Polis, Arturs; Ilin, Alexander

A Relational Model for One-Shot Classification of Images and Pen Strokes

Published in: Neurocomputing

DOI: 10.1016/j.neucom.2022.06.004

Published: 28/08/2022

Document Version
Publisher's PDF, also known as Version of record

Published under the following license: CC BY

Please cite the original version:
Polis, A., & Ilin, A. (2022). A Relational Model for One-Shot Classification of Images and Pen Strokes. Neurocomputing, 501, 1-13. https://doi.org/10.1016/j.neucom.2022.06.004

This material is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of the repository collections is not permitted, except that material may be duplicated by you for your research use or educational purposes in electronic or print form. You must obtain permission for any other use. Electronic or print copies may not be offered, whether for sale or otherwise to anyone who is not an authorised user.
A Relational Model for One-Shot Classification of Images and Pen Strokes

Arturs Polis\textsuperscript{a,}\textsuperscript{*}, Alexander Ilin\textsuperscript{b}

\textsuperscript{a}Aalto University, Espoo, Finland

\textsuperscript{b}Aalto University, Espoo, Finland

\textbf{A R T I C L E   I N F O}

\textbf{Article info}

Received 8 February 2022
Accepted 6 June 2022
Available online 8 June 2022

\textbf{Keywords:}
One-shot learning
Relational learning
Transformer
Graph network
Omniglot

\textbf{A B S T R A C T}

We show that a deep learning model with built-in relational inductive bias can bring benefits to sample-efficient learning, without relying on extensive data augmentation. Our study shows that excellent results can be achieved with a model in which the relational inductive bias is applied to images, while building an efficient one-shot classifier on top of raw strokes is more challenging. The proposed one-shot classification model performs relational matching of a pair of inputs in the form of local and pairwise attention. Our approach solves with almost perfect accuracy the one-shot image classification Omniglot challenge when combined with a Hungarian matching algorithm and attains competitive results on the same task on characters represented as rotation-augmented strokes.

© 2022 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

\section{1. Introduction}

As humans, we model the surrounding world with a help of a relational structure in our minds [1]. We intuitively see the world as composed of meaningful objects and relations. The objects themselves can be combined or decomposed to larger or smaller objects, depending on the task at hand. For example, in the case of car the tasks can range as wide as hailing a cab or fixing a flat tire, clearly requiring different levels of object granularity. It has been argued [2] that one way to deal with the inherent ambiguity of structuring the world in terms of objects is to identify familiar parts from which the objects are composed, clearly in the realm of visual understanding, the human ability to see visual scenes as novel compositions of potentially familiar parts allows us to memorize previously unseen types of objects quickly.

Structuring the world through the lens of object relations is known as relational inductive bias. Recent deep learning models such as Graph Neural Networks [3] and Transformer [4] are explicitly designed to model relations between parts of the model input. In this paper, we explore how such relational approaches can help in sample-efficient learning, that is learning to generalize from few training examples.

A suitable benchmark for our study is the Omniglot [2] dataset of hand-drawn characters (Fig. 1). Each character in the dataset is available both as a bitmap image as well as a set of corresponding pen stroke coordinates in space and time. Treating characters as a set of strokes is a natural candidate for an object-based representation which can be processed by relational models such as graph neural networks. Each stroke can be thought of as a separate object instance from some large library of all possible pen trajectories, and the relative distances between the centers of each stroke can be thought of as relations or edges in a graph. Conversely, a model with a strong relational inductive bias can be expected to infer the stroke structure from the images as well.

In our earlier work [5] we showed that using image based representation of handwritten characters coupled with a Transformer-based model [4] with relational inductive bias yields excellent results on the Omniglot benchmark. In this paper we compare our previous work with experiments on structured, stroke-based inputs and a corresponding graph neural network model. We test whether the stroke-based representation can yield sample-efficient adaptation to new classification tasks, which could motivate the importance of developing such object-based representations. Our results, however, (see Section 4) show quite surprisingly, that applying a relational model to images yields better results compared to the same model trained on the stroke-based representation. We also include a new experiment of generalizing across datasets, with model trained on QuickDraw [6] being evaluated on Omniglot, obtaining good results.

When faced with pairs of unfamiliar objects, when we compare a pair of images to determine whether they both depict the same thing, we may first scan each individual image to get a general idea about the image, and then compare different parts of one image to another to establish whether there exists a correspondence. In our study, we test if we can apply the same basic logic in a deep learn-
ing model. We first apply self-attention on each individual input, followed by cross-attention between two inputs, and finally aggregating the results of the attention operation into a scalar score denoting model confidence. The attention mechanism in our model is borrowed from the Transformer model. Using this model we are able to achieve state of the art accuracy on unaugmented within alphabet classification task on image inputs.

A nice property of the proposed model is the possibility to visualize the similarities between different parts of the compared examples (by visualizing the attention patterns), thus providing an explainable and interpretable one-shot classifier. We observe that the attention patterns obtained on Omniglot are meaningful: the classifier matches parts of the inputs that represent the same strokes. On the other hand, when applied to natural images, it becomes evident that the model pays more attention to similarities in textures rather than semantically similar high-level objects.

We train our model on image inputs as well as strokes and point trajectories, with results showing a similarly interpretable mapping between parts of each input, whether they are individual point-embeddings or local image features. We discover that attention on regions in points and image domains behaves similarly. However image input features are in general more stable with respect to the amount of training data, and do not require augmentation to reach good accuracy. This is somewhat surprising, as one would assume that stroke trajectories contain the same amount of relevant information as the images.

