# SynDaCaTE: A Synthetic Dataset For Evaluating Part-Whole Hierarchical Inference

**Jake Levi** [1] **Mark van der Wilk** [1]

## Abstract

Learning to infer object representations, and in particular part-whole hierarchies, has been the focus of extensive research in computer vision, in pursuit of improving data efficiency, systematic generalisation, and robustness. Models which are *designed* to infer part-whole hierarchies, often referred to as capsule networks, are typically trained end-to-end on supervised tasks such as object classification, in which case it is difficult to evaluate whether such a model *actually* learns to infer part-whole hierarchies, as claimed. To address this difficulty, we present a SYNthetic DAtaset for CApsule Testing and Evaluation, abbreviated as SynDaCaTE, and establish its utility by (1) demonstrating the precise bottleneck in a prominent existing capsule model, and (2) demonstrating that permutation-equivariant self-attention is highly effective for parts-to-wholes inference, which motivates future directions for designing effective inductive biases for computer vision.

## 1. Introduction

In recent years, a dominant trend in machine learning research has been towards finding model architectures that improve with scale (Vaswani et al., 2017), and scaling them up (Sutton, 2019) with billions of parameters (Brown et al., 2020), trillions of tokens (Yang et al., 2024), hundreds of millions of dollars (Maslej et al., 2025), and significant environmental cost (Bengio et al., 2025). There are now strong and increasingly growing financial and environmental incentives to develop machine learning algorithms that achieve similar performance more efficiently.

It is well established that data efficiency can be improved by using more appropriate **inductive biases** (Du et al., 2018; Li et al., 2021), and that models with better data efficiency typically also train faster (Hardt et al., 2016). Developing better inductive biases is therefore a promising strategy for improving the general efficiency of machine learning algorithms. Substantial evidence from the cognitive sciences suggests that object representations (Kahneman et al., 1992) and in particular part-whole hierarchies (Biederman, 1987) are ubiquitous in the human visual system. This has further motivated extensive research in computer vision focusing on object-centric learning (Burgess et al., 2019; Locatello et al., 2020) and models which are designed to learn part-whole hierarchies (Sabour et al., 2017; Hinton et al., 2018), commonly referred to as capsule networks (Hinton et al., 2011).

Despite much early excitement and published work focusing on capsule networks, they have largely been forgotten in favour of convolutional (He et al., 2016; Liu et al., 2022) and attention-based (Dosovitskiy et al., 2020; Liu et al., 2021) vision models. Does this mean that exploring visual part-whole hierarchies as an inductive bias for improving data efficiency is a hopeless endeavour? There is one reason to be hopeful. While it is *claimed* that capsule models learn to infer part-whole hierarchies, to the best of our knowledge, there is no empirical evidence to suggest that they *actually* do this efficiently. Before we can determine whether part-whole hierarchies are a useful inductive bias for improving data efficiency in computer vision, we must first (1) define precisely *what it means* to infer a part-whole hierarchy, and (2) establish a procedure to determine which models have the *capacity* to efficiently learn to infer part-whole hierarchies.

In order to evaluate the accuracy with which a model can infer part-whole hierarchies, we must have access to ground-truth part information, which is typically not available in natural visual datasets. We could manually label an existing visual dataset with part information, but this would be undesirably expensive and ambiguous. Instead, we introduce a synthetic

[1]Department of Computer Science, University of Oxford. Correspondence to: Jake Levi <jake.levi@stcatz.ox.ac.uk>.

*Accepted at Methods and Opportunities at Small Scale (MOSS), ICML 2025*, Vancouver, Canada.

dataset that has optional ground-truth part-information built in. Our proposed dataset is *elementary*, but nonetheless it enables us to draw *unique conclusions*, such as pinpointing the bottleneck in an existing capsule model, and motivating future directions for designing effective inductive biases. More specifically, we make the following **primary contributions**:

1. (§2.1) We introduce a framework which clarifies the meaning of part-whole hierarchical inference.
2. (§2.2) We present a SYNthetic DAtaset for CApsule Testing and Evaluation, abbreviated as SynDaCaTE.
3. (§3) We use our SynDaCaTE dataset to demonstrate that:
   (a) The bottleneck in an existing capsule model (Sabour et al., 2017) is inferring parts from an image (rather than inferring wholes from parts), which has implications for designing capsule networks (discussed further in §4).
   (b) Given explicit part-information, this capsule model is no more efficient than a CNN at inferring wholes from parts.
   (c) A permutation-equivariant SetTransformer (Lee et al., 2019) is a strong baseline for inferring wholes from parts, which motivates future directions for designing effective inductive biases (discussed further in §A).

