2020 25th International Conference on Pattern Recognition (ICPR)
Milan, Italy, Jan 10-15, 2021

Equation Attention Relationship Network (EARN) : A Geometric Deep Metric Framework for Learning Similar Math Expression Embedding

Saleem Ahmed, Kenny Davila, Srirangaraj Setlur, Venu Govindaraju
CSE Department
University at Buffalo
NY, USA
{sahmed9,kennydav,setlur,govind}@buffalo.edu

***Abstract*—Representational Learning in the form of high dimensional embeddings have been used for multiple pattern recognition applications. There has been a significant interest in building embedding based systems for learning representations in the mathematical domain. At the same time, retrieval of structured information such as mathematical expressions is an important need for modern IR systems. In this work, our motivation is to introduce a robust framework for learning representations for similarity based retrieval of mathematical expressions. Given a query by example, the embedding can find the closest matching expression as a function of euclidean distance between them. We leverage recent advancements in image-based and graph-based deep learning algorithms to learn our similarity embeddings. We do this first, by using uni-modal encoders in graph space and image space and then, a multi-modal combination of the same. To overcome the lack of training data, we force the networks to learn a deep metric using triplets generated with a heuristic scoring function. We also adopt a custom strategy for mining hard samples to train our neural networks. Our system produces rankings similar to those generated by the original scoring function, but using only a fraction of the time. Our results establish the viability of using such a multi-modal embedding for this task.**

***Index Terms*—Mathematical Information retrieval; Semi-supervised learning; Graph matching**

I. Introduction

Representational Learning in the form of high dimensional embeddings on a manifold have transformed the landscape of artificial intelligence [1] research. In recent trends, models based on such learnt vector space representations have shown tremendous promise in applications not limited to object recognition, speech and signal processing, natural language processing, multi-modal learning, transfer learning and domain adaptation. In particular, there has been a significant interest in trying to build embedding based systems for the mathematical domain. For example, embeddings have been used to prove mathematical conjectures through a few steps of machine learning [2] and models have been built around embeddings for summarizing mathematical problems [3]. The novelty in such models is the utilization of math expression embeddings.

The code can be found here : https://cse-ai-lab.github.io/MathIR/EARN/

Retrieval of structured information such as mathematical expressions is an important need for modern IR systems. Creating a math aware search engine, which can search and retrieve relevant documents purely based on similar mathematical expressions effectively in real time is still an open challenge. Our core motivation is to introduce a robust framework for representing the domain of symbolic mathematical expressions as a vector space embedding. The objective is that we should be able to use the distance between vectors in this learnt space as a similarity measure between expressions.

We leverage recent advancements in image based and graph based deep learning algorithms to learn our similarity embeddings. We do this first, by using uni-modal encoders in graph space and image space and then, a multi-modal combination of the same. To overcome the lack of training data, we force the networks to learn a deep metric using triplets generated with a computationally expensive scoring function. In the worst case, this heuristic scoring function can take up to full minutes to calculate the similarity between two expressions [4]. In order to improve the training of our networks, we adopt a custom strategy for mining hard samples, both online and offline. Our system based on embeddings can produce rankings similar to those generated by the original scoring function, but using only a fraction of the time. Our results establish the viability of using such a multi-modal embedding for this task.

II. Background

Our work belongs to the field of mathematical information retrieval (MIR), and our proposed methodology uses deep metric learning based on triplets and neural message passing networks to generate embeddings. In this section, we briefly introduce these topics.

A. Math Information Retrieval

Developing a 'math-aware' search engine has been a long standing open problem [5]. The paradigm itself stems from a fundamental symbolic grounding problem: how do we associate logic and semantic meanings to a particular set of symbols? With recent advances in deep learning and hardware capacity, one could claim this problem has been solved to

978-1-7281-8808-9/20/$31.00 ©2020 IEEE

2020 25th International Conference on Pattern Recognition (ICPR) | 978-1-7281-8808-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICPR48806.2021.9412619

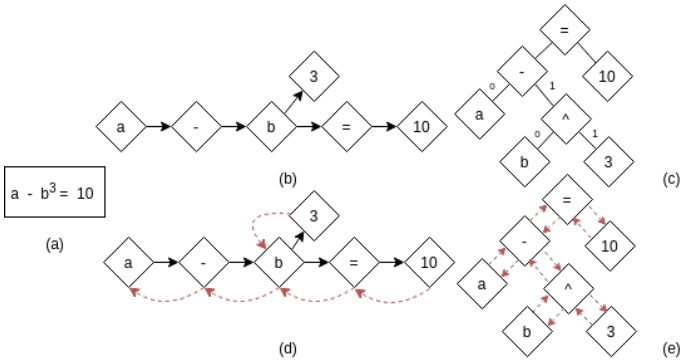

Fig. 1: Mathematical Expression Representations. The expression (a) $a - b^3 = 10$ with associated (b) Symbol Layout Tree (SLT), (c) Operator Tree (OPT), (d) Symbol Layout Graph (SLG), and (e) Operator Graph (OPG). SLTs/SLGs represent formula appearance by the spatial position of symbols on writing lines. OPTs/OPGs represent mathematical operations in expressions.

a certain extent for natural language [6]. However, math expressions are more complex. The symbols themselves derive meaning from the context of the expression that they are a part of, which in turn derives meaning from the domain they are associated with. It is hard even for humans to agree on, what constitutes 'similar' math expressions. For example, for the expressions, $\frac{x}{2}, 0.5x, x^2$, and $2x$, one could say the first two are the most similar because they are equivalent, but then the first, third and fourth have the most similar symbols. Also, one could argue that the last three have the most similar operation.