Our main contributions are as follows:

- Showing that a relational model can have excellent performance without the need for data augmentation, obtaining next to perfect accuracy on the Full Omniglot within-alphabet classification task.
- Interpretability by visualizing the attention masks developed during classification.
- We show that the proposed model works well on different modalities: strokes, trajectories of points, and images.

2. Related Work

2.1. Relational Learning

Graph Neural Networks (GNN) [3,7,8] have been recently shown to be effective on different types of relational data. We use Graph Matching Networks (GMN) [9] for our baseline. GMN compares pairs of graph inputs by embedding each graph using gated aggregation [7] and learning a relative embedding distance between the two graphs. It also supports optional attention between nodes of the two input graphs.

Our data is easy to represent as a fully-connected graph of characters strokes or alternatively image patches. For this reason we use attention-based message passing in our model. Our approach is related to SuperGlue [10], which treats local image features as relational inputs, and learns to match multiple key-points between two images. Each input graph is composed of local features extracted from two images in the pair. The resulting graph has two types of edges: edges between the nodes corresponding to the local features of the same image, and edges that link the local features of one image to those of the other. Message passing in each iteration is performed using self-attention followed by cross-attention. We extend the above approach by allowing image matching, as well as matching of stroke and point-trajectory based inputs. We treat local image features as “object” representations similarly to Relation Networks [11]. Relational models with Transformer-based attention have recently been shown to succeed in other vision tasks such as supervised image classification [12] and unsupervised feature learning [13].

The original contribution of our approach is in combining the ideas from SuperGlue with Transformer-based image classification. Standard Visual Transformer [12] works on individual images, with network output aggregated into a single “class token” per image. SuperGlue, on the other hand, takes two images as an input, and is designed to output (sparse) matching key-point locations between the two images, without aggregating the output. In our work, we perform attentive matching inspired by SuperGlue, followed by the final aggregation similar to the one in Visual Transformer.

In addition, methods tailored for working with point cloud based inputs have emerged recently [14–16]. Our model is not designed exclusively for handling point-based input, and instead treats trajectories of points as one of the possible input modalities.

2.2. Few-Shot Learning

A number of methods have been proposed for one- and few-shot classification of images. The methods include meta-learning [17], metric-based methods [18], as well as relational methods [19–21]. The first set of methods seeks to learn a way to quickly optimize the model to work well on unseen examples once these
examples have been provided. The second set of approaches aims to learn an embedding in some metric space, where nearby embeddings correspond to similar examples. The third set of approaches takes a relational approach by comparing the individual data points to each other. In addition, some methods model each few-shot classification episode as a graph [22,23] with each input corresponding to a single node in that graph, which is not the approach we take, since we model each input and/or pair of inputs as a graph instead.

Typically, few-shot learning models use episodic learning, where training data is split into n-way classification episodes, and the loss is optimized for the classification accuracy averaged across the episodes. Unlike many other few-shot learning models, we train on the entire training set, similar to [24], instead of using episodes.

Contrary to Prototypical Networks [18] our model can learn a pairwise similarity between pairs of inputs, instead of relying on the absolute L2 distance between input embeddings. ARC [19] learns a shared, pair-wise similarity embedding by using a recurrent network to iteratively compare two images using attention-based “glimpses” from one image to another, the shared embedding is then converted to a matching score in the range from 0 to 1. Unlike ARC, we model the input pair as two sub-graphs with two types of edges, where each sub-graph corresponds to the respective input. In addition, our aggregation approach can be thought of as conceptually similar to the one in [20], where the score of the match is calculated jointly for the pair. However, we are not using convolutional layers in the aggregation stage, allowing us to use input modalities other than images. PARN [21] extends [20] by introducing 1) deformable convolutions which expand the relative importance of the class-dependent features and 2) self and cross-correlation layers that use cosine-based attention for comparing local features. Unlike the above, our model can work on non-convolutional features. However, our motivation for using self and cross-attention layers is similar to the use of self and cross-correlation layers in PARN.

2.3. Stroke-based models

There exists a number of models that rely on hand-written character stroke data. The original BPL model [2] published with Omniglot dataset employs a probabilistic model of characters based on individual strokes, utilizing both the strokes and the images to learn the model. Other non-neural methods, such as [25] have been used for classifying character strokes without relying on any image input. In addition, some recent work [26,27] aims to model images as motor programs, where the pen stroke sequences are generated autoregressively conditioned on the image input.

3. Relational Modelling of Images and Strokes

We consider a task of n-way one-shot classification. The training is performed on pairs of inputs, A and B, from the training set. The model needs to decide if the pair is from the same class or not. The test set is composed of classes disjoint from the training set. The evaluation procedure uses testing episodes, where for each episode we sample n distinct classes from the test set and the corresponding support examples, so that the labels of support examples are known. We then pick n query examples corresponding to those classes, with per-example labels unknown to the model. During each testing episode we pair each query input with each support input and pass them to our model. For each pair, the model outputs pair-wise similarity. Each query example is labeled according to the most similar support example, as calculated by the model. The testing accuracy of the model is then defined as the fraction of the correctly assigned labels.

3.1. AttentiveMatcher

As humans, when we compare two unfamiliar images to see if they show the same item, we may first look at each image to verify its contents, and then compare parts of each image to one other to see if all parts match. The same idea informs the approach presented in this paper: our model uses self-attention to learn the structure of each image, and cross-attention to match the corresponding parts between the two images, then the model aggregates this information to determine if images belong to the same class. With the above intuition in mind we call our model AttentiveMatcher.

