Smart Irrigation IoT Solution using Transfer Learning for Neural Networks

Ali Risheh¹, Amirmohammad Jalili² and Ehsan Nazerfard³

¹ Department of Computer Engineering, Amirkabir university, Tehran, Iran, ali.risheh@aut.ac.ir
² Department of Computer Engineering, Amirkabir university, Tehran, Iran, amirm.jalili@aut.ac.ir
³ Department of Computer Engineering, Amirkabir university, Tehran, Iran, nazerfard@aut.ac.ir

Abstract— In this paper we develop a reliable system for smart irrigation of greenhouses using artificial neural networks, and an IoT architecture. Our solution uses four sensors in different layers of soil to predict future moisture. Using a dataset we collected by running experiments on different soils, we show high performance of neural networks compared to existing alternative method of support vector regression. To reduce the processing power of neural network for the IoT edge devices, we propose using transfer learning. Transfer learning also speeds up training performance with small amount of training data, and allows integrating climate sensors to a pre-trained model, which are the other two challenges of smart irrigation of greenhouses. Our proposed IoT architecture shows a complete solution for smart irrigation.

Index Terms — Automation Irrigation System, IoT, Machine Learning, Transfer Learning

I. INTRODUCTION

A. Motivation

Smart irrigation is a major revolution in agriculture to replace costly and error prone human labor. In fact, smart irrigation is the most important part of smart agriculture due to limited water resources and importance of irrigation process on product quality. A particular sector of the agriculture industry which benefits most from smart irrigation is the greenhouse industry; this is because greenhouse crops are more sensitive to environment parameters, especially soil moisture, requiring real-time control and adjustment of irrigation faucets. Consequently, human operated irrigation of greenhouses is very costly and any human error results in large damage to the products. While early solutions to reduce human cost are based on remote irrigation keeping human control in the loop, smart irrigation aims at providing a human-free automatic solution. However, a smart irrigation solution that addresses the practical challenges of the greenhouse industry is not currently available.

Reliability is the main practical challenge of any engineering solution to smart irrigation for greenhouses. In practice, the soil is divided into multiple layers, often 4 or 5 layers each with 10 to 15 cm thickness in our experiments. The moisture level of deepest layer should be constantly within 5% of its accepted level for a quality product, and any 20% deviation from the accepted level results in the crop death.

Internet of Things (IoT) and machine learning are two technology advancements that enable smart irrigation. IoT enables system data collection via sensors and remote system control via actuators [1], [2], [5]. In the context of smart irrigation, an IoT system collects moisture level in all soil layers, and also other environmental variables such as temperature and LDR (light dependent resistor) to automatically control the irrigation faucets using IoT communication technologies. Machine learning enables accurate prediction of variables to control a system. For smart irrigation, machine learning allows prediction of soil moisture and appropriate control of the faucets. Artificial neural network models allow for high accurate prediction to satisfy greenhouse reliability constraints.

There are three other practical challenges in applying machine learning in an IoT solution for greenhouses: limited data collection period (often a week or less), low processing power at IoT edge devices, and flexibility in adopting new environmental sensors. The first results in limited samples while training an artificial neural network from scratch requires high amount of training data. The last one requires flexibility of the machine learning (and the IoT solution) to add new sensors. Transfer learning can address both of the above challenges.

Domain adaptation in transfer learning allows adapting an existing model for other environments (different soil type, different type of greenhouse, etc) to a new environment using small training samples. It also reduces the training processing power needed. Moreover, domain extension in transfer learning allows adding new sensors by extending the structure of the existing artificial neural network. There is a small literature on machine learning and IoT for smart irrigation. [4] uses recurrent neural network to predict required irrigation period. However, the model requires ~ 1000 hours as the IoT solution uses a single sensor on the top soil layer. [5] uses a different approach by trying to replicate farmer’s behavior by directly observing that. Of course, the farmer’s behavior is not necessarily the best approach for irrigation and an IoT solution using sensory data can provide superior results. [6] advocates adding temperature sensors to moisture sensors and runs a regression for prediction. We also add environmental sensors to our moisture data but use an artificial neural network for better prediction. IoT architecture for supporting smart irrigation is presented in [7] using Arduino and conditional states to control the faucets. However, the solution does not

1In opposite to farmlands, human labors often do not live in proximity of the greenhouses.
use full potential of embedded systems in state of the art IoT solutions which we will discuss.