Zanibbi and Blostein [5] analyzed the two major representations used by the MIR community. The Symbol Layout Tree (SLT) and the Operator Tree (OPT) as depicted in Figure 1(b) and 1(c). The SLT is based on modeling the relative visual position between symbols leading to deep trees with few branches. It starts with the left-most symbol on the main baseline of the mathematical expression, and it connects every other symbol using edges labeled as: Next, Above, Pre-above, Below, Pre-Below, Over, Under, Within, and Element [4]. The OPT is built around the hierarchy of operations in a formula resulting in shallow trees with many branches. Edge labels in OPTs indicate descendant position. OPT does have operator information that the SLT lacks. For our purpose, we use this operator 'type' information as the two edge types: commutative and non-commutative [4].

Researchers have previously proposed a host of methods for the formula retrieval task. Most of these approaches are based on heuristic methods [7] either in the text or tree matching domain. Text based methods generally treat the formula as a string and use algorithms like, term frequency-inverse document frequency (TF-IDF) for ranking [8] or using the largest common sub-string between the query formula and each indexed expression [9]. The tree based models try to overcome this by encoding the structural information [10], using partial sub-tree matching [11] or even a combination of retrieval and scoring methods [4].

Davila et al. in *Tangent-S* [4] proposed a pair-based index model, with a 3 phase approach: selecting candidates, finding and scoring the largest common sub-tree between query and each candidate, and finally a regression model to predict relevance based on the structural matching scores. A symbol pair is represented by the tuple (A,D,R), where A and D are the ancestor and descendant symbols on the tree, and R is the sequence of edge labels in the path from A to D. In retrieval time, these tuples are extracted from the query and are used to find candidate matches on the index. The second step uses a detailed alignment algorithm which finds the maximum common sub-tree between the query and each candidate. This is done while enforcing intricate rules governing math expressions such as unifying symbols of same 'type' (e.g. $x + y$ can be matched to $a + b$). Afterwards, candidates are re-ranked by computing three scores: *Maximum Subtree Similarity (MSS)*, *precision of candidate nodes matched with unification*, and *recall of query nodes matched without unification*. Calculation of these scores can be prohibitively expensive for large expressions. Thus the first step helps to score only the most promising candidates per query. Note that we use modified versions of these scoring functions to train our model.

Mansouri et.al. in *Tangent-CFT* [12] use the same (A,D,R) tuples of *Tangent-S* [4] to train a language model by treating them as tokens (e.g. words). Using this model they learn a vector representation for each tuple, and the average of these tuple vectors is used to represent the entire formula. Notice that the graph modality of our approach attempts to learn the final vector embedding from the graph structure directly.

Other recent works have used symbol pairs from a generic Line-of-Sight graph representation of mathematical expressions for visual search [13]. This model is agnostic to the math domain, but still matches expressions based on their visual similarity. Our image-based model attempts to capture additional information from the visual domain.

B. Triplet Learning

The triplet loss [14] learning objective introduced a semi self-supervised methodology for training models given the relative similarity between samples. Given an anchor a, a positive sample p and a negative sample n, a triplet is defined by their corresponding embeddings f^a, f^p, and f^n. The objective of the loss is to make the distance between f^a and f^n larger than the distance between f^a and f^p by a given margin α. For a batch of size k, it can be computed as follows:

$$L_{triplet} = \sum_{i=1}^{k} [||f_i^a - f_i^p||_2^2 - ||f_i^a - f_i^n||_2^2 + \alpha] \qquad (1)$$

The model is trained by optimizing parameters to minimize this loss. For a given anchor, this loss will draw the similar (positive) samples closer and will push away the less similar (negative) samples. The beauty of the formulation lies in the fact that we do not directly need absolute labels for each training sample. All we require is the information about

relative similarity between the given training samples. Since we have a heuristic scoring function which can calculate the relative similarity between mathematical expressions [4], this loss suits our objective perfectly.

C. Message Passing Neural Networks for Graphs

The message-passing algorithm was proposed for performing inference on graphical models, such as Bayesian networks and Markov random fields. It calculates the marginal distribution for each unobserved node (or variable), conditional on any observed nodes (or variables). Gilmer et. al. proposed a deep learning framework [15] for message passing neural networks (MPNN) and an aggregation procedure to compute a function of entire graph inputs. Intuitively, such a method gives us the capacity to model the innate inductive biases of a system. Convolution layers or sequential units like an RNN can easily model spatial or sequential inductive bias. But, it is very hard to learn such underlying relationships in the data, which one can only express as a graph [16].