## 2. Methods

### 2.1. A Framework For Mereological Inference

In general, an image may depict a set of several top-level objects, each of which may be considered a "whole" object containing a set of "parts". Each of those parts may be considered a whole containing its own set of parts, and so on for several layers in a "part-whole hierarchy". Each object (part or whole) in an image may be described by a discrete "class" label and a continuous "pose" vector. The class describes the type of the object, and the pose describes everything which is needed to render the object as seen in the image given its class, which might include $(x, y)$-position, size, rotation, brightness, and so on. In this simplified framework we do not consider noise, or interactions between objects such as reflection, illumination, or shadow. We define the **generalised pose** of an object to be a one-hot encoding of its class concatenated with its pose. We will say "inferring an object" as a shorthand for inferring its generalised pose[1]. We define **inferring a part-whole hierarchy** as inferring all objects (wholes and parts) from an image.

The formal study of part-whole hierarchies is known as *mereology* (Cotnoir & Varzi, 2021), and we will refer to inferring part-whole hierarchies as *mereological inference*. We will refer to models which have the capacity to efficiently learn mereological inference as having *mereological capacity*. In general, a model which has mereological capacity *must* be able to efficiently learn to solve two distinct sub-tasks:

1. **Image-to-parts**: infer a set of parts $\mathcal{P}$ from an image $I \in \mathbb{R}^{C \times H \times W}$.
2. **Parts-to-wholes**: infer a set of wholes $\mathcal{W}$ from a set of parts $\mathcal{P}$.

### 2.2. The SynDaCaTE Dataset

As previously mentioned, we want to determine which models have mereological capacity, and in order to do so we need ground-truth part information. To this end, we introduce a SYNthetic DAtaset for CApsule Testing and Evaluation, abbreviated as SynDaCaTE. Example images from different SynDaCaTE tasks are shown in Figure 5 in §C.

In total across all SynDaCaTE tasks, there are 21 types of object in three categories, which are lines, characters, and words. A full description of the dimensionality and meaning of the poses of objects in each category is presented in §C.1.

The sampling procedure for images in SynDaCaTE is described in §C.2. When each image is sampled, we can control various parameters (such as the maximum and minimum number of objects in an image), and observe various types of data about objects, parts, and images. By appropriately controlling parameters and computing inputs and targets from the available data, we can define many different tasks, which can be used to explore a variety of different research questions. We consider several descriptively named tasks including ImToClass, ImToParts, PreTrainedPartsToClass, PartsToChars, and PartsToClass. All tasks have 60k synthetically generated training samples and 10k test samples. A full description of these and some other tasks, including the type and meaning of inputs and targets in all tasks, is included in §C.3.

[1]Inferring generalised pose is more general than object classification, detection, and segmentation. A class, bounding box, and segmentation mask could all be computed from a generalised pose, however the generalised pose (in general) contains more information than is available in those labels.

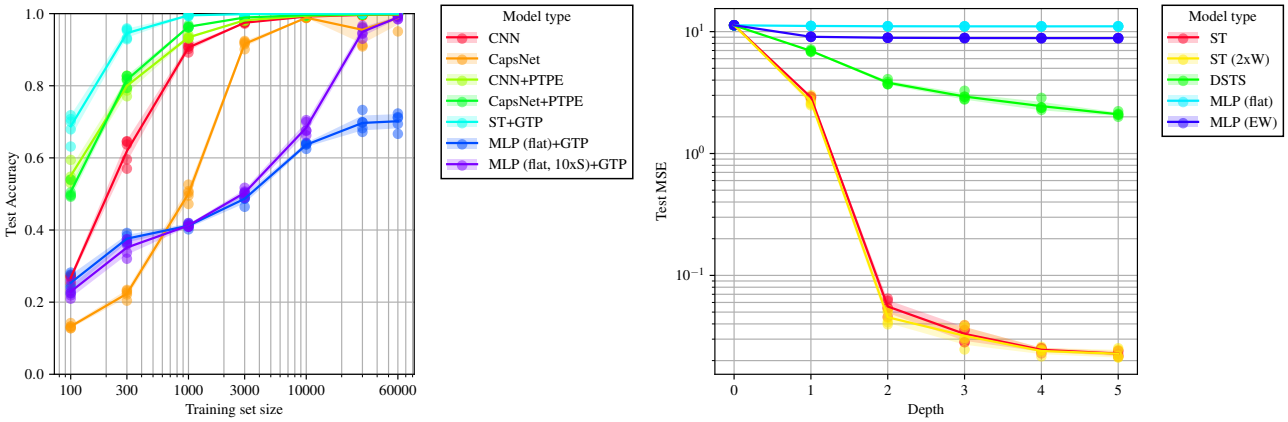


*Figure 1.* **Left**: comparing data efficiency of various models trained on single-object classifcation, from images (ImToClass), pre-trained part encodings (PreTrainedPartsToClass, "+PTPE"), and ground-truth sets of parts (PartsToClass, "+GTP"). **Right**: comparing MSE of various models as a function of depth, trained to predict sets of wholes from sets of parts (PartsToChars). **Abbreviations**: pre-trained part encodings (PTPE), ground-truth parts (GTP), SetTransformer (ST), DeepSetToSet (DSTS), element-wise (EW), $10\times$ gradient steps (10xS), $2\times$ width (2xW). All experiments are repeated with 5 different random seeds.