Our approach is relational, because it relies on the pairwise message-passing between parts of the inputs, similar to earlier work on Relational Networks [11,20], where the output of the convolutional network at each grid location was used as a feature input to a Relational Module comparing the local features to one other. In our model, the final match/no-match decision between the inputs is thus a function of two types of relational operations – matching the local features within each input and matching the local features between the two inputs.

The training of AttentiveMatcher is performed on pairs of inputs, A and B, to predict whether they belong to the same class or not. The architecture of the model is illustrated in Fig. 2a. In the case of images as input, we use a convolutional network to embed the images, obtaining a grid of 16 × 16 super-pixels, which we treat as 256 features. Similar to the hybrid model in [12] we use these features as inputs to the Transformer layers. When the inputs are strokes or points, we embed each stroke or point using an MLP, resulting in a variable number of features per input.

First, we separately perform self-attention on the features coming from the two inputs, with queries Q, keys K and values V coming from the same input:

\[ m_{ij} = \text{mha}(Q = a_i, K = A^{(i)}, V = A^{(i)}) , \]

where mha is the standard multi-head attention from Transformer, \( m_{ij} \) is the message to position \( i \), \( a_i \) is a vector of local features at position \( i \) and \( A^{(i)} \) is a matrix of stacked local features at all positions. All local features are then updated as

\[ a_i^{(i)} \leftarrow a_i^{(i)} + \text{LN} ( f(a_i^{(i)}, m_{..})) . \]

where \( f \) is an MLP that receives local features \( a_i^{(i)} \) and the message \( m_{..} \) as inputs and LN denotes layer normalization [28].

Next, we perform cross-attention between the features in the two inputs, with queries \( Q \) coming from one input, while keys \( K \) and values \( V \) coming from the other one:

\[ \mu_{i} = \text{mha}(Q = a_i^{(i)}, K = B^{(i)}, V = B^{(i)}) , \]

where \( \mu_{i} \) is the message to position \( i \) in input \( A \) and \( B \) is a matrix of stacked local features of all positions in input \( B \). Messages to positions of input \( B \) are computed symmetrically. Next, the local features are updated using the cross-attention messages:

\[ a_i^{(i+1)} = a_i^{(i)} + \text{LN} ( g(a_i^{(i)}, \mu_{..})) , \]

where \( g \) is modeled with an MLP. The MLPs for \( f \) and \( g \) contain one hidden layer followed by a ReLU activation.

We use the Transformer [4] attention layers with the modifications inspired by [10]. Unlike in Transformer, where the outputs of the multi-head attention blocks are passed through a feed-forward block directly, we first concatenate the attention outputs with the input features before passing them to the feed-forward block fol-
A single scalar score predicting the match between across both inputs. We use the binary cross-entropy loss to tune the parameters of the model.

\[ L_{\text{binary}}(\mathbf{g}_a, \mathbf{g}_b) = -\sum_{i} \left( y_{ij} \log p_{ij} + (1 - y_{ij}) \log (1 - p_{ij}) \right) \]

where \( y_{ij} \) is the label indicating whether the \( i \)-th example from \( A \) matches \( j \)-th example from \( B \) and \( p_{ij} = \sigma(\mathbf{g}_a^T \mathbf{g}_b) \).

The testing episodes are constructed by pairing each query input with each support input, resulting in \( N \times N \) comparison pairs for each episode. Each query example is assigned a label of the support example with the highest score \( s \). The model accuracy is defined as the fraction of the labels correctly assigned.

To compare our model with ARC [19], which is the previous state of the art for the task, we also experimented in the scenario assumed by ARC when the query examples in one episode are known to belong to distinct classes. We used Hungarian matching [29] which assigns each support example to a unique query example in the episode, while maximizing overall score.

4. Experiments and results

4.1. Omniglot challenge

Lake et al. [2] proposed Omniglot benchmark to measure progress of generalization from few examples in various tasks such as classification, generation of new examples and parsing objects into parts. Omniglot dataset contains characters from 50 alphabets, 1623 character classes in total, each character represented by only 20 hand-drawn instances, available both in image and pen trajectory formats (a few samples are shown in Fig. 1). The dataset is split into a 30-alphabet Background set (with 964 classes) used for training, and a 20-alphabet Evaluation set (with 659 classes). Recently, the more challenging, “Minimal” [30] version of the training data has been proposed, containing two five-alphabet subsets of the Background set. The evaluation procedure of the \( N \)-way one-shot classification task consists of multiple episodes. Each episode is a classification task with \( N \) classes from the same alphabet. The classifier can use \( N \) support examples (one example per class) to assign \( N \) query examples to one of the \( N \) classes.

The authors of Omniglot developed a generative probabilistic model called Bayesian Program Learning (BPL), as a non-neural baseline for Omniglot tasks. BPL has held the state of the art for the most challenging setup: the 20-way one-shot classification task without the use of data augmentations during training, it also generalizes very well from the Minimal training set (see Table 1).

The authors argue in [30] that the challenge remains unsolved because the human level on the within-alphabet classification has been out of reach for deep learning models that do not rely on heavy data augmentation. Most of the published research on Omniglot classification addresses a simpler between-alphabets classification task, in which testing episodes contain characters from different alphabets, making the examples easier to distinguish.