B. Contribution

Our goal is an automatic reliable and flexible irrigation for greenhouses discussed in the previous section. We provide a smart irrigation using an IoT based system plus a neural network model. We rely on moisture sensors in all soil levels (4 sensors) to achieve high accuracy in prediction, mean squared error (MSE) < 0.05. Our solution allows fast adaptation to new environments and flexibility in adding new environmental sensors using transfer learning.

To this end, we take the following steps (The rest of the paper is organized as following):

- We first collect data from two different soil types and show they have different characteristics. (section II. Data Description)
- Next, we train an artificial neural network for each soil separately that achieves our 0.05 MSE target. We compare the results with a support vector regression (SVR) and show SVR requires higher training data to achieve the same MSE. (section III. Training)
- Next, we illustrate how to deploy transfer learning to use the neural network trained for soil one, for training a neural network for soil two using few of its samples. (section IV. Transfer Learning)
- Next, a transfer learning approach is presented to add new sensors. Particularly this approach allows the system to adapt itself without any disruption in its performance. (section V. Climate-Smart Agriculture)
- Finally, an IoT architecture will be introduced including end point sensors and actuators (faucet), a coordinator, a cloud server, and a communication protocol. The system is designed to be flexible for adding new sensors. (section VI. IoT Solution)

II. DATA DESCRIPTION

In this section, datasets which were gathered from two different pots, denoted by Soil1 and Soil2, will be analyzed. Four soil moisture sensors are installed at a depth of 10cm from each other in both soils. We denote these sensors by Moisture0 (in first layer), to Moisture 3. To better highlight the discrepancy, none of soils were irrigated during the time intervals. The data is stored once every 2-3 minutes. The total amount of data collected is 26,000 points. Our datasets are available online publicly, with updates of more samples at [8]. Table I is an example of our dataset.

| year | month | day | hour | minute | second |
|------|-------|-----|------|--------|--------|
| 2020 | 3     | 11  | 14   | 44     | 39     |
| Moisture0 | Moisture1 | Moisture2 | Moisture3 |
| 0.59 | 0.63 | 0.51 | 0.45 |

Table I: The Moisture0 in table is in the top layer of soil and the others increase in depth, respectively.

In order to compare soils (Soil1 and Soil2) and show their differences, we first plot the moisture level for each soil (Figure 1). The first layer of Soil1 loses water slowly, whereas in Soil2 the water remains longer in the first layer. We also see water passes second and third layers faster in Soil1, but in Soil2 the moisture slowly reaches the last layer. These show clearly that the rate of water absorption varies between the two pots. Moreover, the absorption rate across different layers of each soil are different (similar observations are made in [9]).

III. TRAINING

In this section, we train different models to predict the moisture in the deepest layer in the future from current moisture in all layers using two supervised learning methods: support vector regression and neural network. We denote the period of collecting data with \( T \) (2-3 minutes). The goal is to predict the moisture of the deepest layer at \( 3T \) (6-9 minutes ahead) to avoid overflow. So inputs to the training model are \( m_0^1, m_1^1, m_2^1, m_3^1 \) and output of prediction is \( m_3^{3T} \), where \( m_i^j \) is the moisture of layer \( i \) in time \( j \). All models are trained by sum of errors, in addition, MSE is represented as another metric to better show the convergence. MSE is multiplied by 10 in all graphs to better represent the trend.

A. Support Vector Regression

In this section we train a Support Vector Regression (SVR) model. Due to its simplicity, the model does not need high processing power and this system can be used on network
edges specially in IoT communication technologies in which internet is not available. We will observe however the model requires large training samples for good performance. To address this shortcoming, we will study neural networks and transfer learning ideas in next sections.

First we rewrite the problem statement in machine learning language with SVR model [10]. The prediction model is a linear function $f$ of the following form

$$f(x) = w \cdot x + b \quad \text{with} \quad w \in R^4, b \in R \quad (1)$$

The training problem is then the following optimization problem

$$\min_{w \in R^4} \sum_{i=1}^{N} (y_i - w \cdot x_i - b) \quad (2)$$

We first train the model using Adam optimizer with 4000 samples in train set, 1000 samples in test set and 20 epochs. The model converges to local minimum after 20 epochs. The model and code are available in [11]. On convergence the model has the following test error which is in the acceptable range of 5% error for us.