### 2.3. Models

Vision models (including capsule networks) are most commonly evaluated on object classification, so we start in §3 by considering ImToClass. For ImToClass and PreTrainedPartsToClass we will compare a prominent capsule architecture (Sabour et al., 2017) against a lightweight modernised CNN architecture, containing a CoordConv layer (Liu et al., 2018) followed by a CNN with ReZero residual connections (He et al., 2016; Bachlechner et al., 2021), depthwise-separable convolutions (Chollet, 2017) with zero-padding to preserve dimensions, inverted bottlenecks (Vaswani et al., 2017; Liu et al., 2022), and a stage-design (Liu et al., 2022) which divides the network into stages, each of which starts with a strided dense convolution followed by several residual blocks. The final stage is followed by global average pooling and a linear layer.

For the ImToParts task we train the same modernised CNN architecture, except without global average pooling.

We study full parts-to-wholes inference in isolation using the PartsToChars task, and compare a SetTransformer (Lee et al., 2019), a modified DeepSet (Zaheer et al., 2017) which predicts a separate output element for each input element (using a permutation-invariant embedding of the input set) that we refer to as DeepSetToSet, an MLP which predicts output set elements independently as an element-wise function of the input set, and an MLP which predicts the flattened output set as a function of the flattened input set. Both MLPs use ReZero residual connections (He et al., 2016; Bachlechner et al., 2021).

We relate our previous results using PartsToClass, comparing a SetTransformer and the flattened MLP. The SetTransformer uses global average pooling across the set-dimension after the final attention and MLP blocks, followed by a linear layer.

### 2.4. Loss Functions

For ImToClass, PreTrainedPartsToClass, and PartsToClass we use the cross-entropy loss. For ImToParts we use the Chamfer MSE loss for set-prediction (Zhang et al., 2019). For PartsToChars we use the MSE loss averaged over the output set.

## 3. Experiments

**ImToClass.** We start by comparing the data-efficiency of our modernised CNN against CapsNet (Sabour et al., 2017) in single-object classification. Both models are trained on the ImToClass task for 5k gradient steps, on subsets of the training set with sample sizes ranging from 100 to 60k. The results are shown in red and orange in Figure 1 (left). Immediately we can see that our CNN is more data-efficient than CapsNet, achieving significantly higher accuracy at smaller dataset sizes.

**ImToParts.** Given that the CapsNet fails to learn efficiently from small sample sizes, we want to understand if the CapsNet is failing in image-to-parts inference, parts-to-wholes inference, or both. To this end, we train CNNs to predict sets of part poses from an image. We train each CNN for 100 epochs on the ImToParts task, using the full 60k training set. Across 5

random seeds, the CNNs reach an average test-set MSE of $0.0269 \pm 0.0026$, with the best model reaching 0.02329. We can qualitatively assess the performance of a model trained on ImToParts by evaluating the model on an image, feeding the predictions of the model to the rendering pipeline used to create the dataset, and comparing against the ground-truth targets. The predictions of the best CNN trained on ImToParts are shown in Figure 2 in §B, and are reasonably accurate, although not perfect. We create the PreTrainedPartsToClass task by iterating through ImToClass, feeding each image through the best CNN trained on ImToParts, and replacing the image with the activations in the final hidden layer of the CNN.

**PreTrainedPartsToClass.** Figure 1 (left) also shows the data-efficiency of (randomly initialised) CNNs and CapsNets trained on PreTrainedPartsToClass for 5k gradient steps, in light green and dark green. When given access to part-information, both models are significantly more data-efficient compared to training from images, and also have very similar performance to each other, which suggests at least three important conclusions:

1. The bottleneck in CapsNet is inferring *parts from images*, rather than inferring wholes from parts (because the decrease in performance after removing pre-trained *part-information* is much worse in CapsNet than in CNN).
2. A naive capsule model (CapsNet) is not substantially more efficient at learning parts-to-wholes inference than a CNN.
3. Part-information provides a useful representation for data-efficient classification, even though the part-representations are noisy, and contain no explicit class-information.

**PartsToChars.** Should we expect that, in general, a CNN (or a CapsNet which is not substantially better) has *the best* possible inductive bias for inferring wholes from parts? We begin to explore this question by turning to the PartsToChars task, in which the input is a randomly ordered ground-truth set of poses of parts of between one and three objects, and the target is the set of generalised poses of the objects. The input is designed to represent some of the challenges faced by the receptive fields of neurons. Firstly, as an object transforms relative to a retina or camera (for example by rotating or moving closer), part-representations may move into the receptive fields of other neurons in more complex ways than simple translation. Secondly, the receptive field of any neuron might observe parts belonging to multiple distinct objects.

