CS 502 Project report — COMETT: COMET with a Transformer backbone

Hamza Laarous Salim Cherkaoui Tran Minh Son Le (Group 16)

Abstract

In this project, we attempt to improve the COMET framework from "Concept Learners for Few-Shot Learning" (Cao et al., 2021) by integrating a transformer model in place of the original independent concept learners. Our goal is to benchmark the modified COMET model against both the original framework and other baseline models. We focus on two primary datasets for this endeavor: Tabula Muris, a comprehensive collection of single-cell transcriptome data, and SwissProt, a well-curated protein sequence database. The outcomes of our study, including results and implementation details, can be accessed at our project repository: https://github.com/Hamzezi/DLBioProject/.

1. Introduction

Few-shot learning is a pivotal framework in machine learning, aiming to train models to understand data representations from minimal data. This approach is especially relevant in real-world scenarios where data sparsity is common, and many classes may have few or no observations. The COMET framework, designed with this consideration, features a unique capability to attend to multiple concepts simultaneously, providing it with human-like concept detection abilities.

The primary goal of this architecture is not only to detect these concepts but also to extract meaningful representations from them, thereby simplifying downstream tasks. This approach offers the added advantage of interpretability, as the concepts learned are understandable by humans. In the original paper (Cao et al., 2021), COMET's ability to discern various concepts, such as beaks, wings, and feet from the CUB dataset—an image dataset annotated with 200 bird species—was demonstrated.

However, a potential concern arises with the independent learning of concepts, which may limit the model's ability to contextualize and integrate these concepts effectively. To address these limitations, our project proposes the integration of a dedicated transformer model for concept learning. Transformer models, which have proven their ability in contextual understanding, are expected to enhance the synergy and contextual awareness among the learned concepts.

For our evaluation, the primary testing ground will be the Tabula Muris dataset, utilized in the original COMET study. This will allow for a direct comparison with the existing framework. We will also extend our testing to the SwissProt dataset, broadening our scope to assess our model against various baseline models in different biomedicine contexts. This dual-dataset approach will enable us to comprehensively evaluate the efficacy of our modifications across familiar and novel few-shot learning scenarios.

Related work. Meta-learning and few-shot learning are fields with extensive research. There are several directions of FSL research (Finn et al., 2017; Chen et al., 2019), one of which is metric-based learning. In (Vinyals et al., 2016), a set-to-set attention mechanism is introduced. Meanwhile, (Snell et al., 2017) learn class prototypes via ProtoNets. (Ye et al., 2020) improve upon the idea by introducing learnable set-to-set functions to enhance the discriminative ability of prototypes for target tasks. One such function could be transformers (Vaswani et al., 2017). On the other hand, (Cao et al., 2021) propose the COMET method to modify the ProtoNet by using multiple concepts (and hence metric spaces) per input. The advantage of this approach is the inherent interpretability thanks to the pre-defined human-intepretable concepts (e.g., anatomical parts of animals). The idea of concept learning is not new in deep learning. For instance, the field of computer vision also makes use of concepts for, e.g., object detection (Carion et al., 2020). Inspired by this idea, we modify COMET's backbone with a transformer encoder.

2. Method: COMET with a Transformer backbone

2.1. COMET

In this section, we explain COMET (Cao et al., 2021). In COMET, the input vector is divided into concepts (subspaces) according to pre-defined masks (or random masks). Each concept is then transformed independently by a corresponding concept learner. Each concept learner can have their own weights, or the weights can be shared. Formally:

