. Banerjee S. 2013 Scaling in the immune system. PhD thesis, University of New Mexico, USA.
53. Banerjee S, Moses M. 2009 A hybrid agent based and differential equation model of body size effects on pathogen replication and immune system response. In *Lecture notes in computer science*, vol. 5666. *Artificial immune systems, 8th international conference, ICARIS 2009* (eds PS Andrews *et al.*), pp. 14–18. Berlin, Germany: Springer.
54. Banerjee S, Moses M. 2010 Modular RADAR: an immune system inspired search and response strategy for distributed systems. In *Lecture notes in computer science*, vol. 6209. *Artificial immune systems, 9th international conference, ICARIS 2010* (eds E Hart *et al.*), pp. 116–129. Berlin, Germany: Springer.
55. Clapham HE 2013 Modelling dengue infection dynamics and the impact of control measures. Ph.D. thesis, Imperial College London.
56. MinhNguyen N *et al.* 2013 Host and viral features of human dengue cases shape the population of infected and infectious Aedes aegypti mosquitoes. *Proc. Natl Acad. Sci. USA* **110**, 9072–9077. (doi:10.1073/pnas.1303395110)
57. Hassan HK, Cupp EW, Hill GE, Katholi CR, Klingler K, Unnasch TR. 2003 Avian host preference by vectors of eastern equine encephalomyelitis virus. *Am. J. Trop. Med. Hyg.* **69**, 641–647.
58. Kilpatrick AM, Kramer LD, Jones MJ, Marra PP, Daszak P. 2006 West Nile virus epidemics in North America are driven by shifts in mosquito feeding behavior. *PLoS Biol.* **4**, 606–610. (doi:10.1371/journal.pbio.0040082)
59. Farajollahi A, Fonseca DM, Kramer LD, Marm Kilpatrick A. 2011 'Bird biting' mosquitoes and human disease: a review of the role of Culex pipiens complex mosquitoes in epidemiology. *Infect. Genet. Evol.* **11**, 1577–1585. (doi:10.1016/j.meegid.2011.08.013)
60. Damuth J. 1981 Population density and body size in mammals. *Nature* **290**, 699–700. (doi:10.1038/290699a0)
61. Dobson A. 2004 Population dynamics of pathogens with multiple host species. *Am. Nat.* **164**, 64–78. (doi:10.1086/424681)
62. Ferguson NM, Cucunubá ZM, Dorigatti I, Nedjati-Gilani GL, Donnelly CA, Basáñez MG, Nouvellet P, Lessler J. 2016 Countering the Zika epidemic in Latin America. *Science* **353**, 353–354. (doi:10.1126/science.aag0219)
63. Castillo-Chavez C, Curtiss R, Daszak P, Levin SA, Patterson-Lomba O, Perrings C, Poste G, Towers S. 2015 Beyond Ebola: lessons to mitigate future pandemics. *Lancet Global Health* **3**, e354–e355. (doi:10.1016/S2214-109X(15)00068-6)
64. Clapham HE, Quyen TH, Kien DTH, Dorigatti I, Simmons CP, Ferguson NM. 2016 Modelling virus and antibody dynamics during dengue virus infection suggests a role for antibody in virus clearance. *PLoS Comput. Biol.* **12**, e1004951. (doi:10.1371/journal.pcbi.1004951)

BMC Research Notes

RESEARCH NOTE

Open Access

dsSurvival: Privacy preserving survival models for federated individual patient meta-analysis in DataSHIELD

Soumya Banerjee[1*†], Ghislain N. Sofack[2,3†], Thodoris Papakonstantinou[2,3], Demetris Avraam[4,5], Paul Burton[4], Daniela Zöller[2,3†] and Tom R. P. Bishop[1†]

Abstract

Objective: Achieving sufficient statistical power in a survival analysis usually requires large amounts of data from different sites. Sensitivity of individual-level data, ethical and practical considerations regarding data sharing across institutions could be a potential challenge for achieving this added power. Hence we implemented a federated meta-analysis approach of survival models in DataSHIELD, where only anonymous aggregated data are shared across institutions, while simultaneously allowing for exploratory, interactive modelling. In this case, meta-analysis techniques to combine analysis results from each site are a solution, but an analytic workflow involving local analysis undertaken at individual studies hinders exploration. Thus, the aim is to provide a framework for performing meta-analysis of Cox regression models across institutions without manual analysis steps for the data providers.

Results: We introduce a package (dsSurvival) which allows privacy preserving meta-analysis of survival models, including the calculation of hazard ratios. Our tool can be of great use in biomedical research where there is a need for building survival models and there are privacy concerns about sharing data.

Keywords: Survival analysis, Meta-analysis, Federated analysis

Introduction

Survival models are widely used in biomedical research for analyzing survival data [1]. These models help researchers compare the effect of exposures on mortality or other outcomes of interest [2]. The Cox proportional hazards model [3] is one of the most popular survival analysis models and was primarily developed to determine the importance of predictors in survival, by using covariate information to make individual predictions [4].

Achieving sufficient power in survival analysis usually requires large amounts of data from several sites or institutions. Multi-site analysis across studies with different population characteristics help us understand how diseases affect different populations and what it is about these populations that cause these differences. However, the number of cases at a single site is often rather small, making statistical analysis challenging. Also due to the sensitivity of individual-level biomedical data, ethical and practical considerations related to data transmission, and institutional policies, it may sometimes be difficult to share individual-level data [5].

In consortia, this issue is often addressed by manual analysis in each site, followed by a manual meta-analysis of the analysis results from the individual sites. This process is very time-consuming and error-prone, making exploratory analysis (e.g., for understanding different

†Soumya Banerjee and Ghislain N. Sofack are equal first authors

†Daniela Zöller and Tom R. P. Bishop are equal senior authors

*Correspondence: sb2333@cam.ac.uk

[1] Medical Research Council Epidemiology Unit, University of Cambridge School of Clinical Medicine, Cambridge, United Kingdom
Full list of author information is available at the end of the article

effect patterns in each site) impractical. As an alternative, the DataSHIELD framework can be used.

DataSHIELD is a framework that enables the remote and privacy preserving analysis of sensitive research data [6]. The framework is based on the programming language R [7, 8]. In each site, specifically requested aggregated anonymous analysis results can be requested, which are then combined in a central analysis server. The requirement is that the analysis be privacy preserving and be conducted across globally distributed cohorts.

We have implemented a meta-analysis approach based on the Cox-model in DataSHIELD using individual patient data that are distributed across several sites, without moving those data to a central site i.e., the individual-level data remain within each site and only non-disclosive aggregated data are shared. Our software package for DataSHIELD allows building of survival models and analyzing results in a federated privacy preserving fashion.

Remote federated meta-analysis allows the analysis to come to the data and enables multiple research groups to collate their data [7, 8]. This is an alternative to literature based meta-analysis since study variables and outcomes can be harmonised [9]. Our package offers considerable advantages over: (1) literature based meta-analysis, which suffers from publication bias as well as restricting the analytic endpoints you may wish to use; (2) central pooling of data, which provides important governance challenges and can engender privacy risks; and (3) asking researchers in each location to do local analyses based on a shared analysis plan, which all too often demands numerous emails, with repeated reminders, to disseminate analytic protocols and return results for meta-analysis, which is typically time-consuming and can be error prone. Our tool can be of great use in domains where there is a need for building survival models and there are privacy concerns about sharing data.

Main text

Basics of survival analysis

Survival analysis can be used to analyze clinical data if there are records of patient mortality and time to event data. The key quantity is a survival function:

$$S(t) = Pr(T > t) \tag{1}$$

where $S(t)$ is the survival function, t is the current time, and T is a random variable denoting time of death. $Pr()$ is the probability that the time of death is greater than time t i.e., the probability of surviving till time t.

The instantaneous hazard $[\lambda(t)]$ is the probability of death occurring within time period $[\,t, t + \delta t\,]$ given survival till time t. This is related to the survival function as follows:

$$S(t) = \exp\left(-\int_0^t \lambda(z)dz\right) \tag{2}$$

The proportional hazards model assumes that the effect of covariates is proportional to the hazard. This is modelled as follows using the hazard function $\lambda(t)$:

$$\lambda(t|X_i) = \lambda_0(t)\exp\left(\sum_{j=1}^{p}\beta_j X_{ij}\right) \tag{3}$$

where $\lambda_0(t)$ is called the baseline hazard and $\lambda(t)$ is the hazard at time t. β denotes the parameters and X_{ij} is the jth covariate for the i th subject. We aim to meta-analyze these log hazard ratios.

Implementation

DataSHIELD operates on a distributed architecture that only allows restricted computation. DataSHIELD has a client-server architecture (Fig. 1). There are multiple servers located on separate sites and there is a single analysis client. Assign functions in DataSHIELD perform computation and ultimately create objects that persist on the servers and are not shared with the analysis client. These server-side objects can then be used for subsequent computations. Aggregate functions in DataSHIELD perform computation on each site, check for disclosure risks, and send aggregated results back to a client. The results do not persist on the servers, but can be saved on the client. This is shown in Fig. 1.

The communication between the client and server for the survival models is shown in Fig. 2 for an assign function [*ds.Surv()* to create survival objects on the servers] and aggregate function [*ds.coxphSLMA()* to perform a meta-analysis of Cox regression models]. This shows an asynchronous mode of operation in DataSHIELD where multiple parties (sites) perform secure computation. The client-side package is called dsSurvivalClient and the server-side package is called dsSurvival.

The server-side package dsSurvival 1.0.0 contains the functions *SurvDS()* and *coxphSLMADS()*. These functions are configured to reside in modified R environments located behind a firewall at each institution and process the individual-level data at each distinct repository.

dsSurvivalClient contains the functions *ds.Surv()* and *ds.coxphSLMA()*. These functions reside on the conventional R environment of the analyst. The *ds.Surv()* (assign function) calls the server-side function *SurvDS()* to assign survival objects in each site. This can then be used as the response variable in the *ds.coxphSLMA()* (aggregate) function. The *ds.coxphSLMA()* function calls and controls the corresponding server-side functions *coxphSLMADS()* and performs the regression

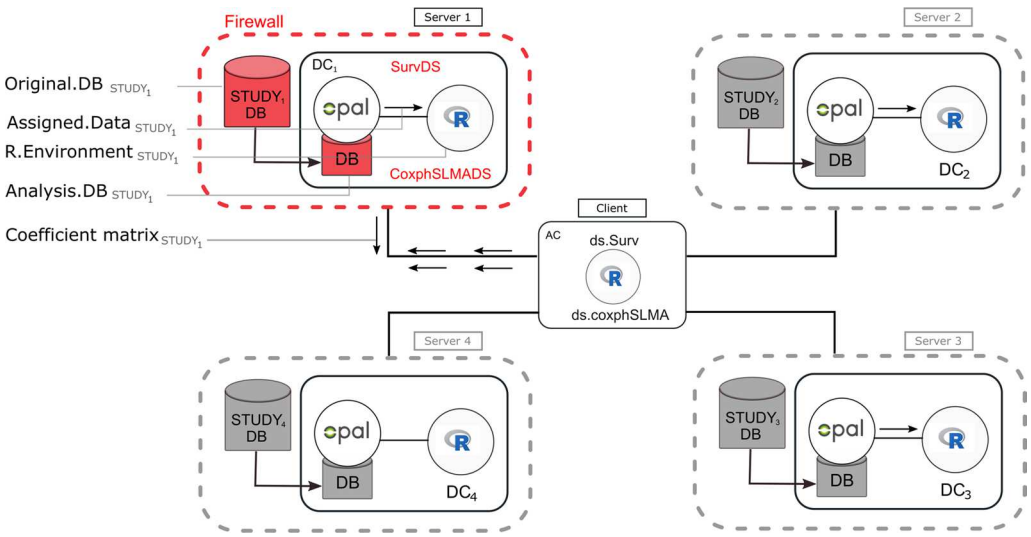

Fig. 1 Client-server architecture of DataSHIELD. The diagram shows four study sites/servers (DC) each having data stored in the Original.DB. The analyst (client) sends commands from the analysis computer (AC) to each study site to request the specific data (Assigned.Data) to be analyzed. This could be all the variables or specific variables stored in Analysis.DB. R commands are also sent from the analysis computer to every study telling it to create survival objects and fit the Cox proportional hazard model. Each site responds to instructions sent by creating the survival object and fitting the model. This fitting is carried out in the R environment of each study. The coefficient matrices, standard errors, and odds ratios from each site are then pooled and meta-analyzed using fixed optimization methods, and only non-disclosive statistics are returned to the analyst

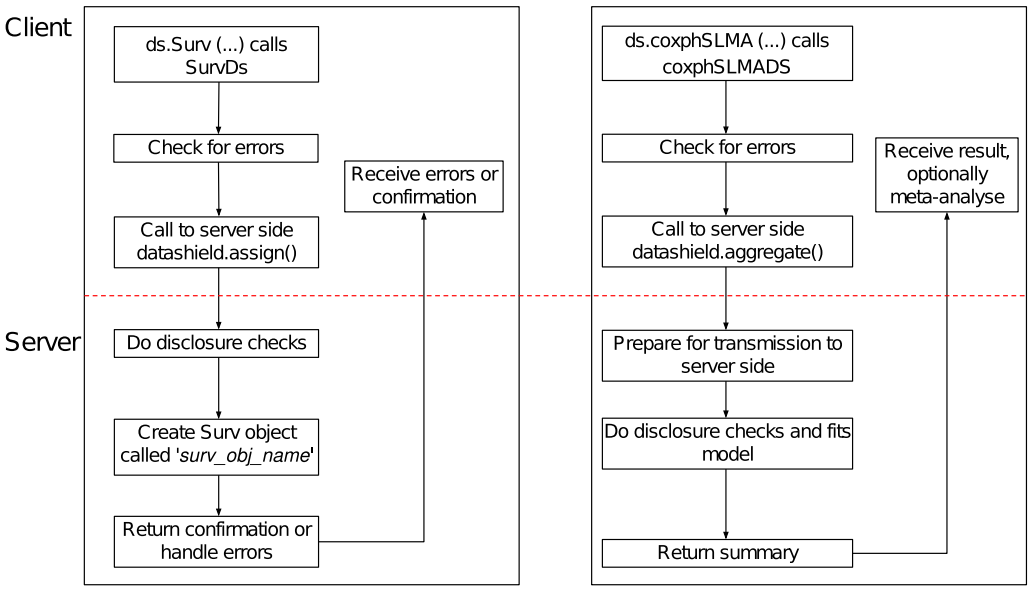

Fig. 2 Architecture of client and server side functions for building survival models in dsSurvival. Left panel: an assign function for creating a server-side survival object using *ds.Surv()*. Right panel: an aggregate function for a Cox proportional hazards model using *ds.coxphSLMA()*

analysis at different sites. These functions implement study-level meta-analysis (SLMA). The estimates from each site are combined and then pooled using fixed effects or random effects meta-analysis.

Computational pipeline and use case

We outline the development and code for implementing survival models (Cox regression) and meta-analysis of hazard ratios in our package (dsSurvival).

A tutorial in bookdown format is available here: https://neelsoumya.github.io/dsSurvivalbookdown/

In the following, we demonstrate the computational steps using synthetic data. The first step is using DataSHIELD to connect to the server and loading the survival data. We assume that the reader is familiar with these details. We show the steps using synthetic data. There are 3 data sets that are held on the same server but can be considered to be on separate servers/sites.

The variable *EVENT* holds the event information and variables *STARTTIME* and *ENDTIME* hold the time information. There is also age and gender information in variables named age and female, respectively. We will look at how age and gender affect survival time and then meta-analyze the hazard ratios. For details on how to setup the variables, please see the bookdown above.

Algorithm 1: Workflow for performing privacy preserving survival analysis.

```
# %Create survival object
dsSurvivalClient::ds.Surv(time='STARTTIME', time2='ENDTIME', event = 'EVENT',
 objectname='surv_object', type='counting')
# %use constructed Surv object in ds.coxph.SLMA()
coxph_model_full = dsSurvivalClient::ds.coxph.SLMA(formula = 'surv_object ~ D$age+D$female')
# %Summary of survival objects
# We can also summarize a server-side object of type survival::Surv() to provide a non-disclosive
 summary of the server-side object.
dsSurvivalClient::ds.coxphSummary(x = 'coxph_serverside')
# %Diagnostics for Cox proportional hazards models
# There are functions to test for the assumptions of Cox proportional hazards models.
dsSurvivalClient::ds.coxphSLMAassign(formula = 'surv_object ~ D$age+D$female', objectname =
 'coxph_serverside')
dsSurvivalClient::ds.cox.zphSLMA(fit = 'coxph_serverside')
```

The log-hazard ratios and their standard errors from each study can be found after running *ds.coxphSLMA()*. The hazard ratios can then be meta-analyzed using the *metafor* package [10]. Fig. 3 shows an example forest plot with meta-analysed hazard ratios. The plot shows the log hazard ratios corresponding to age in the survival model.

There are two options to generate the survival object. The analyst can generate it separately or inline [for example, by the following command: *dsSurvivalClient::ds.coxph.SLMA(formula = 'survival::Surv(time=SURVTIME,event=EVENT) ~ D$age+D$female')*]. If a survival object is generated separately, it is stored on the server and can be used later in an assign function [*ds.coxphSLMAassign()*]. This allows the survival model to be stored on the server and can be used later for diagnostics.

Preserving privacy and disclosure checks

Disclosure checks are an integral part of DataSHIELD and dsSurvival. dsSurvival leverages the DataSHIELD framework to ensure that multiple parties perform secure computation and only the relevant aggregated statistical details are shared. We disallow any Cox models where the number of covariate terms are greater than a fraction (default set to 20%) of the number of data points. The number of data points is the number of entries (for all patients) in the survival data. This fraction can be also be changed by the data custodian or administrator in DataSHIELD. We also deny any access to the baseline hazard function.

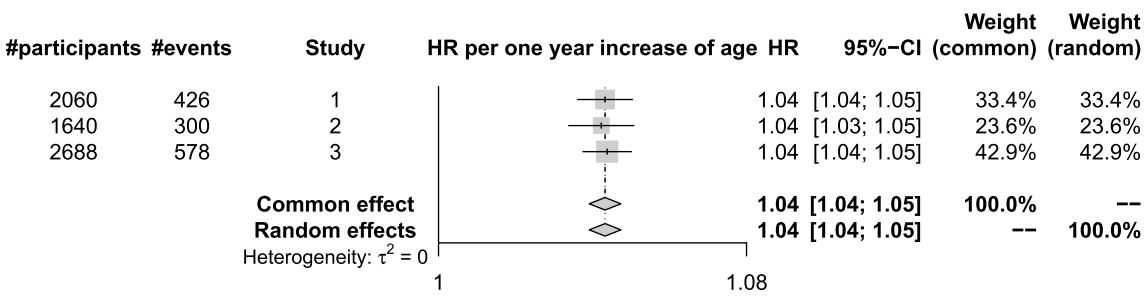

Fig. 3 A plot showing the meta-analyzed hazard ratios generated from dsSurvival. A Cox proportional hazards model was fit to synthetic data. The hazard ratios correspond to age in a survival model

Diagnostics for Cox proportional hazards models

We generate diagnostics for Cox models using the function dsSurvivalClient::ds.cox.zphSLMA(). These diagnostics can allow an analyst to determine if the proportional hazards assumption in Cox proportional hazards models is satisfied. If the p-values returned by dsSurvivalClient::ds.cox.zphSLMA() are greater than 0.05 for a covariate, then the proportional hazards assumption is likely correct for that covariate.

If the proportional hazards assumptions are violated, then the analyst may wish to modify the model. Modifications may include introducing strata or using time-dependent covariates.

Discussion and conclusion

dsSurvival is a DataSHIELD package for privacy preserving meta-analysis of survival data distributed across different sites. dsSurvival also performs federated calculation of hazard ratios. Its implementation relies exclusively on the distributed algorithm of the DataSHIELD environment. DataSHIELD facilitates important research particularly amongst institutions that are not allowed to transmit patient-level data to an outside server.

Previously building survival models in DataSHIELD involved using approximations like piecewise exponential models. This involves defining time buckets and is an additional burden on the researcher. A lack of familiarity with this approach also makes people less trusting of the results.

Previous work has looked at reducing the dimensions of a survival model and the reduced feature space model is then shared amongst multiple parties [11]. Survival analysis is also possible in DataSHIELD using dsSwissKnife [12]. However, our package offers advantages such as storing the model on the server-side, diagnostics, integration with client-side meta-analysis and future plans to add in more functionality such as survival curves.

We have released an R package for privacy preserving survival analysis in DataSHIELD. Our tool can be of great use in domains where there is a need for building survival models and there are privacy concerns about sharing data. We hope this suite of tools and tutorials will serve as a guideline on how to use survival analysis in a federated environment.

Limitations

Our approach implements study-level meta-analysis. This is a computationally faster approach but is also a limitation, especially if the units of meta-analysis are centres within a study. This may reduce the number of events per center and normality approximations implicit in two-stage meta-analysis may be violated. In the future we will implement functionality of iteratively fitting a single model across all studies. We will also develop plotting of privacy preserving survival curves and the ability to have time-dependent covariates in survival models. Our package also does not return Schoenfeld or Martingale residuals (due to privacy concerns), which are used as diagnostics for survival models. Finally in the future we will apply our package on real world data and solve any practical issues that arise.

Abbreviation

SLMA: Study level meta-analysis.

Acknowledgements

We acknowledge the help and support of the DataSHIELD technical team. We are especially grateful to Stephen Sharp, Elaine Smith, Stuart Wheater, Patricia Ryser-Welch, Sharreen Tan and Wolfgang Viechtbauer for fruitful discussions and feedback.

Author contributions

SB and GS carried out the analysis and implementation, participated in the design of the study and drafted the manuscript. TP, DA and PB gave critical comments and edited the manuscript. TB and DZ directed the study. All the authors read and approved the final manucript.

Funding

This work was funded by EUCAN-Connect under the European Union's Horizon 2020 research and innovation programme (Grant Agreement No. 824989). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript. The views expressed are those of the authors and not necessarily those of the funders.

Availibility of data and materials
This study does not generate any data. A tutorial in bookdown format with code, diagnostics, plots and synthetic data is available here: https://neelsoumya.github.io/dsSurvivalbookdown/. All code is available from the following repositories: https://github.com/neelsoumya/dsSurvivalClient/.https://github.com/neelsoumya/dsSurvival/. Our package can provide figures for downstream analysis. An example script for generating forest plots for hazard ratios is available here: https://github.com/neelsoumya/dsSurvival/blob/main/forestplot_FINAL.R.

Declarations

Ethics approval and consent to participate
No ethics approval and consent to participate was necessary.

Competing interests
All authors declare they have no competing interests to disclose.

Author details
[1]Medical Research Council Epidemiology Unit, University of Cambridge School of Clinical Medicine, Cambridge, United Kingdom. [2]Faculty of Medicine and Medical Center, Institute of Medical Biometry and Statistics, University of Freiburg, Breisgau, Germany. [3]Freiburg Center for Data Analysis and Modelling, University of Freiburg, Breisgau, Germany. [4]Population Health Sciences Institute, Newcastle University, Newcastle, United Kingdom. [5]Department of Public Health, University of Copenhagen, Copenhagen, Denmark.