Because the input is a randomly ordered set, we primarily compare set-function models trained on this task. We want to know which inductive bias is most effective for this task, and for any model, it is useful to know how performance varies with important hyperparameters such as depth. To this end, we plot test MSE as a function of depth for several models in Figure 1 (right). All models are trained for 100 epochs. The SetTransformer is strikingly effective at this task, outperforming all other baselines by more than an order of magnitude at depths $\geq 2$, but *not* with a single layer of self-attention. Does the SetTransformer rely on some fundamentally useful computation that *needs* $\geq 2$ layers of self-attention, or does the shallow SetTransformer simply not have enough parameters? We address this question by sweeping over a SetTransformer with double the width ($\approx 4\times$ parameters), also shown in Figure 1 (right). Despite the extra parameters, the performance is remarkably insensitive to width, which suggests that $\geq 2$ self-attention layers *are* in some way fundamentally useful. Future work may examine this through the lens of mechanistic interpretability, which has shown that "Induction heads" also arise in transformers with no fewer than two attention layers (Elhage et al., 2021). Rendered predictions from the best SetTransformer and DeepSetToSet trained on PartsToChars are displayed in Figures 3 and 4 in §B.

**PartsToClass.** Finally, we return to data-efficiency and single-object classification, by comparing the performance of a SetTransformer and a flattened MLP on PartsToClass. The results for various training sample sizes after training for 5k gradient steps are shown in Figure 1 (left). Once again, the SetTransformer outperforms all other baselines, especially for small training sample sizes. The flattened MLP with 5k gradient steps does not achieve good accuracy for any training sample size. Using 50k gradient steps improves performance for larger but not smaller training sample sizes, which aligns with theoretical results connecting generalisation performance to convergence speed (Hardt et al., 2016).

## 4. Conclusion

Although our proposed SynDaCaTE dataset is elementary, its thoughtful design has allowed us to draw unique conclusions about an existing capsule model, and about the effectiveness of permutation-equivariant SetTransformers in parts-to-wholes inference. Our conclusions about CapsNet have strong implications for future work exploring capsule networks, including that image-to-parts inference should be considered explicitly and empirically (possibly using our dataset) and not taken for granted. Our findings about parts-to-wholes inference with SetTransformers motivate future directions for designing effective inductive biases for computer vision, which (because of space constraints) we discuss further in §A.

## Reproducability

Our code is available at <https://github.com/jakelevi1996/syndacate-public>. We provide a full description of hyperparameters in §D.

## Acknowledgements

We would like to thank Christian Rupprecht and Christian Schroeder de Witt for their helpful feedback, discussion, and advice. We would like to thank the reviewers for their helpful reviews. This work was supported by the EPSRC Centre for Doctoral Training in Autonomous Intelligent Machines and Systems (grant number EP/S024050/1).

## References

Bachlechner, T., Majumder, B. P., Mao, H., Cottrell, G., and McAuley, J. Rezero is all you need: Fast convergence at large depth. In *Uncertainty in Artificial Intelligence*, pp. 1352–1361. PMLR, 2021.

Bahdanau, D., Murty, S., Noukhovitch, M., Nguyen, T. H., de Vries, H., and Courville, A. Systematic generalization: what is required and can it be learned? *arXiv preprint arXiv:1811.12889*, 2018.

Bengio, Y., Mindermann, S., Privitera, D., Besiroglu, T., Bommasani, R., Casper, S., Choi, Y., Fox, P., Garfinkel, B., Goldfarb, D., et al. International ai safety report. *arXiv preprint arXiv:2501.17805*, 2025.

Biederman, I. Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2):115, 1987.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Burgess, C. P., Matthey, L., Watters, N., Kabra, R., Higgins, I., Botvinick, M., and Lerchner, A. Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019.

Chollet, F. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.

Cotnoir, A. J. and Varzi, A. C. *Mereology*. Oxford University Press, 2021.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Du, S. S., Wang, Y., Zhai, X., Balakrishnan, S., Salakhutdinov, R. R., and Singh, A. How many samples are needed to estimate a convolutional neural network? *Advances in Neural Information Processing Systems*, 31, 2018.

Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., DasSarma, N., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., and Olah, C. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

Hardt, M., Recht, B., and Singer, Y. Train faster, generalize better: Stability of stochastic gradient descent. In *International conference on machine learning*, pp. 1225–1234. PMLR, 2016.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hinton, G. E., Krizhevsky, A., and Wang, S. D. Transforming auto-encoders. In *Artificial Neural Networks and Machine Learning–ICANN 2011: 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I 21*, pp. 44–51. Springer, 2011.

Hinton, G. E., Sabour, S., and Frosst, N. Matrix capsules with em routing. In *International conference on learning representations*, 2018.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. pmlr, 2015.

Kahneman, D., Treisman, A., and Gibbs, B. J. The reviewing of object files: Object-specific integration of information. *Cognitive psychology*, 24(2):175–219, 1992.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., and Teh, Y. W. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pp. 3744–3753. PMLR, 2019.

Li, Z., Zhang, Y., and Arora, S. Why are convolutional nets more sample-efficient than fully-connected nets? In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=uCY5MuAxcxU.

Liu, R., Lehman, J., Molino, P., Petroski Such, F., Frank, E., Sergeev, A., and Yosinski, J. An intriguing failing of convolutional neural networks and the coordconv solution. *Advances in neural information processing systems*, 31, 2018.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012–10022, 2021.