We follow [2] and use the Background set for training and Evaluation set for validation and testing. For evaluation and results reporting, we use pre-defined episodes from the Evaluation set with \( N = 20 \), which were proposed by the authors of Omniglot [2]. We validate our models by forming random within alphabet episodes from the ten alphabets not in the pre-defined test set. We used accuracy on our validation set as the early stopping criterion.

4.1.1. The Graph Matching Baseline

To evaluate how well stroke-based representation works in a one-shot setting, we re-implemented and trained GMN [9] on strokes processed in the same way as in BPL. In the GMN baseline,
we represent our input as two fully-connected graphs: \( G_A = (V_A, E_A) \), and \( G_B = (V_B, E_B) \), where each graph \( G \) is composed of a set of nodes \( V \) and edges \( E \). While we have access to stroke data, we have no canonical way of separating the stroke information into a graph structure. We used interpolated strokes of fixed length of 10 points per stroke to construct graph nodes and edges. We treated zero-centered strokes as nodes and we constructed a bi-directional edge between each pair of nodes by taking as the feature of the edge the difference of the centers of mass of the corresponding strokes. This relational representation preserves the original information about the shape of each stroke, but has an added benefit of being translation-invariant w.r.t. the original input. We also performed experiments with representation where strokes were further divided into sub-strokes, but this did not yield noticeable improvements.

To make the decision whether a pair of characters belong to the same class, we match their stroke-based representations using a Graph Matching Network. The model learns an embedding of each graph in the input pair by performing \( T \) rounds of message passing. Each node and each edge in the model have an associated set of features. The features of the nodes are updated on each iteration, while edge features stay the same. After the final message-passing iteration, the node features are aggregated into an embedding vector (see Eq. 2) that can be compared to an embedding vector of another input graph using Euclidean distance, thus allowing to measure how similar the input graphs are to each other. The GMN architecture also has optional cross-graph attention, where at each message passing step, cross-attention between the input graphs \( G_A \) and \( G_B \) is added to messages within the graph, which we implemented as described in [9].

We ran experiments on the GMN baseline both with and without cross-graph attention. When training the model without attention, we apply the loss to all combinations of pairs \((A, B)\) in each mini-batch. When cross-graph attention is enabled, we first pair together as many inputs from the same class as possible, and then randomly pair the remaining inputs. Please see A.1 for GMN implementation and training details.

### 4.1.3. AttentiveMatcher Training Details

We embed the input for AttentiveMatcher in the following manner:

- For image inputs, the features are obtained through a four-layer convolutional network commonly used in few-shot learning models [18,20]. The first three layers have 64 channels and the fifth layer has \( d_f \) output channels. We use kernel of size \( 3 \times 3 \), with padding and the stride of one. We pooled the output of the first two layers using \( 2 \times 2 \) MaxPool operation, resulting in a \( d_f \times 16 \times 16 \) convolutional feature map which we flatten into a sequence of 256 local features of size \( d_f \) that we use as an input to AttentiveMatcher. Each layer was followed by Batch Normalization [31] and ReLU non-linearity.

- Each point feature is embedded using an MLP with one hidden-layer of size \( 2 \times d_f \), followed by a ReLU non-linearity and an optional Dropout [32] layer.

We set \( d_f = 128 \), \( d_A = 2048 \) for image inputs, \( d_f = 64 \), \( d_A = 1024 \) for points inputs and \( d_f = 256 \), \( d_A = 512 \) for stroke inputs. We use a batch-size of 256 for all input types. We used both positional encoding [4] and layer normalization. We use Adam [33] optimizer, with starting learning rate 0.0001, we decrease the learning rate by the factor of 0.1 every 100 epochs, in addition we use a weight decay of 0.001. Because of training the model in the pairwise fashion, we sorted each batch so that there are as many comparisons between the inputs belonging to the same class as possi-

---

Table 1

| Model                  | Full | Minimal | Full | Minimal |
|------------------------|------|---------|------|---------|
| Human                  | 95.5 | –       | –    | –       |
| BPL                    | 96.7 | 95.8    | –    | –       |
| Prototypical Networks  | 86.3 | 69.9    | 94   | –       |
| ARC                    | –    | –       | 97.5 | –       |
| ARC (full context)     | –    | –       | 98.5 | –       |
| GMN with attention – strokes | 71.2 | 49.1    | 74.7 | 54.6    |
| GMN without attention – strokes | 79   | 54.9    | 86.5 | 61.6    |
| AttentiveMatcher (ours) – strokes | 75 ± 0.66 | 45.96 ± 0.46 | 78.4 ± 0.72 | 59.12 ± 0.52 |
| AttentiveMatcher (ours) – points | 83.82 ± 2.67 | 43.32 ± 2.06 | 92.62 ± 1.61 | 68.35 ± 2.67 |
| AttentiveMatcher (ours) – images | 97.65 ± 0.4 | 79.95 ± 2.02 | 97.46 ± 0.55 | 94.84 ± 0.7 |
| AttentiveMatcher (ours) – images (full context) | 99.77 ± 0.25 | 84.27 ± 2.08 | 99.29 ± 0.27 | 98.41 ± 0.5 |

1 https://github.com/brendenlake/BPL
ble. This was done because when ordering the batch at random it is much more likely for each pair being compared to contain inputs from different classes, thus preventing the model from receiving a positive training signal frequently enough, resulting in much longer convergence times. We clipped training gradients at 2.5 for images and points and 10 for strokes. We use pair-wise aggregation (2) and pair-wise loss for strokes, and score-based aggregation and binary cross-entropy loss (1) for points and images, setting γ = 40 in the pair-wise loss. When calculating the binary cross-entropy loss we weighed the positive pairs by a factor of 40 to make the loss calculation more balanced.

