A Machine Learning Framework for Volume Prediction

Umutcan Önal[1(✉)] and Zafeirakis Zafeirakopoulos[2]

[1] Gebze Technical University, Gebze, Turkey
umutcanonal@gmail.com
[2] Institute of Information Technologies, Gebze Technical University, Gebze, Turkey

Abstract. Computing the exact volume of a polytope is a #P-hard problem, which makes the computation for high dimensional polytopes computationally expensive. Due to this cost of computation, randomized approximation algorithms is an acceptable solution in practical applications. On the other hand, machine learning techniques, such as neural networks, saw a lot of success in recent years. We propose machine learning approaches to volume prediction and volume comparison. We employ various network architectures such as feed-forward networks, autoencoders and end-to-end networks. We develop different types of models with these architectures that emphasize different parts of the problem, such as representation of polytopes, volume comparison between polytopes and volume prediction. Our results have varying rate of success depending on model and experimentation parameters. This work intends to start the discussion about applying machine learning techniques to computationally hard geometric problems.

Keywords: Machine learning · Autoencoders · Neural networks · Polytope · Volume

1 Introduction

Recent success of machine learning and deep learning brought a lot of new applications to various fields, where machine learning was not used until recently. Computational geometry and algebra is one of the fields where machine learning is not applied sufficiently.

In this work, we employ random forests and neural networks for a geometric problem, namely volume computation. Among neural networks, autoencoders play a special role in the proposed framework. We focus on two problems related to polytope volume computation, namely volume comparison and volume prediction. The first problem is a binary classification problem in machine learning terminology. We consider pairs of polytopes and label them according to whether the first or the second polytope have larger volume. The second problem is a typical regression problem, where the model tries to predict a continuous value (namely the polytope's volume) for given input.

I. Kotsireas et al. (Eds.): SEA^2 2019, LNCS 11544, pp. 408–423, 2019.
https://doi.org/10.1007/978-3-030-34029-2_27

Random Forest. Random forests are ensembles of multiple tree predictors which are trained with randomly sampled independent features from the original feature set. Every predictor generates an output based on their own feature set and the output of the forest is decided by majority voting from predictor outputs. If the tree is for a regression problem instead of a classification problem, the output will be the mean of the tree outputs. It is a flexible algorithm that has a good potential to generalize the data. However, it also has some caveats such as depth of the trees. One of the major problems with decision trees is that they tend to overfit if the depth is not limited to some degree. Random forests can overcome this problem to some extend, however it is still one of the important parameters to decide during training.

Neural Networks. Neural Networks or Artificial Neural Networks is a machine learning technique that uses a network of neuron-like units to learn a model. After Hinton et al. [12] introduced the back-propagation method, it became possible to construct multiple layered networks, which are called Deep Neural Networks . A basic form of these networks is called multilayer perceptron or Feed-forward Neural Network. Layers are densely connected in this type of networks which means every perceptron is connected to all of the perceptrons in the previous and the next layer in the network. A big success came for neural networks when Krizhevsky et al. [10] improved considerably the accuracy on ImageNet database with Convolutional Neural Network (CNN) in 2012 . This type of network skips some of the connections in order to simulate the convolution operation extracting features from images. The success of this model attracted a lot of attention and new convolutional networks such as GoogleNet [15], VGGNet [13], ResNet [6] followed. The same year, Hinton et al. [7] presented another state-of-the-art Deep Neural Network application for speech recognition.

Recurrent neural networks (RNN) [8], another important type of neural networks models, rely on using a feedback mechanism in order to remember previous states. This mechanism allows the network to sequentially process data and learn spatial or time based information. Even though it is used mostly for sequential problems, such as speech recognition or music composition, it can be used for other types of tasks to process data sequentially.

At first glance an RNN does not look like a Deep Neural Network. But when unfolded its depth becomes visible. This is due to the feedback mechanism that gives the previous output as an input as well.

Recurrent Neural Networks are powerful tools in theory, however, they are not working as well in practice. Back-Propagation Through Time algorithms apply the idea of back-propagation of errors in feed forward networks. Errors are back-propagated in time instead of between the layers of the RNN. This propagated error tends to blow up or vanish if the sequence is too long, i.e., the network is too deep. The Long-Short Term Memory (LSTM) architecture is proposed as remedy for this problem [8]. The LSTM architecture helps to prevent losing information due to long time lag, thus allows such networks to work with long sequences.

Autoencoders. Autoencoders are a type of neural networks which is used to extract features from data in a unsupervised fashion. A particularly powerful aspect of neural networks is feature extraction without any human supervision. This unsupervised feature extraction enables inference capacity of neural networks given enough data, although it obscures the exact decision making process. This idea is presented by Dai and Le [3] for a sentiment analysis model.

A network can be considered as an autoencoder when the input and the output of the network is the same. The network only learns the patterns and relations in the data itself and the weights or the intermediate output of this network can be used as a set of extracted features.

An encoder-decoder model takes an input of variable length and encodes it in the encoder layer into a fixed-size vector. Later, this fixed-size vector is decoded to output which has variable length again. Sutskever et al. [14] and Cho et al. [2] proposed this model (with slightly different implementations) for problems which require to convert a sequence to another sequence, e.g., machine translation or speech recognition.