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., and Xie, S. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11976–11986, 2022.

Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., and Kipf, T. Object-centric learning with slot attention. *Advances in neural information processing systems*, 33:11525–11538, 2020.

Maslej, N., Fattorini, L., Perrault, R., Gil, Y., Parli, V., Kariuki, N., Capstick, E., Reuel, A., Brynjolfsson, E., Etchemendy, J., Ligett, K., Lyons, T., Manyika, J., Niebles, J. C., Shoham, Y., Wald, R., Walsh, T., Hamrah, A., Santarlasci, L., Lotufo, J. B., Rome, A., Shi, A., and Oak, S. Artificial intelligence index report 2025, 2025. URL https://arxiv.org/abs/2504.07139.

Ramachandran, P., Parmar, N., Vaswani, A., Bello, I., Levskaya, A., and Shlens, J. Stand-alone self-attention in vision models. *Advances in neural information processing systems*, 32, 2019.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. Zero-shot text-to-image generation. In *International conference on machine learning*, pp. 8821–8831. Pmlr, 2021.

Sabour, S., Frosst, N., and Hinton, G. E. Dynamic routing between capsules. *Advances in neural information processing systems*, 30, 2017.

Shaw, P., Uszkoreit, J., and Vaswani, A. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018.

Sutton, R. The bitter lesson. *Incomplete Ideas (blog)*, 13(1):38, 2019.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wiedemer, T., Brady, J., Panfilov, A., Juhos, A., Bethge, M., and Brendel, W. Provable compositional generalization for object-centric learning. *arXiv preprint arXiv:2310.05327*, 2023.

Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. Deep sets. *Advances in neural information processing systems*, 30, 2017.

Zhang, Y., Hare, J., and Prugel-Bennett, A. Deep set prediction networks. *Advances in Neural Information Processing Systems*, 32, 2019.

## A. Future work

**Overview.** Our findings in §3 (which were facilitated by our proposed synthetic dataset) motivate several promising directions for future work. We first discuss *intuition* for designing effective inductive biases for computer vision, followed by possible extensions to the SynDaCaTE dataset.

**Visual Inductive Biases.** We demonstrated in §3 that a permutation-invariant SetTransformer was significantly more data-efficient than other baselines at classifying objects from a ground-truth *set* of parts (which was designed to represent the receptive field of a visual neuron). However, a general-purpose vision model must take its input from images. How should we best encompass our findings in a general-purpose vision model? A simple approach would be to use our lightweight modernised CNN as a starting point, and replace depthwise-separable convolutions with sliding-window self-attention[2] and point-wise MLPs in alternating residual blocks. These changes maintain the overall structure of a CNN, while modifying the inductive bias only within local neighbourhoods. The resulting model resembles Stand-Alone Self-Attention (Ramachandran et al., 2019) without normalisation layers (Ioffe & Szegedy, 2015) or layer-wise relative positional embeddings (Shaw et al., 2018) and with ReZero residual connections (Ramesh et al., 2021) and an initial CoordConv layer (Liu et al., 2018) instead. Strided dense convolutions for inter-stage downsampling could be replaced with local linear average pooling, or local attention-weighted average pooling. Our preliminary experiments with this model, which we refer to as MereoFormer, demonstrate promising mereological capacity, however a remaining practical challenge is to improve the efficiency of our software implementation.

**Extending SynDaCaTE.** The *simplicity* of our SynDaCaTE dataset has allowed us to draw fundamental scientific conclusions without superfluous additional complexity. In future, we will want to draw analogous conclusions about more complex visual environments which are more relevant to practical real-world applications of computer vision. This motivates developing the SynDaCaTE dataset in several directions:

1. **Generalisation**: The training and test splits are currently sampled from the same distribution. These distributions could be structurally modified to support investigations into systematic (Bahdanau et al., 2018) and compositional (Wiedemer et al., 2023) generalisation.
2. **Classes**: The SynDaCaTE dataset contains a small number of object classes. More object classes (both wholes and parts) could be added, which may also support investigating meta-learning and transfer-learning of new object classes.
3. **3D**: The SynDaCaTE dataset is purely 2D, whereas the natural visual world is 3D. Synthetic 3D part-whole hierarchies could be designed and implemented using a 3D rendering pipeline, to explore the extent to which mereological capacity of various models is consistent between 2D and 3D visual environments.
4. **Video**: SynDaCaTE contains only static images, and our experiments only considered models which were trained on supervised learning tasks. The SynDaCaTE dataset could be extended to include *videos* of moving objects, which may be used to explore the extent to which models which have mereological capacity are also able to learn meaningful representations of parts and wholes (which for example could be linearly decoded from hidden representations) from *unsupervised* next-frame video prediction. We might refer to such a dataset as "Video SynDaCaTE", or more concisely as "VinDaCaTE".
5. **Interaction**: VinDaCaTE could be designed to optionally include interactions between objects in a video, which could be used to systematically investigate how well various models are able to learn to understand such interactions.
6. **Noise**: All inputs and targets in SynDaCaTE are noise-free by design. Future work may explore the effects of various noise distributions on the data-efficiency and final performance of vision models trained on image-to-parts, parts-to-wholes, and end-to-end learning tasks using the SynDaCaTE dataset.