Recent works focus on inductive and transductive learning tasks like node classification, graph classification, or link prediction. The graph 'convolutional' network, proposed by Kipf and Welling [17], learns locally shared parameters across a graph, proving to be a seminal work for the node classification task. Li et al. proposed attention based pooling models to generate graph level embeddings [18] while using global shared parameters in a gated propagation model. There has also been interest in trying to learn similarity between graphs using MPNNs. Li et al. proposed a model for retrieval and matching of graph structured objects, that uses a triplet learning objective [19] similar to our model.

D. Embeddings

An embedding is a vector from a manifold which is a topological space with n-dimensions. Simple euclidean geometry (e.g. euclidean distances) can be used to analyze data on this manifold. Many works aim to represent real world concepts in a vector space. A well-known example is *word2vec* [20] which does this for words, and facilitates generation of natural language, retrieval of similar documents, and other language based tasks. In this work, we develop a system to learn such vectors representing entire math expressions, and we use distances between embeddings to do MIR.

There have been multiple ways of defining the embedding function for nodes in a graph. Works like Isomap [21], which is a widely used isometry embedding method, exhaustively preserves distance patterns (lengths of shortest paths). Other works like Poincare embedding [22] creates latent spaces to capture the nodes in a graph. Also, struc2vec [23] preserves hierarchies and local structures in a graph. One could use more complex ways of capturing the topological information in the mathematical domains at the node level. For our current study however, we use a straightforward annotation scheme and focus more on building the embedding at the entire expression level. We describe this in the next section.

III. Methodology

We propose a multi-modal learner of similarity between math expressions. We train separate models on two modalities (images and graphs) and then further combine their predictions into the third and final model which leverages both pixel inputs and graph based inputs. Figure 2 shows all three components of our multi-modal learner. In this section, we discuss our custom graph-based representation for mathematical expressions. Then, we describe the metrics and procedures used to create triplets. We follow this with a description of each uni-modal encoder of math expressions as well as the combined multi-modal model. Finally, we describe our retrieval method.

A. Un-Directed Math Graphs

We convert the original SLT/OPT representations used by *Tangent-S* [4] and *Tangent-CFT* [12], depicted in Figure 1 (b) and (c), to un-directed graphs as shown in Figure 1 (d) and (e). We do this by adding the 'reverse' edge (shown in dotted arrows) so that messages can be propagated in both directions. We call these variants the Symbol Layout Graph (SLG), and Operator Graph (OPG).

B. Symmetric Scores

The three scores used by the fine-grain matching step of *Tangent-S* [4] (MSS, precision, and recall) are excellent similarity markers, and we use them to generate triplets. However, these scores do not represent symmetric distances. For two given expressions, they will vary based on which one is the query. Therefore, they are not directly viable in a triplet setting. To capture a smooth bijective, we introduce the *symmetric* variant of these scores: *sym-MSS* and *sym-Recall*. Essentially, we smooth the scores by using the harmonic mean of two scores computed by considering each expression once as the query and once as the candidate.

C. Model Objective and Triplets for Math Expression

Depending on how easily a network can learn to differentiate between the positive and negative samples in a triplet, we can classify them as 'easy' or 'hard' triplets. In this domain, triplets involving partially similar expressions are usually the hardest ones. Showing the network more 'hard' samples makes it learn faster and more accurately, since the loss contribution to the gradient is larger [14]. Given the size of our dataset, we opted for a viable strategy to generate easy and hard triplets that avoids exploding the training space. We utilise the concepts of both offline and online hard-mining of triplets to train the network. During the offline selection, our aim is to choose every expression as an anchor once as shown in Algorithm 1. During the online mining, we use Algorithm 2 to prioritize training on the hardest triplets.

D. Math Expressions as an Image

This module learns the relative similarity between mathematical expressions in the image space. Designing an image-based feature extractor allows us to further extract an additional modality of information from the visual representation

