Fast visual grounding in interaction: bringing few-shot learning with neural networks to an interactive robot

José Miguel Cano Santín[1] **Simon Dobnik**[1,2] **Mehdi Ghanimifard**[1,2]
[1]Department of Philosophy, Linguistics and Theory of Science (FLoV)
[2]Centre for Linguistic Theory and Studies in Probability (CLASP)
University of Gothenburg, Sweden
[1]`jmcs990@gmail.com` [2]`{simon.dobnik,mehdi.ghanimifard}@gu.se`

Abstract

The major shortcomings of using neural networks with situated agents are that in incremental interaction very few learning examples are available and that their visual sensory representations are quite different from image caption datasets. In this work we adapt and evaluate a few-shot learning approach, Matching Networks (Vinyals et al., 2016), to conversational strategies of a robot interacting with a human tutor in order to efficiently learn to categorise objects that are presented to it and also investigate to what degree transfer learning from pre-trained models on images from different contexts can improve its performance. We discuss the implications of such learning on the nature of semantic representations the system has learned.

1 Introduction

Robots need to ground real world objects and entities that they see with their cameras to natural language that they hear or generate when interacting with a human tutor. There are four properties that distinguish this kind of machine learning from the approaches that are based on a corpus: (i) agents have to make reliable classifications already after being exposed to a few examples; (ii) they may utilise background knowledge; (iii) learning is not done in large batches of examples but in small increments as the interaction unfolds; and (iv) the mechanisms of interaction are used to control the rate of learning.

Matching Networks (Vinyals et al., 2016) is a neural network algorithm designed for one-shot and few-shot learning of a classifier in a standard dataset. For object classification, their principal advantage is their capability to learn objects from few labelled instances rapidly. This property makes them a possible candidate for modelling meaning representations for robot interactions. The learning algorithm for Matching Networks fits in the supervised learning paradigm. Its central idea is influenced from the meta-learning paradigm with memory-augmentation (Santoro et al., 2016) which has not been evaluated for interactive scenarios. In this work, we implement Matching Networks and evaluate its performance in critical interactive scenarios which includes both offline and online learning with a simulated interactive agent.[1]

The contributions of this paper are as follows: (i) we create our own implementation of the Matching Network model from (Vinyals et al., 2016) and (ii) integrate it with interactive strategies of a situated agent; (iii) we test the performance of such a simulated agent (a) on baseline recognition of objects presented to the agent, (b) where the support set of image categories is from another context, and (c) where new object categories must be learned; (iv) we discuss implications of interactive learning using this model for representations of meaning and semantics of natural language and outline several future experiments in which the model could be exploited and improved.

2 Matching Networks

Network Architecture Matching Networks are composed of four sub-modules: (1) convolutional networks $\texttt{ConvNet}_\theta(\cdot)$ to extract basic visual features from an image, (2) support set embeddings $\texttt{g}_\theta(\cdot)$ to encode visual features of the few-shots of labelled instances in memory, (3) target embeddings $\texttt{f}_\theta(\cdot)$ to encode visual features of a new image, (4) the matching layer $\texttt{att}(\cdot,\cdot)$, which compares the target representation with support embeddings in a memory. The output of the matching layer can be interpreted as attention on categories in the support set.

As shown in Figure 1, the Matching Networks

[1]Additional performance metrics of our implementation on Omniglot (Lake et al., 2015) and ImageNet (Russakovsky et al., 2015) are reported in Appendix.

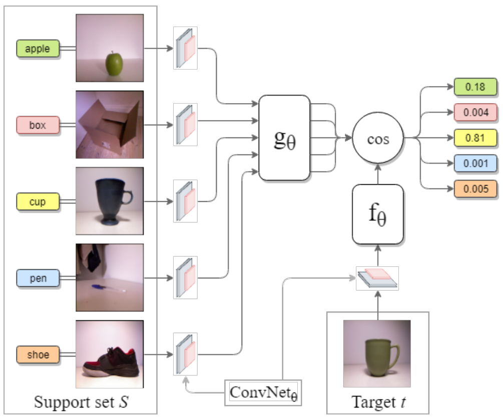

Figure 1: A diagram demonstrating a one-shot learning task with five labelled categories. First, five labelled images of the support set (S) are embedded in memory. Then, an unlabelled target image (t) is also encoded. The matching layer computes the cosine similarity between the embeddings of t and the embeddings of categories in the support set and aggregates attention scores over the categories in the output.