Received: 5 January 2022 Accepted: 24 May 2022
Published online: 03 June 2022

References

1. Altman D, De Stavola B, Love S, Stepniewska K. Review of survival analyses published in cancer journals. Br J Cancer. 1995;72:511–8.
2. Machin D, Cheung YB, Parmar M. Survival analysis: a practical approach. Hoboken: Wiley; 2006.
3. Cox DR. Regression models and life-tables. J R Stat Soc. 1972;34:187–202.
4. Hartmann O, Schuetz P, Albrich WC, Anker SD, Mueller B, et al. Time-dependent cox regression: serial measurement of the cardiovascular biomarker proadrenomedullin improves survival prediction in patients with lower respiratory tract infection. Int J Cardiol. 2012;161:166–73.
5. Blasimme A, Fadda M, Schneider M, Vayena E. Data sharing for precision medicine: policy lessons and future directions. Health Aff. 2018;37:702–9.
6. Gaye A, Marcon Y, Isaeva J, LaFlamme P, Turner A, et al. Datashield: taking the analysis to the data, not the data to the analysis. Int J Epidemiol. 2014;43:1929–44.
7. Banerjee S, Bishop T. dsSynthetic: synthetic data generation for the DataSHIELD federated analysis system, DataSHIELD | DataSHIELD | Newcastle University. https://osf.io/tkxqm.
8. EUCAN Connect. https://www.eucanconnect.eu/. Accessed May 2022.
9. Pastorino S, Bishop T, Crozier SR, Granström C, Kordas K, et al. Associations between maternal physical activity in early and late pregnancy and offspring birth size: remote federated individual level meta-analysis from eight cohort studies. BJOG Int J Obstet Gynaecol. 2019;126:459–70.
10. Viechtbauer W. Conducting meta-analyses in R with the metafor Package. J Stat Softw. 2010;36:1–48.
11. Yu S, Fung G, Rosales R, Krishnan S, Rao RB, et al. Privacy-preserving cox regression for survival analysis. Proc ACM SIGKDD Int Conf Knowl Discov Data Min https://doi.org/10.1145/1401890.1402013. (2008).
12. Dragan I, Sparsø T, Kuznetsov D, Slieker R, Ibberson M, dsSwissKnife: An R package for federated data analysis. bioRxiv : 2020.11.17.386813. (2020).

Publisher's Note

npj | Schizophrenia

www.nature.com/npjschz

ARTICLE OPEN

Check for updates

A class-contrastive human-interpretable machine learning approach to predict mortality in severe mental illness

Soumya Banerjee[1]✉, Pietro Lio[2], Peter B. Jones[1,3] and Rudolf N. Cardinal[1,3]

Machine learning (ML), one aspect of artificial intelligence (AI), involves computer algorithms that train themselves. They have been widely applied in the healthcare domain. However, many trained ML algorithms operate as 'black boxes', producing a prediction from input data without a clear explanation of their workings. Non-transparent predictions are of limited utility in many clinical domains, where decisions must be justifiable. Here, we apply class-contrastive counterfactual reasoning to ML to demonstrate how specific changes in inputs lead to different predictions of mortality in people with severe mental illness (SMI), a major public health challenge. We produce predictions accompanied by visual and textual explanations as to how the prediction would have differed given specific changes to the input. We apply it to routinely collected data from a mental health secondary care provider in patients with schizophrenia. Using a data structuring framework informed by clinical knowledge, we captured information on physical health, mental health, and social predisposing factors. We then trained an ML algorithm and other statistical learning techniques to predict the risk of death. The ML algorithm predicted mortality with an area under receiver operating characteristic curve (AUROC) of 0.80 (95% confidence intervals [0.78, 0.82]). We used class-contrastive analysis to produce explanations for the model predictions. We outline the scenarios in which class-contrastive analysis is likely to be successful in producing explanations for model predictions. Our aim is not to advocate for a particular model but show an application of the class-contrastive analysis technique to electronic healthcare record data for a disease of public health significance. In patients with schizophrenia, our work suggests that use or prescription of medications like antidepressants was associated with lower risk of death. Abuse of alcohol/drugs and a diagnosis of delirium were associated with higher risk of death. Our ML models highlight the role of co-morbidities in determining mortality in patients with schizophrenia and the need to manage co-morbidities in these patients. We hope that some of these bio-social factors can be targeted therapeutically by either patient-level or service-level interventions. Our approach combines clinical knowledge, health data, and statistical learning, to make predictions interpretable to clinicians using class-contrastive reasoning. This is a step towards interpretable AI in the management of patients with schizophrenia and potentially other diseases.

npj Schizophrenia (2021)7:60 ; https://doi.org/10.1038/s41537-021-00191-y

INTRODUCTION

In this article we apply a recent development in machine learning, termed class-contrastive analysis, to the major public health problem of premature mortality in schizophrenia. Schizophrenia affects approximately 0.5% of the population[1]. It is a severe mental illness (SMI), along with bipolar affective disorder, personality disorders, and recurrent depressive disorder. Patients with schizophrenia or other SMI more broadly have substantially increased mortality and reduced life expectancy, often due to physical co-morbidities[2–5]. A challenge for clinical practice is therefore to identify patients at particularly high risk of adverse events (including premature death) and seek to intervene early.

Machine learning (ML) offers a potential route to risk prediction in the field of SMI as elsewhere. ML, a sub-field of artificial intelligence (AI), involves computer algorithms that automatically adjust in response to training data ('train themselves'). Their attractiveness relates to their ability to create predictive models from complex data sets with minimal human intervention, to a degree that may exceed the accuracy of classical statistical models such as logistic regression. ML achieves this through a variety of techniques. In a basic technique such as a multi-layer artificial neural network, for example, a layer of hidden nodes is trained by the algorithm to respond to weighted combinations of inputs; there may be further such layers responding to weighted combinations of the first layer, and these layers may interact recurrently. The "output" layer, giving the prediction or classification, responds to weighted combinations of preceding nodes. The algorithm seeks to minimize output prediction error. As a result, the output may predict accurately (if validated on independent data to avoid overfitting), but it may be very hard for a human to discern how the decision was reached. Clinically, it may be impractical to rely on such a black box predictor.

Here, we develop ML models of mortality in schizophrenia and apply the technique of class-contrastive reasoning to improve their explicability. Class-contrastive reasoning is a technique from the social sciences[6,7]: the contrast is to an alternative class of exemplars. An example of a class-contrastive explanation is: 'The selected patient is at high risk of mortality because the patient has dementia in Alzheimer's disease and has cardiovascular disease. If the patient did not have both of these characteristics, the predicted risk would be much lower.'

We apply an ML and class-contrastive framework to data on clinically relevant bio-social variables spanning physical health, mental health, personal history, and social predisposing factors. We collate this information from an electronic clinical records system and use clinician knowledge to transform them into features that are used to train an ML system (Fig. 1). We use the

[1]Department of Psychiatry, University of Cambridge, Cambridge, UK. [2]Department of Computer Science and Technology, University of Cambridge, Cambridge, UK. [3]Cambridgeshire and Peterborough NHS Foundation Trust, Cambridgeshire, UK. ✉email: sb2333@cam.ac.uk

Published in partnership with the Schizophrenia International Research Society

npj nature partner journals

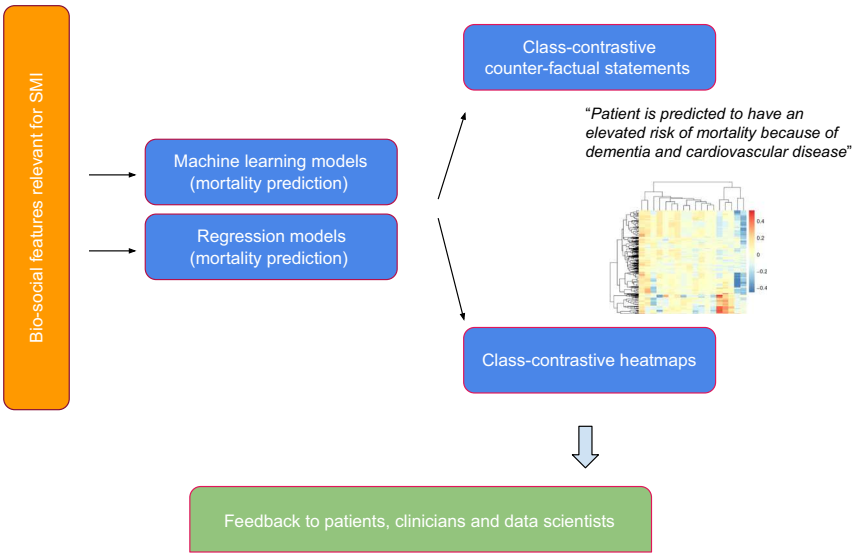

Fig. 1 Overview of approach. Bio-social factors that are relevant in severe mental illnesses (SMI) are derived from the electronic healthcare record system and used as inputs to statistical and machine learning algorithms. The algorithms are made interpretable using class-contrastive reasoning. The class-contrastive textual statements and heatmaps aid the understanding of models by domain experts such as clinicians, patients, and data scientists.

ML model to predict mortality and apply class-contrastive reasoning to explain the model.

We also use a visualization technique for machine learning models (class-contrastive heatmaps) that allows us to map the effect of changing a set of features.

Our approach can be helpful when explicit causal structure is modelled, and when there are a few features that are binary (categorical) in nature. Our approach may not be successful when there are continuous features or many features (hundreds). If features are hard to define and have to be discovered (for example, by semi-supervised techniques), our approach may not be helpful. Most of our features are binary (categorical) in nature and hence the class-contrastive approach could be applied successfully.

The class-contrastive approach may also be used to evaluate the practical limits of explainability of some models. For example, if a model has hundreds or thousands of features, it may be computationally intractable to exhaustively explore how changing combinations of these features affects the model output.

Our aim is not to advocate for a particular statistical model or black box model. Our objective is to give an example of how class contrastive reasoning can be used to explain black box models with binary categorical features in real-world electronic healthcare record data, in a disease of public health significance.

Our aim is not to exhaustively compare all possible statistical models but merely briefly survey and analyse some techniques. We note that our aim is not to demonstrate that some machine learning models can perform better than others.

We show a practical demonstration on a clinical dataset in a disease of public health relevance. We also outline the instances in which class-contrastive reasoning can be successfully applied to electronic health care record data. We suggest class-contrastive reasoning as a method to begin understanding ML and statistical models that have non-linearities. To the best of our knowledge this is the first application of this technique to real world electronic healthcare record data.

Our work is a step towards personalised medicine and interpretable AI in mental health and has the potential to be applicable more broadly in healthcare.

RESULTS

Summary of results

We used a range of statistical and ML techniques to predict mortality in patients with schizophrenia. Class-contrastive reasoning and class-contrastive heatmaps were used to generate human-oriented explanations of statistical and ML model predictions.

Abuse of alcohol and drugs, and a diagnosis of delirium were risk factors for mortality (across all our techniques). The use of antidepressants was associated with lower risk of death via all our techniques.

The machine learning models emphasized combinations of features (like Alzheimer's disease) with other co-morbidities. This highlights the role of co-morbidities in determining mortality in patients with SMI and the need to manage these co-morbidities.

Survival analysis and standardized mortality ratios

Our explainable machine learning techniques complement classical statistical analysis like survival models and standardised mortality ratios. In this section, we outline these approaches.

For survival analysis, we use age (feature scaled) and the bio-social features (as outlined in Subsection Data input to statistical algorithms) as input features for patients with schizophrenia. For patients with schizophrenia, we show the hazard ratios associated with each feature in Fig. 2 using a Cox proportional hazards model. The use of second-generation antipsychotics (SGA) and antidepressants was associated with reduced risk of death in patients with schizophrenia. Alcohol/substance abuse was associated with an elevated risk of death consistent with a previous study[8]. A diagnosis of delirium was similarly associated with increased mortality.

The standardized mortality ratio (SMR) for patients with schizophrenia was 7.4 (95% confidence interval: [5.5, 9.2]). This is consistent with SMRs reported in the UK[8].

Logistic regression models

We used a logistic regression model to predict mortality in patients with schizophrenia. We show the odds ratios and their confidence intervals in Fig. 3. Age, diagnosis of delirium and alcohol/substance abuse were associated with a high risk of death.

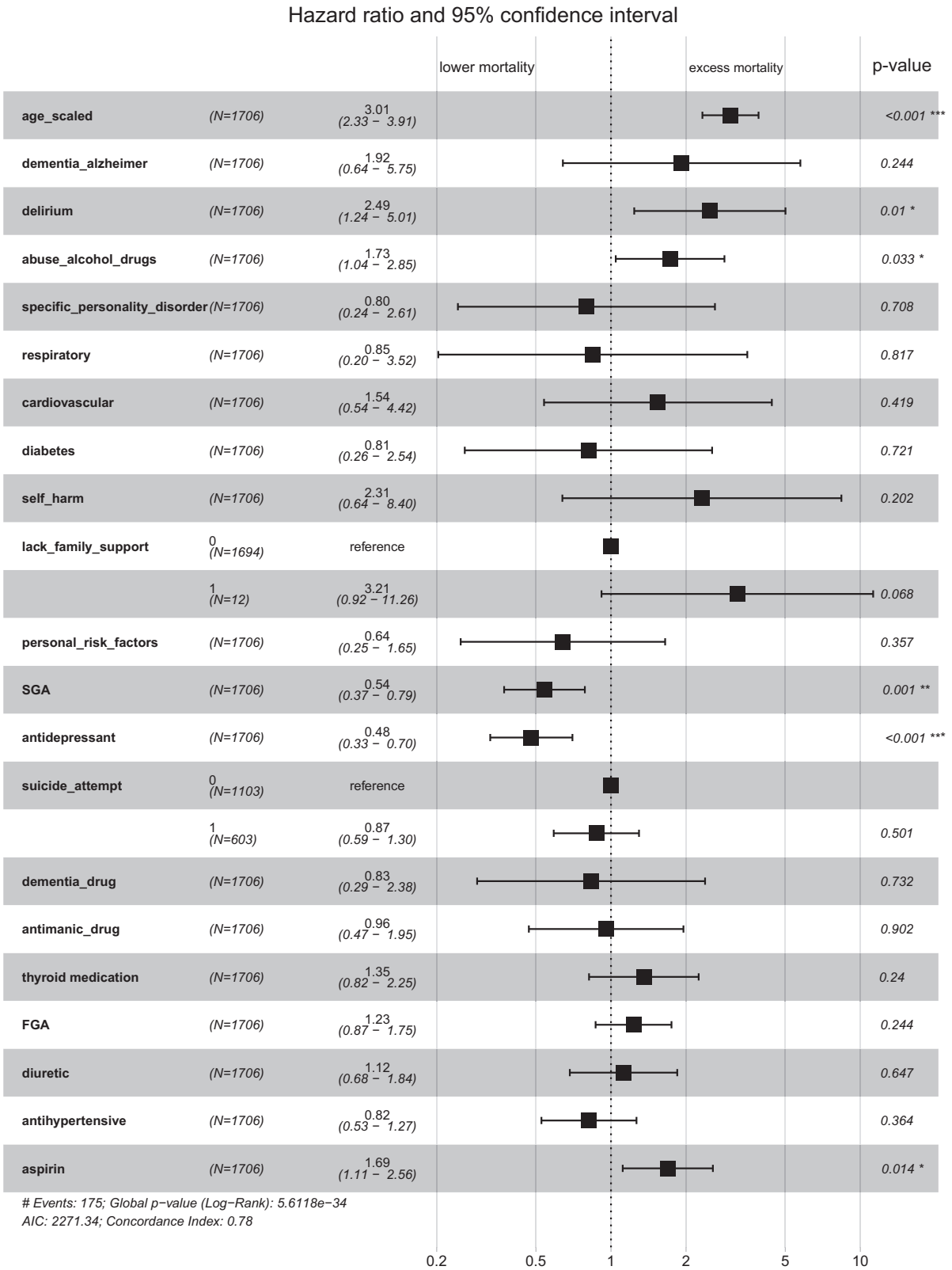

Fig. 2 Survival analysis for patients with schizophrenia using a Cox proportional hazards model. We show the hazard ratios associated with each feature and the confidence intervals. Use of second-generation antipsychotics (SGA) and antidepressants was associated with reduced risk of death in patients with schizophrenia. Alcohol/substance abuse and a diagnosis of delirium were associated with increased mortality. FGA: first-generation antipsychotics.

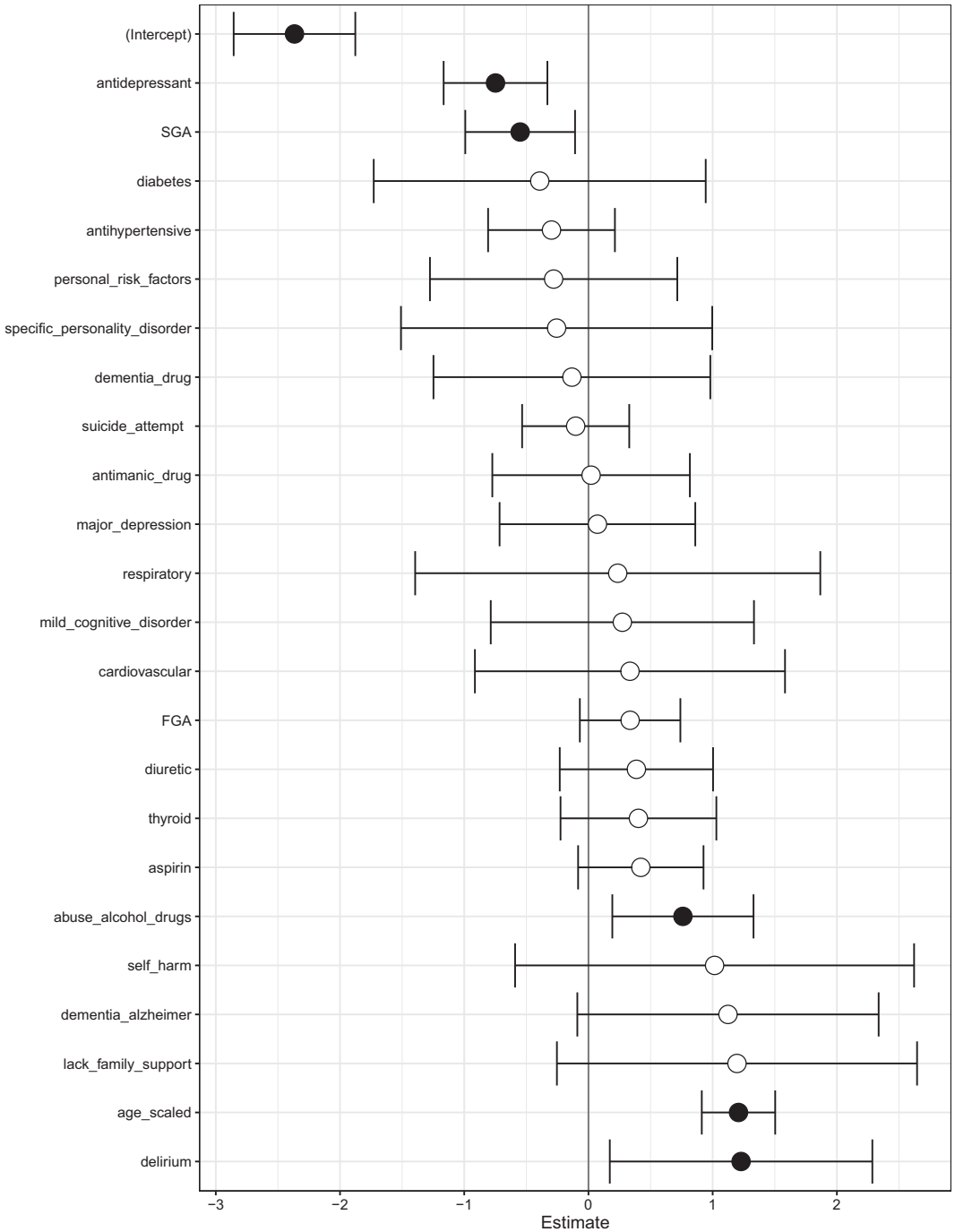

Fig. 3 Logistic regression model to predict mortality in patients with schizophrenia. Log odds ratio of features in a logistic regression model to predict mortality in patients with schizophrenia. Shown are confidence intervals and statistical significance (filled dark circles: p-value < 0.05, open circles: not significant). Age, alcohol/substance abuse and a diagnosis of delirium were associated with a high risk of death. The use of second-generation antipsychotics (SGA) and antidepressants were associated with reduced risk of death.

Use of second-generation antipsychotics and antidepressants were associated with a reduced risk of death.

Class-contrastive heatmaps and counter-factual statements for logistic regression

The class-contrastive explanatory technique is applicable to machine learning models and statistical models such as logistic regression. We first demonstrate our approach by using class-contrastive reasoning on the logistic regression model for predicting mortality. We show the amount of change (predicted by the trained logistic regression model on the test set) in the probability of death by changing one particular feature from 0 to 1 (in the test set). We visualize this using a heatmap (Fig. 4) where rows represent patients and columns represent features in the test set that have been changed. Predictions are made using the trained logistic regression model on the test set.

The class-contrastive heatmap shows patient-specific predictions. Predictions for individual patients are made in the following

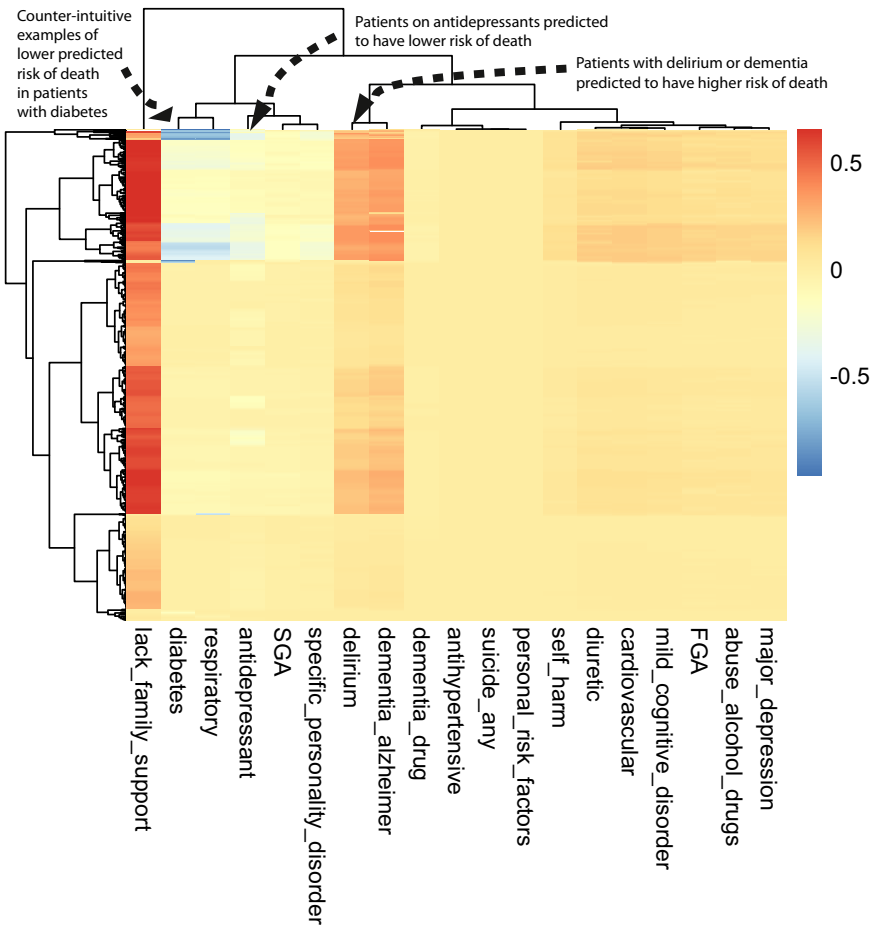

Fig. 4 Class-contrastive heatmap for the logistic regression model. Visualization of the amount of change predicted in the probability of death by setting a particular feature to 1 versus 0 (using a logistic regression model). Rows represent patients and columns represent features. Predictions are made on the test set using the trained logistic regression model. The arrows indicate groups of patients with low predicted risk of death (shown in blue on the heatmap) and high predicted risk of death (shown in red). The arrow on the top left indicates a group of patients having a counter-intuitive characteristic of having diabetes and still have low predicted risk of death. The second arrow on the top left indicates another group of patients on antidepressants. These patients are predicted (using the logistic regression model) to have a lower risk of death. There is a third group of patients with delirium or dementia in Alzheimer's disease (shown with an arrow) who are predicted to have a higher risk of death. The heatmap also shows a hierarchical clustering dendrogram which is performed using an Euclidean distance metric and complete linkage. FGA: first-generation antipsychotics, SGA: second-generation antipsychotics.

way: the trained logistic regression model makes a prediction for the probability of death based on the modified features as input. This process is repeated for each patient and each feature.