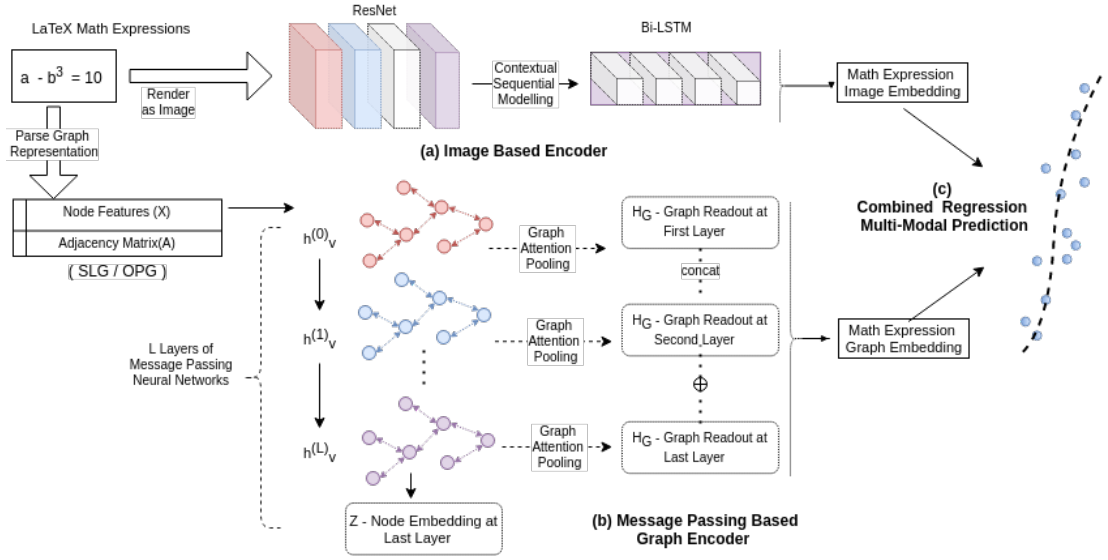

Fig. 2: Proposed Model For Learning Equation Similarity. The figure illustrates uni-modal encoders in (a) the pixel space (b) graph space and then the final multi-modal representation learnt (c) combining both the models. The combined model uses distances from both modalities and both base SLG/OPG representations, hence has 4 features and bias for the regression model.

Algorithm 1: Offline Hard-Mining of Triplets

Result: Offline subset of 3 million Math Triplets
Data: $\mathbb{E}$ - dataset of all expressions
for *Each sequential batch* b_i *of 1000 expressions in* $\mathbb{E}$ **do**
  for *Each sequential split* s_j *of 100 expressions in* b_i **do**
    Select all expressions in s_j as Anchors and the remaining 900 as Candidates;
    Score Candidates w.r.t Anchors;
    for *Each Anchor* **do**
      Select 10 candidates as matches, 90% with highest score, 10% with lowest score ;
      From the matches, choose the positive and negative, depending on which expression has a better score;
    end
  end
end

Algorithm 2: Online Hard-Mining of Triplets

Result: Online Triplets used for Model Training
Data: $\mathbb{T}$ - set of offline hard triplets
for *Each random big-batch* t_i *of 1000 triplets in* $\mathbb{T}$ **do**
  Generate embeddings for each triplet in t_i ;
  Select sub-set $\delta \in t_i$ for which $L_{triplet}$ in Equation 1 is $Positive$;
  Let δ_s be the list of top 800 triplets from δ sorted by decreasing $L_{triplet}$;
  for *Each sequential batch* $\delta_i \in \delta_s$ *of 100 triplets* **do**
    Run Training loop over δ_i;
    Pass to next big-batch if all triplets in δ_s are correct;
  end
end

of the formulas. In this work, we assume that inputs are noise-free binary images of typeset mathematical expressions.

As seen in Figure 2(a) the backbone of the feature extraction layers are a standard ResNet [24] and bi-LSTM layers to model the sequential nature of the data. We modify the ResNet architecture and update the input shape for the rectangular equation images. A convolution operator learns shared weights only based on spatial similarity. The output feature-maps from the ResNet thus lack the capacity to learn the sequential pattern of symbols. The bi-directional LSTM explicitly models this sequential nature $H = Seq.(P)$ after the spatial feature extraction stage. Each column in a feature map $p_i \in P$ is used as a frame of the sequence to generate the final image expression embedding.

Note that the original scoring functions used to create the triplets still need a tree-based input. For images, we can train two parallel versions, one trained using triplets from the SLT representation of the corresponding math expressions, and one for OPT. The resulting embeddings will be different given that similarity in SLT and OPT spaces are also different.

E. Math Expressions as a Graph

Our graph based architecture is a rendition of the neural message-passing models we covered in Section II-C. We have to decide on particulars such as edge representation vectors, node representation vectors, number of message passing layers, the message aggregation and propagation functions, and a pooling mechanism for graph level outputs.

Edges in a graph represent the relationship between two nodes. In this case, they either try to capture spatial information (SLG) or the operator relationship (OPG) as described in section II-A. We use a one-hot encoding scheme to represent edges as vectors. The goal is for the model to learn different weights for each type of SLG/OPG edge type.

TABLE I: Annotation construction for SLG and OPG.

SLG Vector Positions	Feature	OPG Vector Positions
[0-6]	Node Type	[0-19]
[7-16]	Fence Character	[20-29]
[17-25]	Row Size	[30-39]
[26-34]	Col Size	[40-49]
[35-816]	Symbol Vocab	[50-1963]

Nodes in a graph represent the central entities, in this case the mathematical symbols. The Tangent-S SLT/OPT [4] symbols have a $type$ and $value$ notation, which we use to construct a custom feature vector for each node. This vector is a concatenation of five one hot encoded vectors as shown in Table I, where each row in the table represents one property. The first and last vector represent the $type$ and $value$ of a symbol. The OPG representation has more node types (e.g. more specific sub-types for elements within fence characters [4]). It also captures information of math functions like sin which get treated as text nodes in SLG. The middle rows in Table I are dedicated vectors to capture matrix type expressions (An expression in multiple rows enclosed by one or two fence characters such as square/curly/angular brackets). This is based on similar representation schemes used for such formulae in the language model of *Tangent-CFT* [12] where they found that encoding dedicated information about matrices helps in telling these apart from similar symbols occurring in non-matrix formula expressions. The final concatenated vector serves as the initial node feature vector.

