CS 502 Project Report: Group 23

Sebastián Breguel (366776) Kai Mashimo (375674) Pierre Lardet (376393)

Abstract

The field of Deep Learning has seen tremendous growth in recent years, but many state-of-the-art methods require abundant labelled data to achieve high performance. The field of meta-learning attempts to address this challenge by creating data-efficient models. We implement RelationNet and compare the performance with other meta-learning algorithms in the few-shot learning context. The benchmark includes metric-based and optimisation-based algorithms such as Prototypical Networks (ProtoNet), Matching Networks (MatchingNet), Model-Agnostic Meta-learning (MAML) and Baseline++. These methods were evaluated on two biomedical datasets: Tabula Muris and Swiss-Prot. We find that RelationNet underperforms compared to other models in nearly all settings, and demonstrates a need for further in order to obtain results seen in the original RelationNet paper (Sung et al., 2017) in the biomedical domain.

1. Introduction

1.1. Motivation

In the rapidly evolving field of deep learning, many capable models and architectures have emerged as hallmarks of a new era of technology. However, these methods require vast quantities of data and computational power to train. In many real-world settings, such as in the biomedical domain, large volumes of annotated data are often either too expensive or infeasible to obtain.

Among the many emerging methods addressing these challenges is meta-learning (Hospedales et al., 2021) which aims to develop more data-efficient learning processes, such as those exhibited by humans. It proposes a paradigm shift towards a more natural, example-efficient learning approach, much like how a child can differentiate between animals with few examples. Analogously, meta-learning aims to develop architectures that perform well in a given task without requiring a large dataset of training data specific to the task.

In this project (option 2), the task focuses on few-shot learning classification over two biomedical datasets. Here, we propose the addition of a metric-based method - Relational Networks (Sung et al., 2017). Section 1.2 provides a brief overview of the meta-learning problem, Section 1.3 presents some related work exploring the different methods used in this benchmark. Section 2 discusses the methodology of the implemented algorithms (RelationNet). Section 3 outlines the experimental procedure followed, the datasets and the results obtained followed by a discussion. Section 4 summarises our conclusions. All the code used in this paper is available in a public Github repository. [1]

1.2. Meta-Learning Problem Formulation

In supervised learning/'standard' deep learning, we aim to minimise a loss function $\mathcal{L}$ over a model of parameters θ, using a set of labelled data $\mathcal{D} = \{(x_1, y_1), \dots (x_n, y_n)\}$. The data is split into a train and test set, where the model is trained "from scratch" using the training data and then evaluated on the test set. By contrast, in meta-learning we define a set of tasks τ, assumed to share some underlying structure or similar distributions. This implies that learning how to solve one task should aid in solving the others. Therefore, the purpose of a meta-learning algorithm is to share knowledge learnt across tasks.

The set of tasks is split into meta-training and meta-testing tasks. Each meta-training task is further split into training D^{train} and validation D^{val} sets, which in few-shot learning are termed the *support* and *query* sets respectively. During meta-training, sets of tasks are randomly sampled, as well as random support and query samples for each class. Each random sampling is called an *episode*, and this method of training is called episodic training. Subsequently, meta-testing is conducted on the meta-test tasks to evaluate the model's proficiency in 'learning to learn'.

1.3. Related Work

The traditional taxonomy (Hospedales et al., 2021) of meta-learning specifies three approaches: metric-based, optimization-based, and model-based methods. Metric learning methods aim to learn a feature representation on which classification can be performed based on a metric

[1] https://github.com/kaimashimo/dl4bm-project

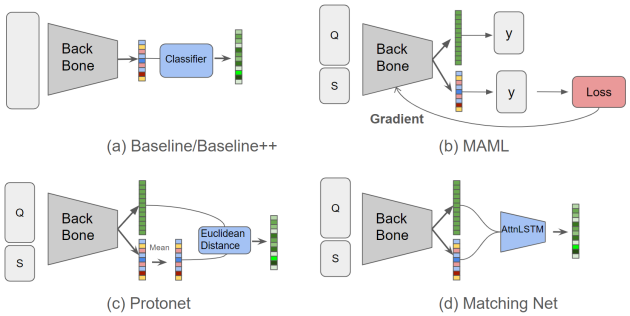

Figure 1. Different meta-learning methods and their architectures

such as Euclidean distance (as in Protonet (Snell et al., 2017)) or a non-parametric method (as in Matching networks (Vinyals et al., 2016)). Optimization-based methods learn shared knowledge across tasks by finding a set of initial parameters of the model which is easily adapted to each new task. The optimization can be performed at different levels: MAML (Finn et al., 2017) is a second-order method that optimizes the initial condition or Reptile (Nichol et al., 2018) an adaptation of MAML in first-order. Model-based methods embed knowledge learned across tasks in a feed-forward NN, which can be a Convolutional, Recurrent, or other. Memory-augmented neural networks (Santoro et al., 2016) incorporate a memory buffer that allows reading and writing information during the learning process.