Sequential problems, usually do not have fixed-length input, while most neural networks require fixed-length input. Since the representation of a polytope is a sequence, encoder-decoder based models are applicable for the problem we study. If the input and output of an encoder-decoder model is the same, it is called autoencoder and is one of the methods for feature extraction.

1.1 Polytopes

Polytopes have two different and equivalent representations. They can be represented as the intersection of a finite set of half-spaces (H-polytope) or as the convex hull of a finite set of vertices (V-polytope). A detailed description of the related theory is out of the scope of this paper, for an excellent presentation see [17].

In this work we consider V-polytopes, namely polytopes given by their vertices. Given the vertices, one can represent the polytope as a list of lists. The length of this list is the number of vertices and the length of each element is the (ambient) dimension of the polytope. The fact that both dimension and number of vertices varies, is one of the main obstacles in using machine learning for polytope related problems. As discussed above, a common practice is the use of autoencoders to extract a feature set that represents the data well enough (or even better) in order to improve the success of the machine learning model. We will present different approaches on how to encode polytopes (and their vertices).

One of the fundamental problems in computational geometry is to compute the volume of a polytope. Volume computation has many applications ranging from financial models [1] to systems biology [9]. Dyer et al. [4] proved that exact volume computation is #P-hard. This lead to the development of randomized and approximation algorithms in order to compute the volume of high dimensional polytopes. The current state-of-the-art is presented in Emiris et al. [5].

We choose volume computation as the problem of interest in this work because:

– an approximation is the best we expect in polynomial time,
– it has natural prediction (regression) and comparison (classification) versions
– there are reliable means to produce data.

1.2 Our Goal and Structure of the Paper

The goal of this work is is to propose a framework allowing the use of machine learning for geometric and algebraic problems. Among the abundance of geometric and algebraic problems, that can be considered for such applications, we choose to focus on polytopes and the computation of their volume. The reason is that this problem already encounters some of the fundamental obstacles in applying machine learning to geometric and algebraic problems, namely that the dimensions of the input are not fixed.

The rest of this paper is structure as follows: In Sect. 2 we present the proposed models and in Sect. 3 we describe the datasets we used. Section 4 contains an analysis of the experiments and the experimental comparison of the proposed models. Finally, in Sect. 5 we conclude the paper.

2 Description of the Models

In this work we consider V-polytopes, namely polytopes given by their vertices. Given the vertices, one can represent the polytope as a list of lists. But a list of lists is ordered, while the vertices of the polytope do not have a natural order. Even though it is possible to give an arbitrary order, since this will not be natural, it may cause failure in our models. For the rest of this work, we consider the polytope given by the list of its vertices. We will present different approaches on how to encode polytopes (and their vertices).

2.1 Encoding a Polytope

An essential difficulty of the encoding of a polytope is the variable length of input both in terms of how many vertices a polytope has and what is the dimension of the polytope.

We will indicate whether a model assumes the dimension is fixed or variable, by the first letter of the model name, i.e., `F` or `V` respectively.

Fixed Dimension. As we saw in Sect. 1, traditionally the dimensions of the input tensor are fixed, while with RNNs we can have one varying dimension if we think that dimension as timesteps. In order to exploit existing models, we have to restrict the generality of the problem.

If we fix an upper bound on the dimension of the polytope, then we can pad with zeroes the vertices of any polytope of dimension smaller than the bound. This allows us to use standard RNN/LSTM solutions and build end-to-end models with ease.

Variable Dimension - Autoencoder for Vertices. One way to achive encoding without fixing the dimension is by using an autoencoder for vertices. This autoencoder works in a way similar to word embeddings in NLP. It is used to encode vertices into a fixed size representation, in order to fix the vertex dimension. This way we don't lose dimension information. However the lack of context, i.e., that a set of vertices belongs to a polytope, may hinder the performance. After encoding, we use the encoded vertices to construct polytopes again. We use this approach as a preporcessing step, before training an end-to-end model for fixed dimensional polytopes.

Polytopes with Fixed Dimensions and Number of Vertices. Inspired by our fixed dimensional model, we also train a model with both fixed dimension and number of vertices. We achieve this by padding data with zeros up to some maximum number of vertices or dimension. One of the important aspects of training with this data is that it removes the need of sequential processing with RNNs. We only used feed forward networks with 2 inputs for different polytopes. This approach is much simpler in many ways compared to the previously mentioned models, but it lacks generality and can only be used for polytopes within its predefined limits.

Flatten. Another way to use RNN/LSTM to encode a polytope is to flatten its list of vertices. One of the major problems with this approach is the loss of dimension information. We use this approach in modular models.

2.2 The Problems

In this work we consider two problems related to volumes of polytopes. The first is the regression problem of approximating the volume of a given V-polytope. The second is the classification problem of deciding if polytope A has larger volume than polytope B, given two V-polytopes A and B.

Volume Prediction. Volume prediction is the main problem we are trying to solve in this work. This is a regression problem. In the first set of models we predict the volume of a given polytope, while in the second set of models we predict simultaneously the volumes of a pair of polytopes.

Volume Comparison. In order to define a classification problem related to the regression problem of volume prediction, we study the volume comparison problem. The goal is to observe the performance of the models in comparing volumes. Given two polytopes as input, the model compares the volumes and labels the pair of polytopes as 1 if the volume of the first polytope is larger and 0 otherwise. This is a binary classification problem.