As we see in Figure 2(b), once we have these initial feature representations and the adjacency matrix, our next step is to define a function mapping it to a continuous latent space. Let $\mathbb{G}= (\mathbf{V}, \mathbf{E})$ be a graph representing a math formula, where each node $v \in V$ has a feature vector x_v and each edge $e \in E$ connects two nodes. Let $f : v \rightarrow z_v$ be a mapping function from a graph node to a d-dimensional vector representation from R^d. Here, $z_v \in R_d$ can also be considered as the position of node v in a latent continuous space. Preferably, the mapping function should capture the structure and properties of the graph which is translated as geometric distances in the latent space. This is facilitated by multiple layers of neural message passing. Each layer in the graph model can be described as a function f with two inputs: the features from the previous layer, and the graph adjacency matrix.

$$H(l+1) = f(H(l), A), \;\; where \tag{2}$$

$$H(0) = X \; and \; H(L) = Z \tag{3}$$

l being the layer number, X being the original node feature annotations (i.e input for layer 0), A being the adjacency matrix, Z being outputs at final layer L.

The authors of GCN [17] proposed the following propagation rule for each layer, to get the readout per node (v) at the $(l+1)^{th}$ layer :

$$h_v^{(l+1)} = \rho(\sum_{j \in N_v} \frac{1}{c_v} W^{(l)} h_j^{(l)}) \tag{4}$$

where ρ is a non-linearity function, C_v is the normalization constant calculated from the degree and adjacency matrix, $W^{(l)}$ is the weight kernel learnt for the $|V| \times d$ feature matrix over input feature space h.

This weight $W^{(l)}$ is shared by all edges in layer l. We need to have the functions learn weights per edge-type, to learn different relationships. So our final update rule becomes;

$$h_v^{(l+1)} = \rho(W_0^{(l)} h_v^{(l)} + \sum_{r \in R} \sum_{j \in N_i^r} \frac{1}{c_{v,r}} W_r^{(l)} h_j^{(l)}) \tag{5}$$

where N_v^r denotes the set of neighbor indices of node v under relation $r \in R$ and $c_{v,r}$ is a normalization constant. For entity classification, the R-GCN [25] paper uses $c_{v,r} = |N_v^r|$.

The message vector is constructed as a concatenation of the updated node features at the l^{th} layer and the one hot encoding of edge vector between neighbouring nodes of the incoming messages. After L layers of message passing, our final readout Z is still a $|V| \times d$ vector. To aggregate each of these vectors into a single graph embedding at layer L, we utilise the global attention pooling layer [18] :

$$H_g = \sum_{k=1}^{N_v} softmax(fgate(H_v^L)) ffeat(H_v^L) \tag{6}$$

where $fgate$ is a function that computes attention scores for each feature and $ffeat$ is a function that transforms the final readout $Z = H_L^v$ into the embedding dimension applied to each feature before combining them with attention scores.

The final embedding output vector in a multi-layer model for math expressions is created as a concatenated output of the global attention pooling outputs at each subsequent layer (see Figure 2(b)). We found this hierarchical read out benefits a lot in capturing information from each stage of message passing and helps improve the similarity score for partial matching. Intuitively, smaller message passing steps means that the embedding readout at that layer would represent a sub-graph entity, since it has information about only a few hop neighbours at each node. Instead, if we take the output of only the last layer, our embedding is a result of only the maximum possible message passing steps. Say, for an 8 layer model, our embedding would try to perfect the representation of every 8 neighbours, as if they were the smallest units. Thus, we see that full matches between larger graphs improve, but partial matches are lost. Having information from each layer of message passing mitigates this.

To optimize this model, we also combine a binary cross entropy criterion. This helps the model learn and converge better. Using Z, we calculate the loss from the initial concatenated binary annotation that we created for the node.

F. Combining Image and Graph based Models

We combine our uni-modal trained encoders using a simple linear regression model as depicted in Figure 2(c). The current method is kept similar to the combination model used by *Tangent-S* [4] to facilitate a direct comparison of results. The embeddings represent expressions on different manifolds and

combining the uni-modal vectors directly would not be as useful. But distances between embeddings on each of these separate manifolds should convey similarity between two expressions. Thus, where the heuristic baseline combinations use scores as the features, our model uses the distances between graph-based and image-based embeddings as the features. We use 4 distances in total, the SLG and OPG for both graph and image embeddings. The linear regressor is trained using relevance assessments provided by human annotators.

G. Mathematical Expression Ranking

Our final objective is to retrieve the most similar expression given a query-by-example. To keep things comparable with previous works, we find the top 1000 candidates for each query for ranking evaluation. For an embedding-based model this is a straightforward computation of the distance between embeddings. After training a model, we use it to exhaustively generate embeddings for the entire collection. Then, we simply consider the top 1000 expressions closest to the given query embedding, and we rank them by increasing distance. In this work, we simply adopt a brute force exhaustive method to find the distance between a query and each candidate in the collection. To deploy the system in a real-world setting, it would be trivial to use an index optimized for K-NN search, but this exercise was outside of the scope of this work.

