Towards a framework for incorporating data acquisition cost in predictive time series models

Adrian Stetco
University of Manchester
mihai.stetco@manchester.ac.uk

Razvan Mosincat
University of Bergen
Razvan.Mosincat@uib.no

Goran Nenadic
University of Manchester
g.nenadic@manchester.ac.uk

John Keane
University of Manchester
john.keane@manchester.ac.uk

ABSTRACT

In many real world applications of machine learning, the cost of acquiring measurements varies significantly. As businesses operate in resource-constrained environments, a question that arises is which combination of signals results in the most accurate models given a fixed budget. Conversely, if more accurate models are needed, what signals increase accuracy at lowest cost? This paper introduces a three-stage framework for integrating data acquisition cost into time-series applications based on signal ablation. Stage 1 constructs a time-series predictive model utilizing all relevant signals to assess the maximum accuracy achievable with an unlimited budget. Stage 2 randomizes each signal independently, computing the degradation in accuracy incurred on the underlying model. This process is achieved either by shuffling of its measurements or random sampling from signal's distribution. Utilizing the resulting estimated signals accuracy and their acquisition cost, Stage 3 solves the knapsack-like combinatorial optimization problem of picking the right combination of signals maximizing total accuracy given any fixed budget. The proposed framework is showcased on a synthetic time-series dataset which allows us to control the cost distribution in order to understand how the system works in different scenarios.

KEYWORDS

Time Series, Cost-efficient, Neural Networks, Knapsack, Combinatorial Optimization.

1 Introduction

From healthcare to finance and engineering, more data is being accumulated, filtered and analyzed, improving decision making across all domains. Increasingly, models are becoming multi-modal, making use of signals from disparate sources, varying widely in type, structure and value. Organizations making use of this data are operating within cost constraints and have to make trade-offs when it comes to model development and data collection budget.

The concept of cost comes in many ways, from monetary to the cost of time lost (e.g. downtime), cost of invasiveness and so on. In healthcare, for example, the cost of taking a blood sample is less than the cost of taking an MRI in terms of monetary cost, but arguably higher in terms of invasiveness. Condition monitoring of wind farms utilizes airborne and subaquatic drones to take image samples of offshore wind turbines. These images, complemented with other sensor data (e.g. electrical signals), can be fed into predictive models making real-time assessments about the structural integrity of the towers, blades or subsea cables. While measuring and transmitting electric signals is cheap, the monetary cost of sending robots offshore can be high. It is thus useful to have a framework for understanding how much accuracy is brought by these signals and how can the budget be allocated in such a way as to maximize prediction capability.

This paper introduces a three-stage framework for addressing the inclusion and optimization of data acquisition cost when considering predictive, time series models. It starts by investigating what the potential maximum accuracy is, considering all relevant signals used together. Stage 2 perturbs (ablates) the signals one by one, computing the drop in accuracy due to each, which is essentially assessing their importance according to the underlying model. The final stage selects the best combination of signals that maximize accuracy for any given budget. The framework is showcased on a synthetic time-series dataset formed by sine functions. A set of independent sine functions differing in phase and frequency is used to generate a dependent variable, a relationship which is captured using a three-layer, feedforward neural network. As the target variable is a weighted sum of its inputs, this allows control of the weights and tests the framework's ability to uncover the importance of each signal.

The paper is organized as follows: we discuss work on cost frameworks for machine learning and relevant optimization methods in *Section 2*; the three-stage framework for incorporating acquisition cost in time series applications is described in *Section 3*; *Section 4* presents the experimental results on a synthetic time-series dataset; *Section 5* concludes the paper and discusses future work.

2 Background

Cost is an essential, yet often overlooked facet of machine learning application in real-world scenarios. There is a multitude of types of costs arising in model development and deployment such as the cost of labelling the data for supervised learning, the cost of misclassification errors, cost of tests, cost of intervention, cost of data acquisition, cost of computation, human-computer interaction cost, cost of instability etc. [1]. While the cost of misclassification has been addressed extensively in the literature - termed cost-sensitive learning [2] - a framework which integrates all types of costs is currently lacking.