4.2. Quantitative Results on Omniglot

Table 1 contains the obtained results on the Omniglot datasets. We show results for both the Full Background set, as well as the averaged values for two five-alphabet Minimal Background sets. We run our experiments on both unaugmented and augmented versions of the training data. The augmentation we perform is done by adding extra classes to the training set, using rotations at 90° increments, similarly to [18], resulting in the total of 3856 training classes in the augmented full Background set.

AttentiveMatcher, trained on unaugmented Background images outperformed other published results, including BPL [2] and Prototypical Networks [18] and achieved perfect accuracy for some of the training seeds and close to perfect accuracy on average after applying Hungarian algorithm at test time, denoted by "full context" in the results. When training on the Minimal set, our model benefits from augmentation, and when full context is used, outperforms the previous state of the art, BPL. The model trained on points using data augmentation (three extra rotations per character) shows a comparable accuracy to our image model, while the unaugmented results on points fall significantly short of the corresponding image results. This is somewhat surprising, as one would assume that stroke trajectories contain the same amount of relevant information as the images. A likely reason for images performing better is the translation invariance induced by the convolutional network helping model to generalize better when compared to point-features prepared by simple MLPs.

When evaluating our model trained on the Minimal set, we start to see the effects of scarcity of the training data, and increased benefits of data augmentation, with our unaugmented image-based model performing much worse than the augmented model. Furthermore, the point model performs significantly worse on the Minimal set compared to both the image-based model and point-based model trained on the full Background set, confirming that the point-based model does not generalize as well as the image model. The results on image data, especially on a challenging Minimal dataset indicate that the convolutional inductive bias helps in generalizing when data is scarce.

All AttentiveMatcher models were trained with 15 different random seeds, and we show the results with 95% confidence intervals applied to mean accuracies obtained from models trained on each seed. The values for the models trained on the two Minimal Background sets are obtained by averaging the results for both versions of Minimal dataset.

The GMN baseline model built on top of stroke-based representations showed poor performance, with both versions of the model reaching significantly lower generalization accuracies than AttentiveMatcher model with points and images as the input. However, the AttentiveMatcher model with fixed-length strokes as input behaves similarly to the GMN model. We do note that a non-neural BPL [2] model exceeds human level performance, while modelling strokes and sub-strokes internally, which means that it is in principle possible to build a good stroke-based model for this task. Nevertheless, our experiments have shown that building a stroke-based neural architecture is challenging. In addition, we note that the parametrized attention used in the AttentiveMatcher can handle matching corresponding strokes better than the attention used in the GMN.

4.3. Qualitative Results on Omniglot

The model allows us to visualize the attention, as shown in Fig. 3. The cross-attention patterns obtained on Omniglot are meaningful: the classifier matches parts of its two inputs that represent the same strokes. Furthermore, similar attention patterns are learned on both the image and points data. First row shows images with superimposed color-mapped trajectories for which we show attention. Cross-attention is shown in both directions. Each odd column shows the attention in one direction (i.e. from A to B), each following (even) column shows attention in the opposite direction (i.e. from B to A).

In Fig. 4 we plot two matrices for support-query matching scores (1 = match, 0 = no-match) for a 10-way episode for both points and images inputs. When inspecting the model results, we noticed that the pairwise matching scores show high certainty of a match for visually similar characters, even though the characters do not share the same label.

4.4. Generalization Between Datasets

In order to see how well the features learned by the AttentiveMatcher generalize across datasets we performed an experiment where we trained the model on QuickDraw [6] human-drawn sketch dataset, and evaluated the resulting model on Omniglot data without any weight updates. QuickDraw dataset contains 345 classes with 70,000 hand-drawn examples per class. As can be seen in Fig. 5, there is a lot of variation between examples belonging to each class, which makes the task of matching examples from the same class more challenging.

We trained the model on 28 × 28 images, using the same 300 classes as in [27], we additionally applied data augmentation by using 90°, 180°, and 270° rotations making three new classes per each existing one. The augmentation benefited generalization because similar looking Omniglot characters can have different meaning depending on their orientation. To speed up training, we used the first 35000 examples from each class. We used the validation accuracy on the 45 remaining QuickDraw classes as our stopping criteria. We evaluated the trained model on 1000 randomly sampled within-alphabet 5-way and 20-way test episodes, where for each alphabet in the Evaluation set 50 episodes were sampled. We set the learning rate to 0.00001, with other training parameters same as in the models trained on Omniglot images. As shown in Table 2, our model performed well when compared to the recent SketchEmbedNet [27] baseline. It is worth noting, that the authors of SketchEmbedNet trained a linear n-way classifier for each evaluation episode, which is an easier task than our approach. We used full context evaluation in our experiments.

4.5. Experiments with minilImageNet

We also test the AttentiveMatcher on natural images using the popular minilImageNet [34] few-shot learning benchmark. We use the splits proposed by Ravi and Larochelle [35], consisting of 64 training classes, 16 validation classes, and 20 test classes. We apply standard pre-processing of 84 × 84 center crops and standard image normalization used with ImageNet. As can be seen from Fig. 6, the model is less certain about its matches in the 5-way minilImageNet classification, compared to the 10-way classification of Omniglot in Fig. 4.
When training on the miniImageNet data, for the embedding network, we set \(d_f = 64\), \(d_e = 256\), apply pooling to the first two convolutional layers and used dropout of 0.1. We also use attention dropout, which randomly removes a fraction of attention weights in the multi-head attention module, setting the attention dropout rate to 0.1. We use Adam optimizer, with a constant learning rate 0.0001. We use positional encoding, and sort our mini-batches so that images belonging to the same class were paired whenever possible. Layer normalization was not used. We clip the gradients at 2.5, same as with Omniglot. However, unlike with Omniglot we do not weigh the loss terms.

