Attentional Meta-learners for Few-shot Polythetic Classification

Ben Day [*1] **Ramon Viñas** [*1] **Nikola Simidjievski** [1] **Pietro Liò** [1]

Abstract

Polythetic classifications, based on shared patterns of features that need neither be universal nor constant among members of a class, are common in the natural world and greatly outnumber monothetic classifications over a set of features. We show that threshold meta-learners, such as Prototypical Networks, require an embedding dimension that is exponential in the number of task-relevant features to emulate these functions. In contrast, attentional classifiers, such as Matching Networks, are polythetic by default and able to solve these problems with a linear embedding dimension. However, we find that in the presence of task-irrelevant features, inherent to meta-learning problems, attentional models are susceptible to misclassification. To address this challenge, we propose a self-attention feature-selection mechanism that adaptively dilutes non-discriminative features. We demonstrate the effectiveness of our approach in meta-learning Boolean functions, and synthetic and real-world few-shot learning tasks.

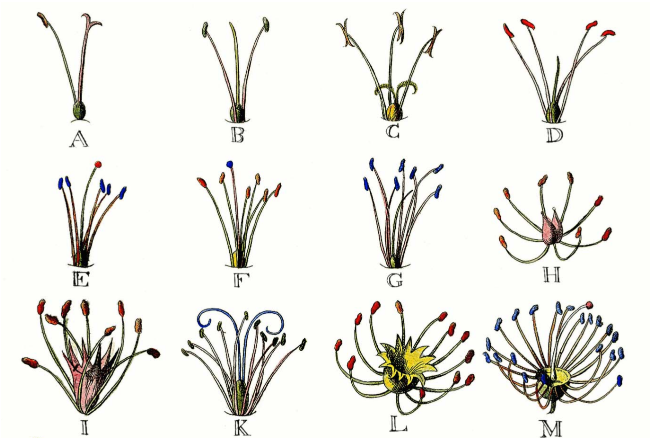


Figure 1: Section of *Methodus Plantarum Sexualis*, Georg Ehret, illustrating *Systema Naturae* (Linnaeus, 1735). The defining attributes of a class do not need to be exclusive or universal, and useful classifications are often contextual.

1. Introduction

Classification meta-learning is typically approached from the perspective of few-shot learning: Can we train a model that generalises to unseen 'natural' classes at test time? For example, in the Omniglot task (Lake et al., 2011) we have access to a labelled set of handwritten characters during training and we are tasked with distinguishing new characters, from unseen writing systems, at test time. From this perspective each example is associated with a consistent class and members of that class share a common set of properties (e.g. all handwritten characters have the shape of the underlying character class). Alternatively, we may consider meta-learning over *unseen ways of categorising*: Can we train a model on character recognition that generalises to alphabet recognition? or to distinguishing upper from lower case letters? In this setting, features need to be understood in relation to a given classification: For instance, when tasked with distinguishing equids (horses, zebras, donkeys) from big cats, the presence of stripes on both zebra and tigers is irrelevant, and potentially misleading. On the other hand, stripes are the key to distinguishing horses from zebra.

Understanding features in the context of a classification is central to the concepts of monothetic and polythetic classes recognised in the fields of taxonomy and knowledge organisation. *Monothetic classifications* are based on universal attributes: there is at least one necessary and sufficient attribute for class membership. *Polythetic classifications* are instead based on combinations of attributes, none of which are sufficient in isolation to indicate membership and, potentially, none of which are necessary. Carl Linneaus, inventor of the binomial nomenclature for species and "father of taxonomy," recognised that natural orders could not be defined monothetically, lacking features that were unique and constant over families and that, until such features could be found, such classifications were necessarily polythetic (Stevens, 1998; Hjørland, 2017).

[*]Equal contribution [1]Department of Computer Science and Technology, University of Cambridge, Cambridge, UK. Correspondence to: Ben Day <bjd39@cam.ac.uk.edu>, Ramon Viñas <rv340@cam.ac.uk.edu>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

Figure 1 illustrates Linnaeus' system for classifying plants, which relies on polythetic classifications and is still in use today. Consider, for example, that we can distinguish $\mathbb{A}$ from $\mathbb{B}$ by the number of filaments, but not $\mathbb{B}$ from $\mathbb{C}$; we can distinguish $\mathbb{B}$ from $\mathbb{C}$ based on whether there are split anthers (the ends), but not $\mathbb{C}$ from $\mathbb{A}$, and so on. To recognise classes it is necessary to consider their attributes in the context of other attributes.

Threshold functions are a related concept in Boolean algebra. A threshold function evaluates positively if a weighted sum of binary inputs crosses some threshold. Lacking a preferred feature basis, we can identify monothetic classifications with threshold functions, as threshold functions that are polythetic in one basis, such as logical OR(x, y), may be recast in a basis where they are monothetic e.g. binary OR(x, y) evaluates as the unary MIN$(x + y, 1)$. We can compare the frequency of monothetic and polythetic classifications in this general case. The number of binary inputs of length n is 2^n and the total number of Boolean functions is the number of binary labellings of these inputs, 2^{2^n}, whereas the number of threshold functions grows only singly exponentially ($\leq 2^{n^2}$), and therefore monothetic classifications represent a vanishingly small proportion of the total (Irmatov, 1993; Gruzling, 2007).

This work explores meta-learning for polythetic classification. Specifically, we

- consider the limitations of widely used threshold classifiers, such as Prototypical Networks (Snell et al., 2017), and how they are able to learn and approximate non-threshold functions in practice;
- show that simple alternatives based on attention, such as Matching Networks (Vinyals et al., 2016), are polythetic by default but susceptible to misclassification due to excessive sensitivity;
- characterise the challenge of spurious correlations in irrelevant features for attentional classifiers;
- and propose a simple solution to this challenge that is non-parametric and based on self-attention.

Throughout, we evidence these findings and the effectiveness of our proposals with experiments on synthetic and real-world few-shot learning tasks.

2. Background

Problem formulation. We are interested in few-shot classification: provided with a small number of labelled points $\mathbb{S} = \{\boldsymbol{x}_i, y_i\}_{i\in\mathbb{I}_\mathbb{S}}$, the support set, with feature vectors $\boldsymbol{x}_i \in \mathbb{R}^n$ and labels $y_i \in \{1, \dots, K\}$, we want to predict the labels of the query set $\mathbb{Q} = \{\boldsymbol{x}_j\}_{j\in\mathbb{I}_\mathbb{Q}}$. $\mathbb{S}_k$ denotes the set of support elements with label k, and $\mathbb{I}_\star$ is the index set of the subscript e.g. $\mathbb{I}_\mathbb{S}$ is the index set of the support. The label space is arbitrary and potentially unique to a task (also referred to as an episode) — both the number of classes, k, and assigned labels may vary over tasks — and, importantly, examples that share labels under one categorisation will not necessarily share labels under another. We will refer to classification functions over n features simply as classifications, and in meta-learning we are often interested in classifications that only depend on some features, α, and not the remainder, $\beta = n - \alpha$.

Classifiers. Deep neural models for problems of this kind are usually equipped with either a threshold classifier or an attentional classifier. Threshold classifiers are based, as the name suggests, on using thresholds to partition a space into regions associated with each class. Prototypical Networks (Snell et al., 2017) are an example of such a model. This model embeds examples using a learned neural function, f_ϕ, finds the average embedding for each class in the support, $\boldsymbol{c}_k$, and classifies queries based on their proximity (measured with a distance function d) to these class prototypes:

$$\boldsymbol{c}_k = \frac{1}{|\mathbb{S}_k|}\sum_{i\in\mathbb{I}_{\mathbb{S}_k}} f_\phi(\boldsymbol{x}_i) \quad ; \tag{1}$$

$$p_\phi(y = k|\boldsymbol{x}) = \frac{\exp(-d(f_\phi(\boldsymbol{x}), \boldsymbol{c}_k))}{\sum_{k'}\exp(-d(f_\phi(\boldsymbol{x}), \boldsymbol{c}_{k'}))} \tag{2}$$

The key advantage of such an approach is that salient class features are preserved when forming the prototype while irrelevant aspects of particular examples are washed out. Attentional classifiers instead use a similarity function to directly compare queries with each example in the support. For example, Matching Networks (Vinyals et al., 2016) also learn embedding functions, but each query is compared with every member of the support to weight a sum over their labels, which we can write simply as $\hat{y} = \sum_{i\in I_S} a(\hat{x}, x_i)y_i$. In the popular terminology of transformers we may write the embedded queries as $\mathbf{Q}$, the embedded support as keys $\mathbf{K}$, and their labels as values $\mathbf{V}$, and dot-product attention classification with temperature τ^{-1} as

$$\text{DotAttn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}, \tau) = \text{softmax}(\tau\mathbf{Q}\mathbf{K}^T)\mathbf{V} \in \mathbb{R}^{|\mathbb{Q}|\times k}. \tag{3}$$

Attentional classifiers are more sensitive to variations within a class at the cost of additional computation.

Boolean tasks. In comparing these classifiers we make repeated use of tasks based on Boolean functions, and on exclusive-OR (XOR) in particular. For a binary feature vector $\boldsymbol{x} \in \{-1, 1\}^n$, the number $\chi_\mathbb{A}(\boldsymbol{x}) = \prod_{i\in\mathbb{A}} x_i$ is the parity function or exclusive-or (XOR) over the bits $(x_i)_{i\in\mathbb{A}}$. We write a parity function of α bits XOR$_\alpha$. The set of parity functions over n bits form a linearly independent basis (O'Donnell, 2014) and, as such, being able to model

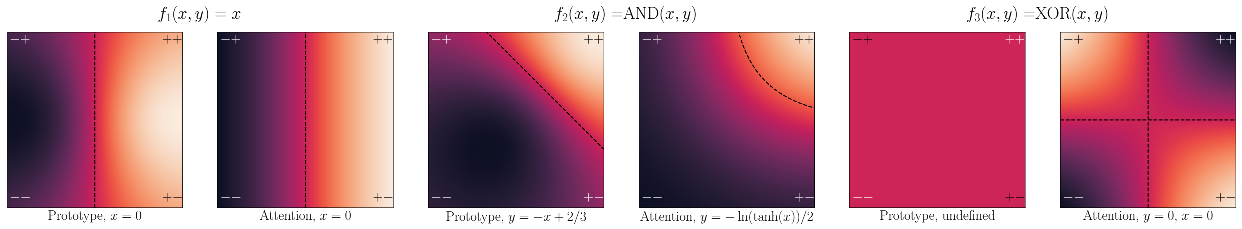


Figure 2: Confidence heatmaps and decision boundaries of prototype and attention classifiers on 2-variable Boolean functions. The attention classifier shown uses a temperature of 1, lower temperatures 'harden' the classification and produce decision boundaries more closely aligned with the axes (see Appendix B). As XOR(x, y) is not a threshold function, simple prototypes fail to produce a correct classification scheme, in this case the prototypes are equal $(= (0, 0))$ and there is no decision boundary.

the partition functions guarantees that one can model any other Boolean function over that domain. Put another way, the decision boundaries of XOR$_\alpha$ are at least as complicated as those for any other α-variable Boolean function: there is one between every possible pair of feature vectors. For these reasons, XOR is our polythetic function of choice in derivations, examples, and experiments.

3. Challenges in meta-learning polythetic classifications

Threshold and attentional classifiers have their own strengths and, in meta-learning polythetic classifications, their own weaknesses. Threshold classifiers are insufficiently flexible and attentional classifiers are prone to misclassification.