The cost of data acquisition is an important cost to be modelled as its application space is potentially immense. Consider, for example, the healthcare domain where electronic patient records document illness progress. The illness trajectory prediction becomes more accurate with more health tests, but they come with wide-ranging costs and thus could be constrained by the available budget. Healthcare providers would benefit from a framework that takes a budget as an input and decides which test to perform in order to maximize health state prediction accuracy. In another context, offshore wind turbines can benefit from the integration of various fixed or mobile sensors. Fixed sensors installed in a turbine can sample data with a relatively low cost, while mobile sensors (such as aerial or subaquatic drones) need to visit the site to take measurements and images, which is a costly and potentially hazardous procedure. In this scenario, wind farm operators would like to know how much increase in diagnostic accuracy could be obtained by sending robots, and thus incurring the cost and increasing the safety risks.

3 Methodology

In our framework, optimization is employed in all three stages (Figure 1). Firstly, the system is optimized to find the minima of the predictive model's loss function using all relevant signals (Stage 1). Secondly, each signal is randomized, by either having its values shuffled or by sampling from a distribution, forcing the model to break the association between a signal and the target variable (Stage 2); a similar approach was introduced in [3] and expanded in [4] and [5]. Estimated accuracy is computed for each shuffled signal before the process goes to the next phase. It is important to note that this computation does not involve re-training the underlying model. The inputs to the final stage (Stage 3) are the cost of acquiring the samples and their estimated accuracy computed at Stage 2. The budget can be specified either as a scalar or a vector (e.g. in future applications where multiple constraints are required). Solving Stage 3 involves a combinatorial optimization problem. The standard one-dimensional Knapsack problem asks the following: given a set of items, each of different weights and values, what is the selection that maximizes total value given a weight constraint? The Multi-Dimensional Knapsack problem (MKP) is described by the following optimization problem [6]:

$$\text{maximize } z = \sum_{j=1}^{N} a_j x_j$$

$$\text{s.t. } \sum_{j=1}^{N} c_{ij} x_j \le b_i, i \in \{1,2,\dots,m\} \quad (1)$$

$$x_j \in \{0, 1\}, \quad j \in \{1,2,\dots,N\} \quad (2)$$

In our framework, we have *N* as the number of signals, *m* is the number of constraints, *a* is the unit of accuracy, *c* is the cost associated with sampling a signal, *b* represents the budget and *z* represents the model's accuracy. It is important to note that accuracy here designates one of the potentially many ways to capture the performance of a predictive system (others include TP, F-Score, RMSE or MAE, etc.). We further contrast and compare an exact dynamic-programming solution (Table 2) to a greedy but faster approach which does not guarantee global optimality (Table 1).

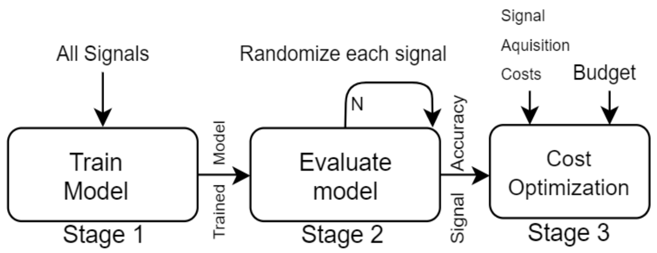

Figure 1. High level framework overview

Stage 1 of our framework consists of a classification scenario where we are tasked with learning a model that takes as an input a vector of data *V*, and outputs a target, categorical variable *T*. Exploring regression schemes where the target variable is continuous will be investigated in the future. The vector of measurements $V_i = \{S_{1i}, \dots, S_{Ni}\}$ is sampled from the feature set *S* of size *N*, each having an associated scalar T_i. It is assumed that the set of features *S* is relevant to predict *T*. If this is not the case, the set of models we train might overfit on noisy data, skewing the results of the analysis. Statistical tools for feature selection such as filters, wrappers or embedded models can be utilized at this step.