Table 3 shows miniImageNet 1-shot, 5-way classification results for models using four-layer CNN for image-features embedding, with 95% confidence intervals on the episode accuracies obtained from 1000 episodes. We can see that the model is functional but cannot beat the state of the art. We also studied the cross-attention patterns (see B) trying to see if one can achieve the same level of correspondence between parts of images being compared as observed in Omniglot experiments. We found that the cross-attention patterns were not meaningful: the model seems to pay more attention to similarities in textures rather than semantically similar high-level objects.

4.6. Discussion on Interpretability

A big benefit of using attention-based models, such as Transformers, is the readily available attention-weights matrix. The attention patterns can be used to examine the model’s inductive
bias on different datasets. As we saw, while the attention patterns on Omniglot images corresponded to how a human would compare character parts to one another, the same was not the case with miniImageNet, with color and texture playing a bigger role than shape and location. This failure of the model to attend to the corresponding image regions both within individual images (such as all patches corresponding to the head of the dog attending to one another), as well as between images (such as attending between the heads of dogs in two different images), can be used to diagnose the issues, such as texture bias [36], preventing the model to reach a higher accuracy.

Matching scores matrices is another valuable indicator on how easy or hard it is for the model to distinguish between classes. A more sparse matching scores matrix indicates that the model is more confident in its predictions. As the scores can be loosely interpreted as matching probabilities for each pair, this information provides a helpful guide for debugging the model. In addition, the pair-wise similarity scores can be compared to subjective scores given by humans.

5. Conclusion

Our model trained on images obtains next to perfect accuracy on the challenging Omniglot within alphabet classification task, exceeding both human level and BPL on unaugmented data, and to the best of our knowledge is the first model to do so. This demonstrates the importance of the convolutional inductive bias in sample efficient settings. At the same time, as humans we find it natural to represent hand-written characters as strokes, and our original expectation was that strokes would perform better than images. However, in our experiments using both Graph Matching Network on top of stroke-based representations, as well as AttentiveMatcher with strokes as input have shown that learning a non-autoregressive and non-generative representation from strokes that is effective for few-shot learning with neural networks is not trivial. However, the non-neural, BPL model by Lake et al. is based on modelling characters in terms of strokes, and it shows excellent classification performance. This supports the notion that a stroke-based representations should be useful for sample-efficient classification and other tasks considered by [2]. It cannot be ruled out that our image-based model performs so well because the convolutional features we use as an input to the AttentiveMatcher help the model learn an internal sub-stroke based representation in a translation equivariant manner. Overall, our model accuracy improved as we moved further away from a pure stroke-based representation, with points outperforming strokes, and images outperforming points. In addition, we have shown that

| Table 2 | Cross-dataset generalization results – model trained on QuickDraw data, evaluated on Omniglot within-alphabet 5-way and 20-way 1-shot classification tasks, 4-convolutional layers embedding. Results show 95% confidence intervals obtained from models initialized from 15 random seeds. |
|---------|-------------------------------------------------------------------------------------------------|
| Model               | Acc. 5-way | Acc. 20-way |
| SketchEmbedNet [27] | 89.16 ± 0.41 | 74.24 ± 0.48 |
| AttentiveMatcher (ours, no aug.) | 88.78 ± 0.5 | 67.11 ± 0.7 |
| AttentiveMatcher (ours, aug.) | 92.37 ± 0.3 | 74.93 ± 0.3 |

| Table 3 | miniImageNet 1-shot 5-way classification, 4-convolutional layers embedding. |
|---------|--------------------------------------------------------------------------------|
| Model               | Accuracy |
| Prototypical Networks [18] | 49.42 ± 0.78 |
| PARN [21]            | 55.22 ± 0.84 |
| AttentiveMatcher (ours) | 49.06 ± 1.3 |

Fig. 5. Some example QuickDraw images and their corresponding classes.

Fig. 6. Matching scores matrix for a single testing episode, miniImageNet. Values below 0.01 not shown.
our model when trained on Omniglot has double benefits of accurate classification and increased interpretability of the matches. We speculate that good results on QuickDraw and Omniglot suggest that the model is good at comparing abstract representations, as one could view sketch drawings as abstract representations of objects shown in natural images [16]. Our results suggest that more research should be done on improved learning of abstract representations from natural images since current models seem to be heavily focused on textures [36].

While the relational inductive bias can be helpful in solving few-shot learning problems, it may not always be possible to obtain a suitable relational representation. Thus, it may be even more important to design models that infer the graph structure from input data, for situations where the ground-truth relations are unavailable. There are still many exciting challenges that remain in adapting relational, graph-based approaches to non-graph data, as well as learning the optimal ways to build graphs from data with ambiguous relational structures. Working in this direction will drive the improvement in deep learning methods for both the relational representation as well as inference.

More work is needed to investigate the potential of this approach on natural images, where it is more challenging to develop meaningful structural and relational representations in terms of objects and object parts. In addition, the relation between model size and interpretability of attention patterns for input matching should be explored. Working in this direction will drive the improvement in deep learning methods for efficient and scalable structural representation.