We observe that a diagnosis of delirium or dementia predisposes a group of patients towards a higher probability of predicted mortality (Fig. 4). Patients (with schizophrenia) who were taking antidepressants were less likely to die during the period observed (Fig. 4). The class-contrastive and counterfactual analysis suggests that antidepressants may be associated with lower mortality in a group of patients (Fig. 4).

The heatmap also highlights counter-intuitive predictions. For example, the heatmap suggests that there is a small sub-group of patients (Fig. 4: top left hand corner indicated with an arrow) who have diabetes and have a lower risk of death. The use of a probability scale illustrates that the effect of each predictor varies in terms of its effect on probability (according to the baseline probability determined by other variables); of course, in log odds terms, changes in a given predictor will have a constant effect across all subjects.

We note that the counter-intuitive observations we observe in the class-contrastive heatmaps (on the test set) may also be as a result of imbalances in the training set. For example, a particular binary feature may be 0 for 100 patients and 1 for 10 patients.

In order to address this, we can add synthetic training data with these imbalances and visualize the class-contrastive predictions on the test set. We can artificially introduce an imbalance (for example, add more zeros than ones to a binary feature) in the test set and training set, and then observe the class contrastive heatmaps.

We note that age is a predictor in all models that we use. However, the class-contrastive heatmaps do not include age. This is because the class-contrastive analysis changes features one at a time (or pairwise), and this can be achieved only for binary categorical features. Hence, the class-contrastive heatmaps show the effect of changing predictors on the model predicted probability of mortality, over and above the contribution of age.

Class-contrastive analysis for machine learning models

We used artificial neural networks to predict mortality in patients with schizophrenia. We performed class-contrastive analysis for this machine learning model (Fig. 5) to make it explainable.

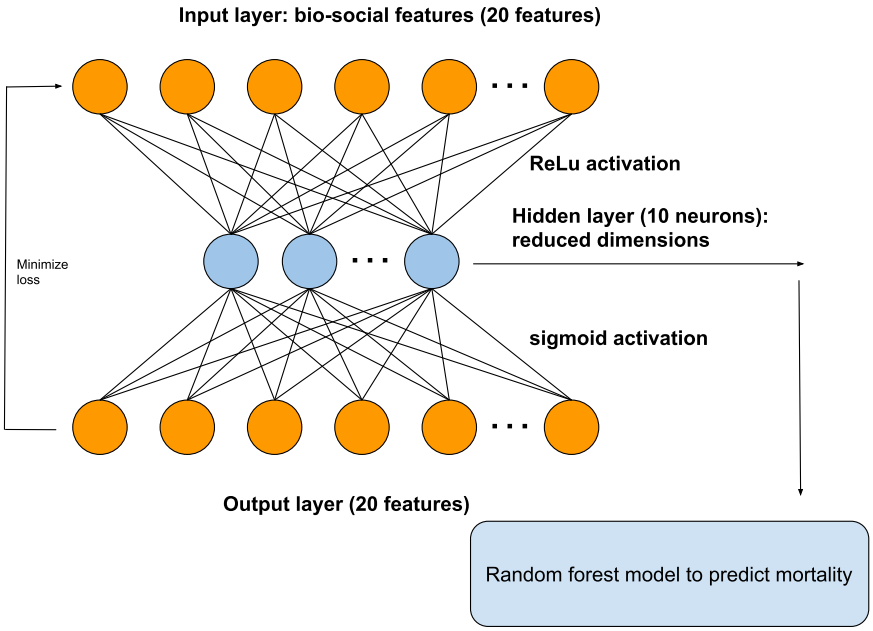

Fig. 5 Architecture of autoencoder. The autoencoder takes as input the bio-social features. The output layer is used to reconstruct the input. The hidden layer of the autoencoder is used for dimensionality reduction. We use the hidden layer as input to a random forest model to predict mortality. The hidden layer is composed of 10 neurons.

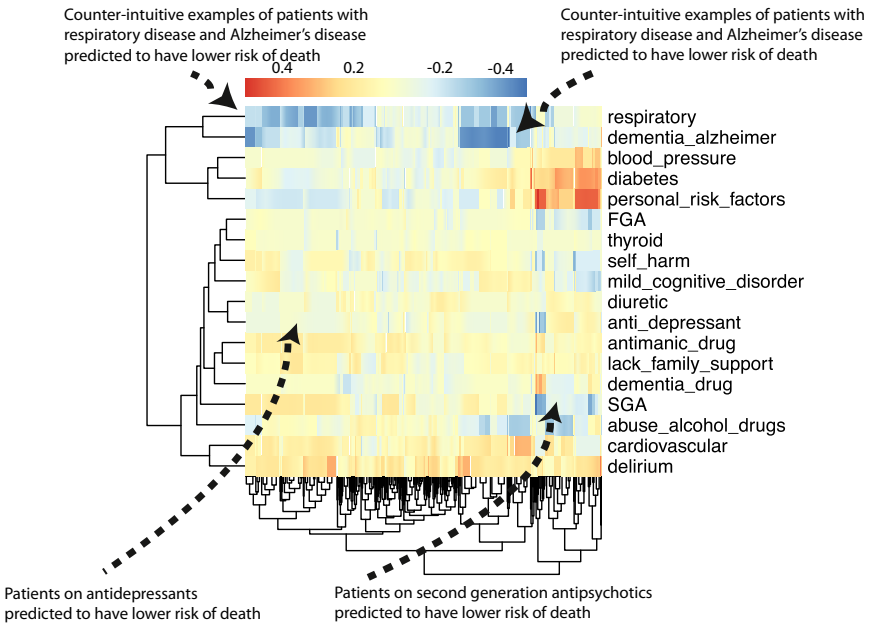

Fig. 6 Class-contrastive heatmap for the deep learning model. Visualization of the amount of change predicted in the probability of death by setting a particular feature to 1 versus 0. Predictions are made on the test set using a random forest model built on top of the autoencoder. Columns represent patients and rows represent features. The arrows at the top indicate counter-intuitive examples. If these patients had a respiratory disease or Alzheimer's disease, the model predicts low risk of death. The arrows at the bottom indicate a group of patients on antidepressants and SGA who are predicted to have low risk of death. The heatmap also shows a hierarchical clustering dendrogram, which is performed using an Euclidean distance metric and complete linkage. We note that even though we cluster the features (columns) we do not aim to imply any similarity between them. FGA: first-generation antipsychotics.

We first show a heatmap for a simple version of class-contrastive reasoning where we mutate only one feature at a time on the test set (Fig. 6). We show the amount of change (predicted by the trained model on the test set) in the probability of death by setting one particular feature to 1 versus 0. We visualize this using a heatmap as before, where rows represent patients and columns represent features. We note that even though we cluster the features, our aim is not to demonstrate any similarity between them.

The heatmap suggests there is a subgroup of patients in whom use or prescription of medications like second-generation antipsychotics (SGA) and antidepressants is associated with a

lower risk of death (Fig. 6). There is another subgroup of patients in whom personal risk factors (ICD-10 coded diagnosis; see 'Methods') are associated with increased risk of mortality.

The class-contrastive heatmaps also reveal counter-intuitive aspects of the data and model. Looking at the effect of individual features in isolation in Fig. 6, we observe small sub-groups of patients in whom having respiratory diseases or having Alzheimer's disease is associated with a lower risk of death (indicated with arrows in Fig. 6).

These counter-intuitive results may be due to the fact that the class-contrastive approach is sensitive to the training data and any imbalances in features. For example, a binary feature may have mostly zeros in the training set. This can lead to a counter-intuitive result on the test set. Correlations across features may also help explain these counter-intuitive results.

We show an additional representative class-contrastive heatmap for the ML model in the Supplementary section (Supplementary Fig. 1). This ML model was run using a different split of the training and test data. This heatmap is consistent with previous results (Fig. 6), with the exception that it shows SGA are associated with an increased probability of mortality (Supplementary Fig. 1, bottom left arrow). This is not consistent with previous results from the logistic regression model and survival analysis for the effect of SGA (Figs. 2 and 3).

Deep learning models combine input features to create higher-order representations using hidden layers. Features are also often correlated and there are non-linearities involved. To account for some higher-order (non-linear) correlations and to better highlight the combinations of features, we simultaneously change all possible combinations of two features from 0 to 1 (in the test set). Specifically, we set a particular combination of two features to 1 simultaneously (versus 0) in the test set. We then repeat this for all possible pairs of features in the test set. We visualize the change in model output on the test set in Fig. 7. This technique can be used to investigate the role of combinations of different features that deep learning models exploit to build higher-order representations.

We found combinations of cardiovascular disease and use of diuretics. Diuretic use was associated with lower risk of mortality in a group of patients with cardiovascular disease (shown in the blue region of the heatmap in the lower right-hand corner which is the region of greatest decrease in predicted probability of death) (Fig. 7). There are also combinations of delirium and dementia in Alzheimer's disease that predispose some patients towards greater risk of mortality (shown in the lower left region of the heatmap in red) (Fig. 7).

Other co-morbidities that are together associated with greater mortality in a sub-group of patients (Fig. 7) included: dementia in Alzheimer's disease with an additional coded diagnosis of cardiovascular disease, and dementia in Alzheimer's disease with a coded history of abuse of alcohol and drugs.

This highlights the role of co-morbidities in determining mortality in a sub-group of patients with SMI and the need for multiple conditions to be managed simultaneously in patients. A class-contrastive statement for one of these patients in this sub-group (Fig. 7) is: 'The selected patient is at high risk of mortality because the patient has dementia in Alzheimer's disease and has cardiovascular disease. If the patient did not have both of these characteristics, the predicted risk would be much lower.'

Our deep learning models emphasize combinations of different features. Therefore, as a very simple approximation, we also fit a more complex logistic regression model with interaction effects. We fit a logistic regression model with main effects and an interaction term between dementia in Alzheimer's disease and cardiovascular disease (Supplementary Fig. 2). The log-odds ratio for this interaction term is greater than 0 although it is not statistically significant. This may suggest that there is only a small sub-group of patients in whom dementia and cardiovascular disease co-occur and predispose towards an increased risk of death. Additional details are available in the Supplementary Section.

Performance

We show the predictive performance of each model in this section. The models we used to predict mortality are:

1. A logistic regression model with the bio-social features as input. The area under receiver operating curve (AUC) from the logistic regression model was 0.68 (95% confidence interval [0.65, 0.70]).
2. An autoencoder with the bio-social features as input. We then used the reduced dimensions from the autoencoder as input features to a random forest model. The predicted AUC from random forests built on top of the autoencoder-reduced dimensions was 0.80 (95% confidence interval [0.78, 0.82]).

We also use other statistical learning techniques to predict mortality and these are discussed in the Supplementary section (Section Additional analysis). We do not aim to exhaustively compare all possible statistical models but merely briefly survey and analyse some techniques. Our aim is to apply class-contrastive analysis to a machine learning model and show that in some scenarios the model predictions can be explained. We note that our aim is not to demonstrate that some machine learning models can perform better than others.

DISCUSSION

Mortality among patients with severe mental illnesses (SMI) is too often premature[3,4]. Routinely collected clinical data can help generate insights that can result in more effective treatment of these patients.

We used routinely collected clinical data in an observational study to answer questions of mortality in patients with schizophrenia. We implemented an interpretable computational framework for integrating clinical data in mental health and interrogating it with statistical and machine learning techniques.

Our framework starts with a database that is a knowledge repository of expertise. This database was created based on consultations with clinicians and maps low-level features (for example, medications such as simvastatin) to broader categories (for example, cardiovascular medication). These features are relevant for patients with schizophrenia and were used to predict mortality.

Our architecture captures clinical information on physical health, mental health, personal history and social predisposing factors to create a profile for a patient. We then used a number of statistical and machine learning techniques to predict mortality using these features.

We make our predictions interpretable by using class-contrastive reasoning[6,7]. Our approach has similarities to case-based reasoning[9] and analogy-based reasoning[10], where predictions are made based on similar patient histories cases. The approach presented here complements other techniques like Shapley explanations that are used to improve the interpretability of machine learning models. Further work is required to ensure the findings from schizophrenia generalise to other types of SMI.

We used a range of statistical and machine learning techniques to predict mortality in patients with schizophrenia. Since machine learning models may also be difficult to explain, we make them explainable using class-contrastive reasoning and class-contrastive heatmaps.

In patients with schizophrenia, abuse of alcohol and drugs, and a diagnosis of delirium were risk factors for mortality (across all techniques). Use or prescription of antidepressants and

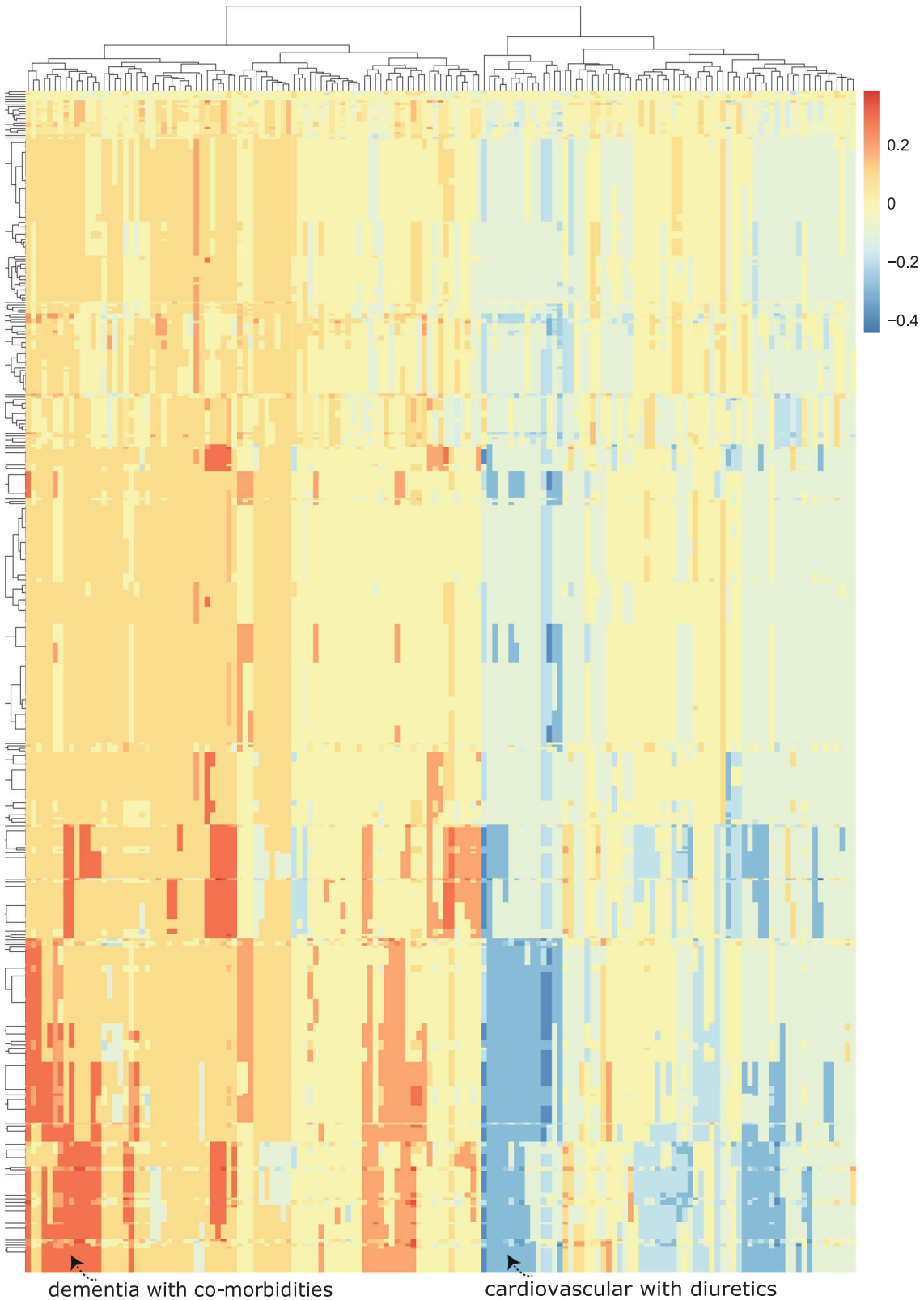

Fig. 7 Advanced class-contrastive heatmap for the deep learning model. Class-contrastive heatmap for deep learning models showing the effect of combinations of features. Visualization of the amount of change predicted in the probability of death by setting a particular combination of two features to 1 simultaneously (versus 0). Rows represent patients and columns represent feature groupings (all combinations of two features). Predictions are made on the test set using a random forest model built on top of the autoencoder. Delirium and dementia in Alzheimer's disease seem to predispose some patients towards greater risk of mortality (shown in the lower left region of the heatmap in red). Diuretics appear to be associated with lower mortality in a group of patients with cardiovascular disease (shown in the blue region in the lower right-hand corner of the heatmap). The heatmap also shows a hierarchical clustering dendrogram which is performed using an Euclidean distance metric and complete linkage.

second-generation antipsychotics (SGA) were associated with lower mortality in our logistic regression and survival models. However, use or prescription of SGA was associated with an increased probability of mortality in one of our ML models.

The logistic regression model predicted that Alzheimer's disease is a risk factor for mortality. The deep learning model emphasized Alzheimer's disease in combination with other co-morbidities. This highlights the role of co-morbidities in determining mortality in patients with SMI and the need to manage them.

The class-contrastive and survival analysis suggest that antidepressants are associated with lower mortality in a group of patients with schizophrenia. Alcohol/substance misuse was consistently associated with elevated mortality, suggesting the requirement to address the needs of so-called "dual diagnosis" patients (with SMI and comorbid substance misuse) as part of a strategy to improve life expectancy in patients with SMI.

The association between delirium and excess mortality is notable but not unexpected[2–4]. A weakness of the current family of models is their lack of temporal structure (for example, consideration of the time between delirium and death) but this finding serves to emphasize that delirium should not be taken lightly.

The association of antidepressant use with reduced mortality was unexpected but consistent across analytical methods. Our data do not support a mechanistic interpretation (for example, mode of death is not recorded in these structured clinical records) but this question would bear further investigation.

Illicit substance abuse and lack of family involvement was associated with increased risk of mortality[8]. Alcohol/substance abuse was also pointed out as a critical factor in our class-contrastive reasoning analysis and survival analysis. Provisioning of family support and involving family members and carers could be part of health management plans[11].

We hope that some of these bio-social factors can be targeted therapeutically by either patient-level interventions (like provisioning of family support[11]) or service-level improvements[12].

Overall, we observed that abuse of alcohol and drugs and a diagnosis of delirium are risk factors for mortality (in both logistic regression models and survival models). The use of SGA and antidepressants were associated with lower mortality from both our logistic regression models and survival models. This may be important given that some clinicians may hesitate to prescribe given what is known about short- to medium-term side effects of these drugs that include adverse impact on cardiovascular risk profiles. While our findings on this and other points is not conclusive evidence of causality, it is in accord with observational clinical data at the national level[13].

The machine learning model emphasized (for example) Alzheimer's disease along with other co-morbidities (Fig. 7). This highlights the role of co-morbidities in determining mortality in patients with SMI and points to the need for multiple conditions to be treated simultaneously in patients. This also suggests that a pragmatic trial of robust management of co-morbidities may be justified.

Interpretability is a major design consideration of machine learning algorithms applied in healthcare. We made our predictions interpretable by using class-contrastive reasoning and counterfactual statements[6].

This approach has the capability to make some black-box models explainable, which might be very useful for clinical decision support systems. We demonstrate the approach here using logistic regression and artificial neural networks. These techniques could ultimately be used to build a conversational AI that could explain its predictions to a clinician.

Our work can also be used to make clinical decision support systems. This may lead to automated alerts in electronic healthcare record systems, after thorough validation in follow-up studies.

Our study is observational in nature and we do not imply causation. Our data is a naturalistic sample from clinics and should not be used to alter clinical practice. Our aim is to raise hypotheses that will need to be tested in randomized controlled trials.

There may also be other unknown confounds and hence causal conclusions cannot be drawn from an observational study. For example, a drug that is associated with better outcomes may be preferred by clinical teams, and a drug that is associated with poorer outcomes may still be prescribed for severely ill patients because it is perceived to be effective.

There may also be under-coding of schizophrenia diagnoses. The data were from a secondary care mental health service provider and may miss important risk factors coded in primary care.

Psychiatric diagnoses are challenging and there can be potential issues related to the reliability of diagnostic categories in SMI. Sampling bias is another issue in real world electronic healthcare record data. For example, it is possible that only the most severely ill patients seek clinical help and/or get referred to secondary care. Hence the data may reflect a category of patients who are more severely ill.

Our data also lacks temporal structure, which is likely to be important in determining progression of disease. The current work relates observable features to the risk of death within the observation period. Clearly this is not as satisfactory as a model that predicts a time-based risk. It would be expected that the temporal risk conferred by different features would vary—for example, diabetes increases cardiovascular risk over decades, whereas delirium is often associated with critical illness and may be associated with an elevated mortality risk that is very immediate or proximal. A comprehensive model might involve autodiscovery of those temporal risk factors, at the price of a considerable increase in model complexity. This will require building more complex recurrent neural network models like long short-term memory models (LSTM), which will require even more data.

Important readouts like statistical significance cannot be judged from the class-contrastive heatmaps. For example, lack of family support appeared to be associated with higher mortality in the class-contrastive heatmap for the logistic regression model (Fig. 4). However, this association was not statistically significant, even though the odds ratio for lack of family support was greater than 1 in a logistic regression model (Fig. 3).

We combined several medications into the category of second-generation antipsychotics, which itself consists of a heterogenous group of medications[13], and made other simplifications in our treatment of medications.

Because of heterogeneity in training data and correlations across features, reproducibility of heatmaps is a limitation. We show an additional representative example in Supplementary Fig. 1. There are a few differences between these heatmaps (Supplementary Fig. 1 and Fig. 6). This heatmap is consistent with previous results (Fig. 6), with the exception that it shows SGA are associated with an increased probability of mortality (Supplementary Fig. 1, bottom left arrow). This is not consistent with previous results from the logistic regression model and survival analysis for the effect of SGA (Figs. 2 and 3). Reconciling these results will require additional analysis and validation in an independent cohort with more patients.