Algorithm 1 Transformer backbone

1: **Input:** input $x \in \mathcal{S}_k$ $(x \in \mathbb{R}^D)$, number of concepts N, number of decoder inputs S
2: **Trainable:** encoder f_{enc}, decoder f_{dec}, decoder inputs $y^{(i)} \in \mathbb{R}^{D/N},\ i = 1 \dots S$
3: Masks $c^{(j)} := \vec{0} \in \{0,1\}^D,\ j = 1 \dots N$
4: $c^{(j)}\left[(j-1)\frac{D}{N} : j\frac{D}{N}\right] = 1$ {zero-indexing}
5: Concepts $\{x^{(j)}\}_{j=1}^N \in \mathbb{R}^{D/N} := x\left[c^{(j)}\right]$ {boolean-indexing}
6: **if** Decoder-only **then**
7: Encoder outputs $\{\hat{x}^{(j)}\}_{j=1}^N := \{x^{(j)}\}_{j=1}^N$
8: **else**
9: Encoder outputs $\{\hat{x}^{(j)}\}_{j=1}^N := f_{\text{enc}}(\{x^{(j)}\}_{j=1}^N)$
10: **end if**
11: **if** Encoder-only **then**
12: **return** $\{\hat{x}^{(j)}\}_{j=1}^N \in \mathbb{R}^{D/N}$
13: **end if**
14: Dec. outputs $\{\hat{y}^{(i)}\}_{i=1}^S := f_{\text{dec}}(\{y^{(i)}\}_{i=1}^S,\ \{\hat{x}^{(j)}\}_{j=1}^N)$
15: **return** $\{\hat{y}^{(i)}\}_{i=1}^S \in \mathbb{R}^{D/N}$

$$x_i, y_i \in \mathcal{S}_k,\ x_i \in \mathbb{R}^D, \qquad i = 1 \dots |\mathcal{S}_k| \quad (1)$$

$$x_i^{(j)} := x_i \circ c^{(j)},\ x_i^{(j)} \in \mathbb{R}^D, \qquad j = 1 \dots N \quad (2)$$

$$\hat{x}_i^{(j)} := f^{(j)}(x_i^{(j)}),\ x_i^{(j)} \in \mathbb{R}^M, \qquad j = 1 \dots N \quad (3)$$

$$p_k^{(j)} = \frac{1}{|\mathcal{S}_k|} \sum_{x_i, y_i \in \mathcal{S}_k} \hat{x}_i^{(j)}, \qquad j = 1 \dots N \quad (4)$$

where $\mathcal{S}_k$ is the support set of class k (each element of which is indexed by i), $\{c^{(j)}\}_{j=1}^N \in \{0,1\}^D$ are one-hot vectors indicating dimensions of x relevant to the concept j (Eq. 2 is a Hadamard product), $f^{(j)}$ is the concept learner of concept j (a neural net backbone), and $p_k^{(j)}$ is the prototype of class k corresponding to concept j. Thus, each class has N concept prototypes. Then, given a query input x_q, one can compute its class probability $p(y_q = k|x_q)$ as:

$$p(y_q = k|x_q) = \frac{\exp(-\sum_j d(\hat{x}_q^{(j)}, p_k^{(j)}))}{\sum_{k'} \exp(-\sum_j d(\hat{x}_q^{(j)}, p_{k'}^{(j)}))}, \quad (5)$$

where d is the Euclidean distance. The loss per query point is then $-\log p(y_q = k|x_q)$. However, Eq. 3 shows there is no interaction between the concepts, and they are not aware of each other, as the $f^{(j)}$ concept learners operate independently. Thus, we propose to replace the concept learners with a single transformer backbone (Vaswani et al., 2017). Thanks to the self-attention mechanism in transformers, each concept can attend to each other, contextualizing the resulting embeddings.

Implementation. In COMET (Cao et al., 2021), the trainable weights of the concept learners $f^{(j)}$ can be independent or shared. In our implementation, we opt to only implement the shared weights version by using a 1x1 1d convolution layer, where the sliding dimension is the concept dimension.

2.2. Transformer backbone

Transformer encoder or decoder. Transformer encoders have been widely used for discriminative NLP tasks (Devlin et al., 2019). Meanwhile, transformer decoders excel in generation tasks (Brown et al., 2020). In this project, we experiment with transformer encoders, decoders, and encoder-decoder models, where the design for each module is directly adapted from (Vaswani et al., 2017). More specifically, by "encoder" we mean a transformer with only self-attention, and "decoder" means a transformer with both self- and cross-attentions. The encoder inputs are the concepts (subspaces of the original input vector); the decoder inputs are learnable concept vectors for self-attention and encoder outputs or encoder inputs (if the model is decoder-only) for cross-attention.