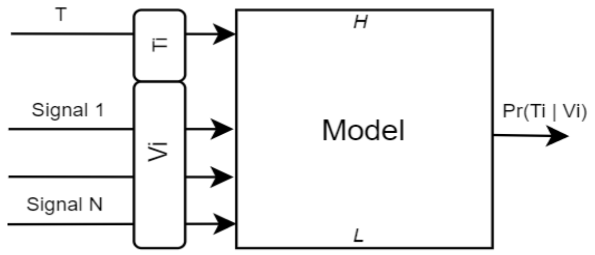

Figure 2. Stage 1: Fitting a model using all the relevant signals

Using a set of fixed hyperparameters *H*, the model $\mathscr{M}$ is trained in a supervised fashion being presented with pairs of the type $[V_i, T_i]$, iteratively minimizing a predefined probabilistic loss function *L* such as cross-entropy (Figure 2). In our case, where the model is a neural network, errors in predicting T_i from V_i are back propagated resulting in changes to the model's parameters. Once the parameters stop changing due to *L* not minimizing further or we have reached the maximum number of predefined epochs, the learning process ends. The training set accuracy of model $\mathscr{M}$ represents the maximum accuracy we can get by using all relevant signals.

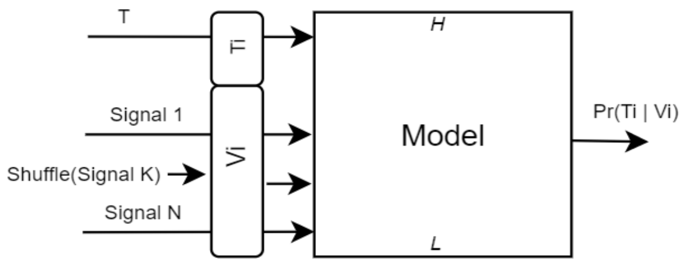

Figure 3. Stage 2: Every signal *k* is shuffled independently with the model evaluated using cross-validation. Resulting accuracy is saved

We now describe Stage 2. As we have a set of *N* signals (from which vectors *V* are sampled), we have 2^*N*-1 potential subsets of signals to fit our model to, which represents a large space to explore. Besides this global optimal solution of O(2^*N*) there is an O(*N*) greedy approach that we have chosen to apply in our experiments. In this scenario, we keep the number of signals fixed to *N* and explore how much each contributes to the accuracy of the model $\mathscr{M}$. One way of achieving this is by repeatedly predicting the test set using the model $\mathscr{M}$ with one of the signals either shuffled or made random noise.

The difference between the accuracy of $\mathscr{M}$ (the model that uses all signals) and the accuracy obtained by shuffling or randomizing signal *k* estimates the impact that this signal has on the total model accuracy and we denote it by A_k. For some machine learning models such as the neural networks used in our experiments, the model training time complexity is not well understood; however, the prediction time complexity is linear in terms of the weights of the network. Figure 3 illustrates how Stage 2 works, with the model being presented with vector V_i containing one component shuffled. The process is repeated for each signal *k* from 1 to *n*, resulting in the vector *A*={$A_1, \dots, A_N$} which accounts for the accuracy of each feature relative to total accuracy.

Stage 3 receives a vector of costs C= {$C_1, \dots, C_N$}, the vector of accuracies *A* computed at Stage 2, and the budget *b*. It then answers which subset of signals fits the budget and maximizes the predictive accuracy. To be additive, these costs need to be in the same unit and to be known when the procedure starts. As an example, the condition monitoring tools attached to drive-trains in a wind turbine are powered by electricity which can be translated into costs per kWh of operation. Operating a drone which takes pictures of blades that feed into a neural network for image detection requires the additional cost of having an engineer operate it (cost of employment).

For permutation feature importance, we used a library available in the Scikit-learn library [7].

For executing Stage 3, Table 1 presents a greedy algorithm that is fast to run (has linear time complexity) but may result in a suboptimal solution. It involves sorting the features based on the accuracy they provide relative to their cost, and adding them one by one to the solution vector as long as it does not overfill the predefined budget. The solution is greedy because it optimizes for the current best feature at every step and cannot look ahead for alternatives. The solution vector tells us which signal to keep and which to discard.

In contrast, the dynamic programming version presented in Table 2 selects the optimal solution but requires a matrix of size *N·(Budget+1)* to store intermediary solutions to its subproblems, where we recall that *n* is the number of signals. At this stage, users will decide whether they want the optimal but computationally demanding version or the fast but suboptimal one, or any flavour in-between [6]. In the following section we experiment with both the greedy and the dynamic solutions for the Knapsack optimization problem.