- Sum of error: 5.1476
- MSE: 0.0298

While with 4k data the model is achieving good enough performance for our application, the training model is very sensitive to number of samples. This is while in practice the number of samples often much less than 4k tested here. To see this sensitivity, we run the following experiment: we start from the pre-trained model for Soil1, and test it with 1k data from Soil2 (transfer learning). The test errors are:

- Sum of error: 20.3211
- MSE: 0.4255

This 40% error in prediction is not acceptable in practice [12], therefore SVR can not be used. To address this issue, we first propose neural network for higher performance in prediction in section III-B. Second in Section IV we will discuss transfer learning which allows using a pre-trained model and adjust it with small samples to a new soil.

**B. Artificial Neural Network**

Artificial neural networks (ANNs) are flexible computing frameworks and universal approximators that can be applied to a wide range of time series forecasting problems with a high degree of accuracy [13]. Single hidden layer feed forward network is the most widely used model form for time series modeling and forecasting [14]. From the result of SVR prediction we expect a single hidden layer neural network should suffice for our purpose. Figure 3 represents our neural network. The model is characterized by a network of three layers of simple processing units connected by acyclic links. The relationship between the output $m_3^T$ and the inputs $(m_0^T, m_1^T, m_2^T, m_3^T)$ has the following mathematical representation:

$$m_3^T = w_0 + \sum_{j=1}^{q} w_{j} \cdot g \left( w_{0j} + \sum_{i=0}^{p} w_{ij} \cdot m_i^T \right)$$

where, $w$ are model parameters often called connection weights; $p = 3$ is the number of input nodes (starts from 0) and $q = 4$ is the number of hidden nodes. Next we define the activation function which is appropriate for this model. Since moisture never becomes negative, we use the Rectified Linear Unit (ReLU) function.

$$\text{Relu}(x) = \max(0, x)$$

We also define a dropout layer as a simple way to prevent neural networks from overfitting [15]. To this end, we choose a dropout rate to specifies the probability at which unit of the second layer is dropped. In Figure 3 the crossed unit is dropped in a training case.

Now we start training with same properties of training as previous part which is based on Adam optimizer using 4000 training data, 1000 test data with 20 epochs. Figure 4 shows the training error of the neural network. Compared to SVR in Figure 2 neural network converges faster.
Now we evaluate the test error for the neural network after 20 epochs:

| Sum of error | MSE           |
|--------------|---------------|
| 0.2306       | 8.4409e-05    |

The result has 3 order of magnitude higher accuracy compared to SVR and it does not do overfitting.

Neural networks are still sensitive to size of training data. For example with 480 samples (one day of sampling) the error results are the following which are above our reliability constraint.

| Sum of error | MSE           |
|--------------|---------------|
| 8.2498       | 0.1275        |

The second problem with neural networks is that they need higher processing power for training. Therefore, it is suitable for edge devices with high processing power. We address these issues and further improve the performance of neural networks with transfer learning in the next section.

IV. TRANSFER LEARNING

To address low performance of neural networks with small data and their high processing cost discussed in previous section, we use transfer learning that allows adopting the model from one soil to another soil while the two soils can have different distribution of underlying data.

Traditional data mining and machine learning algorithms make predictions on the future data using statistical models that are trained on previously collected labeled or unlabeled training data [16], [17], [18]. Semisupervised classification [19], [20], [21], [22] addresses the problem that the labeled data may be too few to build a good classifier, by making use of a large amount of unlabeled data and a small amount of labeled data. Variations of supervised and semisupervised learning for imperfect data sets have been studied; for example [23] have studied how to deal with the noisy class-label problems. Yang et al. considered cost sensitive learning [24] when additional tests can be made to future samples. Nevertheless, these methods assume that the distributions of the labeled and unlabeled data are the same. This is not our case as shown in Section II. Transfer learning allows the domains, tasks, and distributions used in training and testing to be different [25].

We begin using Soil2 dataset. Our pre-trained model (trained with 4000 training data of Soil1) with 1000 test data of Soil2 gives the following error values:

| Sum of error | MSE           |
|--------------|---------------|
| 2.2272       | 0.0124        |

Now we run two experiments. First we train a new model from scratch with 4000 training data of Soil2. Figure 5 shows the results. Second, we transfer the model of Soil1 to Soil2 in the following way. We start from the model of Soil1 (trained in Section III-B) and train with 4000 samples from Soil2. The results are in Figure 6.