2. Method

2.1. Few-Shot Baselines

The benchmark contains various pre-implemented meta-learning algorithms for comparison. These are Baseline, Baseline++ (Chen et al., 2019), MatchingNet (Vinyals et al., 2016), ProtoNet (Snell et al., 2017), and MAML (Finn et al., 2017). These algorithms are metric-based and optimisation-based meta-learning methods. The process followed by each of these methods is illustrated in Figure 1.

Additionally, there are four pre-implemented architectures for the backbone to solve the few-shot learning task - fully connected layers, convolutional layers or Resnet (He et al., 2015). Due to the sequential nature of the biomedical datasets used, we opted for the fully connected layers. This backbone can be formalized as a non-linear embedded function f_θ that maps input information into a latent space, defined by the dimensionality of the last hidden layer.

2.2. RelationNet

Relational Networks (Sung et al., 2017), or simply RelationNet, is a metric-learning method which incorporates ideas found in prototypical networks (Snell et al., 2017). ProtoNet learns an embedding function such that classification can be performed with a fixed nearest-neighbour function. RelationNet extends this idea by employing a learnable *relation* module that acts as a non-linear classifier.

To decide which class a query example belongs to, the relation module is fed the embedded features of each class from the support set concatenated with the embedded example from the query set. The module then outputs a relation score for each class, from which a class can be decided by taking the class of the support example(s) that produced the highest relation score. This can be seen in Figure 2.

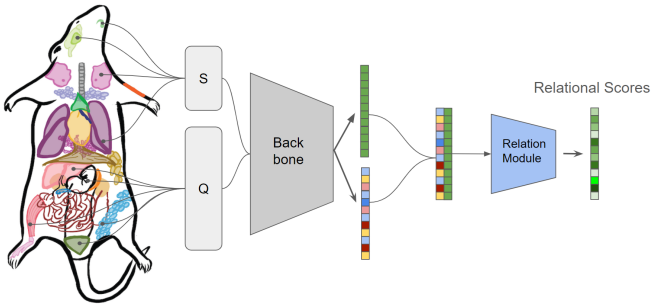

Figure 2. The RelationNet architecture with the backbone and the concatenated feature embeddings for support and query sets

RelationNet can be formalised as having two learnable modules. The first module is the backbone f_θ and the second is the relation module g_ϕ. The backbone maps the support (S) and query (Q) sets into an embedding space before both representations are concatenated and used as input into the relational module. The relational module outputs a relation score (R) for each concatenated input. This can be formalised as seen in Eqn. 1.

$$R = g_\phi(f_\theta(S), f_\theta(Q)) \tag{1}$$

Because the classifier is non-parametric, RelationNet is able to capture non-linear relationships. The initial proposal of RelationNet (Sung et al., 2017) uses convolution layers and two fully connected layers for the relation module, but due the sequential nature of the dataset we adapt the module in this work to use a sequence of linear layers.

In the 1-shot case, the support samples are simply fed through the embedding function to represent the entire class. In the K-shot ($K > 1$) case, similarly to ProtoNet, we take the mean of each embedded feature (element-wise). This is in contrast to the implementation in (Sung et al., 2017) which takes the element-wise sum of each embedded feature to find a prototypical representation of each class. For the 0-shot case, a different embedding function is used on the given semantic class embedding vector. This creates a 'unified and effective approach for both of these two tasks' (Sung et al., 2017), referring to few-shot and 0-shot tasks.

3. Experiments

3.1. Datasets

The few-shot benchmark contains two datasets - **Tabula Muris** (Consortium, 2018) and **SwissProt** (Consortium, 2022). Tabula Muris dataset contains cell-type annotated gene expressions for just over 100K cells, labelled as one of 124 different tissue types. The task is to predict the cell type given the gene expression levels for 2,866 different genes for each cell. The meta-learning task is to generalize across different cell types, given a limited number of examples per cell type. The SwissProt dataset contains embedded amino acid sequences labelled as a part of a gene ontology (GO). The task is to predict the GO label given the embedded amino acid sequence, and the meta-learning task is to generalize across GO labels.