Coupling Prediction with Comparison. We also want observe the effect of comparison and prediction on each other, by combining the two problems in one model. This was the original motivation for studying the classification problem. Generally, the classification problem is easier, and the question is whether predicting and comparing simultaneously, improves the performance of one or both tasks. Having a model performing multiple tasks may be a useful tool.

2.3 Modular Vs End-to-End

Concerning the training of the models, in general, there are two approaches we employ. The first is to train an autoencoder for polytopes and then separately train a network for prediction/comparison. We will mark such solutions by `APC_`. The second approach is to perform end-to-end learning and we will use `E2E_` to mark models following this approach.

An important effect of using modular encoder and regressor models is that it may work with different machine learning algorithms like Random Forest as well. As we will see in Sect. 4, those algorithms may produce better results. However, it is expected to have overall better results with `E2E_` neural network models.

2.4 The Models

In what follows we present different solutions to these problems and for clarity we will name the models in the following way:

- the first part declares if the dimension is fixed or variable by `F` or `V` respectively,
- the second part is `F` if input is flatened, `L` if a list of lists, and `P` if completely padded,
- the third part declares the training scheme by `E2E` or `APC` and
- the fourth part is `C` for volume comparison, `P` volume prediction or `PC` for a coupled model.

For every model, we describe the coupled version `*_*_***_PC` and in the corresponding figures we give also the outline of the comparison only (`*_*_***_C`) and prediction only (`*_*_***_P`) architectures.

In Fig. 1 we see model `F_L_E2E_PC`. The input of the model is a fixed dimensional polytope. The dimensions are fixed by padding the input vectors to the highest dimension in the data. However, it still employs an encoder to generate a fixed representation of a polytope because the number of vertices still variable.

The `F_P_E2E_PC` model (see Fig. 2) requires a fixed number of vertices as well as fixed dimensions. Thus, it only uses dense layers and requires no RNNs to process the input. After the input polytopes are encoded in a fixed size representation, the representation vectors of the two input polytopes are concatenated and fed to the next layer (Fig. 3).

The `V_L_E2E_PC` model (see Fig. 4) differentiates from previous models by keeping both dimension and number of vertices variable. We use two encoders,