compare an unlabelled target image (t) with the labelled images of a support set (S) to determine how similar t is to different categories in S. After encoding images of the support set S with $\texttt{ConvNet}_\theta$, the support set embedding module (g_θ) is responsible to implicitly learn the interdependency between the visual features and categories by creating an embedding space for images where their similarity must correspond to their shared category. After encoding the target image (t) with $\texttt{ConvNet}_\theta$, the target embedding module ($\texttt{f}_\theta$) is expected to learn the projection of convolutional visual features onto the embedding space of categories constructed by (g_θ). After obtaining the output from the embedding functions, the matching layer computes the similarities between the target embedding and the support set embeddings as if it attends on the categories of similar images in the support set:

$$\hat{y}_t = \sum_{i=1}^{n} \texttt{att}(x_i, t) \cdot y_i$$

where, the support set $S = \{(x_i, y_i)\}_{i=1}^{n}$ consists of n instances of labelled images x_i and their one-shot encoded labels y_i. $\hat{y}_t$ represents the predicted distribution of categories $P(\cdot|t, S)$ of t, conditioned on given S.

Training Algorithm Notably, the labelled images in the support set are the augmented memory of the neural networks. The training objective is to minimise the error in categorisation by learning prototype representations of the images of each category in the support set. In principle, components of the model can be trained in an end-to-end fashion. With categorical cross entropy loss, in each supervised learning step, the predicted log-likelihood of the correct labels is back-propagated to all parameters of the modules:

$$loss(y_t, \hat{y}_t) = -\log(y_t \cdot \hat{y}_t)$$

Implementation We have implemented a variation of Matching Networks in TensorFlow (Abadi et al., 2015) with a small number of parameters. In our implementation, g and f are one-layer feed forward networks with ReLU activation and L2 normalisation, ConvNet is VGG16 (Simonyan and Zisserman, 2014) and the matching layer is the cosine similarity of two vectors with softmax normalisation with no trainable parameters.

$$\begin{aligned}
\texttt{ConvNet}_{\theta_0}(x) &= \mathrm{VGG16}_{\theta_0}(x) \\
\mathrm{g}_{\{\mathrm{w}_1,\mathrm{b}_1\}}(x) &= \frac{\texttt{ReLU}(\mathrm{w}_1 \cdot x + \mathrm{b}_1)}{||\texttt{ReLU}(\mathrm{w}_1 \cdot x + \mathrm{b}_1)||_2} \\
\texttt{f}_{\{\mathrm{w}_2,\mathrm{b}_2\}}(x) &= \frac{\texttt{ReLU}(\mathrm{w}_2 \cdot x + \mathrm{b}_2)}{||\texttt{ReLU}(\mathrm{w}_2 \cdot x + \mathrm{b}_2)||_2} \\
\texttt{att}(x_i, t) &= \frac{e^{(\texttt{f}(t)\cdot \mathrm{g}(x_i))}}{\sum_{j=1}^{n} e^{(\texttt{f}(t)\cdot \mathrm{g}(x_j))}}
\end{aligned}$$

In the normalisation of f and g in att, their dot product is equivalent to their cosine similarity. The code of this implementation is openly available.[2]

Differences with (Vinyals et al., 2016) Unlike (Vinyals et al., 2016), in the implementation of g and f we only process one image at a time instead of using bigger networks to process the support set in one go with BiLSTM (for g) and attLSTM (for f) (Vinyals et al., 2016, p. 3). While the incentive behind using LSTMs (Hochreiter and Schmidhuber, 1997) is their ability to encode dependencies between items, they also encode sequential dependencies which are not beneficial for sets of labelled images in S. This was why we do not use LSTMs, despite its promising results in (Vinyals et al., 2016).

Our goal is to use the model in an interactive scenario where the support set grows incrementally. Since the implemented design requires retraining the parameters, smaller models are preferable. Therefore, we chose the simplest solution

[2]https://github.com/jcanosan/Interactive-robot-with-neural-networks

that is sufficient to give reliable results. We simply concatenate the images in the support set vector to retrain the re-parameterised embeddings with a random order of categories.