IV. Experiments

We first discuss our dataset which comes from a sub-task of a large-scale MIR challenge. Then, our three experiments: using math expressions as images, as graphs, and finally using a combination of both representations.

A. Dataset

We use data from the NTCIR-12 MathIR [26] competition. Specifically, we use data from the optional Wikipedia Formula Browsing Task which has a corpus of 319,689 articles from English Wikipedia with more than half a million mathematical expressions. The original task has 40 topics (queries) for isolated formula retrieval: 20 are concrete (without wildcards) and 20 include wildcards. Following other recent studies [11], [12], we focus only on concrete queries.

At the original competition, the top-20 results for each topic from 8 submissions were evaluated for relevance. Each result was assessed by two human evaluators who scored them from 0 (irrelevant) to 2 (relevant). These scores were summed and each formula has a final relevance score between 0 and 4. Any candidate with a score of at least 1 is considered a partially relevant match, and if the score is at least 3 then it is considered a fully relevant match. A total of 2687 relevance assessments were produced by this method.

Expressions with at most 20 SLG nodes cover 92% of the expressions in the corpus. Figure 3 shows the histogram of SLGs graphs sizes for all graphs having between 1 to 70 nodes. The whole dataset has a very skewed distribution with the majority of expressions having between 4 to 8 nodes. The largest expression in the dataset has more than 1500 nodes, but such large expressions are rare.

For our uni-modal experiments using either images or graphs to represent math expressions, we take the whole subset of expressions with up to 20 nodes, and split it into 3 disjoint sets for training, validation and testing. In this setting, the queries (or anchors) are first selected randomly from the available expressions on each split, and they are fixed across conditions. For the last experiment, we use the complete collection and the official benchmark queries.

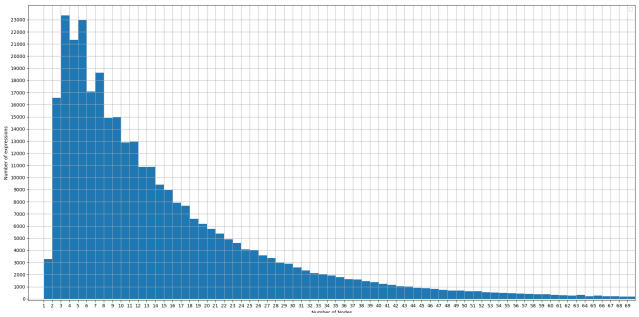

Fig. 3: Distribution of SLGs graphs size with $1-70$ nodes. Each column on x-axis represents 1 node and each row represents 1000 expressions.

B. Evaluation Metrics

We use the Normalized Discounted Cumulative Gain@p ($nDCG_p$) in our evaluation. The premise of DCG is that low relevant matches appearing higher in the rank should get a higher penalty. In this sense, the graded relevance value of each match is reduced logarithmically proportional to their position in the rank. The traditional formula of DCG accumulated at a particular rank position p is defined as :

$$DCG_p = \sum_{i=1}^{p} \frac{rel_i}{log_2(i+1)} = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{log_2(i+1)}$$

Search result lists vary in length depending on the query. As such, the cumulative gain at each position for some p needs to be normalized across queries. This is done by sorting all relevant expressions in the corpus by their relevance related to the query, producing the maximum possible DCG through position p, also called Ideal $DCG(IDCG_p)$. For a query, the normalized DCG, or nDCG, is computed as:

$$nDCG_p = \frac{DCG_p}{IDCG_p}$$

TABLE II: Math Expression as Images: $nDCG_{10}$ for test set consisting of random anchors and *sym-recall* as ideal score

Model(input size)	SLT	OPT
ResNet(100×200)	0.3901	0.3861
ResNet(50×400)	0.3974	0.3882
ResNet(80×250)	0.3926	0.3861
ResNet (50×400) + BiLstm	**0.401**	**0.3996**

TABLE III: Ablation study - $nDCG_{10}$ for test set consisting of random anchors and *sym-recall* as ideal score

Layers # Parameters	4 3059780	6 3695270	8 4326632	10 4953866	4 3063780	7 3701270	8 4334632	10 4963866
# Math Symbols	SLG				OPG			
1-4 Nodes	0.3172	0.3544	0.348	0.411	0.3221	0.297	0.3731	0.402
5-8 Nodes	0.3553	0.3279	0.3539	0.3562	0.3316	0.3147	0.3429	0.3476
9-12 Nodes	0.3448	0.3333	0.3245	0.3731	0.3401	0.3261	0.331	0.3519
13-16 Nodes	0.3321	0.346	0.3677	0.3889	0.3274	0.3301	0.348	0.3497
17-20 Nodes	0.3401	0.3621	0.38	**0.493**	0.3341	0.3545	0.3661	**0.421**

TABLE IV: NTCIR-12 MathIR Wikipedia Formula Browsing Subtask Results: Average Bpref for Concrete Queries