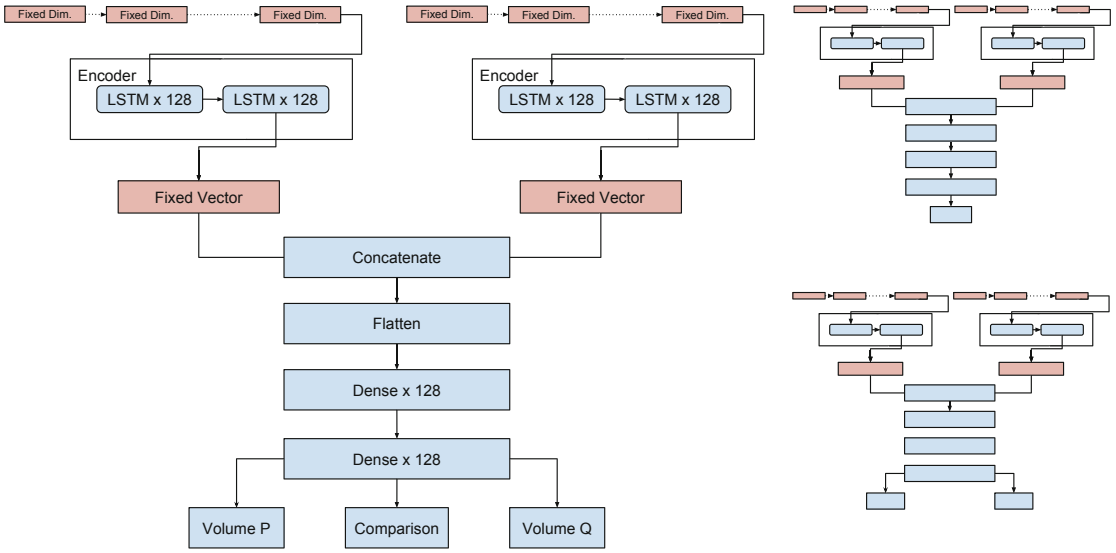

Fig. 1. Fixed dimensional end-to-end models

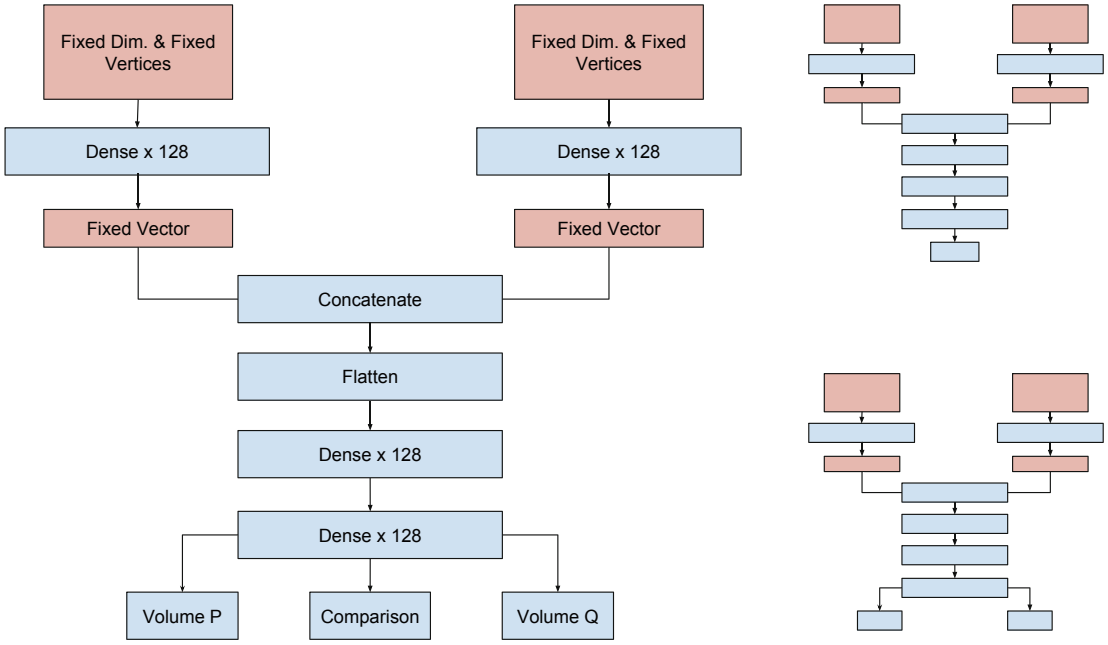

Fig. 2. Padded model

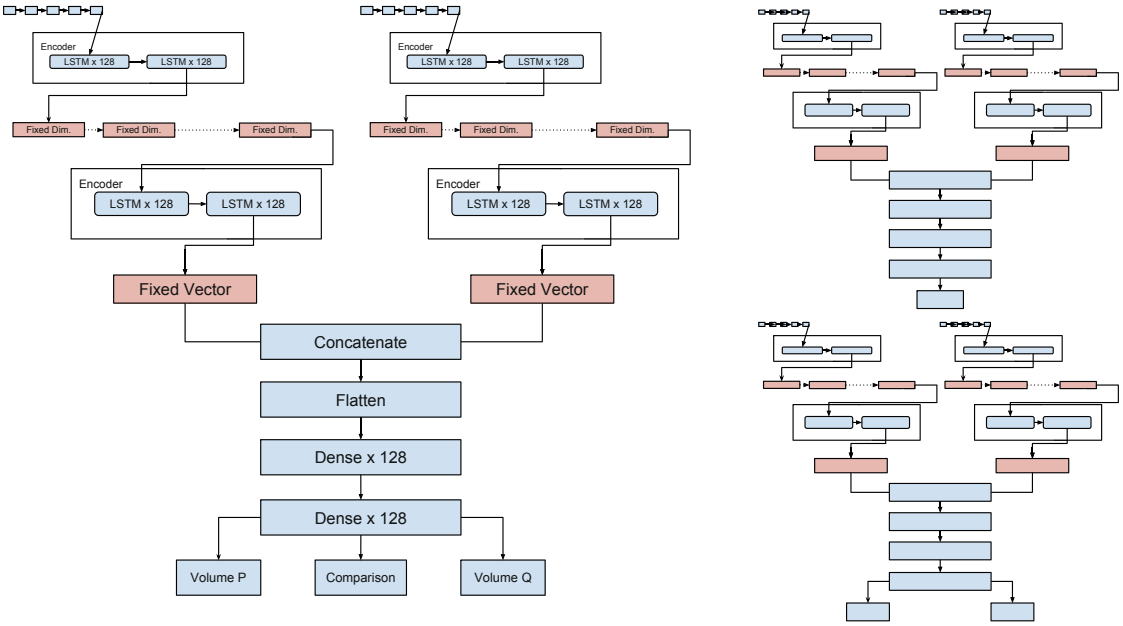

Fig. 3. End-to-end model for both volume predictions and comparison.