[2] In sliding-window self-attention, each pixel is used as a query for keys and values in a local neighbourhood of pixels.

## B. Additional results

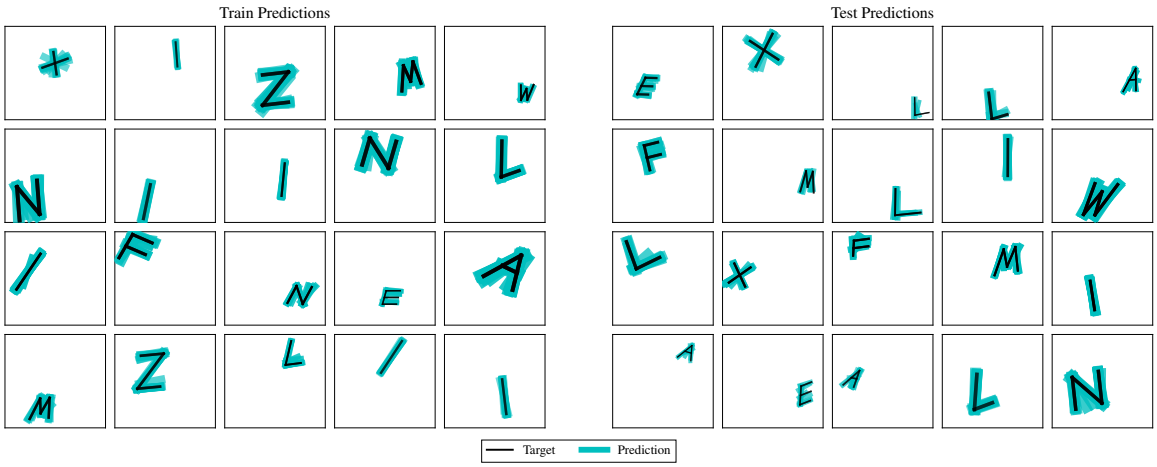


*Figure 2.* Rendered part-predictions of the best CNN trained on ImToParts.

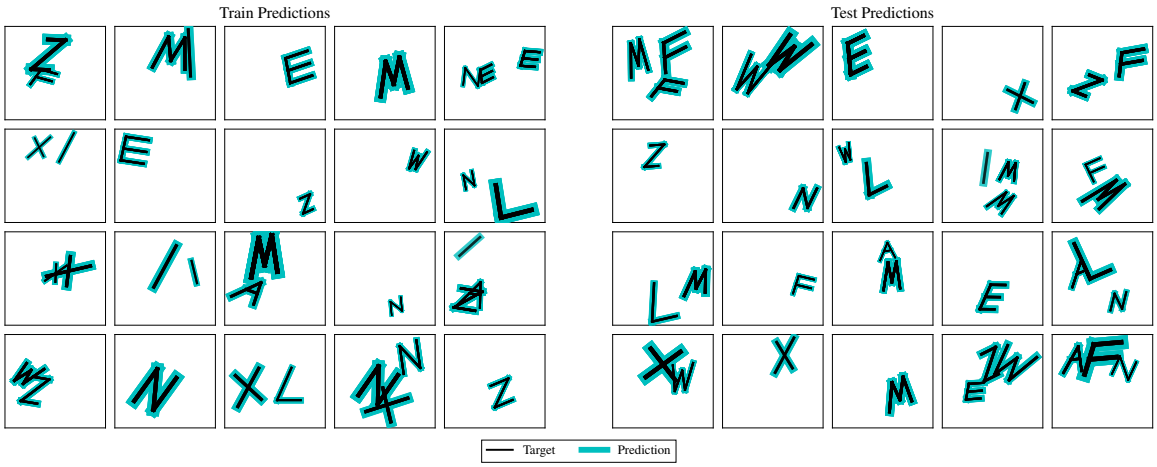


*Figure 3.* Rendered character-predictions of the best SetTransformer trained on PartsToChars.