Transfer of pre-trained visual features We aim to exploit the background knowledge that is transferred from off-the-shelf pre-trained object recognition models trained on images that humans took with their cameras in a variety of settings. VGG16 consist of stacks of convolutional layers with a small receptive field of 3×3, followed by max-pooling layers (see Simonyan and Zisserman, 2014, p. 2). Instead of training the `ConvNet` layers, we use the output of the pre-trained model on the ImageNet dataset (Russakovsky et al., 2015) after the final convolution layer followed by global average pooling as a `ConvNet` module to extract the visual features. We then investigate domain adaptation measures such as fine-tuning on images and experiment with adding new images to the support set as described in Section 4.

We expect that photographic images taken by humans contain different distributions of objects, backgrounds and attention or focus on objects from those that our agent can capture with its camera. Yosinski et al. (2014) point out that the benefit of using pre-trained networks decreases when the task or the data employed in the pre-training stage is very different from the target task. However, they found that "features transferred from distant tasks are better than random weights" when initialising a task and transferred features help improve performance "even after substantial fine-tuning on a new task, which could be a generally useful technique for improving deep neural network performance" (see Yosinski et al., 2014, p. 8).

3 Interactive grounding of visual objects

Robotic systems have to learn constantly new knowledge about their dynamic environment, for example when they encounter new objects. One of the major differences between learning from datasets and learning interactively is that the system must be able to use the knowledge that it has learned very quickly, after seeing only a couple of examples. Therefore, in this respect few-shot learning is ideally suited for this domain. However, an interacting agent with a human tutor can exploit mechanisms of human interaction (*interaction strategies*) to learn faster, for example by being able to control how information is presented to the human or by requesting new information that it is missing (Skočaj et al., 2010).

Our situated agent setup is based on the KILLE framework (Dobnik and de Graaf, 2017). It consists of a stationary Microsoft's Kinect v1 RGB-D sensor connected to an Ubuntu 16.04 system. The sensor is supported by the *Freenect* drivers[3] integrated in the Robot Operating System (ROS) framework (Quigley et al., 2009). Our system could use any RGB camera supported by ROS. Its intended scenario of usage is grounding of (small) objects on a table top that a human tutor presents to it.

Several interaction strategies can be defined in this domain (Skočaj et al., 2009; Dobnik and de Graaf, 2017). Because in this work we explore the integration of the few-shot learning architecture with a focus on data sparseness and its contextual dependence, we investigate two strategies to learn objects. Firstly, the human tutor can present the object to the agent and describe what it is (e.g. *This is an apple*). The image of the object is saved to the dataset. Then, depending on the number of examples that the system has seen of this object category, it either waits to see more examples (e.g. *Please, show me more examples of apple*), *learns a new category* (e.g. *I am learning apple*) if it has five images of this category, or *updates* its knowledge (e.g. *I am updating my systems on apple*) if it has seen more than five images.

The second interaction strategy allows the robot to understand and respond to questions about objects that are presented to it (e.g. *What is this?*). The robot first attempts to classify the object presented and includes the highest scoring label in its answer and the certainty of the guess based on the score of the label (e.g. *This is an apple*; *I think this is an apple*; *I am not sure. Is this an apple?*; *I don't know what is this. Please, tell me.*). The user can then provide feedback and affirm the guess or tell the system the correct label.

The application of the Matching Networks has two stages: training and prediction. When training, we build the support set S with a few (k) images of each labelled class (n). Then, we build a target set t from the images from S to train our model. Hence, we use S as both the support set and the target images in the training phase. When a user asks the system to identify an object, the same support set for training is used (S) but the

[3]https://openkinect.org/

target t is the new input taken with the camera. The target must belong to one of the categories of S.

The online training of the matching network is performed every time the robot needs to *learn a new category* or *update* its knowledge of one of the existing categories. To *learn a new category*, the system collects five images of this category and adds the images as a new category of objects to the support set S which now has an extra category. To *update* the knowledge of an existing category, the system includes in the support set S the new image taken by the camera and trains the model with the new S. This update of S is performed (i) when a user presents an object to the robot and (ii) when the robot incorporates feedback from the user after incorrectly classifying an object or after clarifying a category of an object that it was not sufficiently confident in.