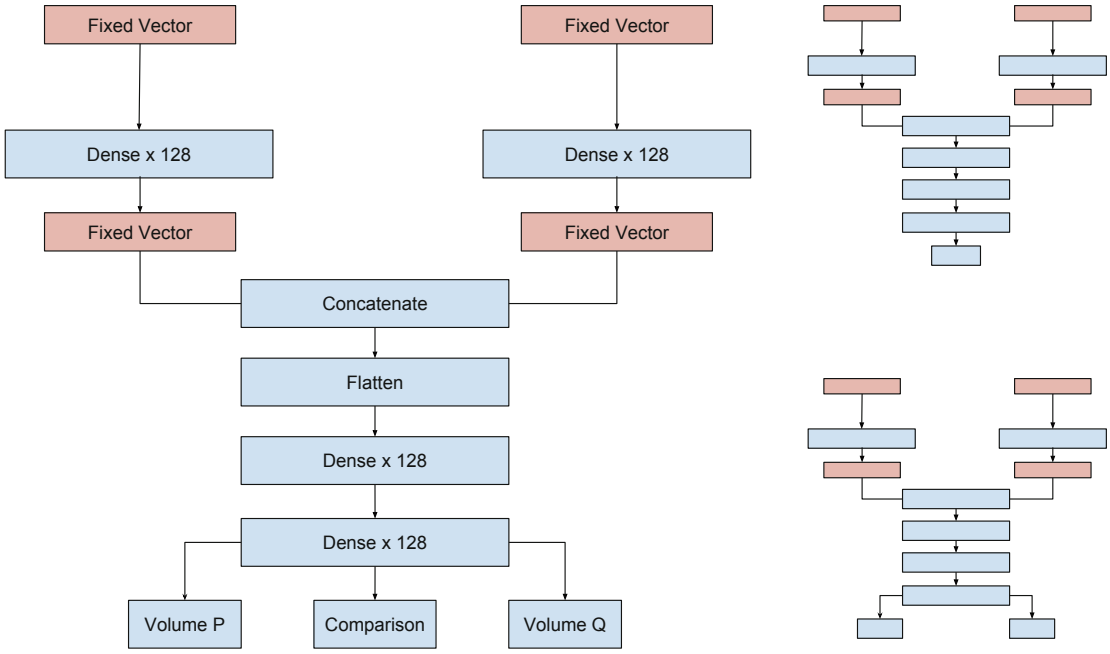

Fig. 4. Feed-forward Neural Network for modular solution.

one for vertices and one for polytopes, in order to achieve this. Apart from having one more encoder for input, this model is similar to `F_L_E2E_PC`.

The last model we introduce is `V_F_APC_PC`. The input of this model is flattened and it is our only modular model. It consists of an autoencoder that takes the input and a regression/classification model. The autoencoder part of the model (see Fig. 5) takes a flattened list of vertices and encodes it into a fixed size vector. Then, this fixed size vector respresentation of the polytope is used as input of the second part of the model. For the second part of the model, any machine learning model can be used. We used Feed-Forward Neural Networks and Random Forest in this part. In Sect. 4, we will indicate what is the model used for the second part in the experiments.

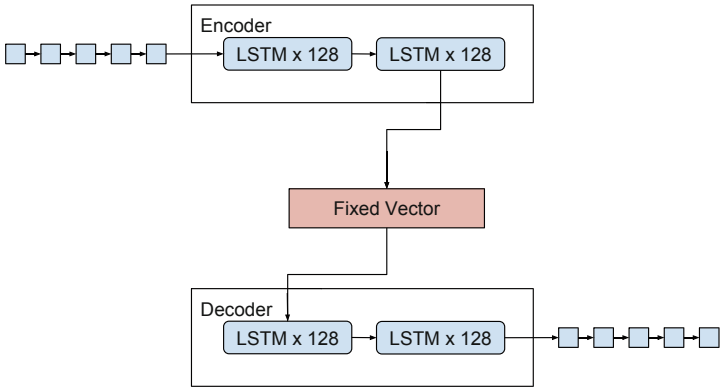

Fig. 5. The autoencoder model using LSTM

3 Description of Data

One of the biggest challenges when designing machine learning experiments is to find data. Although it is easy to create artificial datasets, it is important to make sure that the choice of instances is meaningful.

Data Sources and Generation. In this work, we used two types of data, namely, random polytopes and polytopes from specific families.

A random polytope is generated by first choosing at random (within a predefined range) the number of vertices of the polytope and then create a random polytope with that many vertices using SageMath [16]. The dataset generator creates a random vector with integer coordinates (indicating the numbers of vertices of each polytope in the dataset). The length of this vector indicates the size of the dataset. In this process, the dimension of the polytopes is fixed. Different sets of polytopes of different dimensions are generated by changing this fixed value.

Concerning polytopes from specific families, we use cubes, hypercubes, cross-polytopes and Fano polytopes. We use SageMath to generate cubes, hypercubes, cross-polytopes, while Fano polytopes are taken from [11].

Datasets. We use Fano polytopes paired with each other based on dimension in our experiments. This is a requirement of the comparison problem we described before. This also increases the the number of data instances even though we start with a limited number of polytopes from this data set. We used 200 polytopes and this produced more than 19.000 pairs in the end.

Our second data generator uses predefined families of polytopes in SageMath. We chose several base polytopes, such as unit cube, and we create new polytopes by applying random transformation to these base polytopes. Random transformations include scaling up to 10 times and replacing the center from the origin to another point and applying slight rotations.

A summary of the data can be seen in Table 2. We have three sets of polytopes coming from randomly generated polytopes and a set of polytopes only consists of Fano polytopes. We name these sets as A, B, C an D. Table 2 shows which data set includes which family of polytopes.

In Table 1 we give a statistical analysis of volumes in the datasets we used.

Table 1. Volume statistics

Polytope set	Dimensions	Instances	Min	Max	Mean	Variance
Set A	3,4,5	15000	8.0	1889568.0	113619.3272	62416457013.52322
Set B	3,4,5	7500	8.0	1889568.0	151488.8789	120202038886.83537
Set C	3	2500	8.0	5832.0	1845.8047	3814934.1211
Set D	3,4,5,6	200	6.0	123.0166	28.35704	510.43171

We split all the data sets into three sets before training phases. 30% of the data goes into validation and 10% goes to test set as rest of the data goes into training set. We make this split once and use the same splits in all of the experiments later. We use training and validation during training but we use the test set only for the evaluation of the models.