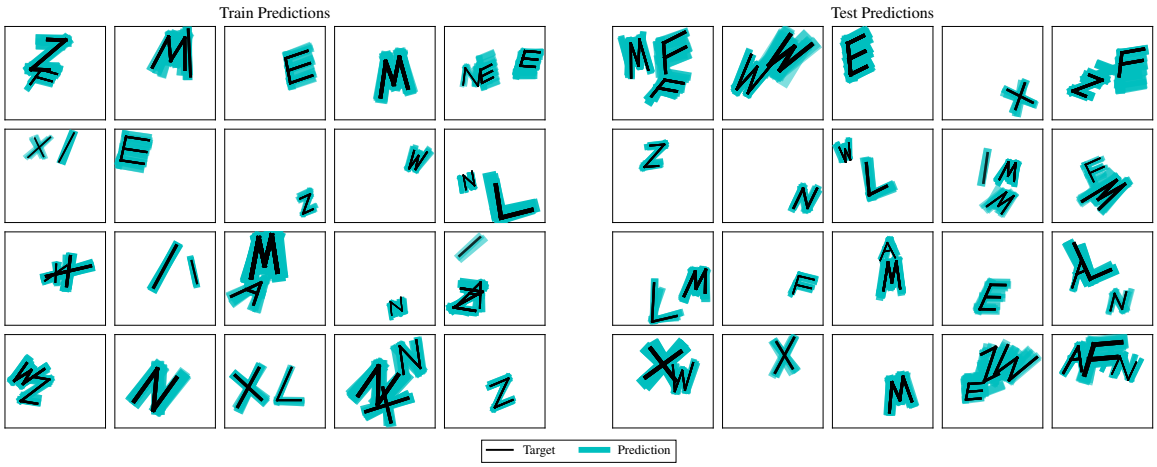


*Figure 4.* Rendered character-predictions of the best DeepSetToSet trained on PartsToChars.

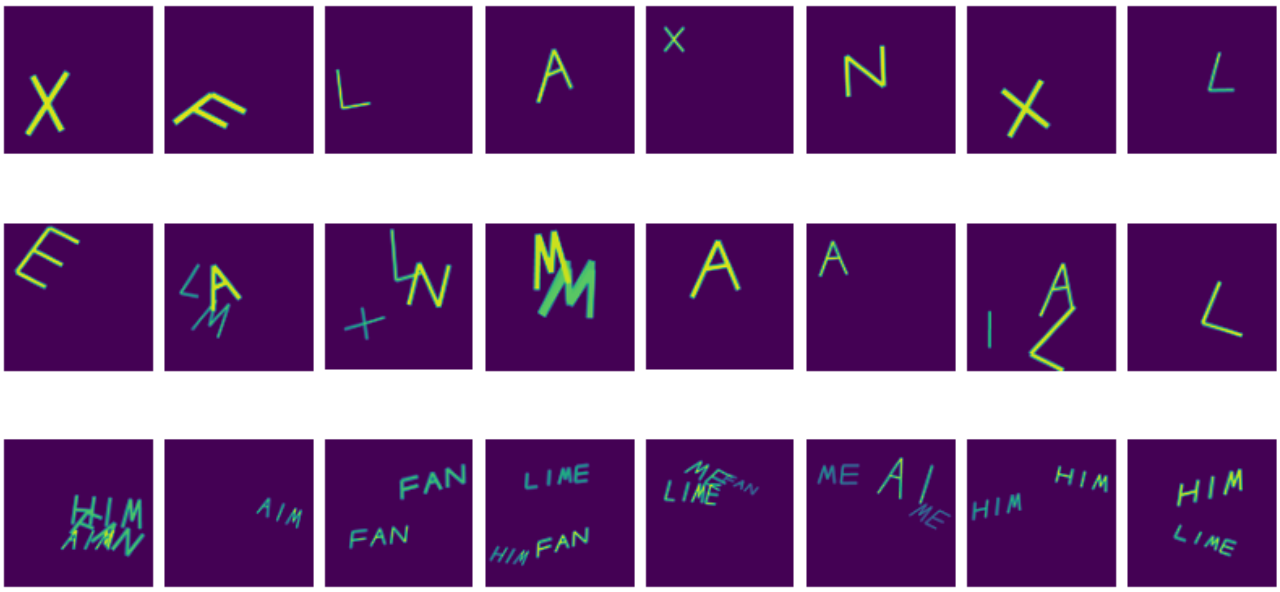


*Figure 5.* Example images from three different SynDaCaTE tasks. **Top row**: ImToClass. **Middle row**: ImToChars. **Bottom row**: Words.

## C. SynDaCaTE Specification

### C.1. Object Types

Across all tasks in SynDaCaTE, there are 21 types of object in three categories:

1. **Lines**: these are the bottom-level parts in all tasks. There is only one type of line. Each line has a 6D pose-vector, with dimensions referring to $x$ and $y$ coordinates of both endpoints, thickness, and brightness. When the ground-truth set of parts is available, each line is included twice, once for each permutation of its endpoints. This is because both permutations of endpoints render identical images, so a single permutation would not be identifiable from a rendered image.

2. **Characters**: these are composite objects composed of lines. There are 10 types of characters. Each character has an 8D pose-vector, with dimensions referring to $x$-position, $y$-position, scale, rotation, wideness, italic, line-thickness, and brightness.

3. **Words**: these are composite objects composed of characters. There are 10 types of words. The dimensionality and meaning of the pose vector of a word is the same as for a character.

### C.2. Sampling Procedure

Each image is sampled according to the following procedure:

1. Sample the number of top-level objects in the image.

2. For each object:

   (a) Sample a discrete class label $c_i$ and continuous pose vector $p_i$.
   (b) Initialise a set of parts (in some canonical pose) as a function of $c_i$.
   (c) Transform each part as a function of $p_i$.
   (d) Recursively initialise the sub-parts of each part until reaching bottom-level parts (lines).