Threshold classifiers. Figure 2 shows the decision boundaries formed by a prototypical threshold classifier and an attentional classifier for a selection of 2-variable Boolean functions, highlighting the problem with using threshold classifiers for polythetic classification: logical XOR(x, y) is not a threshold function and so the prototypes fail to produce a useful decision boundary. This is the perceptron problem identified by Minsky & Papert (1969). However, deep networks using threshold classifiers can learn XOR. This is possible because the network can learn additional pseudo-features for the non-threshold functions it observes. Figure 3 shows an example for 2-variables: one can embed the corners of the square at the corners of a tetrahedron with coordinates $\big(x, y, \text{XOR}(x, y)\big)$ and produce linear thresholds solutions for every 2-variable Boolean function.

There are two problems with this approach, both of which are exacerbated in the meta-learning context: i) the required number of pseudo-variables grows as $\binom{n}{\alpha} \sim \mathcal{O}(n^\alpha)$ to account for all combinations of α active components, and ii) the method does not generalise to unseen non-threshold functions (see Appendix A). The right plot in Figure 3 demonstrates these shortcomings for XOR$_2$: the required number of pseudo-variables is $\binom{n}{2} \sim \mathcal{O}(n^2)$ and initially we find that a quadratic embedding is able to maintain performance, but for longer sequences the number of threshold-functions unseen in training grows and performance degrades.

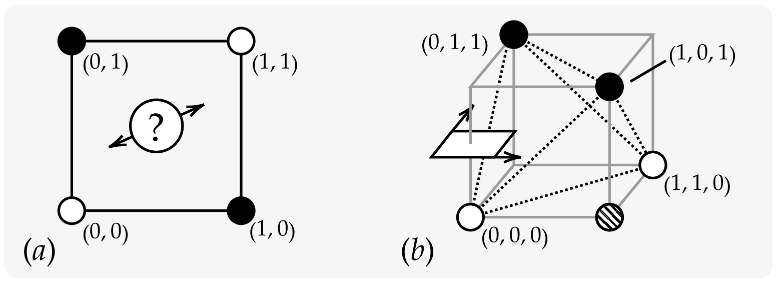
Attentional classifiers. Attentional classifiers avoid these problems — the required embedding dimension is linear in the number of features and the classifier generalises to unseen classifications — but suffer from over-sensitivity to irrelevant features. This results in misclassification, which we quantify in the case of Boolean functions to understand the scaling properties of this problem generally.

Consider classifications over binary feature vectors $\boldsymbol{x} \in \{-1, 1\}^n$, with the class determined by XOR$_\alpha$ over α elements with the remaining $\beta = n - \alpha$ being irrelevant. Assume each of the 2^α variations of the active elements are present with equal frequency, r, for a support set $\mathbb{S}$ of size $|\mathbb{S}| = r2^\alpha$, and that the remaining β elements follow a Bernoulli distribution with probability p. Using an attention classifier of the form $\hat{y} = \sum_{i\in\mathbb{S}} \text{softmax}_i \big(a(\hat{x}, x_i)\big)\, y_i$, where $a(\hat{x}, x_i)$ is a measure of the similarity of $\hat{x}$ and x_i, how likely is it that we misclassify a query drawn from the same distribution as $\mathbb{S}$?

Without loss of generality, we can focus on the positive examples (with label $= 1$) for which a positive output gives the correct classification. Using dot-product attention, the mean and variance of the classifier output, with $\bar{p} = p^2 + (1-p)^2$ and $\bar{q} = 1 - \bar{p}$, are:

$$\mu = r(e - e^{-1})^\alpha \left[\Big(\bar{p}e + \bar{q}e^{-1}\Big)^\beta\right] = r(e - e^{-1})^\alpha [c^\beta], \tag{4}$$

$$\sigma^2 = r(e^2 + e^{-2})^\alpha \left[\Big(\bar{p}e^2 + \bar{q}e^{-2}\Big)^\beta - \Big(\bar{p}e + \bar{q}e^{-1}\Big)^{2\beta}\right] = r(e^2 + e^{-2})^\alpha [d^\beta - c^{2\beta}], \tag{5}$$



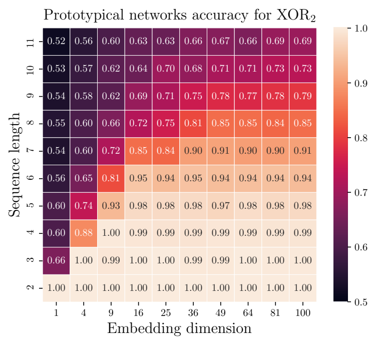


Figure 3: A non-threshold function of 2-variables, the pseudo-variable solution, and Prototypical Network performance on the XOR$_2$ problem. **(a)** XOR(x, y), which does not have a threshold solution in 2 dimensions. **(b)** Appending the pseudo-variable XOR(x, y) gives a 3-dimensional embedding in which all 2-variable Boolean functions have threshold solutions. XOR(x, y) is a pseudo-variable in that it is determined by the other variables and cannot freely vary, for example the hatched circle at $(1, 0, 0)$ cannot occur. *Right:* Accuracy of prototypical networks for the XOR$_2$ problem over sequence length and embedding dimension. Mean over 1000 tasks, $|\mathbb{S}| = 40$. See Appendix F for details.

introducing c and d for compactness. The mean is positive, as desired, but we are interested in the rate of misclassification, which may be interpreted using the scale-free and dimensionless coefficient-of-variation (the ratio of the standard deviation to the mean) where greater variation indicates a greater rate of misclassification. From Equations 4 and 5, we have

$$\frac{\sigma}{\mu} = \frac{1}{\sqrt{r}} \left(\frac{\sqrt{e^2 + e^{-2}}}{e - e^{-1}} \right)^{\alpha} \left(\left(\frac{d}{c^2} \right)^{\beta} - 1 \right)^{1/2}. \tag{6}$$

Starting with the leftmost term, increasing the number of repetitions, r, reduces the relative variability. This aligns with intuition and limits, where an empty support set, $r = 0$, provides no basis on which to make predictions and at the other extreme, $r \gg 2^{\beta}$, the support set is likely to span the input domain reducing classification to look-up. The term raised to α is approximately 3.2, and so the variability increases with the number of active elements. An intuitive explanation is that the number of immediate neighbours of each point grows as $\binom{\alpha}{1} = \alpha$ and this reduces the confidence with which the point is classified, so the barrier to misclassification is reduced. Finally, $d > c^2$ and so the rightmost term is positive and grows exponentially in β, meaning that misclassification increases with the number of irrelevant features. A full derivation, including alternative attention functions, is provided in Appendix C.

The problem of misclassification due to over-sensitivity in attentional classifiers was recognised in the work of Luong et al. (2015) on sequence processing. There the problem was addressed by attending only to a subset of elements within some distance of the target position. However, sets do not have such an ordering and so we instead propose a feature-selection method to resolve the problem more generally.

4. Attentional feature selection

A key challenge of meta-learning is that not all features are relevant in all tasks and that the support is unlikely to span the input domain. The model must choose, using incomplete information, what to focus on and what to ignore by detecting the salient features within and between classes. For monothetic classifications this is straightforward: by definition, averaging highlights necessary features whilst diminishing irrelevant features. Prototype methods rely on this process. In the polythetic case, attentional classifiers have the advantage of being able to learn non-threshold functions without the need for pseudo-variables but do not benefit, as prototypes do, from 'washing-out' irrelevant features through averaging. Indeed, attentional classifiers are susceptible even in the monothetic setting to misclassifying on the basis of closely matching features that are not relevant to the problem (putting a zebra with the big-cats because its stripes match those of a tiger in the support set, for example). Misclassification occurs when irrelevant features overwhelm the signal from the active elements. As this is a problem of highlighting the salient patterns within a set, we propose a self-attention based mechanism for feature selection, presented in Algorithm 1 (and illustrated in Appendix H) with examples given in Figures 4 and 5.

Intuitively, the process exploits the over-representation of patterns within features that are relevant to the classification as compared to patterns within the irrelevant features. We first standardise the features to prevent those common to the entire support, which are not discriminating, from dominating (Line 1) and stabilise, $+\epsilon$, to prevent weakly activated features from being excessively scaled-up.

We then repeatedly self-attend within each separate class

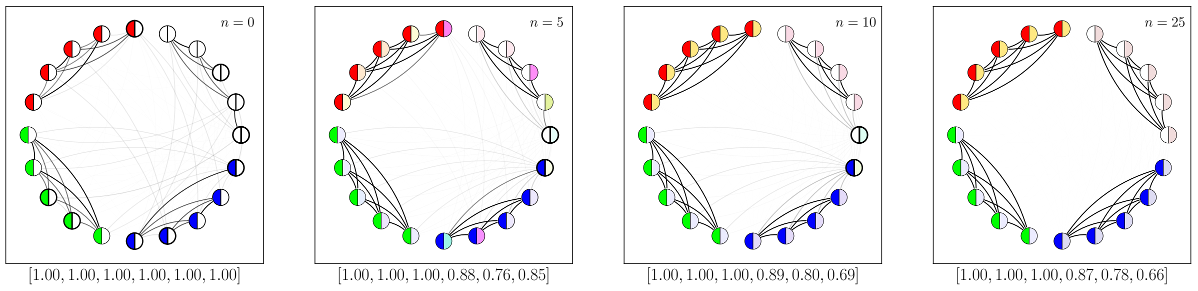


Figure 4: Feature values and attention coefficients during the feature-selection self-attention ($n = \{0, 5, 10, 25\}$) within a class of XOR$_3$. Nodes depict examples of the support set: the colour of the left halves represents the active features; the right halves represents the magnitude of the irrelevant features. Edge width and opacity indicate the attention strength between a pair of nodes. The red, green, blue and white groups, different variants, automatically segregate which preserves their active features while the irrelevant features converge, as shown in the feature scores beneath each plot. In this way, the active features identify themselves.

k of the support set, using dot product attention with scale τ, $\mathbf{X}_k \leftarrow \text{DotAttn}(\mathbf{X}_k, \mathbf{X}_k, \mathbf{X}_k, \tau)$. Self-attention maps elements of a set of vectors to the interior of their convex hull. If every member of a class has some feature in common, the convex hull in that dimension is a point and the features do not change. In the polythetic case it is patterns of features that matter, and by attending more strongly between elements of the support set with such feature-patterns, these too are preserved. Figure 4, for example, shows polythetic variations within a class of XOR$_3$ with three active and three irrelevant features ($\alpha = \beta = 3$). The patterns in the active features self-reinforce, forming cliques of strongly

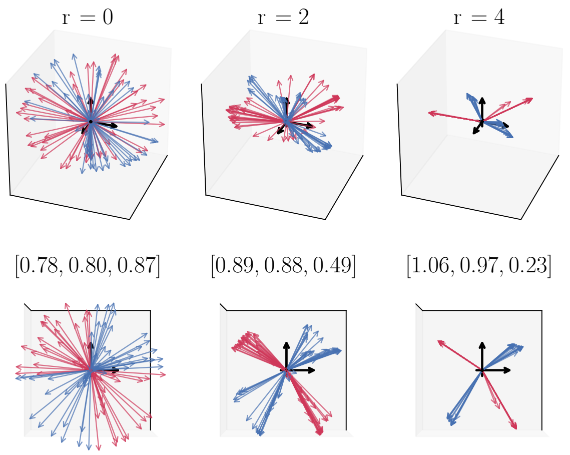


Figure 5: Feature vectors converging under iterated self-attention on a XOR$_2$ classification of vectors uniformly sampled over the sphere, in 3D (top) and down the z-axis (bottom). Colours indicate classes. The vectors quickly align by xy-quadrant and the variation in z is 'washed-out,' also seen in the feature selection scores (mean-absolute-deviation) $[x, y, z]$.