Algorithm 1: Greedy Knapsack1D optimization

```
Input:  Cost = {$C_1 \dots C_n$ }, Accuracy = {$A_1 \dots A_n$}, Budget = b
Output: SolutionVector
1.  AccuracyPerCost = Accuracy/Cost
2.  Index = Index(SortDescending(AccuracyPerCost))
3.  SolutionVector=0; SolutionCost=0
4.  For i from 1 to n:
5.      If (SolutionCost+Cost[Index[i]] > budget):
6.          break
7.      else:
8.          SolutionCost+=Cost[Index[i]]
9.          SolutionVector [Index[i]] = 1
10. end For
11. Return SolutionVector
```

Table 1. Greedy Knapsack1D optimization pseudocode

4 Experimental Results

In this section we showcase the proposed framework and we exemplify the way the target variable is generated using a given set of signals. Knowing beforehand how much influence a signal has on the target variable, we can then show precisely how well the shuffling process helps in uncovering the real impact the feature has during Stage 2 of the framework (Figure 3). In this use case, we generate a set of *N*=20 signals of *size*=10000 made of sines generated using the equation:

$$S_k = a * \sin(b * (x + c)) + d \qquad (3)$$

where a, c, d are sampled independently from normal distributions $N(0,1)$ for each S_k. The constant $b = 2\pi/g$ has g sampled from the same normal distribution, and$x \in [0, X]$.

Algorithm 2: Dynamic Knapsack1D optimization

Input: Cost = $\{C_1 \ldots C_n\}$, Accuracy = $\{A_1 \ldots, A_n\}$, Budget = b
Output: SolutionVector

```
1.  Mat = Zeros[$N$, b+1]
2.  For i from 0 to $N$:
3.    For j from 0 to b+1:
4.      If ( i ==0 and j> C[i]):
5.          Mat[i,j] = A[i]
6.      else if (i>0):
7.          Max_without_current = Mat[i-1, j]
8.          Max_with_current = 0
9.          If (j > C[i]):
10.            Max_with_current = A[i] + Mat[(i-1), j-C[i]]
11.            Mat[i,j] = max(Max_with_current,
           Max_without_current)
12.     end for
13. end for
14. i=$N$; j=b
15. While (i>=0 and j>=0):
16.   If (Mat[i,j] != Mat[i-1,j]:
17.       SolutionVector[i] =  1
18.       j= j- C[i]
19.       i=i-1
20.   else:
21.       i = i-1
22. Return SolutionVector
```

Table 2. Dynamic Knapsack1D optimization pseudocode

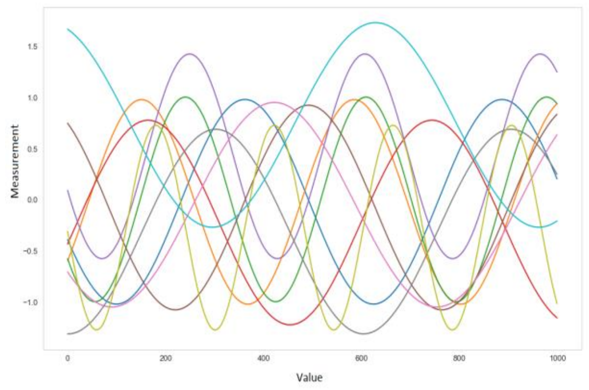

Figure 4. Ten independently generated signals using eq. (3)

One scenario assumes that each feature has the same cost of acquisition, $C(S_k)$ = 500 units for $\forall k \in [0, N]$, summing up to a total cost of 500*20=10000 units, shown in blue in Figure 7. In contrast, we investigate a cost distribution arising from a power law distribution with results marked in red in Figure 7. The resulting matrix contains on each row one of the generated sines. The sines shown in Figure 4 together with the rest of the generated ones in our dataset (N=20 in total) are summed together according to Equation (4), where W^{tr} represents the row vector of a set of predefined mixing weights. We use these weights to control the contribution of a particular sine into the target variable T that we are generating using Equation (4). For our experiments, these weights are sampled from a power law distribution.

$$\mathrm{T} = \mathrm{W}^{\mathrm{tr}} \cdot \mathrm{S} = \sum_{\mathrm{k}=0}^{\mathrm{N}} \mathrm{W}_k^{\mathrm{tr}} * S_k \qquad (4)$$

As we would like to solve a classification problem, we discretize the target variable T into three classes (or bins) - function of their value as shown in Figure 5.