Our results suggest that the class-contrastive approach is sensitive to the training data and any imbalances in features. For example, a particular binary feature may be 0 for 100 patients and 1 for 10 patients. One way to determine this sensitivity is to artificially introduce more zeros and then observe the class contrastive heatmaps.

It is possible that the counter-intuitive observations we see in the class-contrastive heatmaps (on the test set) are likely as a

result of such imbalances in the training set. Because of this, reproducibility of heatmaps is a limitation of our approach.

Our approach can be helpful when explicit causal structure is modelled, and when there are binary (categorical) features and a few features which can be modified at a time. We account for correlations between features by modifying all pairs of features at a time and then observing the effect on model predictions (Fig. 7). However, this approach can become computationally challenging for higher combinations of features (all triples, quadruples, all possible combinations), or as the number of features increase.

In conclusion, our framework combines bio-social factors relevant for SMI with statistical learning, and makes them interpretable using class-contrastive techniques. Our work suggests that medications like antidepressants were associated with a reduced risk of death in a group of patients with schizophrenia. Abuse of alcohol and drugs, and a diagnosis of delirium were risk factors for death.

Our machine learning models highlight the role of co-morbidities in determining mortality in patients with SMI and the need to manage them. We hope that some of these bio-social factors can be targeted therapeutically by either patient-level or service-level interventions.

We complement explainable machine learning techniques with classical statistical analysis like logistic regression, survival models, and standardised mortality ratios. This may be a prudent and pragmatic approach for building explainable models in healthcare. We admit that the distinction between ML models and classical statistical models (like logistic regression) is artificial. Models lie on a continuum and a pragmatic approach towards explainable AI would combine and contrast all of these techniques.

The approach of combining explainable techniques and clinical knowledge with machine learning approaches may be more broadly applicable when data scientists need to work closely with domain experts (clinicians and patients).

Our approach combines clinical knowledge, health data, and statistical learning, to make predictions interpretable to clinicians using class-contrastive reasoning. We view our work as a step towards interpretable AI and personalized medicine for patients with SMI and potentially other diseases.

METHODS

Overview of Methods

We give a brief overview of our approach in this section. Our approach is summarised in Fig. 1.

1. We take de-identified data from an electronic patient record system for mental health.
2. We define a set of high-level features that are in this example time independent. These include age, diagnostic categories (time-independent coded diagnosis at any point during the study period), and medication categories (time-independent prescription of or use of medications). We also include bio-social factors that are important in SMI like information on mental health diagnosis, relevant risk history such as a prior suicide attempt, substance abuse, and social factors such as lack of family support.
3. We use these features to predict death during the time of observation.
4. We use classical statistical models including logistic regression, survival models, and standardised mortality ratios.
5. We then fit machine learning models, comparing predictive accuracy to the classical statistical models.
6. Class-contrastive heatmaps are used to visualize the explanations of the statistical models and machine learning predictions. The corresponding class-contrastive statements also aid human interpretation.

Mental health clinical record database

We used data from the Cambridgeshire and Peterborough NHS Foundation Trust (CPFT) Research Database. This comprises electronic healthcare records from CPFT, the single provider of secondary care mental health services for Cambridgeshire and Peterborough, UK, an area in which ~856,000 people reside. The records are de-identified using the CRATE software[14] under NHS Research Ethics approval (12/EE/0407, 17/EE/0442). The CPFT Research Database operates under UK NHS Research Ethics approvals (REC references 12/EE/0407, 17/EE/0442; IRAS project ID 237953).

Data included patient demographics, mental health and physical co-morbidity diagnoses: these were derived from coded ICD-10 diagnoses and analysis of free text through natural language processing (NLP) tools[15,16].

Dates of death are automatically updated via the National Health Service (NHS) Spine. We considered all patients with coded diagnoses of schizophrenia who had records in the electronic healthcare system from 2013 onwards. There were a total of 1706 patients diagnosed with schizophrenia defined by coded ICD-10 diagnosis (diagnosis code F20). We note there is under-coding of schizophrenia.

Medicine information on prescribed drugs

We extracted medicine information for each patient by using natural language processing on clinical free text data using the GATE software[15,17].

Population mortality data

Population mortality data for England and Wales were used from the Office for National Statistics (ONS)[18].

Data input to statistical algorithms

The features fed in to our statistical and machine learning algorithms included age, gender, high-level diagnosis categories, and medication categories. We also included other bio-social factors important in SMI. All these features are used to predict mortality. The full list of features was as follows:

1. High-level medication categories were created based on domain-specific knowledge from a clinician [RNC]. These medication categories are:
   second-generation antipsychotics (SGA: clozapine, olanzapine, risperidone, quetiapine, aripiprazole, asenapine, amisulpride, iloperidone, lurasidone, paliperidone, sertindole, sulpiride, ziprasidone, zotepine); first-generation antipsychotics (FGA: haloperidol, benperidol, chlorpromazine, flupentixol, fluphenazine, levomepromazine, pericyazine, perphenazine, pimozide, pipotiazine, prochlorperazine, promazine, trifluoperazine, zuclopenthixol); antidepressants (agomelatine, amitriptyline, bupropion, clomipramine, dosulepin, doxepin, duloxetine, imipramine, isocarboxazid, lofepramine, maprotiline, mianserin, mirtazapine, moclobemide, nefazodone, nortriptyline, phenelzine, reboxetine, tranylcypromine, trazodone, trimipramine, tryptophan, sertraline, citalopram, escitalopram, fluoxetine, fluvoxamine, paroxetine, vortioxetine and venlafaxine); diuretics (furosemide); thyroid medication (drug mention of levothyroxine); antimanic drugs (lithium) and medications for dementia (memantine and donepezil).
2. Relevant co-morbidities we included were diabetes (inferred from ICD-10 codes E10, E11, E12, E13 and E14 and any mentions of the drugs metformin and insulin), cardiovascular diseases (inferred from ICD-10 diagnoses codes I10, I11, I26, I82, G45 and drug mentions of atorvastatin, simvastatin and aspirin), respiratory illnesses (J44 and J45) and anti-hypertensives (mentions of the drugs bisoprolol and amlodipine).
3. We included all patients with a coded diagnosis of schizophrenia (F20). For these patients with schizophrenia, we also included any additional coded diagnosis from the following broad diagnostic categories: dementia in Alzheimer's disease (ICD-10 code starting with F00), delirium (F05), mild cognitive disorder (F06.7), depressive disorders (F32, F33) and personality disorders (F60).
4. We also included relevant social factors: lack of family support (ICD-10 chapter code Z63) and personal risk factors (Z91: a code encompassing allergies other than to drugs and biological substances, medication noncompliance, a history of psychological trauma, and unspecified personal risk factors); alcohol and substance abuse (this was inferred from ICD-10 coded diagnoses of Z86.4, F10, F12, F17, F19 and references to thiamine, which is prescribed for alcohol abuse). Other features included are self-harm (ICD-10 codes T39, T50, X60, X61, X62, X63, X64, X78 and Z91.5), non-compliance and personal risk factors (Z91.1), referral to a crisis team at CPFT (recorded in the electronic healthcare record system) and any prior suicide attempt (in the last 6 months or any time in the past) coded in structured risk assessments.

These broad categories constituted our representation of simplified clinician-based knowledge. We used these features (including age of the patient) to predict whether a patient died any time during the time period observed (from first referral to CPFT to the present day). We did not attempt to predict the risk of dying, for instance, 1 year after first referral to CPFT. The features we used to predict mortality were also time-independent. This represents a simplified time-independent model. More detailed modelling would include temporal effects of such predictors.

Age was a predictor in all our models, including survival models. We consider time of death and time of feature collection. The observed outcome (death) was binary and this is the outcome the models are predicting but models do so via a continuous variable related to risk/probability, so this is simultaneously predicted. Our model predictions, if independently validated in another clinical setting, could be converted in to a risk or probability.

All our models, including the machine learning model, include age as a predictor. However, the class-contrastive analysis and the class-contrastive heatmaps do not include age since the feature changes (one at a time or pairwise) can be achieved only for binary (categorical) features. Hence, the class-contrastive heatmaps show the effect of changing predictors on the model prediction, over and above the contribution of age.

Data pre-processing

Diagnostic codes were based on the International Classification of Diseases (ICD-10) coding system[19]. Age of patients was normalised (feature scaled) by subtracting the mean age from the age of each patient and then dividing by the standard deviation. All categorical variables, such as diagnosis and medications (described above), were converted using a one-hot encoding scheme. This is explained in detail in the Supplementary section.

Machine learning and statistical techniques

We performed logistic regression using generalized linear models[20,21]. We used age (feature scaled) as a continuous predictor. There are categorical features (medications, co-morbidities and other social and personal predisposing factors) that were encoded using a one-hot representation.

For our machine learning approach, we used artificial neural networks (autoencoders) to integrate data from different sources giving a holistic picture of mental health, physical health and social factors contributing to mortality in SMI. We use the same set of features for all algorithms.

Artificial neural networks are composed of computational nodes (artificial neurons) that are connected to form a network. Each artificial neuron performs a simple computation (much like logistic regression). The neurons are organised in layers. The input layer takes in the input features, transforms them, and passes it to one or more intermediate layers called hidden layers. The hidden layer performs further transformations and passes the result to the output layer. The final output layer is used to make a prediction (in this case, about mortality).

The autoencoder is a type of artificial neural network that also performs dimensionality reduction since the hidden layer has fewer neurons than the input layer[22]. In our framework, the reduced dimensions of the autoencoder (output of the hidden layer) were used as input to a random forest model to predict mortality (Fig. 5). Random forests are machine learning models that build collections of decision trees[23]. Each decision tree makes a prediction after making a series of choices based on the input data. These decision trees are combined to build a collection (forest) that together has better predictive ability than a single tree.

We split the data into a training set (50%), validation set (25%) and test set (25%). We performed 10-fold cross-validation and regularization to penalize for model complexity. The architecture is summarised in Fig. 5. We used the following models to predict mortality:

1. Logistic regression model with all the original input features;
2. An autoencoder with the bio-social features as input. We then use the reduced dimensions from the autoencoder as input features to a random forest model (Fig. 5).

Machine learning methods. We used an artificial neural network, called an autoencoder, to integrate data from different sources and predict mortality. The input features are age (normalised), gender, diagnosis categories, lifestyle risk factors, social factors and medication categories. We used the same set of features for all algorithms.

Categorical features (such as medication categories) are encoded using a one-hot representation. This involves taking a vector that is as long as the number of unique values of the feature. Each position on this vector corresponds to a unique value that the categorical feature can take. Whenever a categorical feature (say, did a patient take cardiovascular medication) takes on a particular value (say True), we place a 1 ('hot') corresponding to that position on the vector and 0 everywhere else.

We show the architecture of the autoencoder in Fig. 5. The autoencoder is an artificial neural network with an input layer, hidden layer and an output layer. The input layer takes in the bio-social features. The output layer is used to reconstruct the input. The hidden layer of the autoencoder is used for dimensionality reduction.

The autoencoder had one hidden layer of 10 neurons. We used the hidden layer as input to a random forest model to predict mortality. A similar architecture was applied previously to electronic healthcare record data[24]. The choice of an autoencoder allows reduction of the feature space.

An artificial neural network has an input layer, hidden layer(s) and output layer. An activation function is used to project the input data (X) into another feature space using weights (W).

$$f(W \cdot X) \tag{1}$$

The weights W are determined from data using a technique called backpropagation[25].

We used a ReLU (Rectified Linear Unit) activation function for the hidden layer. The form of the ReLU function is shown below:

$$f(x) = \max(0, x) \tag{2}$$

We used a sigmoid activation function for the final layer:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

The output of the sigmoid function is positive even for negative input.

We also experimented with a hyperbolic tangent (tanh) function shown below:

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{4}$$

However, the cross-validation results (see discussion later) were inferior to that of the ReLU activation function.

An artificial feed-forward neural network optimizes a loss function of the form:

$$-\sum_{i=1}^{d} log(P(Y = y_i|x_i, \theta)) + \lambda||\theta|| \tag{5}$$

This is the negative log-likelihood. There are d data points. The i th data point has a label denoted by y_i and input feature vector represented by x_i. The weights of the artificial neural network are represented by a vector θ. λ is a regularization parameter to prevent overfitting and reduce model complexity. λ is usually determined by cross-validation. Shown here is the L_1 norm of the parameter vector (θ). The weights of the artificial neural network are determined using a technique called backpropagation[25].

The autoencoder used a cross-entropy loss function, which is a measure of discrepancy between the input layer and the reconstructed hidden layer. The cross-entropy loss function used for the autoencoder had the following form:

$$\sum_{k=1}^{m} u_k \log v_k + (1 - u_k) \log (1 - v_k) \tag{6}$$

where there are m features in the input layer. u represents the input layer and v represents the hidden layer. The layers are computed by applying the appropriate activation functions (Equations (1), (2) and (3)).

The final cost function is given below:

$$\sum_{k=1}^{m} u_k \log v_k + (1 - u_k) \log (1 - v_k) + \lambda||\theta|| \tag{7}$$

where the vector θ represents all the weights of the artificial neural network. There are m features in the input layer. u represents the input layer and v represents the hidden layer. We added an L_1 penalty term on the weights to perform regularization and prevent overfitting. This is denoted by the term $\lambda||\theta||$. λ is a regularization parameter that we determined by 10-fold cross-validation.

We performed a 50%–25%–25% training-validation-test split of the data. We used the *keras* package[26] with the *Tensorflow* backend[27].

The artificial neural network was trained on the training data for a number of epochs. In one epoch the network is trained on the training dataset. The model fit is then refined over subsequent epochs. Our neural network was trained for 1000 epochs, which was assessed as being sufficient to reach convergence. We used the *Adadelta* method of optimization[28].

We selected all hyperparameters, including the number of neurons in a hidden layer and activation functions, based on a uniform search and 10-fold cross-validation. We split the data into a training set (50%), validation set (25%), and test set (25%). We trained the model on the training set. We carried out cross-validation on the validation set. The architectural parameters and regularization parameters were then selected. This final model was then evaluated on the test set. This process of splitting the data (into training, validation and test sets), training the model and performing cross-validation was repeated 10 times.

We varied the number of neurons in the hidden layer from 2 to 20. For activation functions, we tried sigmoid, rectified linear unit (ReLU) and hyperbolic tangent (tanh). We do not use dropout regularization to keep a simple architecture and simplify the process of model selection. A hidden layer of 10 neurons and ReLU and sigmoid activation functions (for the first and second layers, respectively), were found to have the least cross-validation error.

We repeated the stochastic process of splitting the data into training and test sets and performing cross-validation 10 times. This yielded a mean AUC of 0.80 (95% confidence intervals [0.78, 0.82]).

Class-contrastive reasoning

We explain our models using class-contrastive reasoning and class-contrastive heatmaps. The technique works as follows. The model is trained on the training set. For each patient in the test set, we independently mutate (change from 0 to 1, or 1 to 0) each categorical feature. For each patient in the test set, we use the trained model to compute the change in the predicted probability of death.

We repeat this procedure independently for each feature and each patient in the test set. We do not retrain the model when we mutate the features. The predictions are made using the trained machine learning model on the test set.

We visualize the amount of change in the model predicted probability of mortality, achieved by setting a particular feature to 1 versus 0, using a class-contrastive heatmap. The rows represent patients and columns represent the feature that has been changed from 0 to 1. The heatmaps also show a hierarchical clustering dendrogram, which is performed using an Euclidean distance metric and complete linkage[23].

In another variant, we also simultaneously change all pairs of features in the test set from 0 and 0 to 1 and 1. As before, for each patient in the test set, we use the trained model to compute the change in the predicted probability of death. In this case, the class-contrastive heatmap shows the amount of change in the predicted probability of mortality, achieved by setting a particular combination of features to 1 versus 0. The rows represent patients and columns represent the combination of features that are changed simultaneously.

The class-contrastive heatmap shows patient-specific predictions. Predictions for individual patients are made in the following way: the trained model makes a prediction for the probability of death based on the modified features as input. This process is repeated for each patient and each feature (or feature combination).

Survival analysis and standardized mortality ratios

For survival analysis, we used the entry date (exposure) as the date of referral. In cases where there were multiple referrals for a patient, we considered the earliest date. If this calculated date was earlier than the start date of our mental health clinical database (called RiO), we set it to the start date of RiO (1st December 2012). The event was death. The date of death was derived from the National Health Service (NHS) Spine.

We used a Cox proportional hazards model for patients with schizophrenia, using age (feature scaled) and the bio-social features (as outlined before) as input features.

Standardized mortality ratios (SMR) are a method to standardize and control for age and population structure[29]. We calculated age-standardized mortality ratios (SMR) to standardize and control for age and population structure. For calculating SMRs, we defined five-year age groups (0–4, 5–9, ..., 85–90, and >90 years). Population mortality data was used from the Office for National Statistics (ONS)[18].

We calculated SMRs using the indirect method of standardization[29]. The denominator is the expected number of deaths in the study population and the numerator is the number of observed deaths in the study population.

Hence the indirectly standardized SMR is the ratio of the number of deaths observed in a study population to the number expected if the age-specific rates of a standard population had applied:

$$\mathrm{SMR} = \frac{d}{\sum_{i=1}^{k} n_i R_i} \tag{8}$$

where d is the number of deaths in the study population. Say there are k age groups in the study and standard population. n_i is the number of people in the ith group of the study population and R_i is the crude death rate in the ith group of the standard population. The 95% confidence intervals are $\mathrm{SMR} \pm 1.96 \cdot \mathrm{SE(SMR)}$[29] where SE(SMR) is given by:

$$\mathrm{SE(SMR)} = \frac{\sqrt{O}}{E} \tag{9}$$

Here O is the observed number of deaths in the study population and E is the expected number of deaths in the study population.

Logistic regression models

We used a logistic regression model to predict mortality in patients with schizophrenia. Age (feature scaled) and the bio-social factors were used as input. The model, in R notation, was as follows:

Death ~ age + dementia + delirium + abuse_alcohol_drugs + specific_personality_disorder + respiratory + cardiovascular + diabetes + self_harm + lack_family_support + personal_risk_factors + SGA + antidepressant + suicide_attempt + dementia_drug + antimanic_drug + thyroid + FGA + diuretic + anti_hypertensive + aspirin.

This same model was also fitted using an L_1 regularized logistic regression model (details are available in the Supplementary section, subsection Sensitivity analysis).

We also fitted a logistic regression model with main effects and an interaction term between dementia in Alzheimer's disease and cardiovascular disease. The model, in R notation, was as follows:

Death ~ dementia * cardiovascular + age + dementia + delirium + abuse_alcohol_drugs + specific_personality_disorder + respiratory + cardiovascular + diabetes + self_harm + lack_family_support + personal_risk_factors + SGA + antidepressant + suicide_attempt + dementia_drug + antimanic_drug + thyroid + FGA + diuretic + anti_hypertensive + aspirin.

Software

All software was written in the R[30] and Python programming languages. Generalized linear model (GLM) regression was performed using the *glm* function in R[21,31]. Hierarchical clustering and visualization were performed using heatmaps in the *pheatmap* package[32]. Survival analysis was conducted using the *survminer* package in R[33]. L_1 regularized logistic regression was performed using the *glmnet* package[34].

Reporting summary

Further information on research design is available in the Nature Research Reporting Summary linked to this article.

DATA AVAILABILITY

This study reports on human clinical data which cannot be published directly due to reasonable privacy concerns, as per NHS research ethics approvals and information governance rules. Qualified researchers can apply for data access by submitting an application to the Cambridgeshire and Peterborough NHS Foundation Trust (CPFT).

CODE AVAILABILITY

All software was written in the R[30] and Python programming languages. Generalized linear model (GLM) regression was performed using the *glm* function in R[21,31]. Hierarchical clustering and visualization were performed using heatmaps in the *pheatmap* package[32]. Survival analysis was conducted using the *survminer* package in R[33]. L_1 regularized logistic regression was performed using the *glmnet* package[34]. The deep learning model was built in the Python programming language using the *keras* package[26] with the *Tensorflow* backend[27]. The code used in this study is available from the corresponding author upon reasonable request.

Received: 19 April 2021; Accepted: 10 November 2021;
Published online: 08 December 2021

REFERENCES

1. Goldner, E. M., Hsu, L., Waraich, P. & Somers, J. M. Prevalence and incidence studies of schizophrenic disorders: a systematic review of the literature. *Can. J. Psychiatry* **47**, 833–843 (2002).
2. Hayes, J. F., Marston, L., Walters, K., King, M. B. & Osborn, D. P. Mortality gap for people with bipolar disorder and schizophrenia: UK-based cohort study 2000-2014. *Br. J. Psychiatry* **211**, 175–181 (2017).
3. Chang, C. K. et al. Life expectancy at birth for people with serious mental illness and other major disorders from a secondary mental health care case register in London. *PLoS ONE* 6, e19590 (2011)
4. Olfson., M., Gerhard, T., Huang, C., Crystal, S. & Stroup, T. S. Premature mortality among adults with schizophrenia in the United States. *JAMA Psychiatry* **72**, 1172–1181 (2015).
5. Pedersen, C. B., Mors, O., Bertelsen, A., Waltoft, B.L. & Agerbo, E. A comprehensive nationwide study of the incidence rate and lifetime risk for treated mental disorders. *JAMA Psychiatry* **71**, 573–581 (2014).
6. Sokol, K., Flach, P. Conversational Explanations of Machine Learning Predictions Through Class-contrastive Counterfactual Statements. In: Proc. Twenty-Seventh Int. Jt. Conf. Artif. Intell. California: International Joint Conferences on Artificial Intelligence Organization, 5785–5786. https://doi.org/10.24963/ijcai.2018/836 (2018).
7. Miller, T. Explanation in artificial intelligence: insights from the social sciences. *Artif. Intell.* **267**, 1–38 (2019).
8. Reininghaus, U., Dutta, R., Dazzan, P., Doody, G. A. & Fearon, P. Mortality in schizophrenia and other psychoses: a 10-year follow-up of the AESOP first-episode cohort. *Schizophr. Bull.* **41**, 664–673 (2015).
9. Kolodner, J. *Case-Based Reasoning*. 687. https://doi.org/10.1016/C2009-0-27670-7 (Elsevier Science, 2014).
10. Gentner, D. & Forbus, K. D. Computational models of analogy. *Wiley Interdiscip. Rev. Cogn. Sci.* **2**, 266–276 (2011).
11. Power, P. J., Bell, R. J., Mills, R., Herman-Doig, T. & Davern, M. Suicide prevention in first episode psychosis: the development of a randomised controlled trial of cognitive therapy for acutely suicidal patients with early psychosis. *Aust. N Z J. Psychiatry* **37**, 414–420 (2003).
12. Sahakian, B. J., Bruhl, A. B., Cook, J., Killikelly, C. & Savulich, G. The impact of neuroscience on society: cognitive enhancement in neuropsychiatric disorders and in healthy people. *Philos. Trans. R. Soc. B Biol. Sci.* **370**, 20140214 (2015).
13. Tiihonen, J., Lonnqvist, J., Wahlbeck, K., Klaukka, T. & Niskanen, L. 11-year follow-up of mortality in patients with schizophrenia: a population-based cohort study (FIN11 study). *Lancet* **374**, 620–627 (2009).
14. Cardinal, R. N. Clinical records anonymisation and text extraction (CRATE): an open-source software system. *BMC Med. Inform Decis. Mak* **17**, 50 (2017).
15. Cunningham, H., Tablan, V., Roberts, A. & Bontcheva, K. Getting More Out of Biomedical Documents with GATE's Full Lifecycle Open Source Text Analytics. *PLoS Comput. Biol.* **9**, e1002854 (2013).
16. Wang, T. et al. Implementation of a real-time psychosis risk detection and alerting system based on electronic health records using CogStack. *J. Vis. Exp.* https://doi.org/10.3791/60794 (2020).
17. Sultana, J., Chang., C. K., Hayes, R. D., Broadbent, M. & Stewart, R. Associations between risk of mortality and atypical antipsychotic use in vascular dementia: a clinical cohort study. *Int. J. Geriatr. Psychiatry* **29**, 1249–1254 (2014).
18. ONS. Death registrations summary tables—England and Wales—Office for National Statistics. https://www.ons.gov.uk/peoplepopulationandcommunity/birthsdeathsandmarriages/deaths/datasets (2017).
19. WHO (1992) The ICD-10 classification of mental and behavioural disorders : clinical descriptions and diagnostic guidelines. World Health Organization. Technical report. https://apps.who.int/iris/handle/10665/37958 (1992).
20. Winter, B. Linear models and linear mixed effects models in R with linguistic applications. *arXiv Prepr 1308.5499*. Preprint at https://arxiv.org/abs/1308.5499 (2013).
21. Bates, D., Machler, M., Bolker, B. & Walker, S. Fitting linear mixed-effects models using lme4. *J. Stat Softw.* **67**, 1–48 (2015).
22. Bourlard, H. & Kamp, Y. Auto-association by multilayer perceptrons and singular value decomposition. *Biol. Cybern.* **59**, 291–294 (1988).
23. Gareth, J., Daniela, W., Trevor, H. & Robert, T. Introduction to Statistical Learning with Applications in R. Springer. https://www.statlearning.com/ (2017).
24. Beaulieu-Jones, B. K. & Greene, C. S. Semi-supervised learning of the electronic health record for phenotype stratification. *J Biomed Inform* **64**, 168–178 (2016).
25. Linnainmaa, S. Taylor expansion of the accumulated rounding error. *BIT Numer. Math.* **16**, 146–160 (1976).
26. Chollet, F. keras. https://github.com/keras-team/keras (2015).
27. Abadi, M. et al. TensorFlow: a system for large-scale machine learning. In: Proc. 12th USENIX Conf. Oper. Syst. Des. Implement. (2016).
28. Zeiler, M. D. ADADELTA: An adaptive learning rate method. *arXiv Prepr 1212.5701*. Preprint at https://arxiv.org/abs/1212.5701 (2012).
29. Higham, J., Flowers, J. & Hall, P. Standardisation. Technical report, Eastern Region Public Health Observatory. https://www.scotpho.org.uk/media/1403/inphorm-6-final.pdf (2005).
30. R Core Team. R: a language and environment for statistical computing. https://www.r-project.org/ (2017).
31. Kuznetsova, A., Brockhoff, P. B. & Christensen, R. H. B. lmerTest Package: tests in linear mixed effects models. *J. Stat Softw.* **82**, 1–26 (2017).
32. Kolde, R. Pheatmap: pretty heatmaps. https://cran.r-project.org/package=pheatmap (2018).
33. Kassambara, A. Survminer: survival analysis and visualization. https://github.com/kassambara/survminer (2019).
34. Friedman, J., Hastie, T. & Tibshirani, R. Regularization paths for generalized linear models via coordinate descent. *J. Stat Softw.* **33**, 1–22 (2010).