S.No.	Model	Partially Relevant Matches			Fully Relevant Matches		
#	Bpref@1000	SLT	OPT	Combined	SLT	OPT	Combined
1	MCAT [10]	-	-	0.569	-	-	0.567
2	Approach-0 (3-B) [11]	-	0.595	-	-	0.672	-
3	Language Model Embedding: Tangent-CFT [12](and combos)	0.66	0.66	**0.71**	0.58	0.60	0.60
4	Tangent-S [4] (core)	0.606	0.521	-	0.565	0.583	-
5	Tangent-S [4] (matching)	0.564	0.505	-	0.620	0.589	-
6	Tangent-S [4] (regression)	0.571	0.558	0.587	0.618	0.600	0.636
7	MSS (full-matching)	0.602	0.559	-	0.650	0.617	-
8	sym-MSS (full-matching)	0.566	0.543	-	0.604	0.608	-
9	sym-Recall (full-matching)	0.701	0.560	-	0.625	0.624	-
10	sym-Scores (full-matching, all 6)	-	-	0.688	-	-	0.65
11	**ResNet** (50×400) **+ BiLstm**	0.406	0.421	0.441	0.465	0.483	0.493
12	**Graph Embedding 1-20 Nodes; 10Layers**	0.591	0.579	0.615	0.623	0.58353	0.660
13	**Graph + Image All 4 Regression**	-	-	0.673	-	-	**0.694**

where IDCG is calculated in our work using the top matches based on their *sym-Recall* score.

The second evaluation metric used in our experiments is Bpref. This metric locates the top s judged non-relevant formulas, then compares their ranks against the ranks of the k judged relevant formulas pairwise. The $nDCG_p$ is very sensitive to the level of relevance of each match in the rank. However, it requires all matches in the rank to have an associated or known relevance value. In contrast, the $bpref$ metric only cares to measure the number of known irrelevant matches that are ranked before the known relevant matches and ignores any result which does not have an associated relevance value. We use Bpref to evaluate rankings based on incomplete human relevance assessments from the NTCIR-12 MathIR Wikipedia Formula Browsing Task benchmark.

C. Math Expressions as Images

We render binary images of isolated expressions from our dataset using LaTeX representations. All image models are trained on the full subset of expression with at most 20 nodes.

We experimented with 3 resolutions with the same pixel count : 100×200, 50×400, and 80×250. We then used the best one (50×400) combined with the bi-LSTM layers. For each model, we trained two versions, one using SLT-based triplets and another using OPT-based triplets. The results are reported in Table II. We provide these results in terms of $nDCG_{10}$ taking the *sym-Recall* as the relevance predictor.

D. Math Expressions as Graphs

We want to understand the effect of the number of message passing layers in a MPNN model vs the size of the graphs. We further split the training portion into sub-splits by size: 1-4, 5-8, 9-12, 13-16, 17-20 nodes. We used these splits to evaluate 4 types of models with increasing number of message passing layers: 4, 6, 8, 10. We also trained two versions of each model, one using SLG and one for OPG, for a total of 40 models. We report the results in terms of $nDCG_{10}$ in Table III. We also mention the number of trainable parameters for each model, to verify if any improvement is not just due to an increased representational capacity of the model.

Each layer of an MPNN sends and aggregates messages to one-hop neighbours. After l layers of message passing each node state vector has information about symbols and edges $l-$hops away. So, for a 10 layer model trained on a data consisting of expressions with 17-20 nodes, we expect for the largest expressions that at least one node would learn about every other node in the graph. Increasing number of layers has an improvement, but not for mid-sized graphs. The smallest expressions with 1-4 nodes perform better than mid-sized ones. The smaller expressions could turn out to be sub-graphs for a larger mathematical expression. A model having the capacity to learn such a pattern essentially performs better.

E. Combining Images and Graphs

We report the Bpref scores for the top 1000 results for the 20 concrete queries from the benchmark in Table IV. We use two relevance thresholds (see Section IV-A) to get Bpref for fully relevant matches and partially relevant matches. Results in the 'combined' columns represent models which use information from both SLT/SLG and OPT/OPG. Rows 1-6 of the table provide results from previous works. Interestingly, *Tangent-CFT* [12] provides the best results for partially matching expressions, but our results for exact matching are better.

First, we evaluate an exhaustive version of the 're-ranking' step from *Tangent-S* (see Section II-A) and compare it against our 'symmetric scores'. Rows 7, 8, and 9 (tagged as 'full-matching') in Table IV represent these results. We calculate the score for all the expressions per official query. This is

a slow process and took over a week running on an AMD 1950X 16-Core Processor with 2 threads each. Row 10 in in table IV shows our results for the regression combination of each of the three metrics. For partially relevant matches, The *sym-Recall* full-matching gives us very similar numbers to the combined language model embedding of *Tangent-CFT*.

Next, we evaluate models on human graded relevance. Rows 11 and 12 of Table IV show the evaluation of our best uni-modal architectures. Theses models are trained on whole subset of expressions with 1-20 nodes. The graph based embedding shows quite promising numbers. Row 13 is the farther of embedding distances from both modalities and both representations (see Section III-F). These regression models are all trained on the small subset of the human graded relevance data with a leave-one-out cross validation scheme.