Algorithm 1 Self-attention feature scoring. Scores can be used for rescaling or masking. Note that the z-normalisation is over the entire support set whilst the self-attention is within classes. The choice of dispersion measure is of secondary importance and discussed in the main text.

```
1: Input: Support set $S = \{\mathbf{x}_i, y_i\}_{i \in I_S}$ with class labels $y_i \in \{1, \dots, K\}$ and features $\mathbf{x}_i \in \mathbb{R}^F$, $S_k$ denoting the subset of $S$ containing all samples with $y_i = k$ and $\mathbf{X}_k \in \mathbb{R}^{|S_k| \times F}$ an arbitrarily ordered matrix of feature vectors belonging to $S_k$; small numerical constant, $\epsilon$; attention temperature, $\tau^{-1}$; repetitions, $R$.
2: Output: Feature scores, $\boldsymbol{f} \in \mathbb{R}^F$.
3: $\mathbf{x}_i \leftarrow (\mathbf{x}_i - \mu_X)/(\sigma_X + \epsilon)$        {standardise}
4: for $r \leftarrow 1$ to $R$ do
5:     for $k \leftarrow 1$ to $K$ do
6:         $\mathbf{X}_k \leftarrow \text{softmax}(\tau \mathbf{X}_k \mathbf{X}_k^T)\mathbf{X}_k$        {softmax is row-wise}
7:     end for
8: end for
9: $\boldsymbol{f} \leftarrow \text{dispersion}(\{\mathbf{x}_i\}_{i \in S_i})$        {mean-absolute-deviation, std. dev etc.}
```

connected elements, whilst the irrelevant features decay. This is also evident from Figure 5 in the XOR$_2$ case, illustrating how the feature vectors converge during iterated self-attention. For instance the patterns in (x, y) coordinates of each class in opposite quadrants are preserved, as vectors in these quadrants reinforce each other, while variations in z get diluted. We can measure the degree to which a feature has been preserved with the dispersion. Features are scored by their dispersion over the support set (Line 1) which indicates how well they have been preserved through the self-attention iterations, and thus how relevant they are.

The scores can be used directly to rescale features across the support and query sets before applying the classifier, as in Figure 6, or in top-k selection. We focus on rescaling

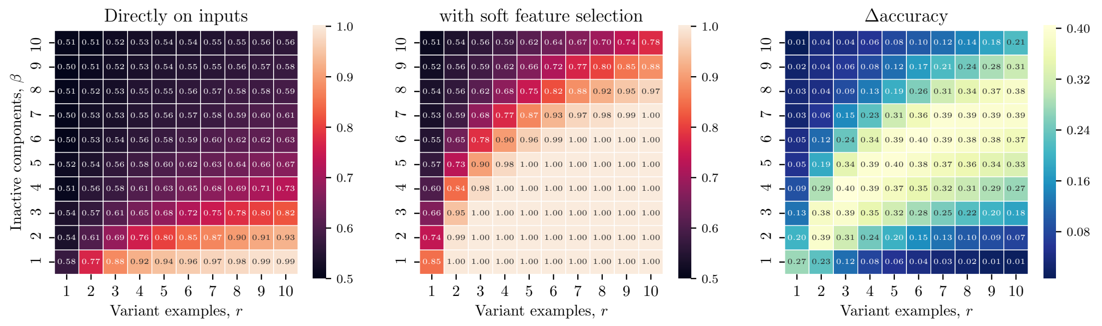


Figure 6: Attention classification of the XOR_4 problem over variant frequency, r, and number of inactive components β. Increasing r assists feature selection, in agreement with the derived misclassification distribution. Soft feature-selection rescales features according to their scores, as determined by the proposed self-attention procedure. This greatly improves performance even at low repetitions, for example at 2 repetitions and 3 inactive components the change in accuracy is $+38$pp. Neither method is effective at high β with low r.

as the method that makes the fewest assumptions about the underlying classification, but using top-k is highly effective when the number of active elements is known, as shown in Appendix D.

5. Experiments

We compare the proposed method (FS) with Prototypical Networks (PN) (Snell et al., 2017), a threshold classifier, and Matching Networks (MN) (Vinyals et al., 2016), an attentional classifier without feature-selection, in a sequence of increasingly complex synthetic and real-world few-shot learning problems. As our approach is non-parametric and operates directly on high-level features, it is agnostic to the choice of feature extractor, and we are free to choose as appropriate e.g. a convolutional neural network for images or a multi-layer perceptron for tabular data, and in all experiments we use the same embedding model for all methods (see experimental details in Appendix F).

Binary strings. We consider meta-learning Boolean functions of $n = \alpha + \beta$ variables. Labels for inputs $\boldsymbol{x} \in \{-1, 1\}^n$ are generated by computing the XOR of a random subset of components of size α. Each variation in the active elements occurs 5 times in the support, $|\mathbb{S}| = 5 \cdot 2^{\alpha}$. The subset of active components is unknown to the meta-learner. Appendix I shows a concrete example of an XOR_2 task. Table 1 summarises the performances of Prototypical Networks and our approach. PN accuracy decreases sharply with sequence length n and the number of embedding units required to effectively solve this problem grows rapidly with n, as shown previously in Figure 3. This suggests that PN are indeed learning pseudo-variables and demonstrates the limitations of threshold classifiers in solving polythetic problems.

Polythetic MNIST. We evaluate the ability of the models to jointly extract high-level features and identify polythetic patterns. We build tasks (episodes) using MNIST digits (LeCun et al., 2010), where an example consists of 4 coloured digits (RGB). An example task is illustrated in Figure 7 and further details are provided in Appendix J. For monothetic tasks, a single high-level feature (e.g. colour of the top-right digit) distinguishes classes. For polythetic tasks, class membership derives from XOR interactions over a subset of features, and the remainder are problem-irrelevant. Table 2 shows the performances on three versions of the polythetic MNIST dataset: clean (excluding non-discriminative dig-

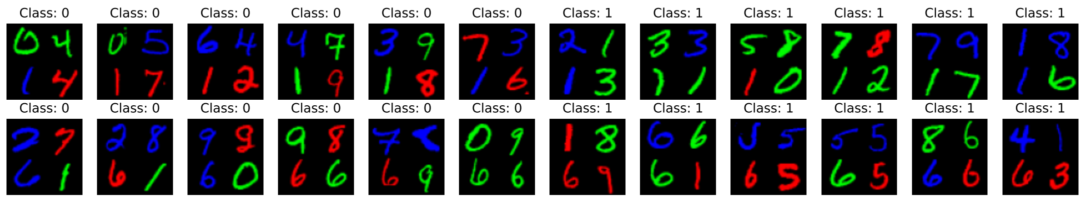


Figure 7: Example of an MNIST polythetic task. Examples from class 0 have either digit 1 (in any colour) in the bottom-left corner with a red digit in the bottom-right corner (top row); or digit 6 in the bottom-left corner with a green digit in the bottom-right (bottom row). Examples from class 1 can have either digit 1 in the bottom-left with a green digit in the bottom-right (top row); or digit 6 in the bottom-left with a red digit in the bottom-right corner (bottom row).

Table 1: Binary strings. Accuracy by embedding dimension for sequences of length $n = 5$ and $n = 10$. Mean and standard error calculated over 1000 tasks.

		$n=5$			$n=10$		
Model	Emb.	XOR_2	XOR_3	XOR_4	XOR_2	XOR_3	XOR_4
PN	1	57.6 ± 0.5	55.3 ± 0.5	60.8 ± 0.7	51.7 ± 0.4	50.2 ± 0.2	50.1 ± 0.2
	n	73.6 ± 0.5	70.3 ± 0.7	91.4 ± 0.4	56.7 ± 0.4	50.1 ± 0.2	50.4 ± 0.2
	n^2	90.4 ± 0.3	77.8 ± 0.7	$\underline{100.0 \pm 0.0}$	62.1 ± 0.4	50.3 ± 0.2	50.6 ± 0.2
FS+MN	n	$\mathbf{99.6 \pm 0.2}$	$\mathbf{100.0 \pm 0.0}$	$\underline{100.0 \pm 0.0}$	$\mathbf{75.9 \pm 1.3}$	$\mathbf{82.6 \pm 1.1}$	$\mathbf{96.3 \pm 0.5}$

Table 2: Polythetic MNIST. Evaluation accuracy on monothetic and polythetic tasks in three settings. Mean and standard error calculated over 1000 tasks.

	Clean		Colourless		Full	
Model	Monothetic	Polythetic	Monothetic	Polythetic	Monothetic	Polythetic
PN	$\mathbf{97.9 \pm 0.1}$	50.6 ± 0.3	92.8 ± 0.3	49.9 ± 0.3	$\mathbf{94.5 \pm 0.3}$	49.8 ± 0.2
MN	79.6 ± 0.6	57.6 ± 0.4	69.7 ± 0.4	61.0 ± 0.5	70.1 ± 0.7	56.6 ± 0.5
FS+MN	96.8 ± 0.1	$\mathbf{98.3 \pm 0.0}$	$\mathbf{94.5 \pm 0.2}$	$\mathbf{98.0 \pm 0.0}$	75.0 ± 0.7	$\mathbf{60.4 \pm 0.7}$

its), colourless (task-irrelevant digits but no colour), and full (both task-irrelevant digits and colour). The models are trained on monothetic tasks and evaluated both on monothetic and polythetic tasks. Protonets excel at identifying monothetic features and ignoring non-discriminative features, but have a close to random performance on polythetic tasks. Conversely, matching networks, which are polythetic classifiers by default, are highly sensitive to task-irrelevant features. The proposed approach (FS) can simultaneously detect salient features and perform polythetic classifications. Furthermore, as shown in Figure 8, we found our classifier to be robust to the rate of polythetic tasks seen during training in a second experiment.

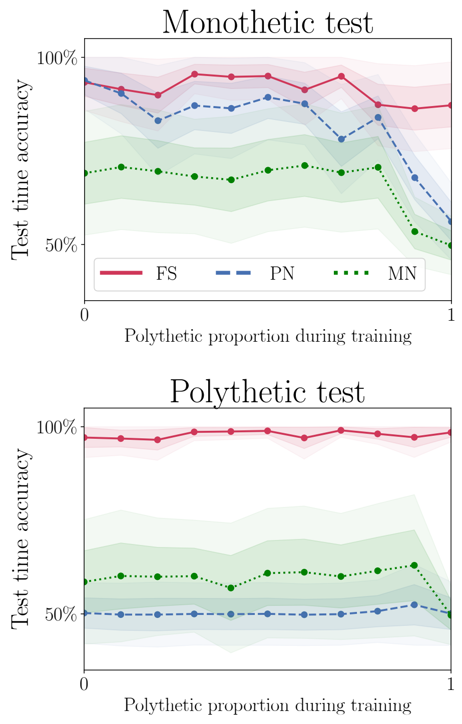


Figure 8: Polythetic MNIST (colourless) by polythetic proportion during training. FS matches or outperforms the other models at all training proportions and is far less affected by the training mix. Mean and standard deviation over 1000 tasks at each proportion.

Omniglot. The Omniglot dataset (Lake et al., 2011) consists of handwritten characters from 50 writing systems with 20 hand drawn examples of each character. Training tasks are formed using examples from 30 of the alphabets and test tasks draw from the other 20. We compare our method to PN, MN, Infinite Mixture Prototypes (IMP) (Allen et al., 2019), and MAML (Finn et al., 2017) with a threshold classifier (Triantafillou et al., 2020). We train the models for character recognition, and additionally evaluate performance on 3-way alphabet recognition (inherently polythetic). Our approach can be used in conjunction with most few-shot learning approaches – we specifically apply FS prior to MN and single-nearest-neighbours (NN), which corresponds to MN with softmax converging to argmax (see Appendix B). Table 3 shows the results of this experiment. The end-to-end trained model (FS+MN) is competitive with PN and IMP, while performing better than MN and

Table 3: Omniglot. 20-way, 5-shot characters; 3-way alphabets. Mean and standard error on 1000 tasks.