ACKNOWLEDGEMENTS

This work was funded by an MRC Mental Health Data Pathfinder grant (MC_PC_17213). P.B.J. is supported by the NIHR Applied Research Collaboration East of England. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript. This research was supported in part by the NIHR Cambridge Biomedical Research Centre. The views expressed are those of the authors and not necessarily those of the MRC, the NHS, the NIHR, or the Department of Health and Social Care. We thank Jenny Nelder and Jonathan Lewis for all their support during this project and Irene Egli for inspiring S.B. to think about patients with schizophrenia. This work is dedicated to the memory of Patrick Winston.

AUTHOR CONTRIBUTIONS

S.B., P.L., P.B.J., and R.N.C. designed the study. S.B. and R.N.C. verified the underlying data. S.B. conducted the analyses and wrote the original draft of the manuscript. All authors edited the manuscript and gave final approval for publication.

COMPETING INTERESTS

R.N.C. consults for Campden Instruments Ltd and receives royalties from Cambridge University Press, Cambridge Enterprise, and Routledge. S.B., P.L., and P.B.J. declare they have no conflicts of interest to disclose.

ADDITIONAL INFORMATION

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41537-021-00191-y.

Correspondence and requests for materials should be addressed to Soumya Banerjee.

Reprints and permission information is available at http://www.nature.com/reprints

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

scientific reports

Check for updates

OPEN

Neural networks for abstraction and reasoning

Mikel Bober-Irizar & Soumya Banerjee

For half a century, artificial intelligence research has attempted to reproduce the human qualities of abstraction and reasoning - creating computer systems that can learn new concepts from a minimal set of examples, in settings where humans find this easy. While specific neural networks are able to solve an impressive range of problems, broad generalisation to situations outside their training data has proved elusive. In this work, we look at several novel approaches for solving the Abstraction & Reasoning Corpus (ARC). This is a dataset of abstract visual reasoning tasks introduced to test algorithms on broad generalization. Despite three international competitions with $100,000 in prizes, the best algorithms still fail to solve a majority of ARC tasks. The best solvers today rely on complex hand-crafted rules, without using machine learning at all. We revisit whether recent advances in neural networks allow progress on this task, or whether an entirely different class of models are required. First, we adapt the DreamCoder neurosymbolic reasoning solver to ARC. DreamCoder automatically writes programs in a bespoke domain-specific language to perform reasoning, using a neural network to mimic human intuition. We present the Perceptual Abstraction and Reasoning Language (PeARL) language, which allows DreamCoder to solve ARC tasks, and propose a new recognition model that allows us to significantly improve on the previous best implementation. We also propose a new encoding and augmentation scheme that allows large language models (LLMs) to solve ARC tasks, and find that the largest models can solve some ARC tasks. LLMs are able to solve a different group of problems to state-of-the-art solvers, and provide an interesting way to complement other approaches. We perform an ensemble analysis, combining systems to achieve better results than any system alone and analysing individual strengths. However, it is sobering to see that approaches based on neural networks still lag behind existing hand-crafted solvers, and we suggest avenues for future improvements. Our findings with the ensemble model may indicate that a diversity of methods might be necessary to solve problems in ARC. Humans likely employ diverse strategies to solve ARC. Studies involving human participants to identify the strategies they employ to solve ARC could provide valuable insights for future AI approaches. Finally, we publish the arckit Python library to make future research on ARC easier.

For the past fifty years, researchers in the field of artificial intelligence (AI) have been striving to replicate human abilities of abstraction and reasoning - developing computer systems that can learn new concepts from a small number of examples, something that humans find relatively easy[1]. While most recent AI research has focused on *narrow intelligence* (solving specific tasks given large amounts of data), we revisit whether new advances can allow computers to extrapolate to new concepts rather than merely interpolate.

This concept has been referred to as *broad generalisation*[2]: humans do this through abstraction and reasoning; drawing analogies to previous situations and thinking logically. While AI systems such as neural networks excel at a wide range of tasks, from protein folding[3] to playing Go[4], their inability to broadly generalise has precluded deployments in the real world, such as in self-driving cars. To enable safer AI, we must understand how to build systems that can reason in unusual situations.

To systematically build and evaluate computer systems that can solve abstract reasoning problems in a human-like intelligent way, we turn to a concrete benchmark. In 2019, the Abstraction and Reasoning Corpus (ARC) was introduced, as an attempt to codify a benchmark of intelligence[2] - a sort of 'IQ Test' for AI. The ARC contains a series of human-designed tasks on grids, which require learning some transformation from a small number of demonstrations. Despite three international competitions with over $100,000 in prize money, performance on ARC has proved elusive, with no single system surpassing 50% accuracy on the ARC-Easy dataset or 20% on the test set[5].

With only a handful of training examples per ARC task, and 10^{900} possible answers (of which exactly one gains credit), traditional machine learning (ML) methods that require large datasets have so far been unable to make progress. Current state-of-the-art approaches to ARC make use of complex hand-crafted algorithms based

University of Cambridge, Cambridge, UK. ✉email: sb2333@cam.ac.uk

on brute force search, without harnessing any ML[6]. This work looks at whether novel machine learning systems that focus on abstraction and reasoning can achieve broad generalisation, using ARC as a focal point.

We investigate two new approaches to ARC, focusing on novel ways to incorporate neural networks to build better abstraction and reasoning solvers.

We adapt the DreamCoder algorithm, a recent state-of-the-art algorithm for program induction, to solve ARC tasks. DreamCoder writes programs in a domain-specific language to solve tasks; we design the **Perceptual Abstraction & Reasoning Language** (PeARL) pure functional domain specific language for this purpose. Our DreamCoder implementation solves $3\times$ more tasks than existing work[7]. We introduce a new framework for solving ARC tasks using large-language models (LLMs), transforming these visual tasks into a textual domain. A detailed evaluation of three model classes[8–10] shows that LLMs can achieve competitive performance with human-crafted systems with the right augmentation and domain transformation. We build an ensemble of multiple ARC solvers that outperforms any individual system, and analyse the diversity and relative strengths of each solver.

We note however that the best systems can currently only solve about 40% of the tasks on the ARC-Hard dataset (which is much lower than human performance), meaning ARC is far from solved. Additionally, the machine-learning based systems in this work still significantly trail the state-of-the-art based on hand-crafted rule-based search[11].

Finally, we publish an open-source Python library for working with ARC to stimulate work in this field.

Background

Since the very first research on artificial intelligence, analogy-making has been considered central to the notion of intelligence. When presented with novel situations (for example; opening a new type of door, or conversing about a new topic), humans effortlessly solve these situations by creating analogies to previous experiences and concepts.

In 1967, Mikhail Bongard made one of the first attempts at identifying this notion of analogy-making in his book *Pattern Recognition*[1]. He noted how scientists such as Alan Turing have long posited the concept of a thinking machine; but while machines can be built to solve specific tasks (such as solving quadratic equations or playing chess), no progress had been made to imitate or even understand the ability of humans to adapt to new situations. Bongard goes on to suggest that pattern recognition, the ability to recognise situations into objects and classes of objects (concepts), is central to the abilities of human intelligence.

Bongard introduced a set of problems (now known as Bongard Problems)[1], where two sets of shapes are presented, and the task is to identify the common factor that differentiates them (see Supplementary Fig. 1). Note that every Bongard problem has a unique transformation, and thus one cannot simply train a classification model to identify a set of transformations - the system must be able to 'invent' them. This rules out classes of models used to solve most AI tasks, which rely on large amounts of training data to spot patterns.

In Douglas Hofstadter's seminal book, *Gödel, Escher, Bach*[12], he writes that *"the skill of solving Bongard problems lies very close to the core of 'pure' intelligence, if there is such a thing"* - this work popularised the Bongard problems and proposed ideas for solving them that are still relevant today.

In the past 50 years, much work has been done to build on these problems. The original collection has been extended to almost 400 Bongard problems by a variety of authors[13]. Many attempts have been made to try and solve Bongard problems computationally, including using neural networks; however, results are still very limited[14–17].

Beyond the difficulties of the task itself, Bongard problems are not well-suited to computer evaluation (discussed further in Section "Comparison with bongard problems"), so new benchmarks have been proposed in recent years that build on Bongard's ideas. One such benchmark is the Abstraction and Reasoning Corpus.

The abstraction and reasoning corpus

In 2019, Chollet published *On the Measure of Intelligence*[2], discussing past and future approaches to general artificial intelligence. Chollet notes that while excellent progress has been made in solving specific tasks to approach or surpass human-level (such as detecting cats and playing Go), these models generally require a huge amount of training and are limited to performing well on situations that they were trained on. The failure of neural network models to perform when extrapolating outside the training data has been widely explored[18].

In contrast, humans have an outstanding ability to solve tasks in highly novel situations with little training data, or indeed tasks that no human has ever solved before.

While machine learning models are often claimed to 'generalise', Chollet defines three types of generalisation: local generalisation, where a system can respond to new examples within an existing domain (for example, an image classifier generalising to a test set); broad generalisation, where a system adapts across a wider set of situations including examples which the system's creator could not have foreseen; and extreme generalisation, "adaptation to unknown unknowns across an unknown range of tasks and domains"[2].

Capturing and comparing a systems' abilities to perform these sorts of broad generalisation tasks is inherently difficult. Despite many decades of research, there is no consensus on how to measure intelligence[19], although the extensive field of psychometric testing has proposed many competing tests for humans.

The Abstraction and Reasoning Corpus (ARC), introduced by Chollet in[2], attempts to provide a benchmark for broad generalisation. By formalising a concrete way to measure generalisation ability, the hope is to foster progress in much the same way as ImageNet transformed image classification.

The ARC dataset consists of 900 hand-crafted tasks, each requiring a solver to perform abstract reasoning. In each task, the solver is first presented with some input grids (usually 3-5) and a corresponding set of output grids. Each grid contains pixels of one of 10 colours, represented by integers 0-9 and with a black 'background'. The grid size varies between each task and, indeed, within a task. Supplementary Fig. 2 and 3 show some example ARC tasks.

Each task represents some common transformation from the input to output grids. A system must reason about the differences in training pairs and abstract a transformation to be applied to new input grids to produce output grids. The system is then presented with one or more test input grids, for which the system can provide up to three predictions. A task is considered solved if any of the three predictions are identical to the correct answer - no partial credit is given for close answers.

The dataset is split into three subsets: a *training set*, *evalution set* and *test set*. The training and evaluation sets each contain 400 unique tasks, and the evaluation set contains tasks that are harder than the training set. The private test set contains a further 100 tasks, which are not publicly available: to evaluate a system on the test set, a researcher must submit code to be executed on an offline system; as a result we focus on the first two datasets.

Notably, algorithms evaluated in this work use unsupervised learning and do not train on labelled data, meaning we use both datasets exclusively for evaluation. As a result, we refer in this work to these datasets as *ARC-Easy* and *ARC-Hard* respectively.

Core knowledge

Each task is designed to take advantage of some of four *Core Knowledge priors*[2]; by defining these explicitly, ARC tries to reduce reliance on whether an algorithm has sufficient 'acquired knowledge' and focus purely on reasoning abilities.

1. **Objectness priors:** Handling objects and their interactions. An algorithm must be able to segment the grid into objects based on space and colour (while accounting for noise and occlusion). Physical contact between objects is a common theme (e.g., 'gravity' transformations and objects growing until they hit other objects).
2. **Goal-directness prior:** Many tasks involve a general notion of 'intentionality', finding the simplest solution to some 'problem' (e.g., drawing the shortest path through a maze rather than a longer one).
3. **Numbers and counting priors:** Some tasks involve counting and basic arithmetic (such as addition and subtraction on numbers below 10), as well as basic set manipulation (such as sorting objects based on some attribute such as size).
4. **Basic geometry and topology priors:** Many ARC tasks rely on geometric transformations, such as translations, rotations, shape scaling, copying objects, and drawing lines.Some examples of these tasks are also shown in Supplementary Fig. 8.

Like Bongard problems, each ARC task is essentially a *few-shot* learning problem. The high dimensionality of the output means that training traditional ML methods on the input/output pairs is impossible; only one of the 10^{900} possible output grids gains credit, with as few as three training examples.

Despite the incredibly challenging nature of the ML problem, *an average human can solve a majority of the tasks in ARC*; this highlights our ability to perform broad generalisation in a way that today's ML systems cannot, and highlights a significant gap in current AI systems.

Comparison with bongard problems

While Bongard problems and ARC are designed to test a system or human's ability to perform inductive reasoning in the presence of few examples, there are key differences. ARC was designed for a modern machine learning paradigm. This makes it more amenable to both designing algorithms to solve it and robustly evaluating those algorithms, compared to Bongard problems and classical psychometric tests designed for humans (such as Raven's Progressive Matrices[20]).

- **Evaluation:** How would we unambiguously determine if a system correctly identified the abstract transformation in a Bongard problem? Determining whether any analogy description is 'correct' could be subjective and time-consuming. In ARC, the problem is modified: the system is instead presented with a few examples of a transformation and tasked with applying the transformation to a new input. The output can then be scored algorithmically (the model was successful if it produces a pixel-perfect output).
- **Data size:** While Bongard published 100 of his problems, ARC has 900 total tasks. More tasks means both that learning systems have more training data, and allows for more precise evaluation. Additionally, 100 tasks are held back as a private test set: by enforcing that these tasks are unseen, one can truly test 'developer-aware' generalisation[2].
- **Problem format:** Distilling the problem of logical reasoning (rather than interpreting or generating images), ARC presents tasks as coloured pixels on a variable-size grid. This means that systems do not have to rely on a computer-vision shape detector: the problem is distilled as far as possible into one of reasoning.These attributes make ARC an excellent testbed for ongoing research into the ability of systems to perform abstraction and reasoning. Benchmarks have led to immense progress in other areas of AI research, such as the ImageNet image classification challenge. A similar benchmark for abstraction and reasoning may accelerate research into broad generalization in machines.

Previous work

Many attempts have been made to computationally solve ARC, primarily through the Kaggle Abstraction & Reasoning Challenge hosted in 2019[5] with a $20,000 prize pool, where the current state-of-the-art was set by Johan Sokrates Wind (also known as Icecuber)[6]. Icecuber implements a Domain-Specific Language (DSL) with 142 handcrafted unary functions on grids. At runtime, the functions are greedily composed on the input grids, with the resulting 'pieces' stored in a directed acyclic graph (DAG). Finally, a solver combines pieces from the DAG to get as close as possible to matching the training examples. Supplementary Fig. 9 shows an outline of the approach; a detailed description is available in Supplementary Materials.

Xu et al. introduced ARGA (Abstract Reasoning with Graph Abstractions)[21]; they extended DSL search by converting ARC grids into an object-graph representation, and operating on these representations instead. Additionally, a series of *constraints* were derived from the training examples to prune the search space, and a 'Tabu list' avoided searching primitives with poor recent performance. Overall, this approach achieved performance comparable to Icecuber limited to a small subset of tasks related to object manipulation.

While ARC was designed to be a machine learning benchmark, the state-of-the-art solutions all rely on entirely handcrafted methods similar to Icecuber. There have been attempts to use ML, but these have been limited to small subsets of the ARC dataset. For example, Golubev et al.solved tasks that rely on **cropping** by extracting features from grids and training a task-specific decision tree classifier to try and predict cropping coordinates (x, y, w, h) for a task's test example[22,23]. This approach generalised to solve 7% of private test set tasks.

In 2021, Alford et al.[7,24] explored the idea of applying the neurosymbolic solver DreamCoder[25] to the ARC dataset, considered much more challenging than previous DreamCoder applications. They selected a subset of 36 ARC-Easy tasks involving symmetry and basic geometric operations (such as rotations, flips and crops), and provided a DSL with 5 basic primitives on grids. They found that DreamCoder could solve some of the tasks and also successfully created new primitives. However, this solution was limited to a small subset of the ARC-Easy dataset, and did not use neural-network-guided search unlike the original neurosymbolic solver DreamCoder paper[25].

In the next section, we review the DreamCoder algorithm in detail, and then revisit whether DreamCoder can be extended to solve much more of ARC by harnessing a more powerful DSL, neural networks and other improvements.

Neurosymbolic programming with Dreamcoder

Within AI, the field of inductive programming[26] describes algorithms that derive *programs* that explain a series of examples. After 30 years of research, many algorithms and approaches have been proposed across a wide range of applications, with the research area far from being solved. In general, inductive programming provides an encouraging research direction for ARC due to its ability to massively prune the search space from "all possible grids" to just those explainable by a programmatic transformation.

The DreamCoder system[25], was a novel approach to inductive programming, which used neural networks to guide its ability to write programs (neurosymbolic programming).

The DreamCoder algorithm can be broken up into multiple phases, and can be thought of as a *wake-sleep* algorithm. During *waking* phase, a generative model writes programs in a domain-specific language (DSL) that attempt to solve tasks. In the two sleeping phases, the programming language is updated to consolidate new information learned in waking, and a separate *recognition model* is trained which learns to guide the search towards promising programs.

These phases are interleaved in several iterations to allow for self-improvement. The result is that DreamCoder can achieve remarkable performance across several domains, such as list processing, reproducing LOGO drawings, and finding regexes[25].

In this section, we provide a detailed explanation of the DreamCoder algorithm, which we later adapt in Section "Adaptation of DreamCoder".

Waking phase

DreamCoder, in a general form, attempts to write programs to solve tasks; we must first define what this means. We define a $taskT(x, y) : p$ as a set of examples (x, y) defined by a program p such that $p(x) = y$. For a list processing task, (x, y) could contain pairs of unsorted and sorted lists, with the correct program p being a sorting algorithm. For tasks without input/output pairs (e.g, generating a program that reproduces a drawing), x can be empty.

The goal of DreamCoder is to inductively reason about the examples in T to produce a candidate program p' which matches the examples. It is important that tasks are *verifiable*: one can check programmatically whether a program is the correct solution to a task. In our list processing example, we can compute whether $p'(x) = y$, even if p is not known; this means that we can apply it to real-world problems where the answer is unknown, but can be verified.

DreamCoder is bootstrapped with a Domain-Specific Language known as a library. Just like any programming language, the library contains a set of functions and values, defined within a functional type-system. Making use of the type-system, we can generate a grammar that recursively defines the set of well-typed programs within the language. For example, consider the following toy language:

```
Types:     int,  str
Values:    1, 2, 'a', 'b'
Functions: add: int -> int -> int
           concat: str -> str -> str
```

In this language, add 1 2 is a well-typed program but add 1 'a' is not, even though it is syntactically correct. With a type-system, one can constrain the search space of programs to valid ones: a powerful tool since the set of syntactically correct programs is commonly astronomically bigger than the set of well-typed programs.

The waking phase of DreamCoder makes great use of this grammar: for each task T, we enumerate a large number of candidate programs $p' \in L$; for each sampled program, we check if it solves the task. As there are infinite possible programs, we must use a heuristic to decide which programs to sample. DreamCoder uses the *Minimum Description Length* (MDL) principle, by computing the entropy of each program and enumerating the programs with the least entropy first. This heuristic is based on the idea that the shortest program that solves a task is the most likely to be the correct one (often compared to Occam's razor).

On its own, the waking phase can be seen as a brute-force search with a powerful and cleverly-defined search space.

Abstraction sleep

The power of the DreamCoder algorithm comes from the two sleep phases, known as *abstraction sleep* and *dreaming sleep*.

Abstraction sleep considers and manipulates the solutions of tasks solved during waking. First, discovered solution programs (**frontiers**) P are inserted into a *version space*; a data structure that efficiently represents possible refactorings of programs.

The version space represents a large set of programs: for example, the functional program $(+\ 1\ 1)$ can be refactored as $((\lambda\ (x)\ (x\ 1\ 1))\ +)$, and both of these would be included in the version space. All programs within $n = 3$ steps of refactoring are considered.

By representing all frontiers along with possible refactorings in the version space, we can pick out common concepts that occur across multiple programs as new *primitives*. These primitives can then be added to our library for the next iteration of waking, adding common concepts to the DSL (learning by analogy to existing tasks). At each iteration, the *k* most common concepts that cause the most reduction in total description length are compressed into new primitives.