Positional encoding and attention masking. Since the attention mechanism is a set-to-set mapping, positional encodings must be supplied to make it sequence-to-sequence. In our setting, since the concepts are subspaces (subsets), we do not use positional encodings. Similarly, masked language modeling (Devlin et al., 2019) or causal language modeling (Brown et al., 2020) are employed for self-supervised language modeling. In our setting, since we only replace the COMET's original backbone (Cao et al., 2021) and do not modify the learning objective, we do not use attention masking. Hence, during training, all concepts are allowed to attend to each other.

Defining the concepts. In COMET (Cao et al., 2021), concepts can be pre-defined or randomized. In our implementation, if the concepts (masks) are not pre-defined, we propose to split the input into equal-size subspaces. For instance, if the input dimension is 1000 and we want to use 20 concepts, each concept would have a dimension of 50. See lines 3 and 4 in Algorithm 1. Furthermore, COMET concept vectors are created by multiplying the input vector with a binary concept mask vector indicating which dimensions belong to the concept. This means that each concept has the same number of dimensions as the input, and dimensions not belonging to the concept are zeroed out. Thus, to reduce the input dimension D and the number of trainable parameters, we propose to instead use the masks to index into the input vector, so that dimensions irrelevant to a concept are simply thrown away.

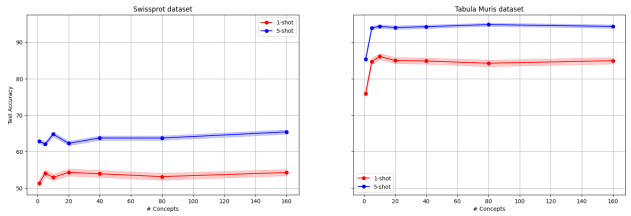

Figure 1. The effect of number of concepts on our transformer backbone test accuracy (test split).

Implementation. Algorithm 1 shows the pseudocode of our proposed transformer backbone, which replaces Eq. 2 and Eq. 3 of the original COMET (Cao et al., 2021). To be more specific, we replace all the concept learners $f^{(j)},\ j = 1 \dots N$ with a single transformer function f. We refer to our method as Transformer. Note that if the input dimension D is not divisible by the number of concepts N, the remainder input dimensions would be thrown away. The design of the transformer decoder f_{dec} with the learnable inputs $y^{(i)}$ is inspired by (Carion et al., 2020), where these learnable decoder inputs can be thought of as "object queries" that cross-attend to the encoder outputs $\hat{x}^{(j)}$.

3. Experiments

3.1. Experimental setup

Datasets. We apply the original COMET (Cao et al., 2021) (referred to as COMET) and a COMET with our transformer backbone (simply referred to as transformer) to two datasets. First, Tabula Muris is utilized for a cell type annotation task across various tissues. Second, SwissProt is used for gene function prediction from sequence information. For both datasets, as explained before, our concepts are either pre-defined, or defined by splitting the input into equal-size subspaces as in lines 3 and 4 of Algorithm 1.

Baselines. We compare the original COMET's and our transformer backbone's performance to five baselines: FineTune/Baseline++ (Chen et al., 2019), Matching Networks (MatchingNet) (Vinyals et al., 2016), Model Agnostic Meta-Learning (MAML) (Finn et al., 2017), and Prototypical Networks (ProtoNet) (Snell et al., 2017). We do not tune the hyperparameters of these methods. Moreover, for COMET, we only tune our transformer implementation and use the default model sizes for the original COMET backbone (which is simply a shared feedforward network for all concepts).

3.2. Ablation & hyperparameter study

Hyperparameters for the transformer backbone. After tuning hyperparameters by using the validation split of both datasets, we find for Tabula Muris (TM) that the optimal configuration for the 1-shot setting involved a transformer encoder with a dimensionality of 64 in its feedforward module, maintaining $N = 10$ concepts, and a dropout rate of 0.2. For the 5-shot TM setting, increasing the feedforward dimension to 128 and the dropout rate to 0.3 yields better performance.