Table 2. Dataset summary

Polytope family	Dimension	Set A	Set B	Set C	Set D
Cube	3	yes	yes	yes	no
Hypercube	4	yes	yes	no	no
Hypercube	5	yes	yes	no	no
Cross-polytope	4	yes	no	no	no
Cross-polytope	5	yes	no	no	no
Random	3	yes	no	no	no
Fano	3,4,5,6	no	no	no	yes

3.1 Normalization

Normalization of data is very important for machine learning in general. Feature values are usually compressed into the $[-1, 1]$ range in order to remove range differences. However, actual values have importance when dealing with geometric and algebraic problems. This lead us to use different normalizations on volume by multiplying it by a scalar value. We have variations of out data sets with different normalization multipliers applied before training (see Table 3).

4 Experimental Results

In this section we present the training setup and results of the experiments.

Metrics and Parameters. Volume prediction is a regression problem and there are several different metrics to measure the performance of a regression model. A standard metric is the mean squared error (MSE). Although we use MSE during training as a loss function, it is hard to compare the performance of different models based on MSE. For example, applying different normalizations results in vastly different values for MSE. The same is true for mean absolute error. Since it is important that the metric is not affected by such changes, we use R^2 (R-squared) as regression measure. In R^2, the best score is 1, while 0 means the model produces the same output regardless of the input. It is possible to get negative values, meaning that the model performs worse than giving always the same output.

For the classification problem, the label distribution is totally balanced, since for every pair of polytopes (A, B), we also include (B, A) in the dataset. The balanced distribution of both labels makes accuracy a good and simple enough metric to measure performance. During training, we use Binary Cross Entropy as loss function.

Another important evaluating criterion to consider is overfitting. Overfitting occurs when a model memorizes too much from the training data and cannot perform well with new inputs. Datasets are split into training, validation and test sets in order to evaluate the fit of the model to the data. We do not see overfit in the experiments as the results from training and testing are similar.

Concerning the rest of the training parameters (unless stated otherwise), we use Adam as optimizer with a learning rate of 0.001. All of the models are trained for 20 epochs with the exception of autoencoders when a model has one. Autoencoders are trained for 30 epochs, with the same optimizer and learning rate.

Modular Framework. The first part of experiments uses randomly generated polytopes and the `V_F_APC_P` model to predict volume. Note that this part of experiments is significantly different from the following part, since it does not perform volume comparison and the datasets used have a wide range of volumes between 0 and 10^6. We use stochastic gradient descent (SGD) as optimizer with

learning rate of 0.321 for autoencoders and Adam with learning rate of 0.001 for feed-forward networks. We train the autoencoder for 10 epochs and the feed-forward networks for 30 epochs. The results can be found in Table 3.

The main observation from this set of experiments is that it is hard to predict volume when the dataset contains polytopes with a wide range of volumes. Even when we use Dataset C, which only consists of cubes, scores are still lower than 0.5 due to the wide range of the volume values. However, results are better compared to results from other datasets as the range of the volume is relatively limited. We also tried to train the models without volume normalization, but the models failed to learn.

Using the insights gained from the first set of experiments, we design a new experiment. In addition to volume prediction, we consider volume comparison as a related problem and we also consider the two problems simultaneously (see Sect. 2). In terms of data, in dataset D we restrict to a specific family of polytopes, with a smaller range of volume values (see Sect. 3).

The modular framework performs better with this new experimentation setup. The coupled model `V_F_APC_PC` has better comparison accuracy compared to `V_F_APC_C` due to the effect of volume prediction. Both `V_F_APC_PC` and `V_F_APC_P` give bad results for volume prediction. When normalization is removed from input data, although volume prediction improves, there is no improvement for volume comparison.

End-to-End. We also experiment with end-to-end network architectures in addition to our modular model with the second experimentation setup.

Fixed dimensional models obtain decent performance in accuracy, but volume prediction is not successful with normalization. An interesting observation is that volume prediction is worse in `F_L_E2E_PC` compared to `F_L_E2E_P`. However, both of them are still below zero which means none of them succeed in volume prediction. Another interesting situation we observe is that `F_L_E2E_PC` has slightly better accuracy for comparison than `F_L_E2E_C`. This becomes even more obvious when we change the normalization multiplier to 0.08 and 1 and leave other parameters such as optimizer and learning rate unchanged. For unnormalized data, `F_L_E2E_PC` reaches 0.9 accuracy and 0.97 volume score. This allows us to assume that the more the model learns from volume prediction the more accurate it will get on comparison. The accuracy and volume scores make it a feasible model for such problems when limiting the dimension is an option.

`F_P_E2E_PC` and its variations are the models with the best performance regardless of normalization. Again, the coupled model `F_P_E2E_PC` has slightly better accuracy compared to the comparison only model `F_P_E2E_C`. However, this time `F_L_E2E_PC` has a volume score above 0, which means that it learns about volume as well (even though not enough). The previous observation, that learning from volume makes comparison better, is supported by the high score of `F_P_E2E_P`. `F_P_E2E_PC` produces close results to `F_P_E2E_P` with different normalization multipliers. The overall performance of the model also improves when there is no normalization on volume. However, an interesting point to consider is

that F_P_E2E_P produce good results even with normalization. This model may be a good solution when the data is within the limits.

The training of V_L_E2E_PC includes an additional encoder, but this time the autoencoder works in a way similar to word embeddings in natural language processing (NLP). We used an autoencoder trained independently to encode our vertices in order to fix their dimension. It is trained for 30 epochs with Adam as optimizer and MSE as loss function. The rest of the model is trained similarly to F_L_E2E_PC as dimensions are fixed now. The reason of such training method instead of a complete end-to-end is that we could not find a way to train such model with LSTM.