The effect of abstraction sleep is to dramatically reduce the depth of search (at the cost of a slight increase in breadth). In practice, this abstraction learning can be extremely powerful: in a LOGO drawing task, the learned concepts included drawing polygons with *n* sides and *l* side length, or drawing a circle with *r* radius[25]. The concepts learnt can even include higher-order functions, such as a radial symmetry function that repeats a function multiple times at different angles. At the next iteration of waking, these concepts form longer, more powerful programs than is possible by DSL search on simple operations.

Dreaming sleep

The final phase of the algorithm is *Dreaming Sleep*. In this phase, we focus on decreasing the breadth of search by intelligently guiding the generative grammar for each task. To do this, we train a neural network *recognition model*, which can directly perform abductive reasoning and infer $T(x, y) \to p$. There are two challenges to overcome in designing such a model.

First, the space of programs is exponentially large, and tasks are very difficult; even a human will often consider and discard multiple candidate hypotheses before arriving at the correct solution. Hence, a neural network which directly predicts the answer is likely to fail.

Instead, the neural network is trained to produce a grammar under which the correct solution *p* has low entropy. Since programs are enumerated in order of entropy (using iterative deepening), this means that a search guided by the recognition network can find the correct solution faster.

The neural network is made up of a feature extractor which converts a task to a fixed-width feature vector, and a *GrammarNet*, which takes the feature vector and produces a volume Q, where $Q_{ijk}(x)$ is the probability of primitive *i* being the *k*th argument to primitive *j*. From Q, we can construct a contextual grammar which assigns likelihoods to programs, and sample from this new grammar during waking. This is shown in Supplementary Fig. 4.

The second issue is that of training data: with only 800 tasks available and no labelled solutions (p) for any of them, we cannot train a complex neural network on our data. To solve this, DreamCoder uses *dreaming* to generate new training tasks (also called Helmholtz enumeration, inspired by Helmholtz machines[27]). To dream up a labelled training task, we randomly sample a program p^* from our existing probabilistic grammar, and sample some input grids x from the empirical distribution of our tasks; we can now construct a new dreamed task $T^*(x, p^*(x)) : p^*$. To train the recognition model, a constant stream of dreamed tasks and associated correct programs can be used for backpropagation.

The role of the recognition model in DreamCoder is very similar to the use of 'policy networks' in other works such as *AlphaGo*[4], which achieved superhuman performance in the board game Go. In AlphaGo, a Monte-Carlo tree search (MCTS) is used to evaluate possible positions on a board, with a policy network suggesting potentially useful moves to evaluate: the neural network acts to dramatically prune the search space and make search feasible.

In DreamCoder, the enumeration engine acts as the MCTS (evaluating programs matching a recursive grammar), while the recognition model assigns weights to the grammar such that more 'promising' programs are evaluated first. *These networks attempt to to mimic a human's ability of intuition*: when a person solves ARC tasks or plays Go, the vast majority of valid operations are immediately discarded as nonsensical.

Methods

Adaptation of DreamCoder

We adapt DreamCoder as an ARC solver, combining the power of DSL search with neural networks. We build on the reference Python/OCaml implementation by Ellis et al.[28], as well as the work of Alford et al.[29]. To allow DreamCoder to effectively solve ARC tasks, we introduce a new recognition model and domain-specific language (PeARL) for operating on grids, described below. Additional adaptations such as a new evaluation module and multicore search were also implemented, and are open-sourced with this work (Section "Software"). We detail the two primary modifications to the DreamCoder architecture below; Supplementary Figure 11 also gives a top-level view of the modules modified from the reference DreamCoder implementation.

ARC recognition model

A core component of the DreamCoder architecture is the **recognition model**. With many primitives available to our model, a brute-force search is costly and cannot go sufficiently deep to find all solutions within the search space. The recognition model is a neural network which attempts to *guide* the search, by estimating which programs are most likely to solve a task before enumeration. It is trained during dreaming sleep, and produces a contextual grammar for each task, which assigns a high probability to the correct solution (Section "Neurosymbolic programming with Dreamcoder").

The recognition model begins with a feature extractor module which converts a task to a fixed-width feature vector. This allows a different feature extractor to be used for different applications (such as LOGO drawings or list processing[25]). At the same time, a common Grammar Network learns to induce grammars for each task based on extracted features. The end-to-end network can be optimised using gradient descent by making the feature extractor differentiable.

ARC tasks have a very different format to previous applications of DreamCoder and thus necessitate a new feature extractor design. The 2D image-like nature of our grids means that we look to image recognition networks as a base, but there are several design challenges to overcome. We discuss these in turn. The final selected architecture is shown in Supplementary Figure 5.

First, the highly variable grid size precludes the use of convolutional neural networks (CNNs), which rely on a fixed-size image input, or at least a minimum-size input when adaptive or global pooling layers are used[30,31]. In our case, we need a network that can effectively operate down to 1×1 grids.

One option is to pad all grids to a sufficiently large fixed size, such as 30×30. However, this risks hurting performance on tiny grids when most of the image is padding (most inputs are much smaller than 30×30). Our network is therefore inspired by Fully Convolutional Networks (FCNs)[32], which remove these limitations. We use a series of convolutional layers; each layer has internal padding such that its output size equals its input size.

Instead of downsampling operations usually employed by CNNs and FCNs to extract larger-scale features (which enforce a minimum input size), we use dilated convolutions[33] in later layers of our network. Dilated convolutions leave gaps between sampled pixels; the resulting behaviour is similar to a convolution on a downsampled image. These layers enable multiple scales of contextual information to be incorporated without the parameter explosion associated with large convolutions. The outputs of the dilated convolutions are added to the feature vectors as residuals, meaning the network can choose not to use them (*e.g.*on small grids).

Another unique attribute of ARC is that we have both input and output grids; rather than extracting features from a single grid, the solution to a task depends on the *relationship* between two grids of potentially different sizes. To generate a single set of features for a *task*, we apply the network to each grid followed by an adaptive average-pooling layer to generate two $64 \times 3 \times 3$ feature maps $M(y)$, $M(x)$. The **difference** between these two feature maps $M(y) - M(x)$ is then passed to a linear layer and averaged across training examples to create a single 256-dimension feature vector for an entire task. Using a spatially adaptive layer allows the network to detect when an object has been *translated* between the input and output grid; this was found to improve performance.

Model training

Due to the small model size, we perform all training on CPU. The recognition model is trained at each wake-sleep cycle on Helmholtz-sampled tasks for 360 seconds; around 3,000 random tasks. The Adam optimiser[34] optimises the sum of two loss functions: the **entropy loss**, which is the overall log-likelihood of the program given the generated grammar, and a **classification loss** that treats the grammar network as an N-classifier where N is the number of primitives and minimises binary cross-entropy. Halfway through training, we anneal the learning rate $10\times$ to improve convergence.

Initially, while the model converged to a relatively low loss, its predictions were not useful on actual ARC tasks. Upon inspection, the dreamed programs were extremely complex and too ambiguous for even a human to solve the generated tasks. To combat this, the Helmholtz sampling procedure was modified to limit any sampled program to a maximum depth of 3 primitives, dramatically reducing the loss.

Supplementary Figure 12 shows the training behaviour of the recognition model. We see that when neural-network-derived grammars are used, the entropy of discovered solutions (description lengths) is reduced by about 30% compared to a prior grammar over all tasks. Solutions with lower description lengths take **exponentially less time to find**, so this improvement is dramatic. We also see that the recognition model learns to classify which primitives might be used to solve a given task. We find empirically that our recognition model reduces the number of programs written before a solution is found by approximately $10\times$.

The PeARL language

DreamCoder aims to perform program induction within some well-defined language, using guided search to improve efficiency. In principle, DreamCoder could be used to write programs in any language with recursive grammar (even Python).

As discussed in Section "Understanding error cases in ARC", searching for all possible Python programs (even a guided one) to discover solutions to ARC tasks would be extremely difficult and inefficient. For this reason, we instead build a *domain-specific language* (DSL) in which a much higher proportion of enumerable programs are likely to be useful. Indeed, current state-of-the-art approaches to ARC mostly involve a DSL in some capacity (see Section "Previous work").

Following existing DreamCoder DSLs for other domains, we designed the **Perceptual Abstraction & Reasoning Language** (PeARL), a bespoke DSL explicitly designed to represent transformations in ARC tasks. PeARL has two constructs: *types* and *primitives*. Types represent data-types and primitives can represent either a *value* or an n-ary function.

We define the following types in PeARL:

grid: A 2D grid of coloured pixels. Each grid contains a size and position, where the position is the original position of that segment (e.g., if the grid has since been cropped).

size, pos: The size or position of a grid, represented by a tuple of integers (x, y). When a subgrid is cropped, it retains a position, which can be used for alignment when recombining subgrids.

colour: A pixel colour, represented by an integer 0-9, where 0 is black.

count: An integer. This type can be used for arithmetic on grid properties, which is useful for some tasks.

A *valid program* in PeARL is any lambda expression of type grid -> grid which type-checks. Following existing DreamCoder DSLs, PeARL programs can use any number of primitives, lists, and higher-order functions. The language is entirely defined by a series of Python functions (one for each primitive), where the Python type-annotations implicitly generate the corresponding grammar. For convenience, a DSL class was implemented, allowing a one-to-one mapping between Python definitions and PeARL, removing additional boilerplate. This enabled rapid prototyping and experimentation.

For example, here is an implementation of a subset of PeARL with two types and two primitives:

```
Colour = NewType("Colour", int) # Colours are ints
class Grid: ... # Implementation of the Grid type is more complex

typemap = {Grid: baseType("grid"), # baseType defines a DreamCoder type
           Colour: baseType("colour")}
dsl = DSL(typemap) # Mapping of python types to DreamCoder types

@dsl.primitive
def topCol(g: Grid) -> Colour:
    return np.argmax(np.bincount(g.grid)[1:])+1

@dsl.primitive
def filterCol(g: Grid, c: Colour) -> Grid:
    filtered = g.grid.copy()
    filtered[filtered != c] = 0
    return g.newgrid(filtered)
```

The DSL is populated with types grid, colour, as well as two primitives:

```
topCol: grid -> colour
filterCol: grid -> colour -> grid
```

An example of a valid program in this DSL is λ: filterCol \$0 (topCol \$0), which represents the operation "remove all colours from the grid except the most common one". The argument \$0 represents the input grid to an ARC task, with the result of the function representing the *output* grid of that task.

A given program f is **correct** if f(x) = y for all (x, y) pairs in a task (such programs are known as **frontiers** for a given task). Hence, the *goal of DreamCoder* is to find frontiers for each task, which can be used to generate predictions on the test examples.

A framework for large-language models

Language models are a class of models designed to statistically model natural language, being trained to predict the next token (*e.g.*words) in a training corpus. Large language models (LLMs) are characterised by their size (containing tens of billions of parameters) and training on a vast corpus of text (usually scraped from the internet). When fine-tuned using reinforcement learning with human feedback, 'chat' models can be conditioned to respond to instructions in a conversational format[35].

Since the development of GPT-3 in 2020, LLMs have gained popularity at a seismic pace, and the ability of LLMs to write code or solve word puzzles is impressive. Researchers from Microsoft have stated that GPT-4[9] "could reasonably be viewed as an early (yet still incomplete) version of an artificial general intelligence (AGI) system", and found that it could solve some reasoning tasks.

Further, work from Webb et al.[36] has recently suggested that "large language models such as GPT-3 have acquired an emergent ability to find zero-shot solutions to a broad range of analogy problems". To demonstrate this, they created a digit matrices problem set inspired by Raven's Progressive Matrices. Each problem is governed by a set of transition rules which can be stacked up to depth 3, and they find that GPT-3 performance largely surpasses human performance.

Many researchers also reject these claims, arguing that these examples do not accurately assess reasoning, that LLMs cannot even solve the letter-string analogies attacked by the Copycat computer program 30 years ago[37,38], and that they still have "a poor grasp of reality"[39].

Given this debate, it is important to evaluate these models and whether they can already be used to solve ARC, which presents a formidable challenge compared to existing reasoning tasks such as in[36]. To do this, we conduct experiments on several state-of-the-art LLMs by a new scheme to translate ARC tasks into a textual domain. We then evaluated them in the same framework as our DreamCoder solution and existing work.

Experimental setup

We evaluate several large language models (LLMs) on a common testing framework, designed to produce a fair comparison with existing ARC solutions. We test the OpenAI GPT series of models through their API, as well as Meta's LLaMA model series. For LLaMA models, we use GPTQ[40] to quantise the models to 4-bit precision, using a group size of 128. This can degrade performance on standard benchmarks but allows even the largest models to be executed on an NVIDIA A100 GPU.

To encode ARC tasks in a textual format, we convert each grid into integer digits (representing colours), with newlines delimiting rows in the grid. The format is selected to match the tokenisation used by each LLM[41], which means that each grid cell corresponds to one token in the model. This is shown in Supplementary Fig. 10. The LLM is fed a prompt explaining the problem, then the inputs and outputs for the training tasks, and the final grid representing the test task must be completed. Some LLMs are fine-tuned for a chat interface rather than text completion; for these models, we represent the input and output grids as a series of user/assistant messages with an overall system message describing the problem. Note that only problems that fit in 2048 tokens (the context window for most models tested) were attempted; in practice, we find that the largest problems are very difficult for LLMs and so this is likely to only marginally decrease performance. Full details of tokenisation and prompting are available in Supplementary Materials.

To combat the disadvantage that 1D sequence models would necessarily have on a 2D task format, we *augment* each task with a transposed version and a 90-degree rotated version. This allows the LLM to attempt each task in a row-major and column-major format, dramatically improving overall performance.

Understanding error cases in ARC

Existing approaches to ARC broadly rely on the principle of program induction: generally, a DSL defines the space of possible solutions that an algorithm could ever produce, while some search process tries to induce programs. While DreamCoders model these two components explicitly, the principle applies to almost all current attempts to solve ARC, and similar reasoning algorithms such as Copycat[38].

Furthermore, when a person attempts an ARC task, we see a similar process: one tries to abstract the training tasks to a 'program' (generally in natural language in one's head, for example "rotate by 90 degrees"); human intuition is the search procedure.

To guide our search for improved algorithms, we design a framework for analysing the shortcomings of existing algorithms. In the context of a program induction algorithm, we propose three classes of failure cases:

- **Class 1:** The algorithm did not find a solution *because the solution was not in the search space*. To resolve this, we need to increase the power of the algorithm by expanding the types of operations it knows (or can construct).
- **Class 2:** The algorithm did not find a solution, *even though it was in the search space*. A prolonged search could have found the correct answer, but computational constraints meant it did not. In this case, we could increase available computation or find a way to guide the search towards promising avenues.
- **Class 3:** The algorithm found a candidate solution, *but it did not generalise*. This is characterised by the finding of some rules that solve the training examples but yield an incorrect answer on the test example - essentially a

false positive. This is perhaps the hardest case, as it cannot be solved by making the algorithm more complete - instead, we need to either reduce the search or design some notion of how plausible a given solution is so that they can be ranked.As an extreme example, we can consider an algorithm which enumerates random Python code to solve ARC tasks. This approach can never have a Class 1 error, because all ARC tasks can be solved in Python. However, it is likely to have many Class 3 errors (e.g., generating programs which simply return the training answers). In practice, we would encounter Class 2 errors because current computers could not hope to enumerate enough programs to find a correct one 'by chance'.

Thus, while in principle a search algorithm sufficiently powerful to solve ARC tasks is easy to design (much like making a Turing-complete programming language is easy) - *the difficulty is how to effectively prune and direct a search* such that it "searches in the right direction" and can solve tasks in a reasonable time. This perhaps approximates something close to human intuition; when presented with an ARC task, a human does not mechanically attempt all possible transformations.

Software

Throughout this work, the ability to quickly analyse and iterate through ARC tasks is instrumental. To this end, we have built the arckit python library, which contains tools to load and manage ARC and evaluate different algorithms. The library comes bundled with the ARC dataset, which can be rapidly loaded in a single line:

```
easy_set, hard_set = arckit.load_data()
```

Each dataset includes 400 Tasks in a TaskSet, allowing them to be looked up by either index or ID. The Task class implements a number of features, such as scoring solutions and generating prompts for LLMs.

A primary strength of ARCkit is fast visualisation to understand the dataset and where algorithms fail; ARCkit can automatically produce vector graphics showing grids or tasks, exported in a PDF or SVG format, arranging grids to fit a specified figure size. This functionality was heavily used to generate the figures in this work. Furthermore, any task can be visualised in an interface using the arctask command or within Python.

The ARCkit library can be installed with the command pip install arckit, and documentation is available at https://github.com/mxbi/arckit. Additionally, we make our code for our DreamCoder implementation available from the following repository: https://github.com/mxbi/dreamcoder-arc.

Results

Primitive design

The design of primitives for PeARL (Section "The PeARL language") has a big effect on the system's performance and is perhaps the most critical implementation detail. If primitives are too specialised or insufficiently powerful, tasks can be unsolvable. Conversely, if primitives are too elemental or there are too many non-useful primitives, the solutions may be too complex to find by search. Our design must balance these aspects for optimal performance (balancing Type 1 and Type 2 errors as described in Section "Understanding error cases in ARC").

In existing applications of DreamCoder, the authors used relatively elemental primitives, highlighting the ability of abstraction sleep to discover higher-order behaviour[25]. For example, given list processing tasks (such as sorting a list or getting the maximum element), basic functional constructs such as map, fold, cons,> were defined, from which DreamCoder was able to learn essential constructs like maximum and eventually sort.

Given the difficulty of ARC tasks, which could limit abstraction sleep's ability to discover operations, a wide range of primitives were implemented in PeARL. To construct the set of primitives in PeARL, we began by implementing basic operations such as rotations, symmetry and composition based on categorising some ARC training tasks in a taxonomy (see Supplementary Fig. 8). To supplement this, we ported a number of primitives from Icecuber, which provide a good starting point for 'powerful' primitives such as gravity, which are likely to encode core knowledge priors (Section Core Knowledge). Most of the primitives are at least loosely based on ones in Icecuber. In addition, we make use of the functional nature of PeARL to add primitives such as list processing (mklist, cons) and higher order functions (mapSplit8) which allow DreamCoder to build new concepts during abstraction sleep.

PeARL has 81 unique primitives; below, we detail some broad categories. The full list is given in Supplementary Materials.

Rigid transformation & Cropping Arbitrary rotations, flips and transpositions are all available, as well as cropping and uncropping operations. Grids can be sliced along an axis (for example, right_half), and can be repeated or mirrored to solve symmetry tasks (these primitives are inspired by Alford et al.[7]).

Composition Grids can be stacked in an arbitrary order, optionally considering original positions. Pixelwise operations can operate on pairs of grids (e.g., pixelwise_and).

Object manipulation A grid can be split into a `list[grid]` in 5 distinct ways: 4/8-connectedness and based on rows, columns and colours. PeARL can apply an arbitrary function to each object in an image with the higher-order function mapSplit8. Gravity can be simulated in a 'Tetris-like' manner, and lines can be drawn between objects.

Inspired by Icecuber, a single object can be selected based on many attributes: size, frequency, density, colour and position. Lists of objects can be composed in order of size (composeGrowing), and PeARL can define and extend lists functionally (mklist, lcons).

Colour manipulation PeARL can erase, filter and remap colours: both targeting a specific colour (c{1-9}), as well as dynamically ({top,rarest},colour).

Morphology PeARL can draw borders around objects, fill holes inside objects, compress blank spaces and repeated rows/columns.

Counting PeARL can count colours, pixels or objects in a grid (`count{Pixels,Colours,Components}`), and use counts to construct new grids of a specified size (`countTo{X,Y,XY}`).

We note that there is likely to be significant improvement possible from a more advanced or more optimized set of primitives; we did not iterate on the selection in this work.

Evaluating DreamCoder

To evaluate our DreamCoder solution, we perform two experiments, one on ARC-Easy and one on ARC-Hard. We run DreamCoder for a single wake-sleep cycle, training the recognition model for 40 minutes and attempting each task for 1 CPU-hour.

Overall, DreamCoder enumerated 3.1 billion programs for 800 tasks, discovering frontiers for 75 and 23 tasks in ARC-Easy and ARC-Hard respectively. Of these frontiers, *70 and 18 tasks were solved respectively*, with a Class 3 error rate (false positive solutions – see Section "Understanding error cases in ARC") of 1.25%. These results improve on those achieved to date by DreamCoder (23 and 4 tasks solved by Alford et al.[7]).

Overall, our implementation of DreamCoder achieves 16.5% accuracy on ARC-Easy and 4.5% on ARC-Hard, improving on other recent systems such as ARGA[21]. However, a large gap remains between DreamCoder and the best hand-crafted solutions such as Icecuber; we discuss this further in Section "Quantitative results across all methods".

Figure 1 shows three examples of PeARL solutions written by DreamCoder. DreamCoder can effectively use higher-order functions, types and combine primitives in interesting ways to discover new functionality. In the next section, we look at whether this performance would be achievable with brute force DSL search over PeARL, or whether dreaming and abstraction sleep bring improvements.

Ablation studies

The DreamCoder algorithm relies on both a good *static* DSL and dynamic learning through abstraction and dreaming sleep. We conducted an ablation study to understand the impact of each learning phase.

We ran three experiments: (i) with the recognition model disabled, (ii) with a simpler context-free recognition model that only models the probability of each primitive, and (iii) with the full contextual recognition model. The context-free model predicts the probability of individual primitives $f(\cdot)$ appearing in the solution, while the contextual recognition model also predicts the probability of every primitive within every context such as $g(\cdot, f(\cdot))$. We allowed 760 CPU-minutes of recognition model training, and ran enumeration for both 1 CPU-minute and 5 CPU-minutes per task. As the recognition model allows us to find solutions earlier in the search space (by suggesting promising candidates), we want to see if the benefit is diminished by simply allowing the search to run for longer.

The first DreamCoder iteration shows the algorithm's behaviour *before* the abstraction sleep runs, with subsequent iterations incorporating compression.

Supplementary Fig. 6 shows the effect of enabling the compression and/or recognition engine on ARC-Easy tasks solved. Compression (abstraction sleep) allows the search to solve a few more tasks by adding more powerful primitives, but the effect is smaller than expected. One explanation is that, unlike the original DreamCoder problems, PeARL was not intentionally designed to include only elementary primitives that compression could build upon, limiting the gain that composition can bring.

On the other hand, enabling the recognition model improved results, solving an additional 24 tasks. *This demonstrates the benefit that neural networks can bring to this problem.* We see that the contextual recognition model slightly outperforms the context-free; the recognition model is to some extent able to learn the relationships between primitives and use this to further narrow the search space. Finally, we see that benefit of enabling the recognition model is much greater than enumerating 5x more programs, which tells us that the recognition model is successful at narrowing down the program search space.

Taking a closer look at the behaviour of abstraction sleep, we can identify why performance sometimes decreases in later iterations. The compression engine emits new primitives combining existing functions: *these new primitives are assigned a lower entropy due to their frequent use in solutions*, and thus end up used in place of simpler counterparts that would solve the same solution - the result is that *solutions to the same tasks tend to get longer* and thus more complicated over time, with more Class 3 errors. This suggests that the Minimum Description Length (MDL) principle used for ranking solutions could be modified to improve performance.

PeARL ablation

The PeARL domain-specific language used in this work contains 81 primitives (full list in Supplementary Material), which DreamCoder can compose to produce solutions. Some of these primitives are quite powerful, with operations that operate on lists and functions (Section "Primitive design"). To understand the effect of these, we analysed the frequency with which DreamCoder used each of the primitives in the 88 discovered solutions across ARC-Easy and ARC-Hard. Overall, 58 of the 81 primitives were used in solutions (a total of 231 times), whereas 23 primitives were never used. Table 1 shows the most important primitives, with the full counts given in Supplementary Material.