With few images and categories the training is fast and the learning is reliable for our robot interaction. However, as S grows, so does the training time. As it is expected that the system would need to constantly learn new objects and categories, this issue will have to be addressed in the future. Having to wait for a long time for retraining the model every time we change the size of the support set is not so good from the perspective of user experience and scalability of the system. One strategy we could take is to implement an additional module of a short-term memory for the current objects and wait with the learning updates until when the robot is not interacting with a user and is resting.

In order to measure the success of learning under different conditions we automatised the testing procedure. To test each configuration with identical data, we created a Small Objects daTAset (SOTA), a collection of 400 images with equal distribution of images over 20 categories.[4] Each image depicts a single small everyday object (such as *apple*, *pen* or *shoe*), which was placed in front of the robot's camera while interacting with our system in real time.

4 Matching networks in interaction

4.1 Baseline performance on SOTA

In this first (baseline) experiment we simulate the object recognition with the Matching Networks outside of the interactive scenario on the collected images of the SOTA dataset.

[4]The dataset is available in our GitHub repository.

		1-shot	**5-shot**	**10-shot**
5 labels	Accuracy	66.0%	90.0%	94.0%
	Encoding	1.12s	1.63s	2.15s
	Training	1.43s	3.57s	7.27s
20 labels	Accuracy	41.5%	71.0%	86.5%
	Encoding	1.41s	1.93s	2.39s
	Training	3.26	12.15s	25.99s

Table 1: SOTA evaluation. The table shows the accuracy of the model evaluated on the same dataset and the time taken for encoding images and training.

The Matching Networks are trained as outlined in Section 3: we build a support set (S) with n categories or labels (5 or 20) and k images per label (1, 5 or 10-shot). During training, the images from S are also used as target images, making the number of training instances of $k \times n$. For instance, when training a model with 20 labels (n) and 5-shot (k), there are 100 training instances of target images. During evaluation, the images that we use as targets are the remaining 10 images per each category from SOTA that have not been used to train the model and so they are unknown to the system.

Results in Table 1 show that the model performs much better as more images are added to each label. However, increasing the number of shots also increases the training time. Although 1-shot per category would be ideal in terms of time performance, the results indicate that we need more images per category to achieve good recognition performance. For balancing both training time and performance, five images per category seems to be the optimal setting.

In this experiment we have not applied any selection on the support dataset. That is, we have taken images from SOTA at random to build S with them (the selection is saved so we can reproduce the same S). Different selections of the support set might improve or worsen the performance of the system. The selection of the categories could be derived by another process modelling how humans categorise and discriminate objects in particular contexts.

4.2 Support set from another context

We argued that it is very useful for a robot to be able to use knowledge from another context. The objective of the experiment is to investigate whether Matching Networks can be benefit from transfer learning on images taken in different contexts with different cameras by human observers,

		5 labels	**20 labels**
S = SOTA-ext	**1-shot**	71.1%	39.4%
t = SOTA-ext	**5-shot**	91.1%	60.0%
S = SOTA-ext	**1-shot**	75.6%	53.3%
t = SOTA	**5-shot**	86.7%	73.9%

Table 2: Transferring knowledge from other domains. The table compares the accuracy of the model trained using the SOTA-external dataset as both a support and a target set and the model trained using SOTA-external as a support and the SOTA as a target.

which can be trained offline rather than in an example-by-example basis. We have created a collection of images from online datasets (ImageNet) and web search which we call SOTA-external. These images depict a single object, each pertaining to one of the 20 categories in SOTA. There are 100 images in total, 5 per label. Unfortunately, due to copyright issues, SOTA-external cannot be published.

We have devised two training strategies. In the *first scenario* we build a support set (*S*) with SOTA-external and use *S* as both support and target (*t*) for training. In the *second scenario* the *S* is again the SOTA-external, but the *t* are the images from SOTA. During evaluation of both, the *S* is always SOTA-external, which represents the knowledge that the robot already has; and the *t* are the images from the SOTA dataset, which represent the objects that our robot is trying to recognise. The objective behind the two scenarios is to test if it is sufficient to train our model using data from another context, or is it better to train the system wit target from our context and use external images as support.