	Characters	Alphabets
PN	**98.6** $\pm$ **0.0**	83.4 $\pm$ 0.3
MN	91.1 $\pm$ 0.1	78.4 $\pm$ 0.3
FS+MN	96.2 $\pm$ 0.0	94.2 $\pm$ 0.2
IMP	**98.6** $\pm$ **0.0**	**96.0** $\pm$ **0.2**
MAML	94.0 $\pm$ 0.1	89.9 $\pm$ 0.3
MN*	97.9 $\pm$ 0.1	81.3 $\pm$ 0.3
NN*	98.3 $\pm$ 0.0	95.7 $\pm$ 0.3
FS+MN*	98.1 $\pm$ 0.0	**96.0** $\pm$ **0.2**
FS+NN*	98.3 $\pm$ 0.0	**96.0** $\pm$ **0.2**

MAML in character recognition. In alphabet recognition, FS+MN performs better than other methods, while being competitive with IMP. We further evaluate the performance when using a pre-trained feature extractor (methods marked with *), obtained by training a PN threshold classifier on character recognition. FS+MN* improves over MN* in both tasks. Compared to PN, the accuracy of FS+MN* and FS+NN* in character recognition is reduced by at most 0.5pp; yet improved in alphabet recognition by 12.6pp, while performing similarly to the more complex IMP.

TieredImageNet. TieredImageNet (Ren et al., 2018) is a subset of ILSVRC-12 (Russakovsky et al., 2015) with polythetic characteristics, with classes grouped into categories corresponding to higher-level nodes in the ImageNet hierarchy. There are 34 categories of 10 to 30 classes each. We compare our method (FS) to PN and MN classifiers. We use a publicly available pre-trained ResNet-12 (Zhang et al., 2020), pre-trained using the training classes in TieredImageNet, as the feature extractor for all models. Table 4 presents the aggregate accuracy while Table 5 shows the head-to-head results of this experiment. FS leads to significant improvements in performance (except C=8/G=5, where the difference between PN and FS is not significant) in this full scale, naturally polythetic problem, particularly in the "more polythetic" case with 10 subgroups.

6. Related work

We characterise meta-learning approaches for few-shot classification. In addition to evaluating their ability to generalise to unseen classes, we investigate how well they can adapt to polythetic tasks (generalisation to unseen ways of cateogorising). Adaptability in few-shot settings has been studied through different paradigms such as: fast weights (Ba et al., 2016); learnable plasticity (Miconi et al., 2018); combining features obtained from different pre-trained networks (Chowdhury et al., 2021) or learned from different data sets (Dvornik et al., 2020); as well as meta-learning. Recent work on meta-learning for few-shot classification includes approaches that are able to quickly adapt through various mechanisms such as recurrent architectures (Santoro et al., 2016; Mishra et al., 2018) for learning parameter updates (Ravi & Larochelle, 2017). Other more general optimisation-based approaches (Finn et al., 2017; Nichol et al., 2018; Rusu et al., 2019), tackle these tasks by explicitly optimising the model's parameters. These, however, are typically model-agnostic and commonly used in conjunction with threshold classifiers (Triantafillou et al., 2020), inheriting their limitations in a polythetic scenarios.

Our work aligns more closely with metric-learning approaches for few-shot classification (Chen et al., 2019) that apply distance functions between queries and the support in a common embedding space (Vinyals et al., 2016; Snell et al., 2017; Sung et al., 2018; Allen et al., 2019; Oreshkin et al., 2018; Zhang et al., 2020). Methods that construct class-wise prototypes from the support (Snell et al., 2017; Ren et al., 2018; Allen et al., 2019) can successfully tackle monothetic tasks, but can struggle with task-adaptiveness in a polythetic context. Attentional meta-classifiers (Vinyals et al., 2016; Kim et al., 2019; Hou et al., 2019; Jiang et al., 2020) adapt to polythetic tasks but lack crucial mechanisms for focusing exclusively on relevant features. Tian et al. (2020) investigate this issue of attentional meta-classifiers through the prism of knowledge distillation (Hinton et al., 2015), where an additional model is sequentially distilled from the original embedding model, leading to better, and arguably more relevant features. The authors show that such an approach, coupled with an attentional classification

Table 4: TieredImageNet. Model accuracy by categories (C) and groups (G) over 500 tasks.

G		C = 2	C = 4	C = 8
5	FS	**83.5** $\pm$ **0.3**	**66.4** $\pm$ **0.3**	48.9 $\pm$ 0.2
	MN	82.7 $\pm$ 0.4	65.4 $\pm$ 0.3	48.3 $\pm$ 0.2
	PN	81.4 $\pm$ 0.3	64.6 $\pm$ 0.3	**49.4** $\pm$ **0.1**
10	FS	**83.6** $\pm$ **0.3**	**65.8** $\pm$ **0.3**	**48.7** $\pm$ **0.1**
	MN	82.9 $\pm$ 0.3	64.9 $\pm$ 0.3	48.0 $\pm$ 0.1
	PN	81.1 $\pm$ 0.3	63.3 $\pm$ 0.3	48.2 $\pm$ 0.1

Table 5: TieredImageNet. Head-to-head comparison over 500 tasks by categories (C) and subgroups (G). **Bold** indicates significance at the $p < 0.001$ level.

			C = 2		C = 4		C = 8	
G	X	Y	X / Y	(tie)	X / Y	(tie)	X / Y	(tie)
5	FS	MN	**230** / 162	(108)	**329** / 101	(70)	**343** / 113	(44)
	FS	PN	**319** / 136	(45)	**339** / 142	(19)	239 / 247	(14)
10	FS	MN	**258** / 185	(57)	**357** / 108	(35)	**413** / 68	(19)
	FS	PN	**374** / 101	(25)	**396** / 99	(5)	**296** / 191	(13)

mechanism such as nearest neighbours (or cosine-similarity classifier) leads to improvements over the non-distilled models. This approach is, in principle, similar to ours, and in terms of performance, inline with our findings (FS + NN* in Table 3). However, unlike Tian et al. (2020), we investigate polythetic tasks in addition to monothetic, and more importantly we do not rely on learning additional models, but rather employ a self-attention mechanism for feature selection, to a similar effect. In this context, our work is remotely related to Škrlj et al. (2020), which analyse self-attention mechanisms for obtaining feature importance estimates, but their scope and applications differ significantly from ours.

Attending over datapoints has been considered previously. Luong et al. (2015) introduced dot-product attention in the context of attending over sequences. Vinyals et al. (2016) considered $\hat{y} = \sum_{i \in I_S} a(\hat{x}, x_i) y_i$ and provided the conditions under which such a model carries out kernel density estimation or k-nearest-neighbours classification. Vaswani et al. (2017) used explicit scaling in hidden attention layers, but scaling a classifying softmax by a temperature parameter dates to the work of Boltzmann and later Gibbs (1902). Plötz & Roth (2018) note that their neural-nearest-neighbours (N^3) block recovers a soft-attention weighting when the number of neighbours is set to 1, and deploy their model on an outlier-detection set-reasoning task.

7. Conclusion

In this work we have articulated the difference between monothetic and polythetic classifications and considered the limitations of standard meta-learning classifiers in the polythetic case. We have shown that threshold classifiers require an embedding space that is exponential in the number of active features and that attentional classifiers are overly sensitive and susceptible to misclassification. To address this, we have proposed an attention based method for feature-selection and demonstrated the effectiveness of our approach in several synthetic and real-world few-shot learning problems. Our approach is simple and can be used in conjunction with most few-shot meta-learners. We expect polythetic meta-learners to find real-world application in domains where data is typically scarce and complex, such as healthcare or bioinformatics. For example, we envision a use in classifying rare diseases from DNA sequences – there are around 7000 rare diseases, affecting $\sim$1/17 of the worldwide population, where mutations often lead to different phenotypes. In such scenarios, few-shot learning approaches able to generalise over unseen combinations of mutations (i.e. ways of categorising), in a similar vein to our binary strings experiment, may lead to better performance in diagnosing rare diseases and shed new insights into their molecular mechanisms.

Reproducibility

The code is available at `https://github.com/rvinas/polythetic_metalearning`. The descriptions of the experimental setups are provided in Section 5 and Appendices F, G, I and J. An illustrative description of our method, discussed in Section 4, is provided in Appendix H.

Acknowledgements

We thank Cristian Bodnar, Shayan Iranipour and Jacob Moss, from University of Cambridge, for their thoughtful feedback and helpful comments. We would also like to thank the reviewers for their constructive feedback and efforts towards improving our paper. We acknowledge the support of Fundación Rafael del Pino (R.V.), and the Mark Foundation for Cancer Research and Cancer Research UK Cambridge Centre [C9685/A25177] (N.S. and P.L.). The authors declare no competing interests.

References

Allen, K. R., Shelhamer, E., Shin, H., and Tenenbaum, J. B. Infinite mixture prototypes for few-shot learning. In Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, volume 97 of Proceedings of Machine Learning Research, pp. 232–241. PMLR, 2019.

Ba, J., Hinton, G. E., Mnih, V., Leibo, J. Z., and Ionescu, C. Using fast weights to attend to the recent past. In NIPS, pp. 4331–4339, 2016.

Chen, W., Liu, Y., Kira, Z., Wang, Y. F., and Huang, J. A closer look at few-shot classification. In 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019, 2019.

Chowdhury, A., Jiang, M., Chaudhuri, S., and Jermaine, C. Few-shot image classification: Just use a library of pre-trained feature extractors and a simple classifier. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 9445–9454, October 2021.

Dvornik, N., Schmid, C., and Mairal, J. Selecting relevant features from a multi-domain representation for few-shot classification. In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J.-M. (eds.), Computer Vision – ECCV 2020, pp. 769–786, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58607-2.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In Precup, D. and Teh, Y. W. (eds.), Proceedings of the 34th International Conference on Machine Learning, volume 70 of Proceedings of Machine Learning Research, pp. 1126–1135. PMLR, 06–11 Aug 2017.

Gibbs, J. W. Elementary principles in statistical mechanics. Charles Scribner's Sons, 1902.

Gruzling, N. Linear separability of the vertices of an n-dimensional hypercube. 2007. doi: 10.24124/2007/bpgub464.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. In NIPS Deep Learning and Representation Learning Workshop, 2015. URL http://arxiv.org/abs/1503.02531.

Hjørland, B. Classification. Knowledge Organization, 44 (2):97–128, 2017. URL http://www.isko.org/cyclo/classification.

Hou, R., Chang, H., Ma, B., Shan, S., and Chen, X. Cross attention network for few-shot classification. In Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, pp. 4005–4016, 2019.

Irmatov, A. A. On the number of threshold functions. Discrete Mathematics and Applications, 3(4):429–432, 1993.

Jiang, Z., Kang, B., Zhou, K., and Feng, J. Few-shot classification via adaptive attention. CoRR, abs/2008.02465, 2020.

Kim, H., Mnih, A., Schwarz, J., Garnelo, M., Eslami, A., Rosenbaum, D., Vinyals, O., and Teh, Y. W. Attentive neural processes. In International Conference on Learning Representations, 2019. URL https://openreview.net/forum?id=SkE6PjC9KX.

Lake, B., Salakhutdinov, R., Gross, J., and Tenenbaum, J. One shot learning of simple visual concepts. In Proceedings of the annual meeting of the cognitive science society, volume 33, 2011.

LeCun, Y., Cortes, C., and Burges, C. MNIST handwritten digit database. ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist, 2, 2010.

Linnaeus, C. Systema Naturae. 1735.