3.2. Procedure

The training procedure for all meta-learning algorithms follows the episodic nature over samples of tasks and classes (Hospedales et al., 2021) seen in all meta-learning algorithms. This differs from the standard training procedure followed by Baseline and Baseline++. Training and testing were performed on a single Nvidia T4 GPU hosted on Google Cloud Platform. The code was written using Python and Pytorch. This Github repo be found here.

The benchmarks consist of the 1, 5, and 10-shot settings over 5 and 20-way classification. For all the experiments, we use Adam (Kingma & Ba, 2014) as our optimizer with a learning rate of 10^{-3}, over 60 epochs for the meta-learning algorithms and 40 epochs for Baseline and Baseline++ models. We use Cross Entropy as our loss function for all models but Matching Networks which uses negative log-likelihood. The epoch with the greatest validation accuracy was selected as the best model, and its performance on the test set is reported in Section 3.3.

We perform hyperparameter tuning on the relation module of RelationNet. While we investigated manually changing the number of epochs and learning rate, we decided to keep these the same as in the other methods as we saw no clear improvement. However, we did conduct hyperparameter tuning on the dropout probability, hidden-layer size, and number of layers for each dataset. We chose not to tune with different backbone architectures in order to maintain a direct comparison with other methods in the benchmark. Additionally, we alter the architecture from containing convolutional layers as in (Sung et al., 2017), to only containing fully connected layers as we are operating on biomedical data, not image data. We then take the best performing model for each dataset and ran it on the benchmark alongside the other methods. Tuning and hyperparameter scripts can be found in our repository.

3.3. Results

We first cover the results of the hyperparameter tuning. We find no clear relationship between test accuracy and the number of hidden layers or hidden layer size. For Tabula Muris, it seems the model performed better in general with fewer and smaller layers - perhaps because Tabula Muris does not require high model complexity for high performance to be attained. This can be seen as a heatmap in Appendix 5.2, Figure 5. However, there was a clearer relationship with dropout rate. Models with lower dropout rate performed better, although this effect was more extreme on the SwissProt dataset as shown in Figure 4. The results of all hyperparameter tuning can be seen in Appendix 5.1, Table 3. We take the best performing models for each dataset: 512 hidden size, with 0.25 dropout and 3 layers for Tabula Muris, and the same architecture with the exception of 4 layers for SwissProt. The high level of agreement between the best model for both datasets may indicate that the tuning was not arbitrary.

METHOD	DATASET	5-WAY	20-WAY
PROTONET	SP	62.69 ± 0.60	$\mathbf{56.01 \pm 0.41}$
MAML	SP	43.50 ± 0.64	—
MATCHINGNET	SP	61.68 ± 0.64	37.29 ± 0.46
BASELINE	SP	$\mathbf{65.37 \pm 0.62}$	—
BASELINE++	SP	59.79 ± 0.63	—
RELATIONNET	SP	43.71 ± 0.65	30.52 ± 0.41
PROTONET	TM	89.55 ± 0.59	71.59 ± 0.37
MAML	TM	86.19 ± 0.66	—
MATCHINGNET	TM	84.36 ± 0.76	60.23 ± 0.41
BASELINE	TM	$\mathbf{89.75 \pm 0.58}$	$\mathbf{74.05 \pm 0.35}$
BASELINE++	TM	81.63 ± 0.74	60.88 ± 0.37
RELATIONNET	TM	82.13 ± 0.90	54.76 ± 0.40

Table 1. Test accuracies/uncertainties across all models on both Tabula Muris (TM) and SwissProt (SP), in both the 5-shot 5-way and 5-shot 20-way settings

We now consider the results across all models. Table 1 shows the performance of all models across both datasets in both the 5-way and 20-way settings (5-shot), and Figure 3 illustrates the trends across increasing n-shot, 5-way settings. In both datasets, we observe unexpected behavior: Baseline outperfoms all other methods in all 5-way settings but 1-shot, in which only ProtoNet performs better. We theorise that these tasks lend themselves well to fine-tuned models, as in transfer learning (Zhuang et al., 2019). The additional structure imposed by meta-learning algorithms does not necessarily help performance in these biomedical settings.

We also find a significant difference in performance between the two datasets. The maximum test accuracy achieved on Tabula Muris was 92% (82.13% for RelationNet), while the maximum on Swissprot was 75.33% (48.02% for RelationNet). Clearly, the SwissProt data provides a harder task for all models than Tabula Muris.