We further prepare a feed-forward neural network containing three dense layers. The choice of the model, its hyperparameters and architecture should match the complexity of the data. Neural networks have been shown to be universal functions approximators [8]. The particular hyperparameters we chose are 2*N neurons for the first dense layer, followed by N neurons for the second dense layer and $\frac{N}{2}$ neurons in the third (note N=20 in our setting). Batch Normalization layers are used to stabilize and increase the rate of convergence of the model. We use Rectifed Linear Units (RELU) as the activation function and the ADAM optimizer [9]. We set the batch size to 256 and we train our model for 512 epochs. This model is trained on 80% of the training data, obtaining a total accuracy *TR_acc.*

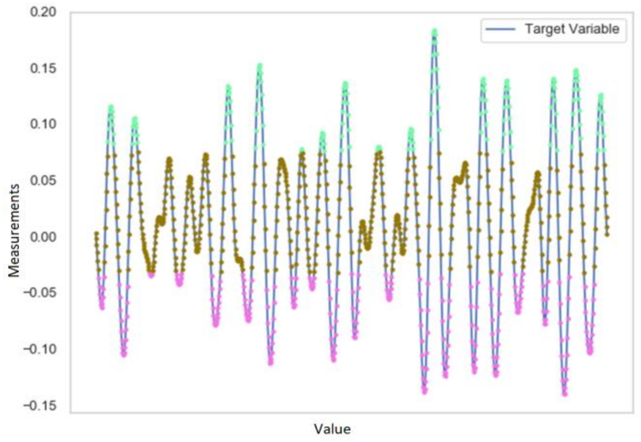

Figure 5. Target variable T, in this case a uniformly weighted sum of sines. Colours represent the three classes to be predicted by our model

When it comes to Stage 2, the same model is used with shuffled (or noisy data) with its prediction being evaluated both on the training data and the test (hold-out) data. This is because we want to evaluate not only how much the model depends on the data it sees during the training but also how well the model generalizes using the given signal. Here, besides shuffling, one could also find the distribution of the signal with maximum likelihood estimation and then sample randomly from that distribution.

In contrast to Molnar [5] who used either the training or the testing set estimates of accuracy for each feature S_k, here we propose the following scheme:

$$A_k \quad = \quad \text{TR\_acc} - \alpha * TR\_\text{acc}\_S_k + (1-\alpha) * TS\_\text{acc}\_S_k \qquad (5)$$

where α (in our case α=0.5) controls how important the training set estimate is compared to the testing set. $TR\_acc\_S_k$ is the training accuracy when the model uses shuffled feature S_k to predict the training data, while similarly, $TS\_acc\_S_k$ is the accuracy of the model on the testing set.

Figure 6 shows in orange the original mixing weights used to generate the target variable T and in blue the A_k, normalized to the sum of 1 for comparison. This confirms that the model learned correctly how the input features contribute to the target variable. In any real-world situation, we would not know the real underlying mixture process that gives rise to the target variable T, and we would have to rely on the estimates produced by shuffling our signals as described at Stage 2.

Using the discovered feature importance A and the cost function C_u (of uniform costs) and C_p (power law costs), we can compute the best accuracy for any given budget, up to the total of 10000 units where we can utilize all the available features. In Figure. 6 one can observe the accuracy taking higher leaps with less budget, which happens because the framework is considering the best features first. While the Greedy and Dynamic solutions perform equally well for signals with equal costs (in blue), when it comes to power law distributed costs (red), the Dynamic solution achieves better scores (Figure 7).

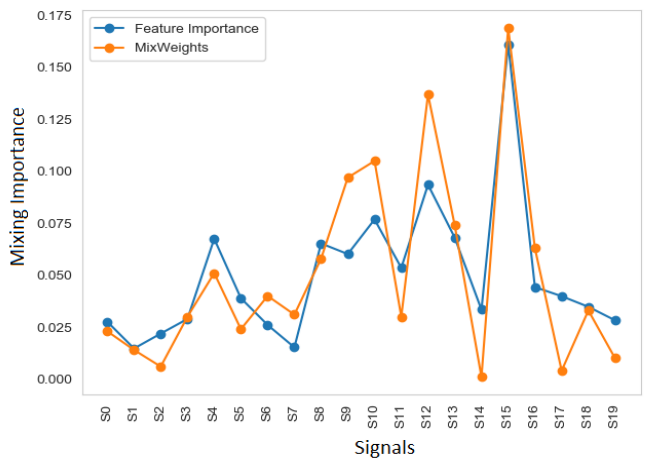

Figure 6. Orange shows the original mixing weights for equation (4) while blue shows the estimated importance of each signal $A(S_k)$ resulting from running Stage 2. The shuffling process identifies signal's importance in our classification task.