Luong, T., Pham, H., and Manning, C. D. Effective approaches to attention-based neural machine translation. In Màrquez, L., Callison-Burch, C., Su, J., Pighin, D., and Marton, Y. (eds.), Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015, pp. 1412–1421. The Association for Computational Linguistics, 2015. doi: 10.18653/v1/d15-1166. URL https://doi.org/10.18653/v1/d15-1166.

Miconi, T., Stanley, K., and Clune, J. Differentiable plasticity: training plastic neural networks with backpropagation. In Dy, J. and Krause, A. (eds.), Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pp. 3559–3568. PMLR, 10–15 Jul 2018.

Minsky, M. and Papert, S. Perceptrons. 1969.

Mishra, N., Rohaninejad, M., Chen, X., and Abbeel, P. A simple neural attentive meta-learner. In International Conference on Learning Representations, 2018.

Nichol, A., Achiam, J., and Schulman, J. On first-order meta-learning algorithms, 2018.

O'Donnell, R. Analysis of Boolean functions. Cambridge University Press, 2014.

Oreshkin, B. N., López, P. R., and Lacoste, A. TADAM: task dependent adaptive metric for improved few-shot learning. In Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada, pp. 719–729, 2018.

Plötz, T. and Roth, S. Neural nearest neighbors networks. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), Advances in Neural Information Processing Systems, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/f0e52b27a7a5d6a1a87373dffa53dbe5-Paper.pdf.

Ravi, S. and Larochelle, H. Optimization as a model for few-shot learning. In ICLR, 2017.

Ren, M., Triantafillou, E., Ravi, S., Snell, J., Swersky, K., Tenenbaum, J. B., Larochelle, H., and Zemel, R. S. Meta-learning for semi-supervised few-shot classification. In 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings, 2018.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. Imagenet large scale visual recognition challenge. International Journal of Computer Vision, 115(3):211–252, Dec 2015. ISSN 1573-1405.

Rusu, A. A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., and Hadsell, R. Meta-learning with latent embedding optimization. In International Conference on Learning Representations, 2019.

Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T. Meta-learning with memory-augmented neural networks. In Balcan, M. F. and Weinberger, K. Q. (eds.), Proceedings of The 33rd International Conference on Machine Learning, volume 48 of Proceedings of Machine Learning Research, pp. 1842–1850, New York, New York, USA, 20–22 Jun 2016. PMLR.

Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/cb8da6767461f2812ae4290eac7cbc42-Paper.pdf.

Stevens, P. Linnaeus, Carl von (1707-78). 1998. doi: 10.4324/9780415249126-Q059-1. URL https://www.rep.routledge.com/articles/biographical/linnaeus-carl-von-1707-78/v-1/sections/the-natural-method.

Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H., and Hospedales, T. M. Learning to compare: Relation network for few-shot learning. In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1199–1208, 2018.

Tian, Y., Wang, Y., Krishnan, D., Tenenbaum, J. B., and Isola, P. Rethinking few-shot image classification: A good embedding is all you need? In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J.-M. (eds.), Computer Vision – ECCV 2020, pp. 266–282, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58568-6.

Triantafillou, E., Zhu, T., Dumoulin, V., Lamblin, P., Evci, U., Xu, K., Goroshin, R., Gelada, C., Swersky, K., Manzagol, P.-A., and Larochelle, H. Meta-dataset: A dataset of datasets for learning to learn from few examples. In International Conference on Learning Representations, 2020.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017.

Vinyals, O., Blundell, C., Lillicrap, T., kavukcuoglu, k., and Wierstra, D. Matching networks for one shot learning. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), Advances in Neural Information Processing Systems, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/90e1357833654983612fb05e3ec9148c-Paper.pdf.

Škrlj, B., Džeroski, S., Lavrač, N., and Petković, M. Feature importance estimation with self-attention networks. In Proceedings of the 24th European Conference on Artificial Intelligence - ECAI 2020, volume 325, pp. 1491 – 1498, 2020. doi: 10.3233/FAIA200256.

Zhang, C., Cai, Y., Lin, G., and Shen, C. Deepemd: Few-shot image classification with differentiable earth mover's distance and structured classifiers. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020.

A. Partitioned XOR performances

We hypothesise that, in order to solve polythetic tasks, prototypical networks need to create pseudo-variables in the embedding space (e.g. XOR for each pair of components for binary strings tasks). To test this, we train a prototypical network on binary strings tasks (where pairs of active components are always chosen from the 5 first components) and evaluate the performance on unseen combinations of components (i.e. combinations of the 5 first components not seen during training). Table 6 shows the accuracies for sequences of length $n = 5 + \beta$ (where β is the number of variables that are always inactive). We attribute the better-than-chance performances (i.e. for 100-dimensional embeddings) to the high-dimensionality of the embeddings – by chance, large numbers of non-linear features (e.g. output by a random feature extractor) will include some features that are useful for the task of interest (i.e. similar to extreme learning machines). Overall, these results highlight the inability of prototypical networks to generalise to unseen combinations and support our hypothesis.

Table 6: Accuracy of prototypical networks on unseen non-threshold functions by embedding dimension. We use sequences of length $n = 5 + \beta$, where β is the number of components that are always inactive. The labels are derived from XOR_2 combinations over the first 5 components and the sets of combinations seen at train and test times are disjoint. In other words, the combinations of variables $(0, 1), (0, 2), (1, 3), (2, 4), (3, 4)$ are only seen active at train time, whereas, the combinations $(0, 3), (0, 4), (1, 2), (1, 4), (2, 3)$ are only seen at test time, with $r = 5$ repetitions for each XOR combination. Each of the 5 first variables has the same expected frequency of being active at train and test times. The inability to generalise in this scenario suggests that protonets need pseudo-variables (i.e. XOR functions applied to each pair of components) to solve the binary strings task.

Emb.	$\beta = 0$	$\beta = 1$	$\beta = 2$
1	48.741 ± 0.276	48.855 ± 0.282	49.558 ± 0.284
10	50.350 ± 0.282	49.397 ± 0.286	49.479 ± 0.265
20	50.705 ± 0.271	51.057 ± 0.247	50.144 ± 0.255
100	52.246 ± 0.256	52.390 ± 0.265	52.573 ± 0.258

B. Temperature in attention classification

The softmax in attention mechanisms permits a temperature scaling that interpolates between argmax and uniform (and argmin.) This is controlled by $T = \frac{1}{\beta}$, as

$$\text{softmax}_i(\boldsymbol{x}, \beta) = \frac{\exp(\beta x_i)}{\sum_j \exp(\beta x_j)}$$

with softmax converging to argmax as $\beta \to \infty$, and to uniform (i.e. all elements equal to the reciprocal of the length of the vector) as $\beta \to 0$.

For attention classifiers, a decrease in temperature increases the model confidence and can cause decision boundaries to move. As $\beta \to \infty$ and the softmax converges to argmax, the classifier tends to the single nearest neighbour classification scheme; for $\beta = 0$ the classifier returns the support set class balance. For Boolean functions, changes in temperature effect the degree to which decision boundaries are axis-aligned. For example, Figure 2 shows the decision boundary for

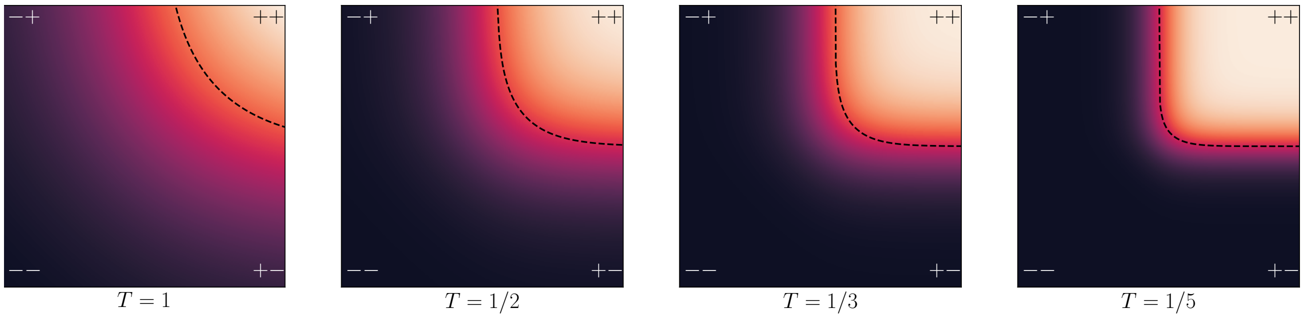


Figure 9: Changes in confidence and decision boundary of the attention classifier with temperature for AND(x, y).

$f(x, y) = \text{AND}(x, y)$ using a softmax temperature of 1 at $y = -\frac{1}{2}\ln(\tanh x)$, which we derive as

$$\begin{aligned}
p(\text{class } 1) &= p(\text{class } 0) \\
\frac{\exp\big(\beta(x+y)\big)}{\sum \exp(...)} &= \frac{\exp\big(\beta(-x-y)\big) + \exp\big(\beta(-x+y)\big) + \exp\big(\beta(x-y)\big)}{\sum \exp(...)} \\
\exp\big(\beta(x+y)\big) - \exp\big(-\beta(x+y)\big) &= \exp\big(\beta(-x+y)\big) + \exp\big(\beta(x-y)\big) \\
\sinh(\beta(x+y)) &= \cosh(\beta(x-y)) \\
\sinh(\beta x)\big(\cosh(\beta y) + \sinh(\beta y)\big) &= \cosh(\beta x)\big(\cosh(\beta y) - \sinh(\beta y)\big) \\
\tanh(\beta x) &= \frac{\cosh(\beta y) + \sinh(\beta y)}{\cosh(\beta y) - \sinh(\beta y)} \\
\tanh(\beta x) &= \exp(-2\beta y) \\
y &= -\frac{1}{2\beta}\ln\big(\tanh(\beta x)\big).
\end{aligned}$$

The effect of decreasing the temperature on the decision boundary is shown in Figure 9.

C. Misclassification distribution full derivation

We first present a more detailed derivation for the case of dot-product attention. We will make repeated use of the Binomial theorem

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^{n-k} y^k = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k}. \tag{7}$$

Recall that we consider classifications over binary feature vectors $\boldsymbol{x} \in \{-1, 1\}^n$, with the class determined by XOR$_\alpha$ over α elements with the remaining $\beta = n - \alpha$ being irrelevant. Assume each of the 2^α variations of the active elements are present with equal frequency, r, for a support set $\mathbb{S}$ of size $|\mathbb{S}| = r2^\alpha$, and that the remaining β elements follow a Bernoulli distribution with probability p. There are $\binom{\alpha}{\delta}$ strings over the active elements that differ from a given example in δ positions. XOR flips the classification with each difference, so if δ is even the class is the same, if δ is odd the class is different (parity).