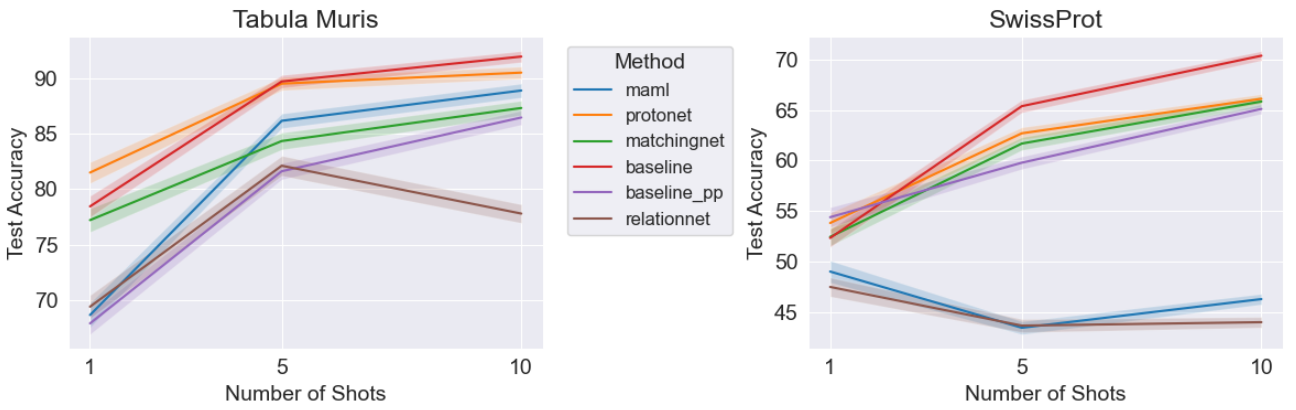

Figure 3. Test accuracies/uncertainties across all methods over the two datasets in 1,5,10-shot and 5-way settings

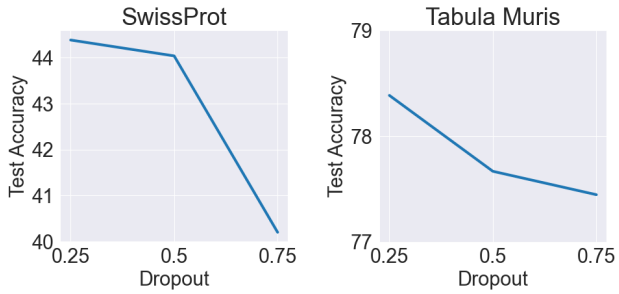

Figure 4. The test accuracies for varying dropout rates in the relation module of RelationNet for both Tabula Muris and SwissProt

RelationNet falls behind all other models in almost all settings. Furthermore, its trajectory across increasing n-shot settings is unstable as seen in Figure 3. With the exception of RelationNet and MAML in the 5-shot setting on SwissProt, all models increase in performance with more data, as expected. This is perhaps due to classification of SwissProt data being a harder task, and therefore RelationNet's additional model complexity makes it even harder for it to learn useful information, especially on a harder task.

We performed an ablation study by comparing the results obtained from RelationNet with ProtoNet. This can be done since both share the same structure other than RelationNet's relational module. ProtoNet performed better than RelationNet in all settings and hence we conclude that the relational module is not helpful to model performance in the settings of the experiments we carried out. This result is in contrast to many of the results seen in the original paper (Sung et al., 2017), which finds improvement over ProtoNet. This difference might be attributed to the difference in domain of the datasets (image vs biomedical), which is further supported by the results in (Cao et al., 2020). Here, the authors find that RelationNet does not perform as well as ProtoNet on biomedical datasets, although the difference is smaller than the results we obtain here.

The results on MAML differed drastically between the two datasets. In Swissprot, we observe unstable results across increasing n-shot similar to RelationNet. This could be attributed to the relative complexity of the Swissprot task, resulting in gradients across tasks bbeing in opposition, making the model's initial conditions deviate from the optimal solution. By contrast, MAML performs well in the 5 and 10-shot settings on Tabula Muris.

4. Conclusion

We observe an overall decrease in performance by RelationNet compared to the other algorithms in this benchmark in nearly all settings. Notably, it performs worse than ProtoNet, its predecessor model. It is also unstable with increasing n-shot, indicating that the additional model complexity hinders its ability to learn. This is in disagreement with results found in (Sung et al., 2017), which was only tested on image classification, but in agreement with (Cao et al., 2020).

It should be noted that RelationNet is also capable in the 0-shot setting, but the datasets in this benchmark do not allow for 0-shot classification as this requires semantic class embeddings.

Avenues of further work include more extensive hyperparameter optimisation, with other backbones being considered, or extending the benchmark to include other biomedical datasets, or datasets from domains other than biomedicine or image classification.