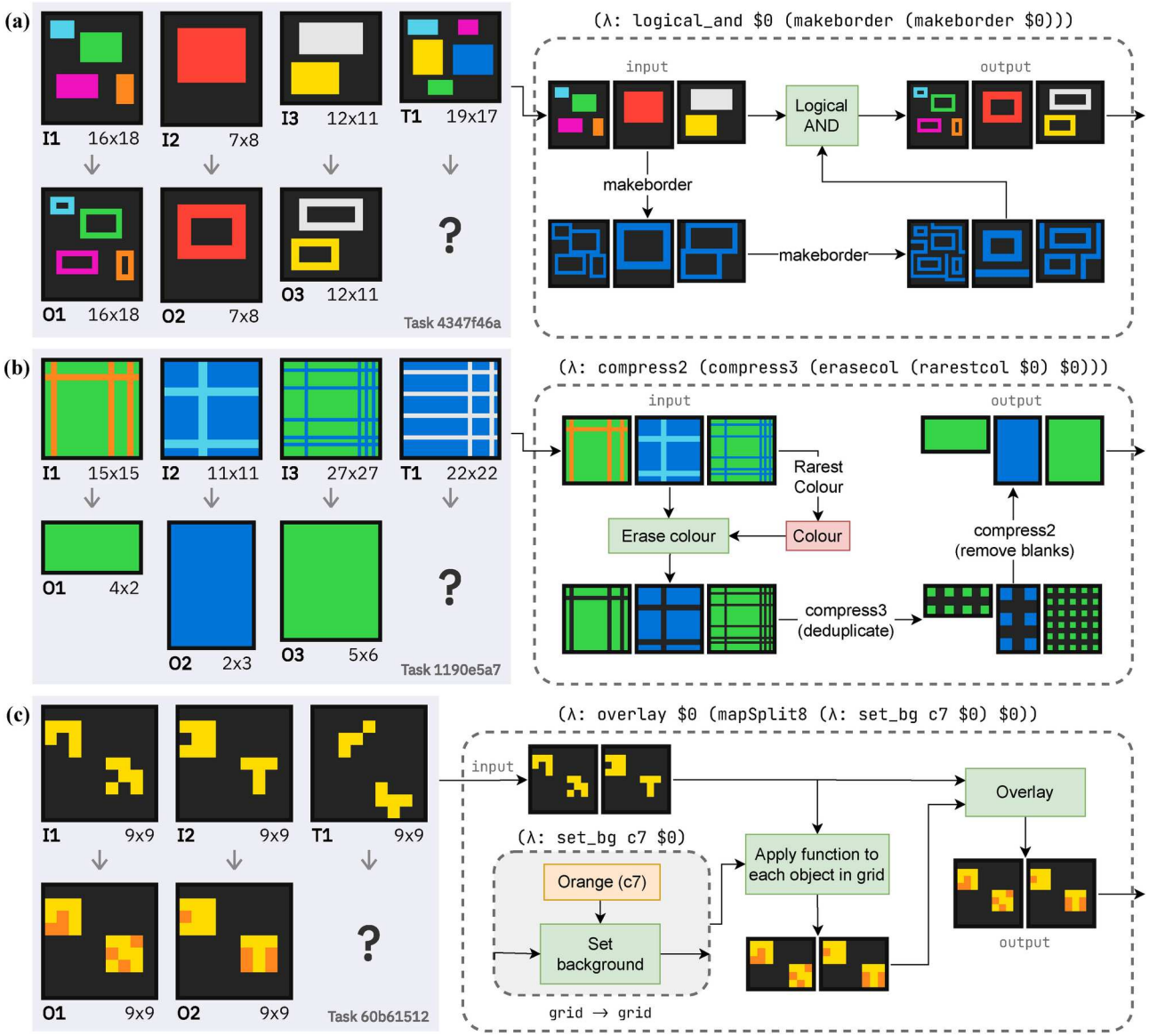

Figure 1. Three ARC tasks solved by DreamCoder (left), with the PeARL programs it generated to solve them illustrated as a computation graph (right), with intermediate outputs shown. (**a**) DreamCoder has no primitive to get the perimeter of objects, so it draws a border around it twice and takes the intersection with the original image. (**b**) Our type-system allows DreamCoder to operate on colours within images. (**c**) DreamCoder can use higher-order functions to apply arbitrary operations to each object in the image.

Additionally, we performed an ablation by removing all primitives that create or operate on lists as well as higher-order functions, leaving 61 primitives remaining. This new primitive set only contained the grid and colour types. The reduced PeARL language was able to achieve 48 and 10 solved tasks on ARC-Easy and ARC-Hard respectively, overall a 35% reduction in performance.

Large language model results

We test a variety of large language models (LLMs) on ARC. Table 2 shows the results on ARC for some popular LLMs. Figure 2 shows the accuracy attained by each LLM on each ARC dataset, broken down into three accuracies: **initial** shows the proportion of tasks which are solved on a first attempt; **augmented** allows three attempts with rotated and transposed tasks following our augmentation scheme. Finally, we consider a 'size accuracy' which shows whether the LLM outputs a grid of the correct size (even if the contents are incorrect).[1]

Several broad conclusions can be drawn from these results. First, we see a prominent performance scaling as the model size increases, especially in the LLaMA models, where progressively larger models are trained

[1] The size of these models are a trade secret, but GPT-4 likely exceeds 175B parameters.

Primitive	Description	Count
`overlay` (G → G → G)	Overlay two grids transparently. If same size, ignore position	19
`ic_compress2` (G → G)	Remove adjacent duplicate rows and columns	14
`mirrorX` (G → G)	Horizontally mirror grid [abc]⟶[abccba]	11
`mirrorY` (G → G)	Vertically mirror grid	11
`ic_compress3` (G → G)	Remove any entirely black rows/columns	9
`ic_erasecol` (G → colour → G)	Remove pixels of specific colour	8
`ic_connectX` (G → G)	Join up any objects of the same colour horizontally	8
`ic_connectXY` (G → G)	Join up objects of same colour horizontally & vertically	7
`setcol` (colour → G → G)	Set all non-black pixels to the specified colour	7
`set_bg` (colour → G → G)	Set black pixels to the specified colour	7
`ic_splitall` (G → list[G])	Split grid based on 4-connected objects	7

Table 1. The primitives used most frequently by DreamCoder in successful solutions, along with their type signatures (where `G` denotes `grid`) and descriptions. 19 primitives have at least 5 usages, with 58 total primitives seen in 88 solutions. The base colour values `c[1-9]` were used a total of 25 times. Details for all primitives are given in Supplementary Materials.

Series	Model	Params	ARC-Easy/400		ARC-Hard/400		Size acc /800
			Init.	Aug.	Init.	Aug.	
Meta	LLaMA-7B[10]	6.7 B	7	13	0	1	201
	LLaMA-13B[10]	13 B	7	15	1	3	334
	LLaMA-33B[10]	33 B	11	24	3	9	429
	LLaMA-65B[10]	65 B	18	37	5	13	408
OpenAI complete	GPT-3 Babbage[8]	1.3 B	2	5	0	0	42
	GPT-3 Curie[8]	6.7 B	1	8	2	2	85
OpenAI chat	GPT-3.5-turbo[42]	—[1]	22	40	4	13	538
	GPT-4[9]	—[1]	59	85	15	32	567

Table 2. Results on ARC for 8 popular LLMs. **Init.** gives the initial 1-shot accuracy, **Aug.** allows the model three guesses using our augmentation scheme. Size acc gives the proportion of tasks in which a correct-size grid was predicted across both datasets.

similarly. It is likely that this trend continues past currently available models, but the extent to which this continues is unclear and an open research question[43]. We also see that the ARC-Hard dataset is much more challenging, with small models solving almost no tasks.

Overall, the GPT-4 model from OpenAI massively outperforms all other LLMs, solving 21% of easy tasks and 8% of hard tasks. We see that across all models, **our augmentation can double accuracy or more** (Figure 2). The smaller GPT-3 models severely underperform, while large LLaMA models have middling performance.

Finally, we see that the size accuracy gives an easier metric that correlates with overall performance: models that solve very few tasks can be compared by their size accuracy. Figures 3 and 4 qualitatively show the types of errors that underperforming LLMs make.

Overall, we see that with our encoding and augmentation scheme, GPT-4 achieves slightly better accuracy than our DreamCoder solution (see Section "Evaluating DreamCoder").

Applicability of LLMs to ARC problems

One fair criticism of evaluating LLM abilities on ARC is that these models are designed to be text-based; they read a stream of tokens. However, we argue that this still provides a useful comparison point: there is no known equivalent to LLMs that could support reasoning with visual input and output data. This is because LLMs derive their abilities from the huge diversity of tasks displayed in their training data (often encompassing a trillion tokens of code, scientific papers, forums, etc.), and performance has been shown to be very dependent on training data[10]. In the visual domain, equivalent datasets that display broad human reasoning capabilities are

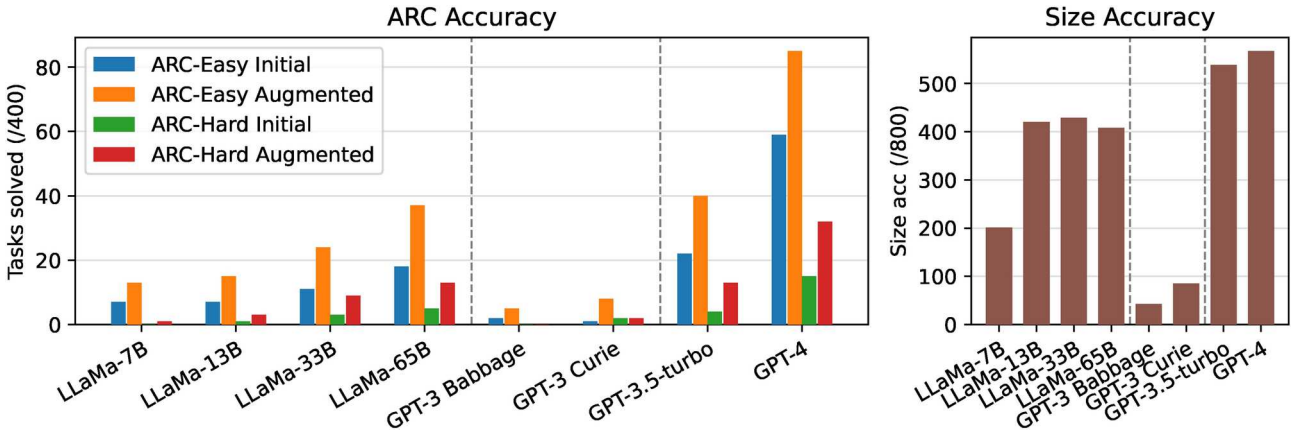

Figure 2. ARC performance of each large language model (LLM), as in Table 2. The number of correctly solved tasks is given for ARC-Easy and ARC-Hard, both on the original task and our 3-shot augmentation scheme. Size accuracy gives the number of tasks for which the LLM was able to output any grid of the correct size across both datasets.

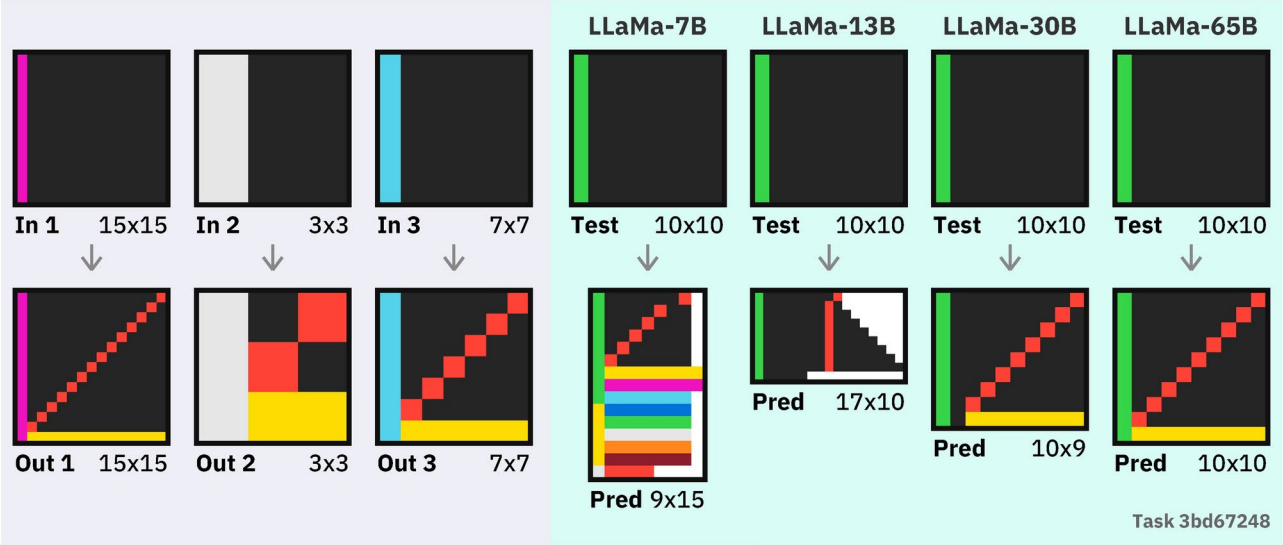

Figure 3. An example task solved by the large-language model LLaMA-65B but not by smaller models; training examples are on the left with predicted answers on the right. We see the smallest two models struggle to produce a coherent answer. The 30B model is almost correct (skipping a row and one yellow pixel), and the largest model solves the task perfectly.

much harder to collect, although some promising work on Large Vision Models has shown abilities on other reasoning tasks[44].

There are several existing approaches that could potentially improve LLM performance. Prompt engineering involves manipulating the prompt format to improve performance on downstream tasks, and can have a large performance gap (the extent to which this could be done was limited by cost considerations). Additionally, we only evaluate the *zero-shot* performance of LLMs; it is possible that advances such as Low-Rank Adaptation (LoRA) fine-tuning of models[45] could be harnessed to condition an LLM to perform better on ARC-specific tasks.

LLMs can be used to propose hypotheses that can then be refined by task-specific solvers[46]. This has recently been applied to a smaller version of ARC called MiniARC[46]. Several other approaches have been suggested, such as learning rules in multiple stages using LLMs[47], using a multi-agent LLM system[48] or representing ARC problems as graphs and using these as a prompt to an LLM[49].

Textual descriptions of how humans solve problems in ARC could also inform future machine learning approaches[50,51]. Studying how humans solve problems in ARC and communicate these solutions to each other in natural language[50] may help us design better AI approaches to solve ARC. Accurate reconstructions of of how humans solve problems in ARC[52,53] can be used to train models[54], which may also open up new approaches

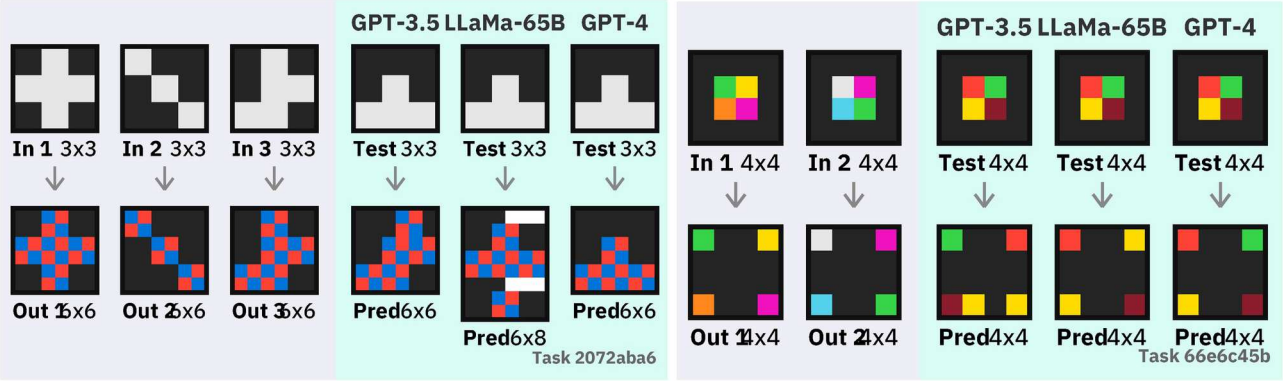

Figure 4. Two ARC-Hard tasks where only GPT-4 produced a correct solution. While the other LLMs often produce somewhat close matches on first inspection, an exactly correct solution is required to solve a task.

to potentially augment human performance with LLMs. Some have suggested that LLMs may be capable of deductive, inductive and abductive learning[55], which is promising.

We discuss some of these further in Section "Discussion".

Ensemble methods

In this work, we evaluate some very different systems as ARC solvers. One interesting question arises when looking at the differences between these systems: are they solving the same tasks, or does each system have different strengths?

Machine learning research has looked at ensemble approaches, where multiple models are combined to produce predictions better than any single model. The success of these approaches relies on *diversity* of models[56], often measured by looking at the correlation of predictions. More diverse models produce more powerful ensembles, even with disparate individual accuracies.

Since we cannot measure correlation between predictions directly, we propose two approaches to measure the similarity between systems attempting ARC, even when models have very different headline performance. First, the overlap between the set of tasks solved by each system (the Szymkiewicz-Simpson coefficient[57])

$$\text{Overlap}(A, B) = \frac{|A \cap B|}{\min(|A|, |B|)}$$

tells us the proportion of tasks in the weaker model solved by the stronger model. When one model solves a subset of the tasks of another, the overlap is 1. This can be seen as a performance-normalised similarity metric: it ignores the overall accuracy difference between the models, and lets us determine if one system is a more powerful version of another, or whether they are solving complementary types of tasks (as opposed to more common metrics such as Jaccard index). We also propose the asymmetric Gain measure,

$$\text{Gain}(A, B) = |A \cup B| - |A|,$$

which gives the additional tasks solvable by adding one model to another model - hence, it reflects the *unique skills* of each model; tasks that cannot be solved by the other model in the pair. The asymmetry means that we can see which model performs the best.

Figure 5 shows both metrics for our systems' predictions on the ARC-Easy set, from which we can make a few key takeaways.

First, we see that LLMs tend to solve different tasks than DSL-based solutions: the overlap between these models is relatively low. In particular, our DreamCoder and GPT-4 solutions, which have comparable accuracy independently, have only a 37% overlap between *which* tasks they can solve - this suggests an ensemble which can effectively select between approaches *could lead to an almost doubling in performance*.

Conversely, we see that different LLMs tend to have similar predictions (with larger models tending to solve a superset of problems solved by their smaller counterparts). Likewise, we see that DreamCoder and Icecuber solve similar types of task: with DreamCoder uniquely solving only 4 tasks and a 94% overlap between them.

Building a heterogenous high-performance ensemble

For classification and regression tasks, ensembles can be built effectively by a weighted average of each system's predictions[56]. In our case, with only exact solutions scoring credit, interpolating solutions would be unlikely to work well.

Instead, we build a *voting* ensemble of systems: on each task, each system can propose some output grids which they 'vote' for, added to a priority queue. When multiple systems generate the same prediction, votes are summed, increasing their priority. Since ARC allows three guesses, we select the three output grids with the most votes. Another difficulty to overcome is the heterogeneity of our systems. For example, DreamCoder only

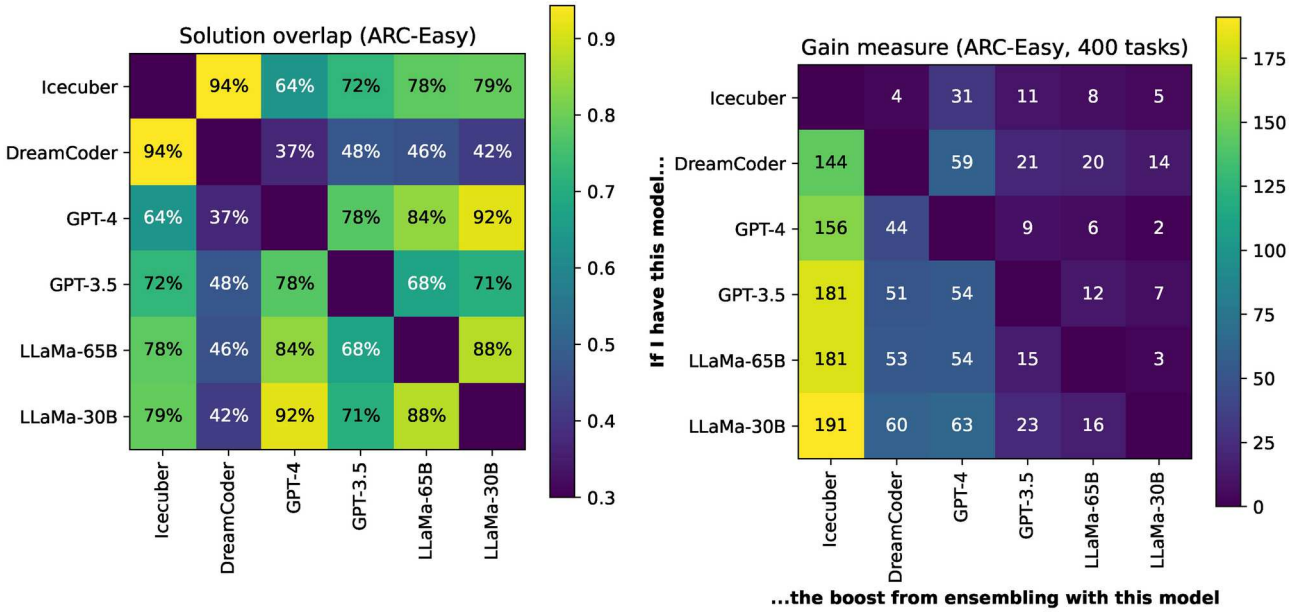

Figure 5. Comparison of the solution similarity of our systems, showing solution overlap (left panel) and gain measure (right panel) between each system: this tells us how similar the predictions are, and how much one could expect to gain by combining systems. Systems are ordered by overall performance, so the top-right triangle in the panel on the right shows the unique skills of the lesser system. The panel on the left shows that DreamCoder and GPT-4 have only 37% overlap between the tasks which they can solve.

produces a prediction when it solves all training tasks: a DreamCoder prediction should be prioritised for the final list. On the other hand, when combining LLMs, we would like to provide higher weight to LLMs with better performance.

We assign distinct weights to each algorithm to build our ensembles: 20 to DreamCoder, 3 to each GPT-4 prediction, and [8, 4, 4] to the three IceCuber predictions for each task (based on inspection of the above figures and minimal manual tuning). Additionally, we discard any LLM predictions which are not grids, the default prediction from Icecuber (predicted when no solution was found), and de-duplicate predictions from DreamCoder (as it tends to find equivalent programs: e.g., when functions are commutative under composition, we have $f(g(x))$ and $g(f(x))$). This process allows us to select the three most promising predictions from up to 9. The performance of our ensembles are discussed in the next section.

Quantitative results across all methods

We now take stock of each system discussed in this report, looking at headline performance on both ARC datasets. For fairness, all systems were evaluated from scratch using a common framework, following the official ARC evaluation procedure. Each system has access to training examples, and can create a maximum of three predictions for each test example. A task is solved if one of the three attempts are **exactly** correct. Figure 6 and Table 3 show these results.

Our new approaches GPT-4 and DreamCoder attain decent performance, improving over ARGA[21]. Notably, DreamCoder solution solves $3\times$ and $9\times$ more ARC-Easy and ARC-Hard tasks respectively than the existing best implementation by Alford et al.[7].