The class is determined by the sign of the sum of contributions at an even distance subtract those at an odd distance. Putting those into a single sum we have

$$\text{class} = \text{sign}\left(\sum_{\delta=0}^{\alpha} r(-1)^\delta \binom{\alpha}{\delta} \exp(\alpha - 2\delta)\right). \tag{8}$$

We can factor out the number of repetitions, r, and as it is positive it doesn't change the sign. We can then rearrange to match the Binomial theorem form and recover the form in the main text:

$$\text{class} = \text{sign}\left(\sum_{\delta=0}^{\alpha}\binom{\alpha}{\delta}e^{\alpha-\delta}\left(\frac{-1}{e}\right)^{\delta}\right) = \text{sign}\left(\left(e-e^{-1}\right)^{\alpha}\right) = +. \tag{9}$$

Next we give the Binomial distribution of the irrelevant feature contribution as $2\mathrm{B}(\beta,\bar{p})-\beta$ with $\bar{p}=p^2-(1-p)^2=2p^2-2p+1$. Something that is important to the behaviour of the mean and variance later, and breaks the usual symmetry of p and $q=(1-p)$, is that $\bar{p}\geq 0.5$ and is quadratic in p defined by the three points $(p,\bar{p})=\{(0,1),(0.5,0.5),(1,1)\}$. $\bar{q}$ has the usual definition $1-\bar{p}$, so $\bar{q}\leq 0.5$ and so on. The contribution at a difference δ is then $\mathrm{X}(\delta)\sim\exp\left(\alpha-2\delta+2\mathrm{B}(\beta,\bar{p})-\beta\right)$. The expectation is computed in the usual way

$$\begin{aligned}\mathbb{E}[\mathrm{X}(\delta)] &= \sum_i \mathrm{P}(\mathrm{X}=x_i)x_i = \sum_{b=0}^{\beta}\mathrm{P}(\mathrm{B}(\beta,\bar{p})=b)\exp\left(\alpha-2\delta+2b-\beta\right)\\ &= \exp\left(\alpha-2\delta\right)\sum_{b=0}^{\beta}\binom{\beta}{b}\bar{p}^b\bar{q}^{(\beta-b)}\exp\left(2b-\beta\right)\\ &= \exp\left(\alpha-2\delta\right)\sum_{b=0}^{\beta}\binom{\beta}{b}(\bar{p}e)^b(\bar{q}e^{-1})^{\beta-b} = \exp\left(\alpha-2\delta\right)\left(\bar{p}e+\bar{q}e^{-1}\right)^{\beta}.\end{aligned}$$

Finding the variance in the traditional way, $\mathrm{Var}[\mathrm{X}(\delta)]=\mathbb{E}[\mathrm{X}(\delta)^2]-\mathbb{E}[\mathrm{X}(\delta)]^2$, first $\mathbb{E}[\mathrm{X}(\delta)^2]$ following the derivation for $\mathbb{E}[\mathrm{X}(\delta)]$:

$$\mathbb{E}[\mathrm{X}(\delta)^2] = \sum_i \mathrm{P}(\mathrm{X}=x_i)x_i^2 = \sum_{b=0}^{\beta}\mathrm{P}(\mathrm{B}(\beta,\bar{p})=b)\exp\left(2\alpha-4\delta+4b-2\beta\right) \tag{10}$$

$$= \exp\left(2\alpha-4\delta\right)\sum_{b=0}^{\beta}\binom{\beta}{b}\bar{p}^b\bar{q}^{(\beta-b)}\exp\left(4b-2\beta\right) \tag{11}$$

$$= \exp\left(2\alpha-4\delta\right)\left(\bar{p}e^2+\bar{q}e^{-2}\right)^{\beta}. \tag{12}$$

From this we can write the variance

$$\mathrm{Var}[\mathrm{X}(\delta)] = \exp\left(2\alpha-4\delta\right)\left(\left(\bar{p}e^2+\bar{q}e^{-2}\right)^{\beta}-\left(\bar{p}e+\bar{q}e^{-1}\right)^{2\beta}\right). \tag{13}$$

Next we want to find the expectation of the sum of the contributions at each difference δ. As pointed out there are $\binom{\alpha}{\delta}$ many strings at a difference of δ. We apply $\mathbb{E}[\sum_i \mathrm{X}_i]=\sum_i\mathbb{E}[\mathrm{X}_i]$ and $\mathrm{Var}[\mathrm{X}-\mathrm{Y}]=\mathrm{Var}[\mathrm{X}]+\mathrm{Var}[\mathrm{Y}]$, and, remembering to change the sign with each increase in δ,

$$\mathbb{E}\left[\textstyle\sum_{i\in S}(-1)^{\delta_i}\mathrm{X}(\delta_i)\right] = \sum_{\delta=0}^{\alpha}r(-1)^{\delta}\binom{\alpha}{\delta}\mathbb{E}[\mathrm{X}(\delta)] \tag{14}$$

$$= r\left(\bar{p}e^2+\bar{q}e^{-2}\right)^{\beta}\sum_{\delta=0}^{\alpha}\binom{\alpha}{\delta}e^{(\alpha-\delta)}\left(\frac{-1}{e}\right)^{\delta} \tag{15}$$

$$= r\left(\bar{p}e^2+\bar{q}e^{-2}\right)^{\beta}\left(e-e^{-1}\right)^{\alpha}, \tag{16}$$

and

$$\mathrm{Var}\left[\textstyle\sum_{i\in S}(-1)^{\delta_i}\mathrm{X}(\delta_i)\right] = \mathrm{Var}\left[\textstyle\sum_{i\in S}\mathrm{X}(\delta_i)\right] = \sum_{\delta=0}^{\alpha}r\binom{\alpha}{\delta}\mathrm{Var}[\mathrm{X}(\delta)] \tag{17}$$

$$= r\left(\left(\bar{p}e^2+\bar{q}e^{-2}\right)^{\beta}-\left(\bar{p}e+\bar{q}e^{-1}\right)^{2\beta}\right)\sum_{\delta=0}^{\alpha}\binom{\alpha}{\delta}\left(e^2\right)^{\alpha-\delta}\left(e^{-2}\right)^{\delta} \tag{18}$$

$$= r\left(\left(\bar{p}e^2+\bar{q}e^{-2}\right)^{\beta}-\left(\bar{p}e+\bar{q}e^{-1}\right)^{2\beta}\right)(e^2+e^{-2})^{\alpha}. \tag{19}$$

C.1. Different attention mechanisms

For dot-product and cosine-similarity attention, ‘angular’ difference mechanisms, we use $(+,-)$ to encode the input variables. This is because we want the score to be greater when the variables are the same and lesser when they are opposed (if we were to use $(1,0)$ we’d have $0\times 0\neq 1\times 1$ and $0\times 1=0\times 0$). With these methods we get

$$f_{\text{dot}}(\delta)=\alpha-2\delta, \tag{20}$$

$$f_{\text{cos}}(\delta)=1-\frac{2\delta}{\alpha}. \tag{21}$$

The change for cosine-similarity introduces a factor of $\exp\left(1/\alpha\right)$ to the mean and $\exp\left(2/\alpha\right)$ to the variance, and the overall picture doesn’t change

$$\mathbb{E}\left[\sum_{i\in S}(-1)^{\delta_i}\mathrm{X}_{\text{cos}}(\delta_i)\right]=r\left(\bar{p}e^2+\bar{q}e^{-2}\right)^{\beta}\left(e^{1/\alpha}-e^{-1/\alpha}\right)^{\alpha} \tag{22}$$

$$\text{Var}\left[\sum_{i\in S}(-1)^{\delta_i}\mathrm{X}_{\text{cos}}(\delta_i)\right]=r\left(\left(\bar{p}e^2+\bar{q}e^{-2}\right)^{\beta}-\left(\bar{p}e+\bar{q}e^{-1}\right)^{2\beta}\right)\left(e^{2/\alpha}+e^{-2/\alpha}\right)^{\alpha}. \tag{23}$$

For squared Euclidean distance attention (which coincides with the ‘Laplace attention’ L1 norm in this encoding), if we encode the inputs as $(1,0)$ we get $f_{L2}(\delta)=-(\sqrt{\delta})^2=-\delta$, introducing a factor of $\exp\left(-\alpha\right)$ to the mean and $\exp\left(-2\alpha\right)$ to the variance, which also doesn’t change the overall picture.

D. Top-k feature selection

Top-k feature selection masks out all but the k highest scoring features. That is $\boldsymbol{x} \leftarrow \boldsymbol{m} \odot \boldsymbol{x}$ with

$$m_i = \begin{cases} 1 & \text{if rank}(\boldsymbol{s})_i \leq k \\ 0 & \text{otherwise} \end{cases} \tag{24}$$

As compared to soft feature selection which rescales as $\boldsymbol{x} \leftarrow \boldsymbol{s} \odot \boldsymbol{x}$, or $\boldsymbol{x} \leftarrow \boldsymbol{s}' \odot \boldsymbol{x}$ with normalised $\boldsymbol{s}'$.

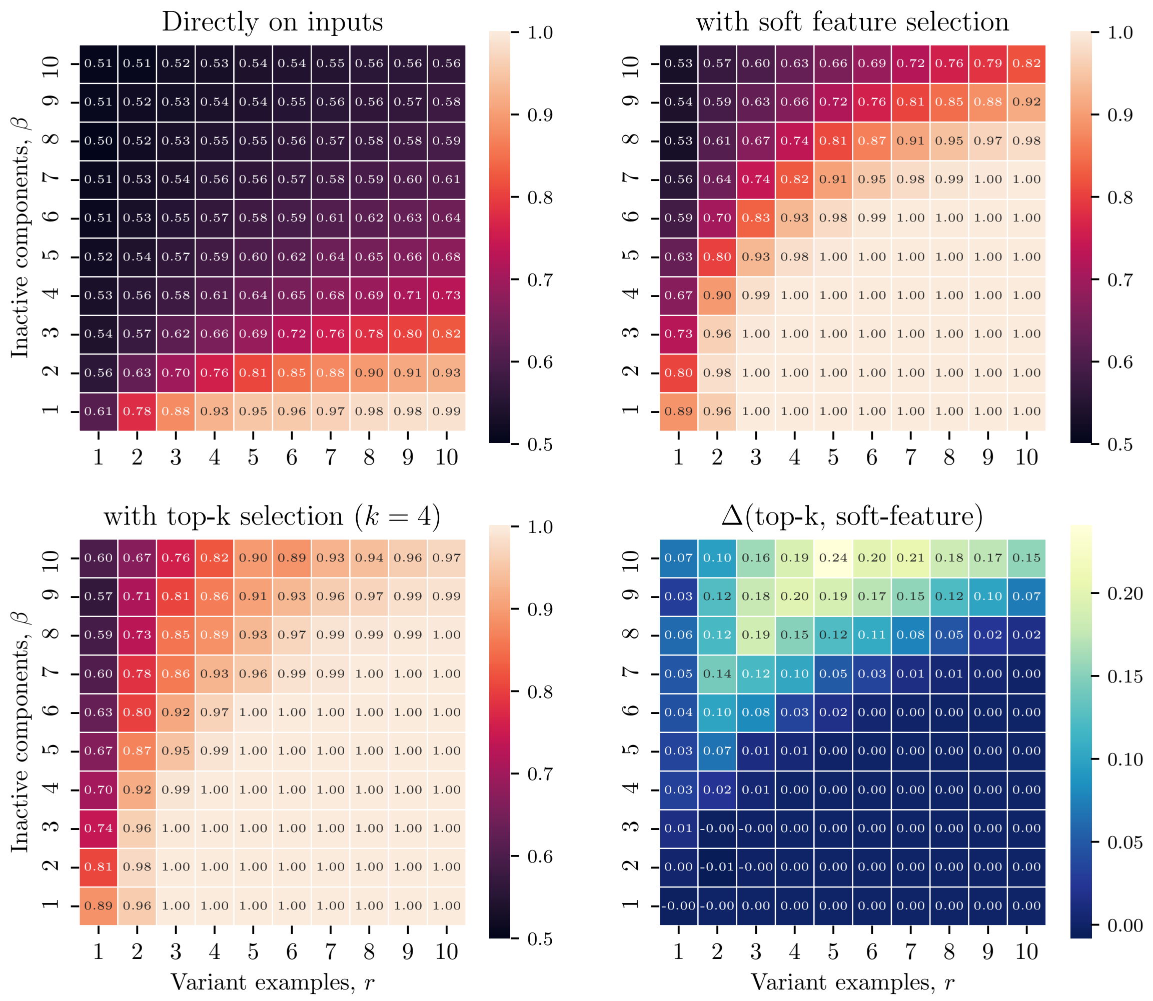


Figure 10: Comparing soft and top-k feature selection in the same setting as Figure 6: classification of XOR$_4$ over variant frequency, r, and number of inactive components β. Examples of a variant satisfy the XOR in the same way, i.e. the active components are equal, but may not have the same inactive components. The top-k version uses a binary mask to leave the k highest scoring features unchanged and zeroing the rest. This produces significant improvements over even the soft feature selection method at high values of β, but requires knowledge of the number of active elements.