3. Aggregate all bottom-level parts from all top-level objects into a single sequence and sort it by depth (in SynDaCaTE we simply equate depth with inverse brightness, so brighter parts are in front).
4. Render the sorted parts in order to form an image.

### C.3. Tasks

We present 7 supervised learning tasks, with inputs $x$ and targets $t$ as follows:

1. **ImToParts**: $x \in [0,1]^{1\times100\times100}$ is an image of a single character, $t \in \mathbb{R}^{9\times6}$ is the set of poses of the parts (lines) in the character. The maximum number of parts per character is 4, each part is represented twice (see above), and the total number of part-poses is padded with zeros up to 9 to ensure that $t$ always contains at least one all-zero pose vector. $t$ contains only 6D poses and no class information (because there is only one type of line). The order of parts in $t$ is randomised.
2. **PartsToChars**: $x \in \mathbb{R}^{25\times6}$ is a set of poses of the parts of between 1 and 3 characters, $t \in \mathbb{R}^{25\times18}$ contains the generalised pose of the character that each part belongs to. The outer dimensions of $x$ and $t$ are padded with zeros up to 25. The order of parts in $x$ is randomised, and the order of characters in $t$ is aligned to the order in $x$. Each generalised pose in $t$ is repeated according to its number of parts.
3. **ImToChars**: $x \in [0,1]^{1\times100\times100}$ is an image of between 1 and 3 characters, $t \in \mathbb{R}^{4\times18}$ contains the generalised poses of the characters, padded with zeros up to an outer dimension of 4. The order of parts in $t$ is randomised.
4. **ImToClass**: $x \in [0,1]^{1\times100\times100}$ is an image of a single character, $t \in \{1,\dots,10\}$ is the class of the character (represented as an integer). Pose information is discarded.
5. **PartsToClass**: $x \in \mathbb{R}^{9\times6}$ is the set of poses of the parts of a single character, $t \in \{1,\dots,10\}$ is the class of the character.
6. **PreTrainedPartsToClass**: $x \in [0,1]^{M\times H'\times W'}$ is the activations in the final hidden layer of a model trained on ImToParts, frozen, and evaluated on an image of a single character, $t \in \{1,\dots,10\}$ is the class of the character.
7. **Words**: $x \in [0,1]^{1\times100\times100}$ is an image of between 1 and 3 words, $t \in \mathbb{R}^{4\times18}$ contains the generalised poses of the words, padded with zeros up to an outer dimension of 4.

# D. Hyperparameters

## D.1. All Models

| | |
|---|---|
| Optimiser | Adam (Kingma & Ba, 2014) |
| Initial learning rate | 1e-3 |
| Final learning rate | 1e-5 |
| Learning rate schedule | Cosine decay |
| Batch size | 100 |

## D.2. Data-efficiency Classification Sweeps

| | | |
|---|---|---|
| CNN (ImToClass) | Kernel size | 5 |
| | Model dimension | 64 |
| | Expand ratio (in residual layers) | 2 |
| | Number of stages | 3 |
| | Number of blocks per stage | 2 |
| | Stride per stage | 2 |
| | Input embedding | CoordConv (Liu et al., 2018) |
| CapsNet (ImToClass) | Routing iterations | 3 |
| | All other hyperparameters | Default* |
| CNN (PreTrainedPartsToClass) | Kernel size | 5 |
| | Model dimension | 64 |
| | Expand ratio (in residual layers) | 2 |
| | Number of stages | 1 |
| | Number of blocks per stage | 3 |
| | Stride per stage | 1 |
| | Input embedding | CoordConv (Liu et al., 2018) |
| CapsNet (PreTrainedPartsToClass) | Routing iterations | 3 |
| | All other hyperparameters | Default* |
| SetTransformer (PartsToClass) | Depth | 5 |
| | Model dimension | 64 |
| | Number of heads | 8 |
| | Expand ratio (in residual layers) | 2 |
| MLP (PartsToClass) | Depth | 5 |
| | Model dimension | 100 |
| | Expand ratio (in residual layers) | 2 |

*For CapsNets, we used the public implementation available at <https://github.com/adambielski/CapsNet-pytorch>, including default hyperparameters.

## D.3. PartsToChars Depth Sweeps

| | | |
|---|---|---|
| SetTransformer | Model dimension | 64 |
| | Number of heads | 8 |
| | Expand ratio (in residual layers) | 2 |
| SetTransformer (2xW) | Model dimension | 128 |
| | Number of heads | 8 |
| | Expand ratio (in residual layers) | 2 |
| DeepSetToSet | Model dimension | 100 |
| MLP | Model dimension | 100 |
| | Expand ratio (in residual layers) | 2 |

### D.4. ImToParts

| | | |
|---|---|---|
| CNN | Kernel size | 5 |
| | Model dimension | 64 |
| | Expand ratio (in residual layers) | 2 |
| | Number of stages | 3 |
| | Number of blocks per stage | 2 |
| | Stride per stage | 2 |
| | Input embedding | CoordConv (Liu et al., 2018) |