**Declaration of Competing Interest**

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

**Acknowledgements**

We thank CSC (IT Center for Science, Finland) for computational resources and the Academy of Finland for the support within the Flagship programme: Finnish Center for Artificial Intelligence (FCAI).

**Appendix A. Architecture and Dataset**

**A.1. Architecture of the Graph Matching Network for One-Shot Classification**

When training the Graph Matching Network [9], we encode the node and edge features using two separate MLPs, with output dimensions $d_f$ and $d_{edge}$. Messages $m_{ij}$ between nodes $j, i$ are obtained by concatenating the two nodes and the corresponding

---

**Fig. A.7.** Omniglot strokes data statistics. Both raw and interpolated data has the same number of strokes per character, shown in (a). After pre-processing, in the resulting interpolated strokes the total number of points is decreased, as seen from comparing (b) and (c).
Fig. B.8. Visualization of cross attention for two miniImageNet images from the same class. Lime-green regions mark super-pixels corresponding to attention from, and red regions mark super-pixels corresponding to attention to. The target regions on the right mostly focus on the correct object, however there is no clear pair-wise correspondence to the regions marked on the left. Please see Fig. B.9 for cross-attention in the opposite direction.
Fig. B.9. Attention on the miniImageNet images in the opposite direction. There is no object/part based correspondence between attention from (lime-green) and attention to (red), and texture appears to play the biggest role for matching. See Fig. B.8 for attention in the opposite direction.
edge and passing them through another MLP. The message dimension is set to $d_m$. The GNN is run for $T$ iterations. At each iteration $t$ of the GNN, the nodes are updated using a GRU-based RNN cell, where node is the hidden state, and the sum of all incoming messages is the input:

$$a_i^{(t+1)} = \text{GRU}(a_i^{(t)}, \sum_{j \in V} m_{ij}^{(t)})$$

The authors of the above architecture also propose a simple form of cross-graph attention between nodes $V_A$ and $V_B$. The cross-graph attention messages $\boldsymbol{m}_{ij}^{(t)}$ between node $a_i$ in graph $G_A$ and node $b_j$ in graph $G_B$ is a weighted element-wise difference:

$$a_i^{(t+1)} = \text{GRU}(a_i^{(t)}, \sum_{j \in V} m_{ij}^{(t)})$$

with other similarity measures than L2 being possible as well. The per-node attention messages are summed, and concatenated to the input of the node update RNN:

$$\mathbf{a}_i^{(t+1)} = \text{GRU}(\mathbf{a}_i^{(t)}, \sum_{j \in V_A} \mathbf{m}_{ij}^{(t)}, \sum_{j \in V_B} \mathbf{m}_{ij}^{(t)})$$

This attention mechanism does not have separate learned transformations. Finally, at the last time-step, $T$, the two graph embeddings $\mathbf{g}_A$ and $\mathbf{g}_B$ of dimension $d_a$ are computed by applying a gated sum to the nodes belonging to each graph, here shown for nodes $a \in V_A$ (but are symmetric for $b \in V_B$):

$$\mathbf{g}_A = f_{emb}\left(\sum_{i \in V_A} \mathbf{f}_{node}(\mathbf{a}_i^{(T)}) \otimes \mathbf{f}_{gaa}(\mathbf{a}_i^{(T)})\right)$$

We use pairwise hinged margin loss that encourages L2 distance between same classes to be small:

$$\mathcal{L}_{pair} = \max(0, \gamma - \|\mathbf{g}_A - \mathbf{g}_B\|^2)$$

Parameter $\gamma$ is the margin, and $l$ is a positive or negative label for the pair, $l = 1$ when $A$ and $B$ have the same class, and $l = -1$ otherwise.

### A.1.1. GNN Training Details

When training GMN without attention we set $d_1 = 256$, $d_{edge} = 32$, $d_a = 128$, $d_b = 512$, $\gamma = 400$, dropout$=0.1$, batchsize$=512$, $T = 4$, with every time-step having separate message MLP weights.

For GMN with cross-graph attention: we set $d_1 = 256$, $d_{edge} = 32$, $d_a = 128$, $d_b = 256$, $\gamma = 400$, dropout is set to zero, batchsize$=256$, $T = 6$. Message MLP weights are shared across time steps.

For both of the above, we clip gradients at 10 and use Adam optimizer with learning rate of 0.0002. Batch normalization was used in all MLPs.

### A.2. Stroke Dataset Statistics

Fig. A.7 shows some statistics for the stroke data. One strokes data point contains strokes trajectories for a single character as sequences of points in three dimensions: $(x, y, t)$, where $t$ denotes time. The beginning and the end of each stroke is marked. The average number of strokes per character is 2.6, points per stroke: 64.5, and points per character: 168.9.

### Appendix B. Result Visualizations

We visualize cross-attention for positive class pairs in miniImageNet dataset. Figs. B.8 and B.9 show cross attention for pairs of images, where the attention seems to be matching somewhat meaningful corresponding regions, however on further inspection the matches are likely to be based on texture similarity.

### References

[1] P.W. Battaglia, J.B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, et al., Relational inductive biases, deep learning, and graph networks, arXiv preprint arXiv:1806.01261 (2018).

[2] B.M. Lake, R. Salakhutdinov, J.B. Tenenbaum, Human-level concept learning through probabilistic program induction, Science 350 (6266) (2015) 1322–1328.

[3] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE Transactions on Neural Networks 20 (1) (2009) 1312–1328.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in neural information processing systems, 2017, pp. 5998–6008.