As shown in Table 2, the performance on five categories is variable: it is better to use external target with 5-shot learning but SOTA target with 1-shot learning. This suggests that for external targets during training, more shots are required possibly because such targets are more variable. However, with more categories it is clear that using images from our robot context (SOTA) as *t* during training is better than using images from SOTA-external. Despite the fact that there is a visible gap in performance, the system still performs acceptably when using only SOTA-external. Compared to the baseline in Table 1, SOTA-external with SOTA targets still improves the performance. Therefore, transferred learning from another context helps, possibly because external images introduce more variation in visual features which allow better discrimination of objects. Hence, if there is a scenario with a new room in a house (e.g. kitchen), the robot could have pre-learned categories for common objects found there (e.g. plate, cup, etc.). Then, we can improve this knowledge with new images and labels from the robot camera. Collections of images from external resources may also be useful to learn new categories.

4.3 New categories of objects

The objective of this experiment is to test how the system learns new categories that it encounters in its dynamic environment. How many images are needed for the robot to recognise a new category and what are the implications of this for modelling the interaction strategies of the robot. We simulate the learning process of each label from SOTA by building a support set *S* which contains 19 labels with five images each, which represent the categories that the robot already knows, plus the remaining label in SOTA which represents the newly learned category. We incrementally increase the number of images in this category from 1 to 5 which gives us five models: 1-shot to 5-shot. The images in *S* are also used as targets for training. In the evaluation phase we take the same *S* and measure the recognition accuracy of the newly learned category by using the remaining 15 images of the same label in SOTA as target images.

The results are shown in Figure 2. We can see that four to five images are necessary for most of the labels. Some labels are clearly easier to learn than others. For instance, *bottle*, *box*, *clothespin* and *marker* do not reach more than 40% accuracy, while *apple*, *book* and *stuffed toy* do not need more than three images to achieve 80% accuracy. The plot also shows some irregular fluctuations in accuracy when adding new images to some labels. For instance, the accuracy of *boot* achieves 73.3% in 3-shot, but then drops again to 46.7%. It appears that some images confuse the model, for instance images with different objects or depicting an object from an unusual perspective. The irregularities could be due to the object background.

5 Discussion and conclusions

We have explored how a neural network algorithm for one-shot learning, Matching Networks (Vinyals et al., 2016), can be used in an interactive scenario between a robot agent and a human tu-

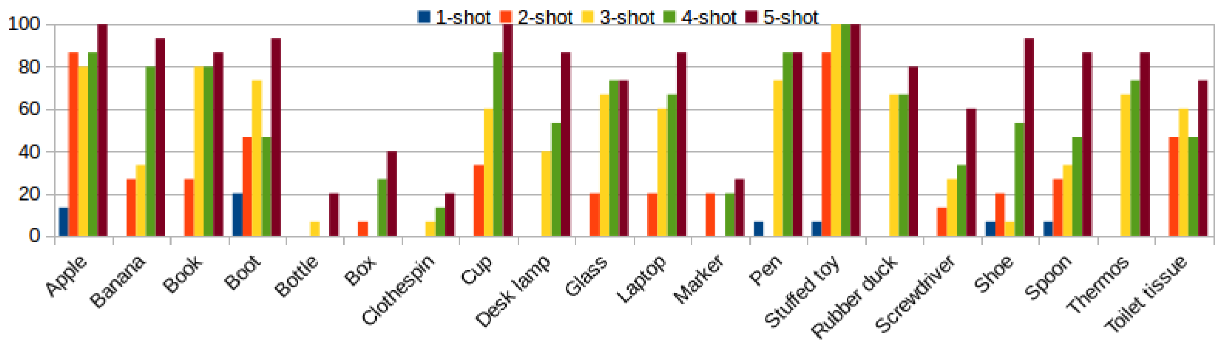

Figure 2: Results on learning new labels. The k-shot learned label is specified under the x axis and each of the bars represent the accuracy of the classification from 1-shot (left) to 5-shot (right).

tor. Our evaluation demonstrates that fast learning with neural networks is possible for this task and it is effective with a few categories.

The Matching Networks do not learn how to directly ground words describing objects in visual features (Roy, 2002; Dobnik, 2009). Instead, it relies on a small support set, contextually introduced objects that it stores, and then learns how to discriminate their categories from one another. This is a particularly appropriate meaning representation for situated agents: (i) the system is able to express gradient beliefs: the outputs of the network are probabilities of the target belonging to each category; (ii) contextual priming information can be integrated by controlling the categories of objects in the support set which demonstrates a potential for modelling top-down attention of an agent (Lavie et al., 2004; Dobnik and Kelleher, 2016); (iii) it models the Saussurean notion of lexical meaning, namely that language is a system of differences without positive terms (Saussure et al., 1983); (iv) as a result the system is very robust and discrimination of categories can already be achieved with a small number of examples.