We can compare the two experiments. First we compare their speeds of convergence which we define as difference in loss per epoch, depicted in Figure 7. The pre-trained model has a faster convergence. Second we compare the performance on initial points. Obviously the pre-trained model of experiment two has much better starting point performance. Third, we compare time to reach local minimum which is...
about 8 epochs for pre-trained model and 18 for experiment 1. Comparing these results shows the transfer learning is superior in performance and achieves our practical reliability constraints ($MSE < 0.05$) at all times for the system.

Next we run another experiment to assess the performance of transfer learning for our use case to compare the number of training data required by a new model. In each round, we add 50 samples to the number of training data of Soil2 and train a new model with previous conditions (epoch=20), then evaluate the 1000 test data of Soil2 and continue as long as the efficiency of the new model is the same as the pre-trained model with 4000 data of Soil1 (until sum of error = 2.2272). We have seen that a pre-trained model with 4,000 training data of Soil1, on 1000 test data of Soil2, had sum of error less than 3, this accuracy is possible with at least 2600 training data of Soil2 for a new model. This number of training data in our experiment indicates about 5 days of data collection to reach an acceptable accuracy. It should be noted that the main model is based on the moisture of different layers. Conditions must be provided to which new variables can be added, and it is usual that not much data is available from the added variable(s).

V. CLIMATE-SMART AGRICULTURE

The Climate-smart agriculture (CSA) concept emerged in 2010 as a response to the imminent threat of climate change. In the original Food and Agriculture Organization (FAO) document (FAO, 2010), the denition of CSA is: Agriculture that sustainably increases productivity, resilience (adaptation), reduces/removes greenhouse gases (GHG) and enhances achievement of national food security and development goals. CSA has become known for the so-called triple win, i.e., working simultaneously to achieve its three objectives (or pillars): adaptation, mitigation, and food security. CSA is also sometimes presented as a mechanism to achieve synergies between the three pillars in a context-specic manner. CSA aims to contribute to sustainable landscapes and food systems as well as to resilience, ecosystem services, and value chains. Since the CSA concept did not arise from the academic community, its underlying concepts were not aligned with existing scientic debates, on e.g., sustainability, food security, resilience, or agroecology. This should not preclude CSA from being analyzed rigorously [26]. The eld of CSA is very broad, we aim here to use climate variables to optimize irrigation, but this is in line with the concept of CSA because it aims at all three goals with the help of climate characteristics that will be discussed. As mentioned in the introduction, many variables (including temperature, light, etc) affect irrigation, our goal is to provide a way for each of these variables to contribute to the irrigation decision. It should be noted that the main model is based on the moisture of different layers. Conditions must be provided to which new variables can be added, and it is usual that not much data is available from the added variable(s).

Suppose a variable such as air temperature is added to the model, in which case the amount of surface evaporation from the soil is affected by this variable, as well as the moisture of the different layers. The most convenient way to make this variable effective is to include it in the model. Changes in the structure of the neural network require a new training process, in which case model, predictions may be hampered by a lack of data from the new variable. The right solution is to train a new model and replace it at the right time, in the continuation of this discussion, a suitable solution is provided for this purpose. Mathematical expression of this method will be with the help of deinitions in transfer learning concept. We assume we had a source domain $D_s$ and a target domain $D_t$.

$$D_s = \{t_1, t_2, t_3, \ldots, t_n\} \text{ and } D_t = \{X_0, t_1, t_2, \ldots, t_n\}$$