E. Convergence of feature vectors across self-attention iterations

Figure 11 illustrates how feature vectors converge on the XOR_2 task across self-attention iterations of the feature selection mechanism.

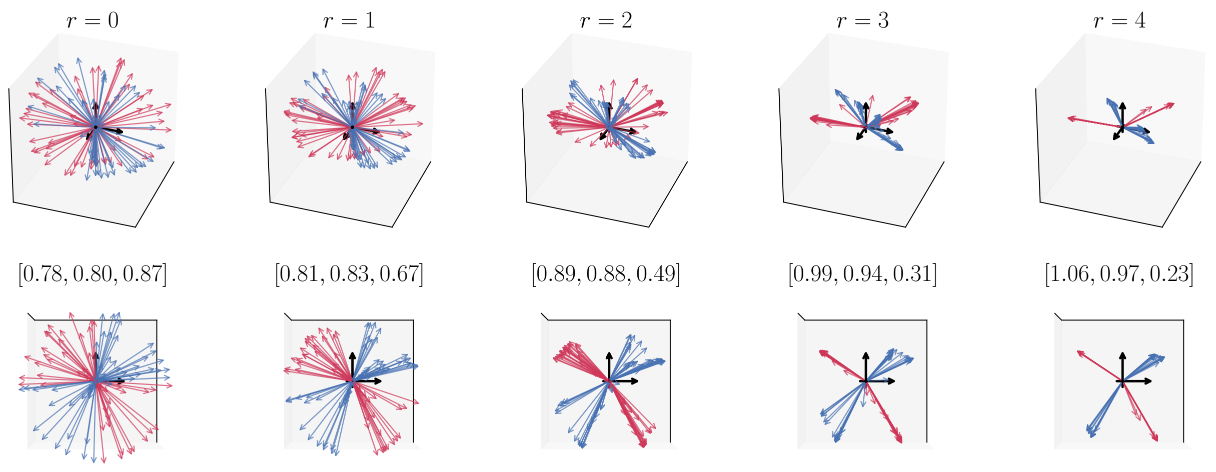


Figure 11: Feature vectors converging under iterated self-attention on a XOR_2 classification of vectors uniformly sampled over the sphere, in 3D (top) and down the z-axis (bottom). Colours indicate classes. The vectors quickly align by xy-quadrant and the variation in z is 'washed-out,' also seen in the feature selection scores (mean-absolute-deviation) $[x, y, z]$.

F. Experimental details

We use the same feature extractor architecture and train loop for all the baselines.

Feature extractor. We leverage a convolutional neural network with 4 blocks as a feature extractor. Each block consists of a convolutional layer (64 output channels and 3×3 filters), followed by batch normalisation (momentum 0.01), a ReLU activation, and 2×2 max pooling:

$$\text{Conv2d}(64, 3 \times 3) \rightarrow \text{BN} \rightarrow \text{ReLU} \rightarrow \text{MaxPool}(2 \times 2)$$

Then, we flatten the output and apply a linear layer to map the data into a 64-dimensional embedding space (unless otherwise stated). As explained in the main manuscript, each method then manipulates these embeddings in different way.

Train loop. We train all models in an episodic manner. At each training iteration, we follow these steps:

- First, we sample task-specific support and query sets. For polythetic MNIST, the support set has 96 samples (2 classes, 2 groups per class, and 24 group-specific examples per group). The query set consists of 32 samples (2 classes, 2 groups per class, and 8 group-specific examples per group). For Omniglot, the support set consists of 5 examples for 20 classes (20-way, 5 shot). The query set consists of 15 examples per class.
- Second, we compute embeddings using the feature extractor and produce class probabilities for the query points. The way in which these probabilities are computed depends on the method (e.g. attentional classification for matching networks or softmax over prototype distances for prototypical networks).
- Finally, we compute the cross entropy for the query examples and optimise the feature extractor via gradient descent. We employ an Adam optimise with learning rate 0.001.

We train the models for 10,000 iterations (i.e. tasks) for all experiments, except for full polythetic MNIST (100,000 tasks). We then compute the performances on a held-out dataset and average the results across 1,000 tasks.

G. Multi-categorical pre-training

In this experiment we first train a classifier in the multi-categorical setting for the full polythetic MNIST task. In this case there are no task-irrelevant digits or colours as the label describes all four digits with their colours: there are four 10-way labels for the digits ($\in \mathbb{R}^{4\times 10}$) and four 3-way labels for the colours ($\in \mathbb{R}^{4\times 3}$) for a combined multi-hot output vector $\in \mathbb{R}^{52}$. The model architectures match that used in the other experiments, see Appendix F, other than using a variation of the MLP head that takes the flattened output from the convolutional network. The variation has two hidden layers with 512 units each and ReLU activations, before linear layers with softmaxes for each of the label heads. The model is pre-trained over 800 batches of 16 examples drawn at random from the label combinations. In the multi-categorical pre-training, the model achieved a validation accuracy of 95.6% on the digit labels and 100% on the colour labels over 1600 validation examples.

The pre-trained model was then used with prototypical and attentional classifiers in the polythetic MNIST few-shot classification setting discussed in Section 5 and detailed further in Appendix F. We compared performance using the multi-headed softmax activations the model was pre-trained with and an elementwise sigmoid for both the monothetic and polythetic settings. The results are presented in Table 7, and conform to the trend we see in other experiments: threshold classification has the advantage in monothetic tasks but perform no better than chance for polythetic tasks. Attentional classifiers are weaker in the monothetic setting, but more than make up for this defeict in the polythetic setting.

Table 7: Polythetic MNIST problem with multicategorical pre-training. Mean over 1000 tasks.

	Softmax		Sigmoid	
Model	Monothetic	Polythetic	Monothetic	Polythetic
Proto.	97.95	50.18	94.64	50.21
Attn.	93.48	93.40	88.16	86.24

H. An illustrative example of the method

Figure 12 provides a more comprehensive overview of the proposed method. In this example, we are interested in distinguishing between big-cats and equids (horses, donkeys, and in the case zebra). We first extract features from all samples in the support and query sets. Here, we imagine these features as corresponding to some general 'cat' properties, patterns (such as stripes, dots etc.) and general 'equid' properties. Next (as presented in lines $2:4$ in Algorithm 1), we perform repeated self-attention with respect to the separate classes of the support set, which yields updated features for each support sample.

We then aggregate the resulting support features with an appropriate dispersion metric, e.g. mean absolute deviation or standard deviation, (line 5 in Algorithm 1) to obtain a vector of feature scores. These scores quantify the relevance of each feature in a given task. Next, we rescale both the query and (initial) support features, i.e. multiplying by the feature scores, to dilute the task-irrelevant and (potentially) misleading features. In this particular example, this would correspond to diluting the 'patterns' feature, since it is irrelevant when distinguishing between cats and equids. Finally, we produce class probabilities via an attentional classifier.

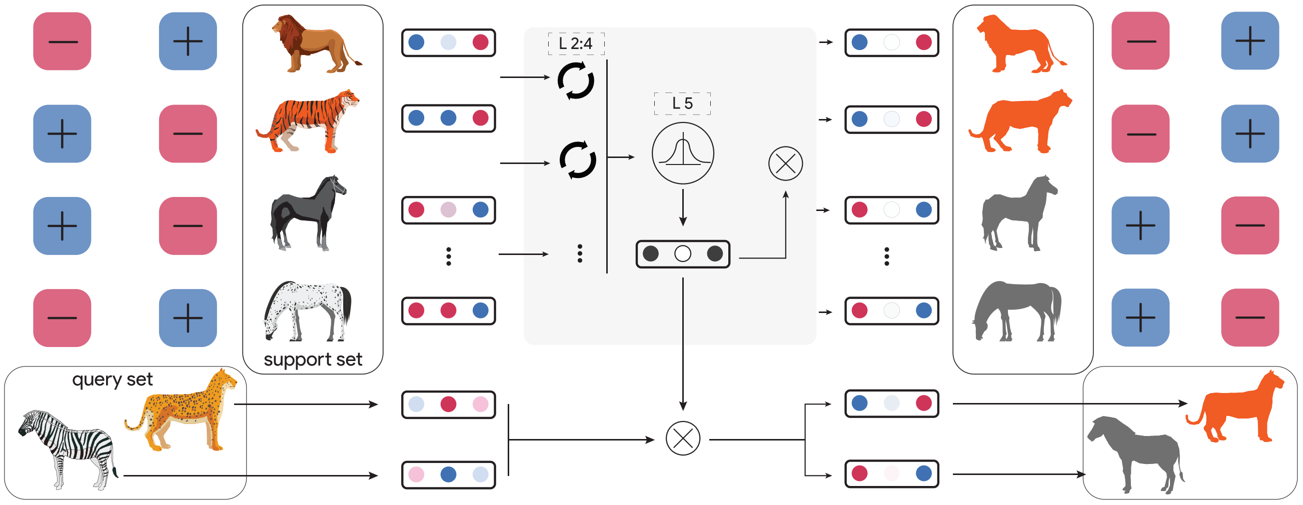


Figure 12: Diagram of the proposed approach. Note how the misleading features of stripes and spots, which may cause misclassification, are diluted through the feature selection process.

I. Construction of binary strings tasks

For binary strings tasks, we construct training tasks by sampling $|\mathbb{S}| = 5 \cdot 2^{\alpha}$ support examples and $|\mathbb{Q}| = 5 \cdot 2^{\alpha}$ query examples. Each example consists of $\alpha + \beta$ bits. The location of the α active bits is completely random and consistent between the support and query sets within the same task. The labels are computed as the XOR over the α active components. The remaining β noisy components are randomly sampled from a Bernoulli distribution with protability 0.5. Figure 13 shows an example of a binary strings task with $\alpha = 2$ active components, $\beta = 3$ noisy bits, and $r = 1$ repetitions.

Support set	**Query set**
$- - - + - \rightarrow -$	$+ + - - - \rightarrow -$
$+ - - - + \rightarrow +$	$- + - + + \rightarrow +$
$+ + + - - \rightarrow +$	$- + + + - \rightarrow +$
$- + + - + \rightarrow -$	$- - + + + \rightarrow -$

Figure 13: Example of an XOR$_{\alpha}$ task with $\alpha = 2$ active components (3rd and 5th bits), $\beta = 3$ noisy bits, and $r = 1$ repetitions for each combination of active components. The support and query sets contain $r2^{\alpha}$ examples each.

J. Construction of polythetic MNIST tasks

For polythetic MNIST tasks, the support set consists of 96 samples, with 48 samples for class 0 and 48 samples for class 1. Each class is further divided into 2 groups of 24 samples and each group is defined by a specific set of traits. The groups are complementary between classes, e.g. red ones and blue zeros for class 0; and blue ones and red zeros for class 1. The set of traits is sampled randomly for each task. The query set is sampled in the same manner, with 2 groups per class and 8 samples per group. Tables 8 and 9 summarise the details of the support and query sets and Figures 14, 15, and 16 show the whole support set for the clean, colourless, and full versions of polythetic MNIST.