Our voting ensemble can combine the strengths of different algorithms, with DreamCoder and GPT-4 ensembling to solve **50% more tasks than either system alone**. Moreover, these systems successfully tackled tasks that stumped Icecuber, the previous state-of-the-art solution. Our final ensemble, which incorporates Icecuber, DreamCoder, and GPT-4, manages to solve 57% and 40% of tasks on the ARC-Easy and ARC-Hard datasets respectively.

Across the board, performance on ARC-Easy correlates well with ARC-Hard, with most systems solving $2-4$ times more easy tasks - a clear outlier here is Icecuber, which solves 160 hard tasks. This could be partially attributed to the fact that **the Icecuber DSL was built on hand-crafted solutions to 100 hard tasks**[6], meaning that the developer designed the language to be good on these tasks.

Collectively, these results show two approaches using neural networks for abstraction and reasoning. We highlight the need for diverse and complementary methods that, when combined, can lead to superior performance.

We note, however, that Icecuber, the current state-of-the-art that uses hand-crafted brute-force search, still retains a commanding lead over neural-network-based solutions.

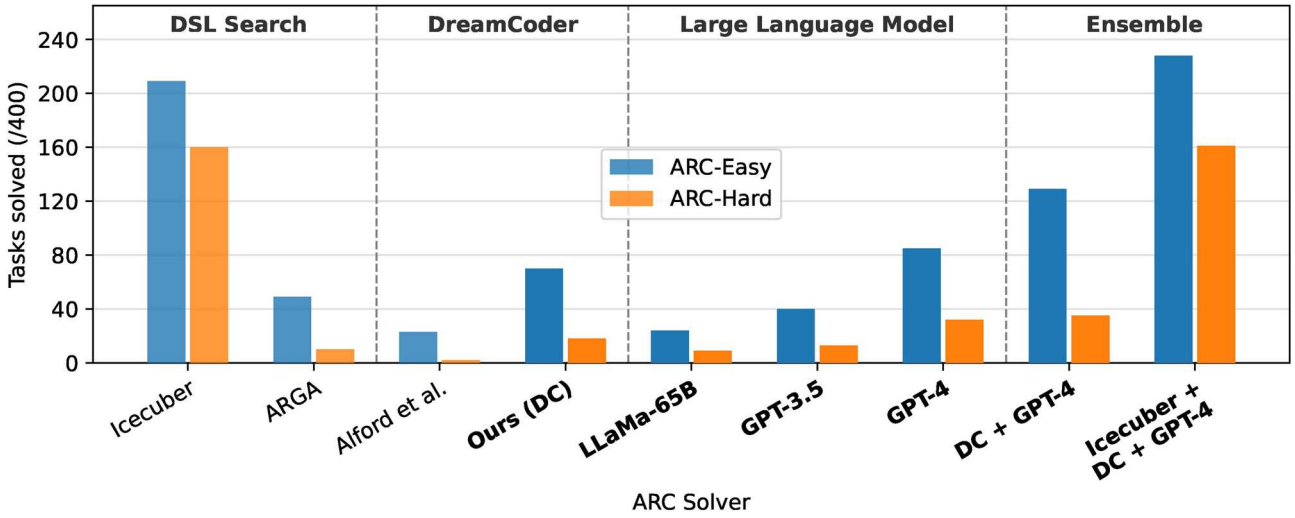

Figure 6. ARC tasks solved by each system in this work on the two available ARC datasets (see Table 3); results from this work in bold. The fully handcrafted Icecuber solver remains the best single solution, with GPT-4 obtaining slightly better performance than DreamCoder. We achieve dramatic gains by ensembling solutions. For the sake of brevity, DreamCoder is abbreviated to DC and Abstract Reasoning with Graph Abstractions[21] is abbreviated to ARGA.

Type	System	ARC-Easy/400	ARC-Hard/400
DSL Search (Section "Previous work")	Icecuber[6]	209	160
	ARGA[21]	49	10
DreamCoder (Section "Adaptation of DreamCoder")	Alford et al.[7]	23	2
	Ours (DC)	70	18
LLM (Section "A framework for large-language models")	**LLaMA-65B**[10]	24	9
	GPT-3.5[42]	40	13
	GPT-4[9]	85	32
Ensemble (Section "Ensemble methods")	**DC + GPT-4**	129	35
	Icecuber + **DC + GPT-4**	**228**	**161**

Table 3. Comparison of headline performance for the systems discussed in this work; all systems have three attempts per task. Systems in **bold** were designed in this project. For the sake of brevity, DreamCoder is abbreviated to DC and Abstract Reasoning with Graph Abstractions[21] is abbreviated to ARGA.

Discussion

Overview

In this work, we have applied a modern machine learning lens to a touchstone problem in AI: searching for systems that can perform *broad generalisation* to abstraction and reasoning tasks. Overall, we find that complex human-crafted solutions based on domain specific languages (DSL) search still provide the best known single-system performance. However, we demonstrate two interesting machine-learning-based solutions, which provide a concrete research direction to make more robust learning machines.

Limitations of our approach

There are several types of ARC tasks which our DreamCoder adaptation cannot solve. For example, *copy-paste* tasks, which rely on copying a pattern from training outputs to test outputs, cannot be solved. This is because DreamCoder considers each input/output pair in isolation, and cannot copy objects from one grid to another. *In-painting* tasks can likely be solved much more effectively by a dedicated algorithm, which tries to identify the tiling or symmetry pattern required to fill in the missing pixels.

Our method for adapting large-language models to ARC is also exploratory, and there is likely to be significant gains attainable from the use of different models, different prompt techniques (for example that allow LLMs to solve problems larger than their context window), or systems that combine mechanistic reasoning with LLMs. We discuss some avenues in Section "Discussion"

Additionally, in this work we report results on the publicly available splits of the ARC dataset (referred to as ARC-Easy and ARC-Hard), as opposed to the private test set hosted on Kaggle[5]. This creates a risk that the hand-crafted DSL (or indeed, training data for an LLM) is biased towards the training set, and so is a significant limitation of the evaluation in this work. The evaluation environment for the private test set has a strict CPU-

time limit as well as no internet access, which would have precluded running DreamCoder or any of the LLMs. In the future, we would like to see a system of improved access to this dataset with more compute availability so that more reliable numbers can be reported.

Qualitative comparison of methods

We conducted an informal analysis of the weaknesses of each algorithm by looking at task successes and failures with reference to the taxonomy in Supplementary Fig. 8. Broadly, we found that Icecuber and DreamCoder had very similar strengths and weaknesses: they were very good at solving tasks involving objects, gravity, symmetry and other tasks which solvable within the DSL, even if the solution involved composing. The DreamCoder DSL is more narrow than that of Icecuber, and so many of the tasks solved by Icecuber were simply not possible in the DreamCoder DSL, suggesting that DSL design is extremely important for program-synthesis approaches. There were a few cases of DreamCoder solving tasks that should have been solvable by Icecuber given sufficient search depth, suggesting that a guided search is useful. DSL-based approaches are unable to solve tasks such as those require copying objects or colours between input and output examples (such as lookup tables), or those requiring learning a more complex pattern (in-painting).

On the other hand, LLMs were very good at tasks where the input and output are very similar (such as inpainting and denoising), and those where the sizes of grids are small. This makes sense as larger problems can be 100x longer than smaller problems, and reasoning over longer inputs can be a challenge for LLMs, as well as producing longer outputs with no mistakes. For a DSL approach, the grid size does not affect the difficulty of the problem. They also struggled at tasks that required 2D manipulation, likely due to the way that LLMs operate on 1D sequences (as discussed in Section "Applicability of LLMs to ARC problems"). Interestingly, there were no obvious categories where LLMs were perfect or totally fail.

In Supplementary Fig. 7, we provide several examples of exemplar tasks that IceCuber, DreamCoder and GPT-4 could solve respectively that the others could not.

Future work

Research on computational abstraction and reasoning is far from complete, so we consider some future directions suggested by our conclusions.

While we improve the accuracy of DreamCoder, it is still far behind the handcrafted Icecuber DSL search[6]. One future improvement would be to extend DreamCoder with the unique features of Icecuber, such as colour normalisation and, in particular, the greedy stacker, which combines intermediate solutions working backwards from the output grid. Execution-guided program synthesis would be one way to achieve a similar effect, where intermediate solutions are scored based on their similarity to the final output and used to guide the search, replacing all-or-nothing evaluation.

There are several research avenues that could improve LLM performance. We only evaluate the in-context learning performance of LLMs; advances such as low-rank adaptation fine-tuning[45] could precondition a model further on ARC priors. Additionally, chain-of-thought prompting and similar techniques, where the LLM is first required to explain the training examples before applying it to the test example, may also improve performance[58,59].

Early experiments have been made with Large Vision Models[44] which suggest that visual datasets are sufficient to train a model with emergent reasoning abilities that could be applied to ARC. While these models warrant investigation, challenges are likely to include the requirement for pixel-perfect outputs as well as collecting a suitable mix of training data.

Finally, an ARC2 dataset is in development by Chollet and Lab42[60], aiming to crowd-source 5,000 tasks. ARC2 aims to be more diverse, with "no specific task type used more than once". This is likely to make ARC even harder, and require greater generalisation abilities.

Concluding remarks

Our implementation of DreamCoder achieves a $3.5\times$ improvement in tasks solved across ARC over the previous best implementation[7,24], through a wide range of improvements, such as the PeARL language and the use of a neural-network recognition model (Section "Adaptation of DreamCoder").

Additionally, we introduced a framework to transform visual ARC problems into a text completion domain suitable for large language models (LLMs), and found that the largest available LLMs trained across huge corpora have sufficiently transferable abilities to begin to solve ARC problems; their accuracy can also be effectively doubled with suitable augmentation. Our results suggest that even larger models could continue to improve attainable accuracy on ARC (Section "A framework for large-language models"). While previous solutions largely rely on human-designed priors in the form of a DSL, the allure of LLMs is that they instead use neural-networks in an end-to-end fashion, and all reasoning priors come from training data.

We also show the effectiveness of ensembling solutions: while each algorithm may specialise on a certain type of task, systems with low false-positive rates can be effectively combined (Section "Ensemble methods"). Rather than trying to create a system that *beats* Icecuber, a useful direction would be to try to build systems that complement the state-of-the-art, leading to a multipronged solution with better generalisation. In particular, we already find that LLMs solve many problems not suitable for domain-specific languages.

It is sobering to observe that both the previous state-of-the-art solutions and the approaches developed in this work achieve only modest accuracy on the ARC dataset (approximately 40% of tasks solved on the ARC-Hard dataset), which is still much less than what humans can solve[2,50,53,61]. This suggests that current machine learning methods may not yet be powerful enough to develop complex concepts and abstractions for the hardest tasks. Significantly more work will be required before we can consider ARC solved. Our ensemble analysis indicates that a diversity of methods might be necessary to solve problems in ARC.

We hope our work will serve as the foundation for future approaches. Neuro-symbolic methods such as DreamCoder may benefit from many improvements, such as execution-guided synthesis[62] and a stronger DSL mimicking Icecuber[11]. Likewise, there are many potential avenues to improve LLM performance, such as chain-of-thought prompting methods[58,59], fine-tuning[45] or constrained generation. Furthermore, studying how humans solve ARC tasks[50,63] could inform the development of future AI systems. Mimicking the diverse strategies humans use to solve ARC may prove helpful in designing next-generation abstraction and reasoning algorithms.

Finally, we see that broad generalisation is still an incredibly challenging goal for AI, even for the most advanced systems. ARC is proving to be an informative benchmark; still unsolved, and motivating research after four years, while at the same time well-defined and objectively evaluable. We hope that research on ARC will one day bring about a broad generalisation machine, in the way that MNIST and ImageNet brought about the rise of modern deep neural networks.

Data availability

All data generated or analysed during this study are included in this published article and its supplementary information files.

Received: 20 April 2024; Accepted: 18 September 2024
Published online: 13 November 2024

References

1. Bongard, M. M., Hawkins, J. K. & Cheron, T. *Pattern Recognition* (Spartan Books, 1968).
2. Chollet, F. On the measure of intelligence. arXiv: 1911.01547. (2019)
3. Jumper, J. *et al.* Highly accurate protein structure prediction with AlphaFold. *Nature* **596**, 583–589 (2021).
4. Silver, D. *et al.* Mastering the game of Go with deep neural networks and tree search. *Nature* **529**, 484–489 (2016).
5. Chollet, F., Tong, K., Reade, W. & Elliott, J. Abstraction and reasoning challenge (2020). https://kaggle.com/competitions/abstraction-and-reasoning-challenge.
6. Wind, J. S. 1st place solution + code and official documentation. Kaggle Forums (2022). https://www.kaggle.com/c/abstraction-and-reasoning-challenge/discussion/154597.
7. Alford, S. A Neurosymbolic Approach to Abstraction and Reasoning. Master's thesis, Massachusetts Institute of Technology (2021).
8. Brown, T.B., Mann, B., Ryder, N., Subbiah, M. & Kaplan, J. et al. Language models are few-shot learners. In: Larochelle H, Ranzato M, Hadsell R, Balcan M, Lin H, editors, 33rd Annual Conference on Neural Information Processing Systems (NeurIPS) (2020).
9. OpenAI. GPT-4 technical report. arXiv:2303.08774. (2023)
10. Touvron, H., Lavril, T., Izacard, G., Martinet, X. & Lachaux, M. A. et al. LLaMA: Open and efficient foundation language models. arXiv:2302.13971 (2023).
11. Wind, J. S. DSL solution to the ARC challenge. https://github.com/top-quarks/ARC-solution.
12. Hofstadter, D. R. Gödel, Escher, Bach: An Eternal Golden Braid. Basic Books (1979).
13. Foundalis, H. Index of bongard problems. https://www.foundalis.com/res/bps/bpidx.htm (2007).
14. Halme, A. Bongard solvers. https://notes.fringeling.com/BongardSolvers/ (2020).
15. Kharagorgiev, S. Solving Bongard problems with deep learning. https://k10v.github.io/2018/02/25/Solving-Bongard-problems-with-deep-learning/ (2018).
16. Depeweg, S., Rothkopf, C.A. & Jäkel, F. Solving Bongard problems with a visual language and pragmatic reasoning. arXiv:1804.04452 (2018).
17. Foundalis, H. E. Phaeaco: A Cognitive Architecture Inspired by Bongard's problems. Ph.D. thesis, Indiana University (2006).
18. Geirhos, R., Temme, C.R.M., Rauber, J., Schütt, H.H., Bethge, M. et al. Generalisation in humans and deep neural networks. In: Bengio S, Wallach HM, Larochelle H, Grauman K, Cesa-Bianchi N et al., editors, 31st Annual Conference on Neural Information Processing Systems (NeurIPS) 2018. pp. 7549–7561. (2018)
19. Legg, S. & Hutter, M. A collection of definitions of intelligence. arXiv:0706.3639 (2007).
20. Raven, J. C. Mental tests used in genetic studies: The performance of related individuals on tests mainly educative and mainly reproductive. Master's thesis, University of London (1936).
21. Xu, Y., Khalil, E. B. & Sanner, S. Graphs, constraints, and search for the abstraction and reasoning corpus. arXiv:arXiv2210.09880 (2022).
22. Golubev, V. 3rd place[short preview+code]. Kaggle Forums. https://www.kaggle.com/competitions/abstraction-and-reasoning-challenge/discussion/154305 (2019).
23. Golubev, V. 7 solved tasks via trees. Kaggle Kernels. https://www.kaggle.com/code/golubev/7-solved-tasks-via-trees/notebook (2019).
24. Banburski, A., Ghandi, A., Alford, S., Dandekar, S. & Chin, P. et al. (2020) Dreaming with ARC. Technical report, Center for Brains, Minds and Machines (CBMM). https://dspace.mit.edu/handle/1721.1/128607.
25. Ellis, K., Wong, C., Nye, M., Sable-Meyer, M. & Cary, L. et al. DreamCoder: Growing generalizable, interpretable knowledge with wake-sleep Bayesian program learning. arXiv:2006.08381 (2020).
26. Muggleton, S. H. Inductive logic programming. *N. Gener. Comput.* **8**, 295–318 (1991).
27. Dayan, P., Hinton, G. E., Neal, R. M. & Zemel, R. S. The Helmholtz machine. *Neural Comput.* **7**, 889–904 (1995).
28. Ellis, K. Dreamcoder official code. GitHub. https://github.com/ellisk42/ec (2022).
29. Alford, S. bidir-synth: DreamCoder and bidirectional program synthesis on ARC. https://github.com/simonalford42/bidir-synth.git (2021).
30. He, K., Zhang, X., Ren, S. & Sun, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **37**, 1904–1916 (2015).
31. Lin, M. & Chen, Q. & Yan, S. Network in network. In *2nd International Conference on Learning Representations, ICLR 2014* (eds Bengio, Y. & LeCun, Y.) 14–16 (AB, Canada, April, Banff, 2014) (2014).
32. Long, J., Shelhamer, E. & Darrell, T. Fully convolutional networks for semantic segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015. IEEE Computer Society, pp. 3431–3440 (2015).
33. Yu, F. & Koltun, V. Multi-scale context aggregation by dilated convolutions. In *4th International Conference on Learning Representations, ICLR 2016* (eds Bengio, Y. & LeCun, Y.) 2–4 (Puerto Rico, May, San Juan, 2016) (2016).
34. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. In: Bengio Y, LeCun Y, editors, 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015. http://arxiv.org/abs/1412.6980 (2015).

35. Ouyang, L., Wu, J., Jiang, X., Almeida, D. & Wainwright, C.L. et al. Training language models to follow instructions with human feedback. arXiv: 2203.02155 (2022).
36. Webb, T., Holyoak, K. J. & Lu, H. Emergent Analogical Reasoning in Large Language Models. arXiv:2212.09196 (2022).
37. Mitchell, M. Can GPT-3 make analogies? https://medium.com/@melaniemitchell.me/can-gpt-3-make-analogies-16436605c446 (2020).
38. Mitchell, M. *Analogy-Making as Perception* (MIT Press, 1993).
39. Marcus, G. & Davis, E. GPT-3, Bloviator: OpenAI's language generator has no idea what it's talking about. MIT Technology Review (2020).
40. Frantar, E., Ashkboos, S., Hoefler, T. & Alistarh, D. GPTQ: Accurate post-training quantization for generative pre-trained transformers. arXiv: 2210.17323 (2023).
41. OpenAI. TikToken. https://github.com/openai/tiktoken (2023).
42. OpenAI. Introducing ChatGPT. https://openai.com/blog/chatgpt (2023).
43. Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E. & Cai, T. et al. Training compute-optimal large language models. arXiv:2203.15556 (2022).
44. Bai, Y., Geng, X., Mangalam, K., Bar, A. & Yuille, A. et al. Sequential modeling enables scalable learning for large vision models. arXiv:2312.00785 (2023).
45. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z. & Li, Y. et al. LoRA: Low-rank adaptation of large language models. In: The Tenth International Conference on Learning Representations, ICLR (2022).
46. Qiu, L., Jiang, L., Lu, X., Sclar, M. & Pyatkin, V. et al. Phenomenal yet puzzling: Testing inductive reasoning capabilities of language models with hypothesis refinement. arXiv:2310.08559 (2023).
47. Zhu, Z., Xue, Y., Chen, X., Zhou, D. & Tang, J. et al. Large language models can learn rules. arXiv:2310.07064 (2023).
48. Tan, J., Min, C. & Motani, M. Large language model (LLM) as a system of multiple expert agents: An approach to solve the abstraction and reasoning corpus (ARC) challenge. arXiv:2310.05146 (2023).
49. Xu, Y., Khalil, E. B. & Sanner, S. Graphs, constraints, and search for the abstraction and reasoning corpus. *Proceedings of the AAAI Conference on Artificial Intelligence* **37**, 4115–4122 (2023).
50. Acquaviva, S., Pu, Y., Kryven, M., Sechopoulos, T., & Wong, C. et al. Communicating natural programs to humans and machines. arXiv:2106.07824 (2021).
51. Wang, R., Zelikman, E., Poesia, G., Pu, Y. & Haber, N. et al. Hypothesis search: Inductive reasoning with language models. arXiv:2309.05660 (2023).
52. Kim, S., Phunyaphibarn, P., Ahn, D. & Kim, S. Playgrounds for abstraction and reasoning. In: NeurIPS Workshop on Neuro Causal and Symbolic AI (2022).
53. Johnson, A., Vong, W. K. & Lake, B. M., Gureckis TM Fast and flexible: Human program induction in abstract reasoning tasks. Proceedings of the 43rd Annual Meeting of the Cognitive Science Society: Comparative Cognition: Animal Minds, CogSci 2021: 2471–2477 (2021).
54. Park, J., Im, J., Hwang, S., Lim, M. & Ualibekova, S. et al. Unraveling the arc puzzle: Mimicking human solutions with object-centric decision transformer (2023). arXiv:2306.08204.
55. Liu, E., Neubig, G. & Andreas, J.: An incomplete loop: Deductive, inductive, and abductive learning in large language models (2024). arXiv:2404.03028.
56. Bober-Irizar, M., Husain, S., Ong, E. J. & Bober, M. Cultivating DNN diversity for large scale video labelling. In: Conference on Computer Vision and Pattern Recognition, CVPR 2017, Youtube-8M Workshop (2017).
57. Vijaymeena, M. & Kavitha, K. A survey on similarity measures in text mining. *Mach. Learn. Appl.: Int. J.* **3**, 19–28 (2016).
58. Wei, J., Wang, X., Schuurmans, D., Bosma, M. & Ichter, B. et al.: Chain-of-thought prompting elicits reasoning in large language models. In: Thirty-sixth Conference on Neural Information Processing Systems (NeurIPS) (2022).
59. Yao, S., Yu, D., Zhao, J., Shafran, I. & Griffiths, T. L. et al. Tree of thoughts: Deliberate problem solving with large language models (2023). arXiv:2305.10601.
60. Lab42 (2023). ARCreate: Crowdsourcing ARC 2. https://arc-editor.lab42.global/.
61. Mitchell, M., Palmarini, A. B. & Moskvichev, A. Comparing Humans, GPT-4, and GPT-4V on abstraction and reasoning tasks (2023). arXiv:2311.09247v2.
62. Chen, X., Liu, C. & Song, D. Execution-guided neural program synthesis. In: International Conference on Learning Representations (2019). https://openreview.net/forum?id=H1gfOiAqYm.
63. Peterson, J. C., Bourgin, D. D., Agrawal, M., Reichman, D. & Griffiths, T. L. Using large-scale experiments and machine learning to discover theories of human decision-making. *Science* **372**, 1209–1214 (2021).

Acknowledgements
We acknowledge the help and support of the Accelerate team. We are especially grateful to Neil Lawrence and Carl Henrik Ek for fruitful discussions and feedback.

Author contributions
MBI carried out the implementation and analysis, participated in the design of the study and wrote the manuscript. SB carried out the analysis, participated in the design of the study and wrote the manuscript. SB directed the study. All authors gave final approval for publication.

Funding
SB acknowledges funding from the Accelerate Programme for Scientific Discovery Research Fellowship. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript. The views expressed are those of the authors and not necessarily those of the funders.

Declarations

Competing interests
All authors declare they have no conflicts of interest to disclose.

Ethical approval
No ethics approval was necessary.

Additional information

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-024-73582-7.

Correspondence and requests for materials should be addressed to S.B.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.