The system could be extended in several ways, which will be the focus of our future work. First, additional experiments will involve a more distant type of knowledge transfer, to offline pre-train the Matching Networks also on datasets of images with large number of categories and then fine-tune the pre-trained model in the local domain. Another extension is to have a process in place to control the size of the support set over time and transfer the matching knowledge to the memory as required. The update procedure that trains the model from scratch every time affects the user experience negatively, especially when the support set becomes large. Additional procedures could be added to deal with this issue: (1) a training process scheduled when the user is not interacting; (2) a training process of a new model during the interaction when this is still using the older model; (3) implementing methods in the framework of Continual Learning to prevent catastrophic forgetting when updating our models (Greco et al., 2019; Hayes et al., 2019). We will also investigate the influence of using different support sets in terms of variation of objects within a category, variation of objects sampled from different views and selection of object categories as referred to in the current conversation. New interactive strategies with the robot will be investigated and implemented as a way to make the interactive learning of our framework more efficient, for example to *unlearn* efficiently incorrectly learned labels (Skočaj et al., 2009; Dobnik and de Graaf, 2017).

As another direction of future work, we noted that a uniform image background in our experiments negatively influences both performance and scalability. This could be countered by automatic localisation and segmentation of relevant regions of images either with bounding boxes around objects (Girshick et al., 2013; Anderson et al., 2018) or using soft-attention over regions of the image (Xu et al., 2015; Lu et al., 2016) to remove the background of the objects.

Finally, further studies of this framework should go beyond learning of visual grounding of objects. One direction of our future work is to learn grounding of relations including spatial relations with few-shot learning and matching knowledge. Another direction is to consider linguistic and distributional knowledge that could be transferred from cross-modal resources (Lazaridou et al., 2014) as an external matching knowledge.

Acknowledgements

We are grateful to the anonymous reviewers for their helpful comments on our earlier draft. The research of Dobnik and Ghanimifard reported in this paper was supported by a grant from the Swedish Research Council (VR project 2014- 39) for the establishment of the Centre for Linguistic Theory and Studies in Probability (CLASP) at the University of Gothenburg.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6077–6086.

Simon Dobnik. 2009. *Teaching mobile robots to use spatial words*. Ph.D. thesis, University of Oxford: Faculty of Linguistics, Philology and Phonetics and The Queen's College, Oxford, United Kingdom.

Simon Dobnik and Erik de Graaf. 2017. Kille: a framework for situated agents for learning language through interaction. In *Proceedings of the 21st Nordic Conference on Computational Linguistics*, pages 162–171, Gothenburg, Sweden. Association for Computational Linguistics.

Simon Dobnik and John D. Kelleher. 2016. A model for attention-driven judgements in Type Theory with Records. In *JerSem: The 20th Workshop on the Semantics and Pragmatics of Dialogue*, volume 20, pages 25–34, New Brunswick, NJ USA.

Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2013. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv*, arXiv:1910.02509 [cs.LG].

Claudio Greco, Barbara Plank, Raquel Fernández, and Raffaella Bernardi. 2019. Psycholinguistics meets continual learning: Measuring catastrophic forgetting in visual question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3601–3605, Florence, Italy. Association for Computational Linguistics.

Tyler L. Hayes, Kushal Kafle, Robik Shrestha, Manoj Acharya, and Christopher Kanan. 2019. REMIND your neural network to prevent catastrophic forgetting. *arXiv*, arXiv:1910.02509 [cs.LG].

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. 2015. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

Nilli Lavie, Aleksandra Hirst, Jan W de Fockert, and Essi Viding. 2004. Load theory of selective attention and cognitive control. *Journal of Experimental Psychology: General*, 133(3):339–354.

Angeliki Lazaridou, Elia Bruni, and Marco Baroni. 2014. Is this a wampimuk? Cross-modal mapping between distributional semantics and the visual world. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1403–1414, Baltimore, Maryland. Association for Computational Linguistics.

Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. 2016. Knowing when to look: adaptive attention via a visual sentinel for image captioning. *arXiv*, arXiv:1612.01887 [cs.CV].

Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. 2009. ROS: an open-source robot operating system. In *ICRA Workshop on Open Source Software*.