In this paper we have shown that in the context of two biomedical datasets, RelationNet's performance is poor and that simpler, more traditional methods, such as those exhibited by the Baseline model, are among the most effective. Clearly, more development is needed in the meta-learning field for effective architectures to be found which are consistently more data-efficient than traditional methods across different domains.

References

Cao, K., Brbic, M., and Leskovec, J. Concept learners for generalizable few-shot learning. *CoRR*, abs/2007.07375, 2020. URL https://arxiv.org/abs/2007.07375.

Chen, W., Liu, Y., Kira, Z., Wang, Y. F., and Huang, J. A closer look at few-shot classification. *CoRR*, abs/1904.04232, 2019. URL http://arxiv.org/abs/1904.04232.

Consortium, T. T. M. Single-cell transcriptomics of 20 mouse organs creates a tabula muris. *Nature*, 562:367–372, 2018. doi: 10.1038/s41586-018-0590-4. URL https://doi.org/10.1038/s41586-018-0590-4.

Consortium, T. U. UniProt: the Universal Protein Knowledgebase in 2023. *Nucleic Acids Research*, 51(D1): D523–D531, 11 2022. ISSN 0305-1048. doi: 10.1093/nar/gkac1052. URL https://doi.org/10.1093/nar/gkac1052.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. *CoRR*, abs/1703.03400, 2017. URL http://arxiv.org/abs/1703.03400.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition, 2015.

Hospedales, T., Antoniou, A., Micaelli, P., and Storkey, A. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):5149–5169, 2021.

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

Nichol, A., Achiam, J., and Schulman, J. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T. Meta-learning with memory-augmented neural networks. pp. 1842–1850, 2016.

Snell, J., Swersky, K., and Zemel, R. S. Prototypical networks for few-shot learning. *CoRR*, abs/1703.05175, 2017. URL http://arxiv.org/abs/1703.05175.

Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H. S., and Hospedales, T. M. Learning to compare: Relation network for few-shot learning. *CoRR*, abs/1711.06025, 2017. URL http://arxiv.org/abs/1711.06025.

Vinyals, O., Blundell, C., Lillicrap, T. P., Kavukcuoglu, K., and Wierstra, D. Matching networks for one shot learning. *CoRR*, abs/1606.04080, 2016. URL http://arxiv.org/abs/1606.04080.

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., and He, Q. A comprehensive survey on transfer learning. *CoRR*, abs/1911.02685, 2019. URL http://arxiv.org/abs/1911.02685.

5. Appendix

5.1. Hyperparameter Tuning: Tabula Muris

Hidden Size	Dropout	Number of Layers		
		2	3	4
256	0.25	79.18	80.12	78.33
	0.5	78.20	77.93	76.83
	0.75	75.30	72.37	73.39
512	0.25	79.72	**82.13**	76.79
	0.5	79.51	77.86	78.08
	0.75	78.8	78.42	78.34
1024	0.25	80.34	72.64	76.15
	0.5	76.81	75.99	77.47
	0.75	77.88	75.94	71.41

Table 2. Test Accuracies for the hyperparameter grid search of the relation module of RelationNet Over. We search the hidden size, dropout rate and number of layers

5.2. Hyperparameter Tuning: SwissProt

Hidden Size	Dropout	Number of Layers		
		2	3	4
256	0.25	42.12	45.17	43.84
	0.5	40.7	44.1	41.06
	0.75	40.03	37.09	37.89
512	0.25	40.02	47.44	**48.24**
	0.5	41.08	46.03	47.35
	0.75	41.84	39.22	41.5
1024	0.25	48.024	24.13	42.44
	0.5	46.87	43.62	44.82
	0.75	42.84	41.91	

Table 3. Test Accuracies for the hyperparameter grid search of the relation module of RelationNet Over. We search the hidden size, dropout rate and number of layers

5.3. Hyperparameter Tuning: Heatmap

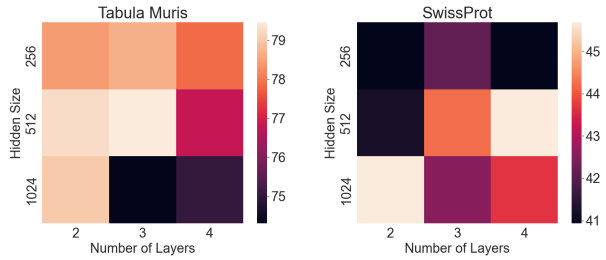

Figure 5. Heatmaps for both Tabula Muris and Swissprot showing the test accuracy over varying hidden sizes and number of layers