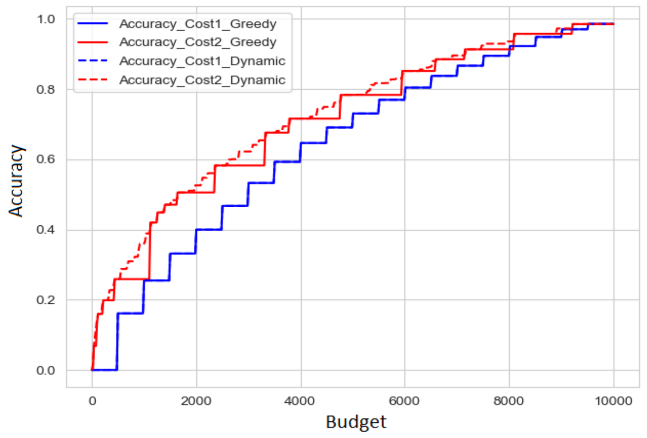

Figure 7. Knapsack solution for each budget up to 10000 units which is the cost of acquiring all the features and getting close to 100% accuracy. Dotted lines (Dynamic Knapsack solutions) are contrasted with straight lines (greedy Knapsack) in two cost scenarios: uniform (blue) and power law (red).

5 Discussion and Conclusion

This paper introduced a novel framework for incorporating feature acquisition cost for time-series applications operating with limited budgets. The approach utilized in our data experiments trades global optimality for speed by adopting an O(N) ablation strategy at Step 2. Where this technique may offer sub-optimal results is when joint signal accuracy may be higher than the sum of independent signal accuracy. Relevant signals should be selected beforehand using expert knowledge and/or statistical feature selection procedures. While brute-force signal combination - O(2^N) - can yield optimal results, other solution space exploration techniques can be utilized. While this work explored cost optimization in a classification environment, it would be interesting to see how it performs in regressions and on real datasets. Further, an interesting area to investigate would be the MKP problem with multiple budget constraints and different kinds of costs.

ACKNOWLEDGEMENTS

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) through grants EP/P009743/1 and EP/L021463/1. The authors are thankful to all colleagues and partners on the HOME Offshore project (http://homeoffshore.org/). R.M. was supported by a Trond Mohn Foundation grant.

REFERENCES

[1] P. D. Turney, "Types of Cost in Inductive Concept Learning," in *Proceedings of the Cost-Sensitive Learning Workshop at the 17th ICML-2000 Conference*, 2000.

[2] S. H. Khan, M. Hayat, M. Bennamoun, F. A. Sohel, and R. Togneri, "Cost-sensitive learning of deep feature representations from imbalanced data," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 29, no. 8, pp.

3573–3587, Aug. 2018.

[3] L. Breiman, “Random Forests,” *Statistics (Ber).*, vol. 45, no. 1, pp. 1–33, 2001.

[4] A. Fisher, C. Rudin, and F. Dominici, “All Models are Wrong, but Many are Useful: Learning a Variable’s Importance by Studying an Entire Class of Prediction Models Simultaneously,” *J. Mach. Learn. Res.*, vol. 20, no. 177, pp. 1–81, Jan. 2018.

[5] Christoph Molnar, “Interpretable Machine Learning: A Guide for Making Black Box Models Explainable,” 2019. [Online]. Available: https://christophm.github.io/interpretable-ml-book/. [Accessed: 31-Jul-2019].

[6] A. Fréville, “The multidimensional 0-1 knapsack problem: An overview,” *European Journal of Operational Research*, vol. 155, no. 1. pp. 1–21, 16-May-2004.

[7] Scikit-learn, “Permutation feature importance,” *Scikit-learn*. [Online]. Available: https://scikit-learn.org/stable/modules/permutation_importance.html#id2.

[8] G. Cybenko, “Approximation by superpositions of a sigmoidal function,” *Math. Control. Signals Syst. Vol.*, vol. 2, pp. 303–314, 1989.

[9] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2014.