Sachin Ravi and Hugo Larochelle. 2017. 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*.

Deb Roy. 2002. Learning visually-grounded words and syntax for a scene description task. *Computer speech and language*, 16(3):353–385.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-learning with memory-augmented neural networks.

In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1842–1850, New York, New York, USA. PMLR.

Ferdinand de Saussure, Charles Bally, Albert Sechehaye, Albert Riedlinger, and Roy Harris. 1983. *Course in general linguistics*. Duckworth, London.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv*, arXiv:1409.1556 [cs.CV].

Danijel Skočaj, Miroslav Janiček, Matej Kristan, Geert-Jan M. Kruijff, Aleš Leonardis, Pierre Lison, Alen Vrečko, and Michael Zillich. 2010. A basic cognitive system for interactive continuous learning of visual concepts. In *ICRA 2010 workshop ICAIR - Interactive Communication for Autonomous Intelligent Robots*, pages 30–36, Anchorage, AK, USA.

Danijel Skočaj, Matej Kristan, and Aleš Leonardis. 2009. Formalization of different learning strategies in a continuous learning framework. In *Proceedings of the Ninth International Conference on Epigenetic Robotics; Modeling Cognitive Development in Robotic Systems*, pages 153–160. Lund University Cognitive Studies.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. 2016. Matching networks for one shot learning. *arXiv*, arXiv:1606.04080 [cs.LG].

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? *arXiv*, arXiv:1411.1792 [cs.LG].

A Appendix

Matching Networks (ours)			
5 labels	**1-shot**	**5-shot**	**10-shot**
Accuracy	96.0%	99.0%	99.0%
Encoding	0.94s	0.86s	0.98s
Training	1.40s	3.77s	6.76s
20 labels	**1-shot**	**5-shot**	**10-shot**
Accuracy	71.0%	93.0%	98.0%
Encoding	0.83s	0.81s	0.82s
Training	3.25	11.81s	22.59s
Vinyals et al. (2016)			
5 labels	**1-shot**	**5-shot**	**10-shot**
Accuracy	98.1%	98.9%	
20 labels	**1-shot**	**5-shot**	**10-shot**
Accuracy	93.8%	98.7%	

Table 3: Validation of our Matching Networks on the Omniglot dataset in comparison to the figures cited in (Vinyals et al., 2016, p. 5) for their model. Encoding is the number of seconds that took to encode the support set images with VGG16. Training is the number of seconds to train the Matching Networks.

Matching Networks (ours)			
5 labels	**1-shot**	**5-shot**	**10-shot**
Accuracy	75.8%	89.8%	98.8%
Encoding	1.12s	1.63s	2.15s
Training	1.43s	3.57s	7.27s
20 labels	**1-shot**	**5-shot**	**10-shot**
Accuracy	52.5%	74.2%	82.6%
Encoding	1.41s	1.93s	2.39s
Training	3.26	12.15s	25.99s
Vinyals et al. (2016)			
5 labels	**1-shot**	**5-shot**	**10-shot**
Accuracy	46.6%	60%	

Table 4: Validation on miniImageNet. Encoding is the number of seconds that took to encode the support set images with VGG16. Training is the number of seconds to train the Matching Networks. Results are compared to the results cited in (Vinyals et al., 2016, p. 7) for their model.

A.1 System validation

Tables 3 and 4 show the results of validation of our Matching Networks and interactive strategies on the standard datasets and in comparison to the implementation in (Vinyals et al., 2016). To this end, we simulated the learning process as we do in Section 4.1 but with two standard offline datasets.

In Table 3 we use the Omniglot dataset (Lake et al., 2015), which consists of 1623 grey-scale images that represent characters from 50 different alphabets. Each of the categories in this dataset contains 20 images of the same character hand-drawn by different people.

In Table 4 we use miniImageNet, a sub-set of ImageNet, containing 60,000 images distributed equally over 100 categories (600 per category). This makes this dataset more suitable for "rapid prototyping and experimentation" than the full dataset (see Vinyals et al., 2016, p. 6). Since the categories used in (Vinyals et al., 2016) were not released with their dataset, our splits are the ones proposed in (Ravi and Larochelle, 2017). The 100 categories are divided into three splits: 64 for training (*train* split), 16 for validation (*val*) and 20 for testing (*test*).