In the case of SwissProt (SW), both the one-shot and five-shot models achieve their best results with a transformer encoder having a feedforward hidden size of 128 and a dropout rate of 0.3.

Additionally, for both datasets, adjusting the learning rate to 0.005, higher than the initially set 0.001, provided the best results.

Ablation: transformer encoder/decoder, number of trainable parameters, and using pre-defined masks. First, we briefly experiment with the transformer type (encoder-only, decoder-only, or encoder-decoder, see Algorithm 1). We found that whenever a decoder is involved (so the decoder inputs $y^{(i)}$ are learned from scratch), the performance is subpar compared to the encoder-only setting, where inputs are available from the dataset. Hence, in this report, we only consider results using an encoder-only transformer. On the other hand, the number of trainable parameters can be controlled by the feedforward hidden size, the number of transformer layers, and the number of concepts N (since we are indexing into the original input x, see line 5 in Algorithm 1). We perform a separate ablation for the number of concepts in the next section. We find that increasing the number of trainable weights tend to hurt the performance. Hence, for the rest of the report, we only use 1 transformer layer, with the feedforward hidden size as described above. Furthermore, with the transformer backbone, we do not find a difference when using pre-defined masks or the ad-hoc masks as in Algorithm 1, suggesting the model's relative robustness against concept definitions.

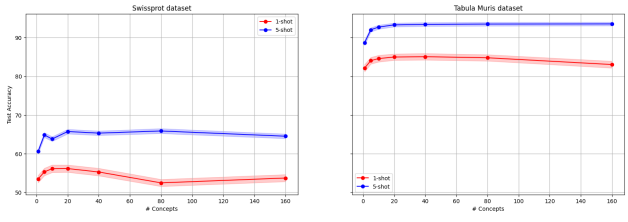

Figure 2. The effect of number of concepts on COMET's accuracy (test split).

Ablation: number of concepts. We systematically evaluate the effect of the number of concepts on COMET's and our transformer implementation's performance on Swissprot

and Tabula Muris datasets (see Figure 2 and Figure 1). In this ablation study, we ignore pre-defined masks (if there are) and generate the masks according to lines 3 and 4 of Algorithm 1, Here, we have a total control over the number of concept we can use. In particular, for COMET, we start from ProtoNet's result that equate to using a single concept in COMET that covers all dimensions of the input. We then gradually increase number of concepts and train and evaluate COMET with the selected number of concepts. For our transformer implementation, we retain default hyperparameters from the codebase but adjusted the feedforward dimension to 128. The results demonstrate that on both domains COMET and the transformer consistently improves performance when increasing the number of concepts. It is worth to note that we observe a drop of performance for a certain amount of concept on COMET on Swissprot. This could be explain by how we define our masking method in algorithm 1 (line 3 and 4), which could be sub-optimal.

	Tabula Muris	
Method	**1-shot**	**5-shot**
Standard NN	$77.0\% \pm 1.0\%$	$88.9\% \pm 0.6\%$
Baseline-Finetune	$71.9\% \pm 1.1\%$	$86.1\% \pm 0.7\%$
MAML	$71.8\% \pm 1.1\%$	$86.6\% \pm 0.7\%$
MatchingNet	$77.2\% \pm 1.0\%$	$85.1\% \pm 0.8\%$
ProtoNet	$81.2\% \pm 0.9\%$	$88.9\% \pm 0.6\%$
Comet	$82.6\% \pm 0.9\%$	$93.2\% \pm 0.5\%$
Transformer	$\mathbf{86.5\% \pm 0.8\%}$	$\mathbf{93.5\% \pm 0.5\%}$

Table 1. Results on 1-shot and 5-shot classification on the Tabula Muris dataset. We report average accuracy and standard deviation over the test dataset.