V_L_E2E_PC and V_L_E2E_C offer acceptable performance for the comparison problem. However the results for volume are not good for either V_L_E2E_P or V_L_E2E_PC. The results do not seem to change much unless normalization is removed. When normalization is removed V_L_E2E_PC has a decrease in its comparison accuracy and increase in volume score. However, these are not especially good results.

4.1 Comparison of the Models

We propose four different approaches to solve volume prediction and volume comparison for polytopes.

We can divide our models into two groups based on their performances. The F_L_E2E_PC and F_P_E2E_PC models produce good results with over 0.90 accuracy and over 0.97 prediction score. On the other hand, V_L_E2E_PC and V_F_APC_PC models produce relatively worse results with around 0.80 accuracy and 0.7 prediction score. One key difference between these two groups is their input data format. Better results come when we fix the data size instead of trying to keep it variable. It is also possible that autoencoders do not learn a good feature set for these problems.

A general observation we can make is that almost all models produce better comparison accuracy and worse R^2 score when both of the problems are combined in a model (coupled models). The only exception is V_L_E2E_PC. We conclude that the two problems can affect each other, improving comparison while learning to predict volumes.

Another interesting result is the effect of volume normalization on all of the models. In contrast to our initial failure without normalization, our second set of experiments without normalization on volume produces better results. Training with lower normalization multiplier produces better results overall and the best results came when normalization was removed by setting the multiplier to 1. All the models can have significant increase in their R^2 scores due to this.

Table 3. Results from experiments with different model, normalization, encoding data, prediction data combinations. We will name the models in the following way: the first part declares if the dimension is fixed or variable by `F` or `V` respectively,the second part is `F` if input is flatened, `L` if a list of lists, and `P` if completely padded, the third part declares the training scheme by `E2E` or `APC` and the fourth part is `C` for volume comparison, `P` volume prediction or `PC` for a coupled model.

Model	Normalization	Enc. Data	Pred. Data	Accuracy	Volume Q	Volume P
V_F_APC_P+NN 2 Layer	0.000001	A	A	N/A	0.10	N/A
V_F_APC_P+NN 2 Layer	0.000001	A	B	N/A	0.40	N/A
V_F_APC_P+NN 2 Layer	0.000001	A	C	N/A	−2.91	N/A
V_F_APC_P+NN 2 Layer	0.000001	B	B	N/A	0.15	N/A
V_F_APC_P+NN 2 Layer	0.000001	B	C	N/A	−0.01	N/A
V_F_APC_P+NN 2 Layer	0.000001	C	C	N/A	0.35	N/A
V_F_APC_P+NN 4 Layer	0.000001	A	A	N/A	0.11	N/A
V_F_APC_P+NN 4 Layer	0.000001	A	B	N/A	0.42	N/A
V_F_APC_P+NN 4 Layer	0.000001	A	C	N/A	0.22	N/A
V_F_APC_P+NN 4 Layer	0.000001	B	B	N/A	0.30	N/A
V_F_APC_P+NN 4 Layer	0.000001	B	C	N/A	−0.11	N/A
V_F_APC_P+NN 4 Layer	0.000001	C	C	N/A	0.30	N/A
V_F_APC_P+RF(100,5)	0.000001	A	A	N/A	0.05	N/A
V_F_APC_P+RF(100,5)	0.000001	A	B	N/A	0.15	N/A
V_F_APC_P+RF(100,5)	0.000001	A	C	N/A	0.39	N/A
V_F_APC_P+RF(100,5)	0.000001	B	B	N/A	0.21	N/A
V_F_APC_P+RF(100,5)	0.000001	B	C	N/A	0.38	N/A
V_F_APC_P+RF(100,5)	0.000001	C	C	N/A	0.36	N/A
V_F_APC_P+RF(100,15)	0.000001	A	A	N/A	0.16	N/A
V_F_APC_P+RF(100,15)	0.000001	A	B	N/A	0.40	N/A
V_F_APC_P+RF(100,15)	0.000001	A	C	N/A	0.49	N/A
V_F_APC_P+RF(100,15)	0.000001	B	B	N/A	0.45	N/A
V_F_APC_P+RF(100,15)	0.000001	B	C	N/A	0.42	N/A
V_F_APC_P+RF(100,15)	0.000001	C	C	N/A	0.46	N/A
F_L_E2E_PC	0.0001	N/A	D	0.8624	−42.48	−43.58
F_L_E2E_C	0.0001	N/A	D	0.8401	N/A	N/A
F_L_E2E_P	0.0001	N/A	D	N/A	−0.1268	−0.0089
F_P_E2E_PC	0.0001	N/A	D	0.9266	0.1986	0.1419
F_P_E2E_C	0.0001	N/A	D	0.9106	N/A	N/A
F_P_E2E_P	0.0001	N/A	D	N/A	0.9785	0.9925
V_L_E2E_PC	0.0001	D	D	0.8510	−4.5031	−8.0108
V_L_E2E_C	0.0001	D	D	8559	N/A	N/A
V_L_E2E_P	0.0001	D	D	N/A	−0.5204	−0.0005
V_F_APC_PC	0.0001	D	D	0.8152	0.1260	−0.5238
V_F_APC_C	0.0001	D	D	0.7849	N/A	N/A
V_F_APC_P	0.0001	D	D	N/A	−0.0301	−0.4150
F_L_E2E_PC	0.008	N/A	D	0.7957	0.0163	0.0115
F_L_E2E_PC	1	N/A	D	0.9084	0.9774	0.9975
F_P_E2E_PC	0.008	N/A	D	0.9111	0.9269	0.9547
F_P_E2E_PC	1	N/A	D	0.9631	0.9980	0.9982
V_L_E2E_PC	0.008	N/A	D	0.8223	−0.1644	−0.2701
V_L_E2E_PC	1	N/A	D	0.7226	0.7013	0.7135
V_F_APC_PC	1	D	D	0.8277	0.5857	0.7754