V. Conclusion

We have proposed a method for encoding math expressions and their similarity in latent space. Our experiments provide the groundwork for leveraging the rapid advancements in geometric deep learning and deep metric learning. Improving the underlying graph embedding, message passing network, aggregation techniques or updated criterion functions can easily be implemented as a plug-and-play updates to improve the results that we have achieved. Combining image and graph based non-euclidean feature extractors in a early/late fusion model would be definite next steps. This in turn also gives us a huge boost for leveraging updated and improved methods from the domains of computer vision and pattern recognition.

VI. Acknowledgements

We thank the reviewers for their valuable inputs. This research was supported by the National Science Foundation under Grant No. 1640867 (OAC/DMR).

References

[1] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, p. 1798–1828, Aug. 2013.

[2] T. Rocktäschel and S. Riedel, "End-to-end differentiable proving," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., 2017, pp. 3788–3800.

[3] K. Yuan, D. He, Z. Jiang, L. Gao, Z. Tang, and C. L. Giles, "Automatic generation of headlines for online math questions," in *AAAI*. AAAI Press, 2020, pp. 9490–9497.

[4] K. Davila and R. Zanibbi, "Layout and semantics: Combining representations for mathematical formula search," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '17. Association for Computing Machinery, 2017, p. 1165–1168.

[5] R. Zanibbi and D. Blostein, "Recognition and retrieval of mathematical expressions," *Int. J. Doc. Anal. Recognit.*, vol. 15, no. 4, p. 331–357, Dec. 2012.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT (1)*, 2019, pp. 4171–4186. [Online]. Available: https://aclweb.org/anthology/papers/N/N19/N19-1423/

[7] R. Zanibbi, K. Davila, A. Kane, and F. W. Tompa, "Multi-stage math formula search: Using appearance-based similarity metrics at scale," in *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '16. Association for Computing Machinery, 2016, p. 145–154.

[8] P. Sojka and M. Líška, "The art of mathematics retrieval," in *Proceedings of the 11th ACM Symposium on Document Engineering*, ser. DocEng '11. Association for Computing Machinery, 2011, p. 57–60.

[9] B. C. Pavan Kumar P., Agarwal A., "A structure based approach for mathematical expression retrieval," in *Multi-disciplinary Trends in Artificial Intelligence. MIWAI 2012. Lecture Notes in Computer Science, vol 7694.* Springer, 2012.

[10] G. Y. Kristianto, M. quoc Nghiem, and A. Aizawa, "The mcat math retrieval system for ntcir-10 math track."

[11] W. Zhong and R. Zanibbi, "Structural similarity search for formulas using leaf-root paths in operator subtrees," in *Advances in Information Retrieval*, L. Azzopardi, B. Stein, N. Fuhr, P. Mayr, C. Hauff, and D. Hiemstra, Eds. Springer International Publishing, 2019, pp. 116–129.

[12] B. Mansouri, S. Rohatgi, D. W. Oard, J. Wu, C. L. Giles, and R. Zanibbi, "Tangent-cft: An embedding model for mathematical formulas," in *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval*, ser. ICTIR '19. Association for Computing Machinery, 2019, p. 11–18.

[13] K. Davila, R. Joshi, S. Setlur, V. Govindaraju, and R. Zanibbi, "Tangent-v: Math formula image search using line-of-sight graphs," in *Advances in Information Retrieval*, L. Azzopardi, B. Stein, N. Fuhr, P. Mayr, C. Hauff, and D. Hiemstra, Eds. Springer International Publishing, 2019, pp. 681–695.

[14] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," *CoRR*, vol. abs/1503.03832, 2015.

[15] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17. JMLR.org, 2017, p. 1263–1272.

[16] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. F. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, Ç. Gülçehre, H. F. Song, A. J. Ballard, J. Gilmer, G. E. Dahl, A. Vaswani, K. R. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu, "Relational inductive biases, deep learning, and graph networks," *CoRR*, vol. abs/1806.01261, 2018.

[17] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *CoRR*, vol. abs/1609.02907, 2016.

[18] Y. Li, D. Tarlow, M. Brockschmidt, and R. S. Zemel, "Gated graph sequence neural networks," *CoRR*, vol. abs/1511.05493, 2016.

[19] Y. Li, C. Gu, T. Dullien, O. Vinyals, and P. Kohli, "Graph matching networks for learning the similarity of graph structured objects," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 2019, pp. 3835–3845.

[20] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 3111–3119.

[21] J. B. Tenenbaum, V. d. Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[22] M. Nickel and D. Kiela, "Poincaré embeddings for learning hierarchical representations," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 6338–6347.

[23] D. R. Figueiredo, L. F. R. Ribeiro, and P. H. P. Saverese, "struc2vec: Learning node representations from structural identity," *CoRR*, vol. abs/1704.03165, 2017.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[25] M. S. Schlichtkrull, T. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *ESWC*, 2018.

[26] R. Zanibbi, A. Aizawa, M. Kohlhase, I. Ounis, G. Topic, and K. Davila, "Ntcir-12 math-ir task overview," in *NTCIR-12*, 2016, p. 299–308.