3.3. Results

Performance comparison. In our study, we evaluate the performance of the original COMET model and our modified version with a transformer backbone on both datasets, Tabula Muris (Figure 1) and Swissprot (Figure 2). We conduct experiments in both 1-shot and 5-shot settings. The results (all evaluated on the test split of each dataset) are noteworthy, particularly when compared with the findings from the original COMET paper.

COMET's performance. On the Tabula Muris dataset, COMET's performance in the 5-shot test is 4.3% better than ProtoNet, and in the 1-shot test, it is 1.4% better. For the Swissprot dataset, COMET shows a 2.6% improvement over ProtoNet in the 5-shot setting and a 7.9% improvement in the 1-shot setting.

Transformer's performance. In the Tabula Muris dataset, the transformer backbone outperforms ProtoNet by 4.6% in the 5-shot test and 5.3% in the 1-shot test. On the Swissprot dataset, the transformer model shows a 1.4% improvement over ProtoNet in the 5-shot test and a 5.8% improvement in the 1-shot test.

COMET and Transformer performances comparison. Comparing COMET with our transformer model on the Tabula Muris dataset, the transformer shows a 0.3% improvement in the 5-shot test and a significant 3.0% improvement in the 1-shot test. On the Swissprot dataset, the transformer model is 1.2% less effective than COMET in the 5-shot test and 2.1% worse in the 1-shot test, yet both remain competitive compared to other baselines. These results indicate that both COMET and the transformer model significantly improve upon the ProtoNet baseline, with the transformer showing particularly promising results in the 1-shot learning tasks. The improvement of the transformer model over COMET, especially in the 1-shot scenarios, suggests that the transformer architecture may offer unique advantages in modeling concept-based dependencies in few-shot learning tasks for biomedical datasets.

	Swissprot	
Method	**1 shot**	**5 shot**
Standard NN	$\mathbf{55.4\% \pm 0.9\%}$	$64.9\% \pm 0.6\%$
Baseline-Finetune	$53.7\% \pm 0.9\%$	$61.1\% \pm 0.6\%$
MAML	$49.8\% \pm 1.0\%$	$42.1\% \pm 0.7\%$
MatchingNet	$53.4\% \pm 1.0\%$	$61.8\% \pm 0.6\%$
ProtoNet	$47.4\% \pm 0.9\%$	$62.5\% \pm 0.6\%$
Comet	$55.3\% \pm 1.0\%$	$\mathbf{65.1\% \pm 0.6\%}$
Transformer	$53.2\% \pm 0.9\%$	$63.9\% \pm 0.6\%$

Table 2. Results on 1-shot and 5-shot classification on the Swissprot dataset. We report average accuracy and standard deviation over the test dataset.

4. Conclusion

In this project, we introduced an re-implementation of the COMET framework, integrating a transformer model to replace the original independent concept learners. Our approach aimed at adressing the lack of contextual understanding between the multiple concepts in COMET. Our experiments, conducted on two distinct datasets - Tabula Muris and SwissProt - demonstrated that our transformer is competitive with the original framework and other baseline models in various scenarios. Notably, in one-shot learning tasks on the Tabula Muris dataset, our model showed a substantial improvement over the original COMET. While our results are promising, there are several avenues for future exploration. The potential of our approach in other domains of few-shot learning, particularly those involving more complex or less structured data, remains to be explored.

References

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

Cao, K., Brbić, M., and Leskovec, J. Concept learners for few-shot learning. In *International Conference on Learning Representations (ICLR)*, 2021.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. End-to-end object detection with transformers. In *European conference on computer vision (ECCV)*, 2020.

Chen, W.-Y., Liu, Y.-C., Kira, Z., Wang, Y.-C. F., and Huang, J.-B. A closer look at few-shot classification. In *International Conference on Learning Representations (ICLR)*, 2019.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning (ICML)*, 2017.

Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. *Advances in neural information processing systems (NeurIPS)*, 2017.

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

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. Matching networks for one shot learning. *Advances in neural information processing systems (NeurIPS)*, 29, 2016.

Ye, H.-J., Hu, H., Zhan, D.-C., and Sha, F. Few-shot learning via embedding adaptation with set-to-set functions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 2020.