5 Conclusion

In this paper we propose a framework allowing the use of machine learning for geometric and algebraic problems. The framework has similarities to models developed for other sequential problems employing RNNs, and in particular autoencoders.

We focus on two volume related problems, namely the prediction of polytope volume and the comparison of volumes of a pair of polytopes. The choice of problems is convenient since it provides naturally a regression problem and a related classification problem.

One of the important differences between the problems and solutions in the literature and the ones presented in this work is the effect of data normalization. In general, data normalization is essential for the success of most machine learning models. On the contrary, for the volume prediction problem and the proposed models, data normalization is catastrophic. If the range of volumes in the training data is too wide and the dataset not large enough, then the model fails to train due to lack of normalization. Nevertheless, as shown in our experiments, if the range is not too wide, unnormalized training data provide the best results.

To the best of our knowledge, this is the first attempt to employ machine learning in this type of problem. Our goal was to show that it is possible to train a model with input of two variable dimensions in a meaningful way. Although we are far from the ideal solution, the results presented here are very encouraging. We detect some important parameters that differentiate the problem from classical machine learning problems and we show how to overcome the issues raised by these differences.

Naturally, we will also continue testing our better performing models with different families of polytopes and more diverse datasets. Such experiments will provide a better understanding on how to solve geometric and algebraic problems with machine learning.

Acknowledgements. This work was supported by the project 117E501 under the program 3001 of the Scientific and Technological Research Council of Turkey.

References

1. Calès, L., Chalkis, A., Emiris, I.Z., Fisikopoulos, V.: Practical volume computation of structured convex bodies, and an application to modeling portfolio dependencies and financial crises. In: Speckmann, B., Tóth, C.D. (eds.) 34th International Symposium on Computational Geometry, SoCG 2018, 11–14 June 2018, Budapest, Hungary. LIPIcs, vol. 99, pp. 19:1–19:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2018). https://doi.org/10.4230/LIPIcs.SoCG.2018.19
2. Cho, K., et al.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1724–1734. Association for Computational Linguistics (2014). https://doi.org/10.3115/v1/D14-1179, http://aclweb.org/anthology/D14-1179

3. Dai, A.M., Le, Q.V.: Semi-supervised sequence learning. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 28, pp. 3079–3087. Curran Associates, Inc. (2015). http://papers.nips.cc/paper/5949-semi-supervised-sequence-learning.pdf
4. Dyer, M.E., Frieze, A.M.: On the complexity of computing the volume of a polyhedron. SIAM J. Comput. **17**(5), 967–974 (1988). https://doi.org/10.1137/0217060
5. Emiris, I.Z., Fisikopoulos, V.: Efficient random-walk methods for approximating polytope volume. In: Proceedings of the Thirtieth Annual Symposium on Computational Geometry, SOCG 2014, pp. 318:318–318:327. ACM, New York (2014). https://doi.org/10.1145/2582112.2582133, http://doi.acm.org/10.1145/2582112.2582133
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition, pp. 770–778, June 2016. https://doi.org/10.1109/CVPR.2016.90
7. Hinton, G., et al.: Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Signal Process. Mag. **29**(6), 82–97 (2012)
8. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
9. Jaekel, U.: A monte carlo method for high-dimensional volume estimation and application to polytopes. In: Sato, M., Matsuoka, S., Sloot, P.M.A., van Albada, G.D., Dongarra, J.J. (eds.) Proceedings of the International Conference on Computational Science, ICCS 2011. Procedia Computer Science. Nanyang Technological University, Singapore, 1–3 June 2011, vol. 4, pp. 1403–1411. Elsevier (2011). https://doi.org/10.1016/j.procs.2011.04.151
10. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems, vol. 25, pp. 1097–1105. Curran Associates, Inc. (2012). http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
11. Paffenholzr, A.: Smooth reflexive lattice polytopes. https://polymake.org/polytopes/paffenholz/www/fano.html
12. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323**, 533 (1986). https://doi.org/10.1038/323533a0
13. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR abs/1409.1556 (2014)
14. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Proceeding of the NIPS. Montreal, CA (2014). http://arxiv.org/abs/1409.3215
15. Szegedy, C., et al.: Going deeper with convolutions. In: Computer Vision and Pattern Recognition (CVPR) (2015). http://arxiv.org/abs/1409.4842
16. The Sage Developers: SageMath, the Sage Mathematics Software System (Version 8.3.0) (2019). https://www.sagemath.org
17. Ziegler, G.M.: Lectures on polytopes. Springer, New York (1995). https://doi.org/10.1007/978-1-4613-8431-1