$\lvert\mathbb{S}\rvert = 96$	Examples	Groups	Examples per group	Example of groups
Class 0	48	2	24	green ones and blue threes
Class 1	48	2	24	blue ones and green threes

Table 8: Support set details

$\lvert\mathbb{Q}\rvert = 32$	Examples	Groups	Examples per group	Example of groups
Class 0	16	2	8	green ones and blue threes
Class 1	16	2	8	blue ones and green threes

Table 9: Query set details

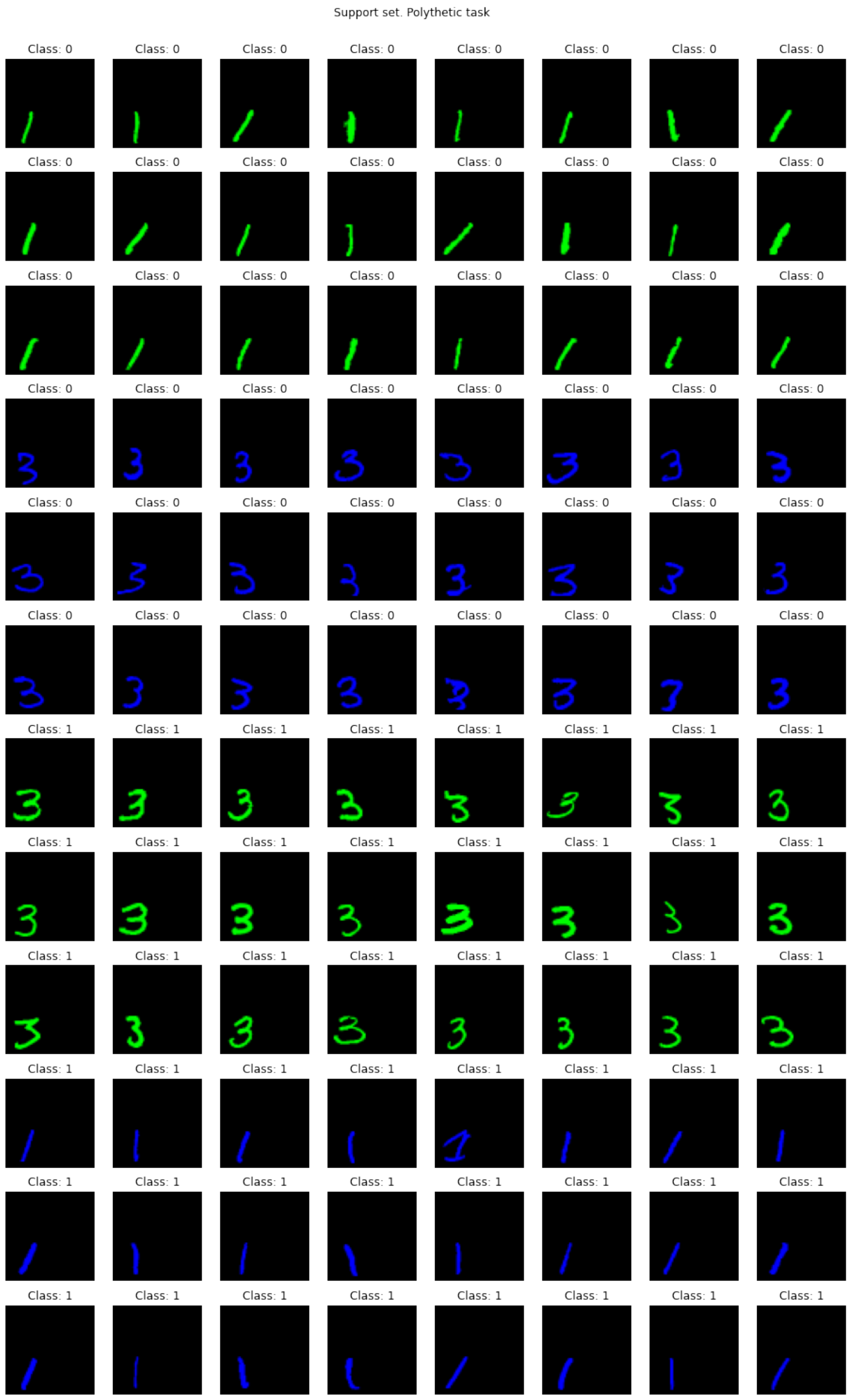


Figure 14: Example of the support set for a polythetic MNIST task (clean).

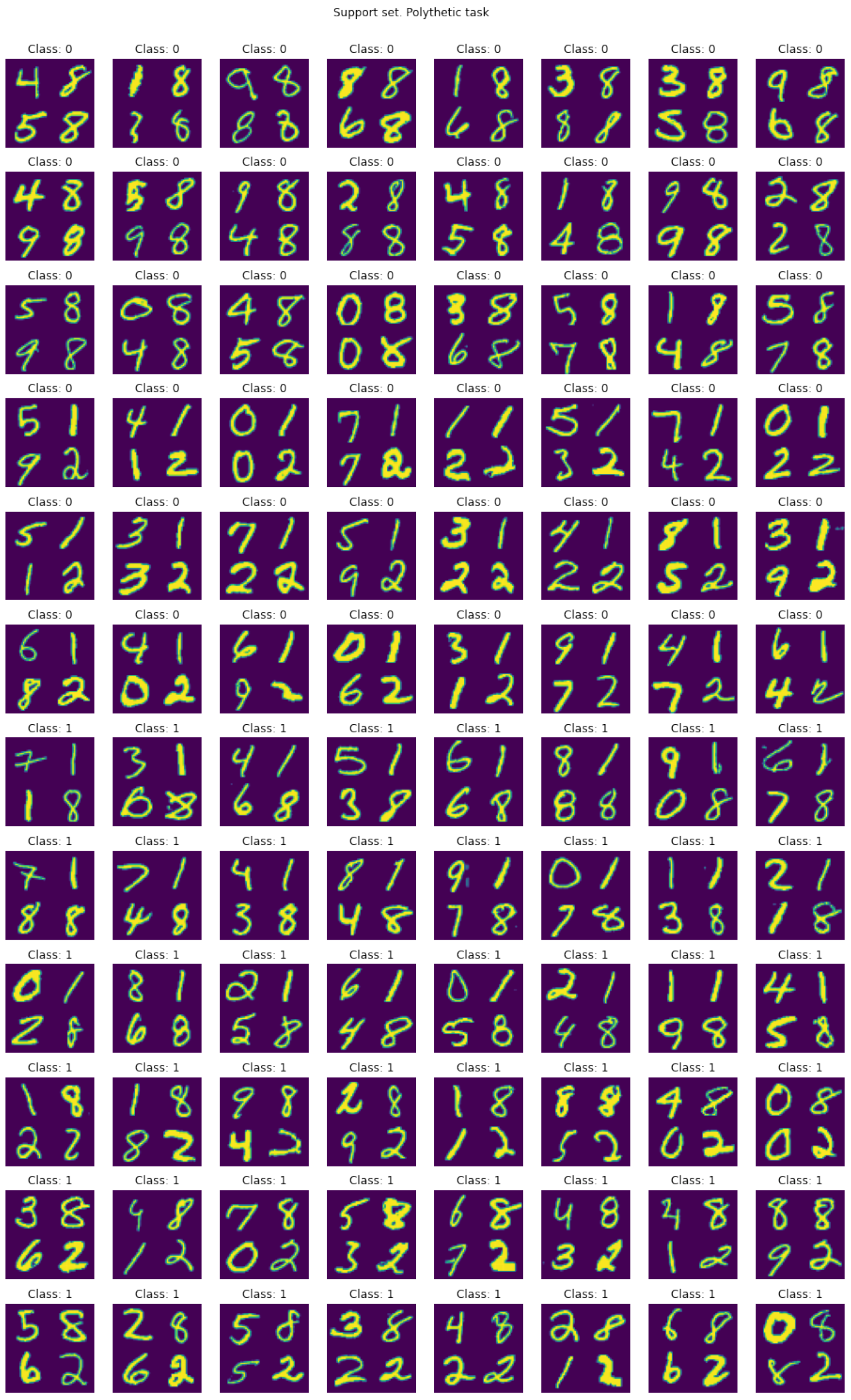


Figure 15: Example of the support set for a polythetic MNIST task (colourless).

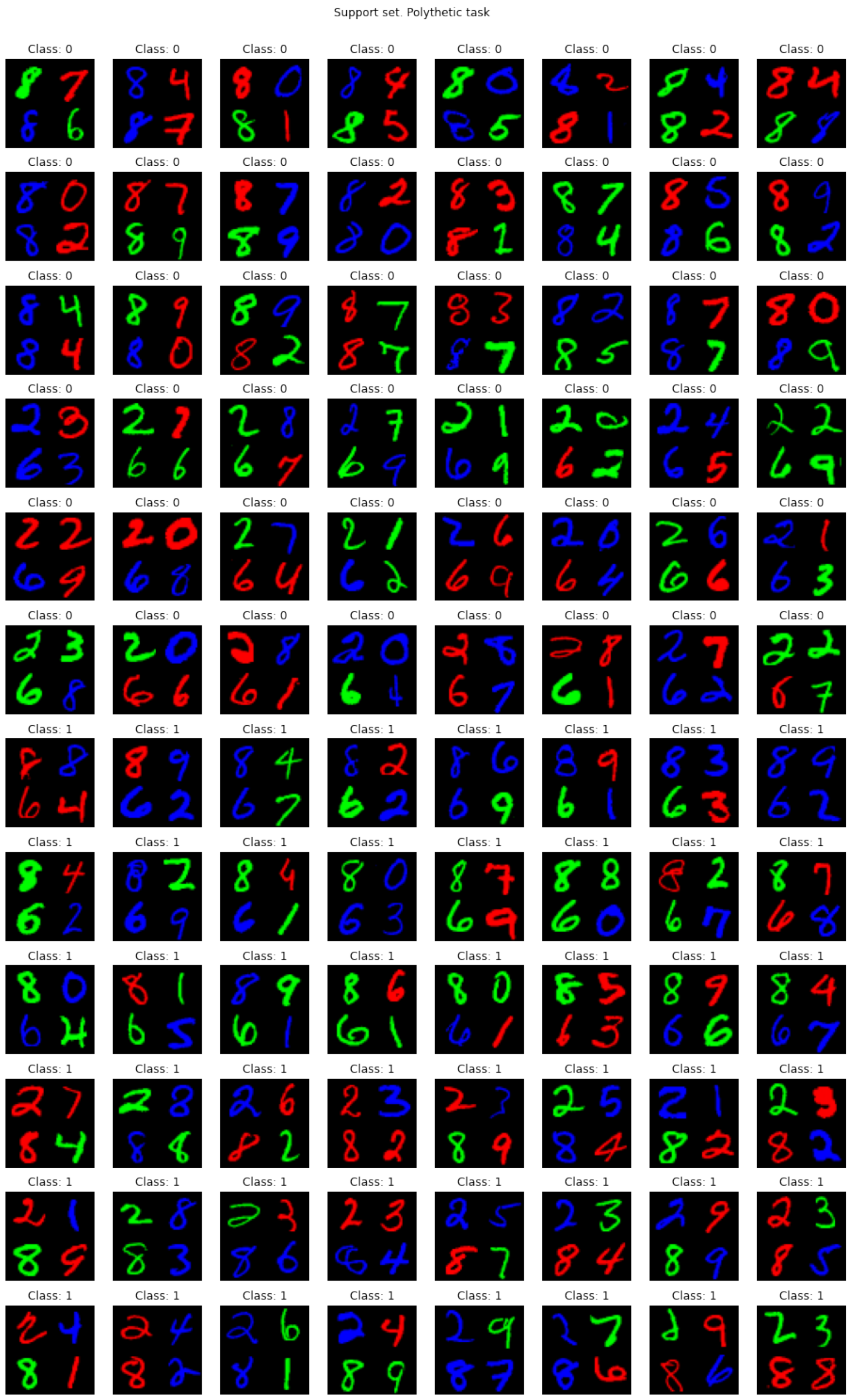


Figure 16: Example of the support set for a polythetic MNIST task (full).