[5] A. Poli, A. Ilin, A relational model for one-shot classification, in: European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, 2021, pp. 647–652.

[6] J. Jongejans, H. Rowley, T. Kawashima, J. Kim, N. Fox-Gieg, The quick, draw?! experiment. (2016), http://quickdraw.withgoogle.com (2016).

[7] Y. Li, D. Tarlow, M. Brockschmidt, R. Zemel, Gated graph sequence neural networks, arXiv preprint arXiv:1511.05493 (2015).

[8] J. Gilmer, S.S. Schoenhoff, P.F. Riley, O. Vinyals, G.E. Dahl, Neural message passing for quantum chemistry, in: International conference on machine learning, PMLR, 2017, pp. 1263–1272.

[9] Y. Li, C. Gu, T. Dullien, O. Vinyals, P. Kohli, Graph matching networks for learning the similarity of graph structured objects, in: International conference on machine learning, PMLR, 2019, pp. 3835–3845.

[10] P.-E. Sarlin, D. DeTone, T. Malisiewicz, A. Rabinovich, SuperGlue: Learning feature matching with graph neural networks, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 4938–4947.

[11] A. Santoro, D. Raposo, D.C. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, T. Lillicrap, A simple neural network module for relational reasoning, in: Advances in neural information processing systems, 2017, pp. 4967–4976.

[12] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al., An image is worth 16x16 words: Transformers for image recognition at scale, in: International Conference on Learning Representations, 2020.

[13] M. Caron, H. Touvron, I. Misra, H. Jégou, J. MAiral, P. Bojanowski, A. Joulin, Emerging properties in self-supervised vision transformers, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 9650–9660.

[14] C.R. Qi, H. Su, K. Mo, L.J. Guibas, Pointnet: Deep learning on point sets for 3d classification and segmentation, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 652–660.

[15] C.R. Qi, L. Yi, H. Su, L.J. Guibas, Pointnet++: Deep hierarchical feature learning on point sets in a metric space, in: Advances in neural information processing systems, 2017, pp. 5099–5108.

[16] Y. Wang, Y. Sun, Z. Liu, S.E. Sarma, M.M. Bronstein, J.M. Solomon, Dynamic graph cnn for learning on point clouds, Acm Transactions On Graphics (tog) 38 (5) (2019) 1–12.

[17] C. Finn, P. Abbeel, S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, in: Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR. org, 2017, pp. 1126–1135.

[18] J. Snell, K. Swersky, R. Zemel, Prototypical networks for few-shot learning, in: Advances in neural information processing systems, 2017, pp. 4077–4087.

[19] P. Shyam, S. Gupta, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al., An image is worth 16x16 words: Transformers for image recognition at scale, in: International Conference on Learning Representations, 2020.

[20] V.G. Satorras, J.R. Estrach, Few-shot learning with graph neural networks, in: International Conference on Learning Representations, 2018.

[21] J. Kim, T. Kim, S. Kim, C.D. Yoo, Edge-labeling graph neural network for few-shot learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 11–20.

[22] Y. Tian, Y. Wang, D. Krishnan, J.B. Tenenbaum, P. Isola, Rethinking few-shot image classification: a good embedding is all you need?, in: European Conference on Computer Vision, Springer, 2020, pp 266–282.
25] L. Van Der Maaten, Learning discriminative fisher kernels., in: ICML, Vol. 11, 2011, pp. 217–224.
[26] D. Ha, D. Eck, A neural representation of sketch drawings, in: International Conference on Learning Representations, 2018. URL: https://openreview.net/forum?id=Hy6GHPkCW.
[27] A. Wang, M. Ren, R. Zemel, Sketchembednet: Learning novel concepts by imitating drawings, in: M. Meila, T. Zhang (Eds.), Proceedings of the 38th International Conference on Machine Learning, Vol. 139 of Proceedings of Machine Learning Research, PMLR, 2021, pp. 10870–10881.
[28] J.L. Ba, J.R. Kiros, G.E. Hinton, Layer normalization, arXiv preprint arXiv:1607.06450 (2016).
[29] H.W. Kuhn, The hungarian method for the assignment problem, Naval research logistics quarterly 2 (1–2) (1955) 83–97.
[30] B.M. Lake, R. Salakhutdinov, J.B. Tenenbaum, The omniglot challenge: a 3-year progress report, Current Opinion in Behavioral Sciences 29 (2019) 97–104.
[31] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: International conference on machine learning, PMLR, 2015, pp. 448–456.
[32] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, The journal of machine learning research 15 (1) (2014) 1929–1958.
[33] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.
[34] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al., Matching networks for one shot learning, in: Advances in neural information processing systems, 2016, pp. 3630–3638.
[35] S. Ravi, H. Larochelle, Optimization as a model for few-shot learning, in: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24–26, 2017, Conference Track Proceedings, 2016.
[36] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F.A. Wichmann, W. Brendel, Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness, in: International Conference on Learning Representations, 2018.

Arturs Polis is a Ph.D. student at the Department of Computer Science of Aalto University. He received his M.Sc. degree in Data Science from the University of Helsinki in 2019. His most recent research interests include unsupervised and weakly-supervised structured and object-based representation learning from static images and videos.

Alexander Ilin is a Professor of Practice in the Department of Computer Science of Aalto University. Prior to that he was a Senior Scientist and Head of ML at the Curious AI company, ML Scientist at Amazon and Chief Scientist at Analyse2. His research interests are in the field of deep representation learning with applications in various domains such as document understanding and industrial process control.