We have generated an artificial neural network (source model) for our $D_s$ and its data is complete and large. We make a copy of our source model with a new neuron in input layer and hidden layer (target model). The method is clear from Figure 9 we just need to find a solution that can replace the new model (target model) at the right time with the old model (source model). Now we start training the second model (new model) with new labeled data, but we use both models to predict final result based on linear regression with $Y_1$ (source model prediction) and $Y_2$ (target model prediction):

$$\alpha Y_1 + \beta Y_2 = Y_{ans}$$  \hspace{1cm} (3)
We initialize $\alpha$ and $\beta$ with 0.999 and 0.001 as we know our first model has been trained before. We will train our models base on $Y_{ans}$ with gradient descend optimization and the linear regression separately. In this section, we wanted to introduce a method so that we no longer need to provide a new system or model to add a new variable, this method helps each greenhouse to install a more accurate system according to its financial capacity. Our main method is to find the relationships between the soil layers, which will definitely be the safest way to predict soil moisture. With this method, you can implement this to be realized, the prediction model must be downloaded out the irrigation management itself in case communication link to server is broken e.g. internet connection is down. For this to be realized, the prediction model must be downloaded to the raspberry pi. The raspberry pi communicates with end point devices using a ZigBee device.

**Algorithm 1:** Formula for add new variable:

Initialize $\alpha = 0.999$, $\beta = 0.001$

while(while(True):
  $Y_1 = Y$ (value from sensor) and train source model
  $Y_2 = Y$ and train target model
compute $Y_1$ and $Y_2$

$Y_{ans} = Y$ and optimize $\alpha Y_1 + \beta Y_2 = Y_{ans}$

If $\beta > 0.9$ :
  Replace target model as the main model
  Break

introduce a method so that we no longer need to provide a new system or model to add a new variable, this method helps each greenhouse to install a more accurate system according to its financial capacity. Our main method is to find the relationships between the soil layers, which will definitely be the safest way to predict soil moisture. With this method, you can implement all the previous articles mentioned in the introduction [5], [6], [7], and each person can add new option to the whole system only by paying for the desired sensor, without having to pay extra money.

**VI. IoT Solution**

Smart Irrigation systems generally include three main technologies: evapotranspiration-based controllers, soil moisture sensor controllers, and rain-sensor-based controllers [27]. The proposed system in this paper is based on soil moisture sensor controllers. But, new variables like temperature data can be added to the system by simply connecting the required sensors. Various architectures for IoT-based agriculture systems using sensors to obtain soil data have been proposed in literature. In [1] different sensors were used to transmit soil data to a central server through serial communication and enable the user to monitor the data. In [7] they used one moisture sensor per valve, Arduinos and a raspberry pi to make possible the automatic control of irrigation. Also, CherryPy server was used to connect the system to the web. In [6] LoRa P2P networks were used to automate the irrigation of crops. Soil data was collected using various types of sensors and then processed and sent to a server using a LoRa gateway. The smart irrigation system architecture proposed in this paper is illustrated in Figure [10]. The system consists of three major subsystems:

- **End points** collect soil moisture data to be processed by the cloud server. They are also connected to actuators to carry out irrigation.
- **Cloud server** is the subsystem that is responsible for receiving sensory data and instigating irrigation activities executed by end points.
- **Coordinator** is the center of a star topology. It acts as a mediator for communications taken place between server and endpoints.

**A. End Point**

Each end point consists of moisture sensors embedded in different depths of the soil that provide sensory data, and Solenoid valves as actuators. Only one set of sensors per homogeneous parcel, where the soil and crop is homogeneous and constant, is used [4]. Doing so makes the system more cost-effective. At the end point’s heart is a low-power and low-cost microcontroller, namely PIC16F688. The end point perceives the irrigation operation as a binary decision sent by the cloud server and actuates it through opening or closing of faucets. Utilizing the microcontrollers enhanced timer/counter, each end point implements a safety mechanism limiting the maximum irrigation time. When an irrigation operation starts, a timer is started and a predefined timeout is set to limit the maximum allowable irrigation duration. When the timeout is exceeded, irrigation is stopped and an error message is sent to the server. This prevents over-irrigation, water runoff, and plant damage in case of server failure. End points send sensory data to the coordinator by means of ZigBee devices.

**B. Coordinator**

Coordinator manages communications between end points and the cloud server. It is the central node of a star topology. Its main component is a raspberry pi responsible for communicating with the cloud server and the end points. It is also capable of running python programs and, therefore, can carry out the irrigation management itself in case communication link to server is broken e.g. internet connection is down. For this to be realized, the prediction model must be downloaded to the raspberry pi. The raspberry pi communicates with end point devices using a ZigBee device.

**C. communication design**

One important aspect of wireless sensor networks is the method for communication between the nodes. Two prominent protocols are CoAP and MQTT. MQTT has been implemented for IoT-based irrigation systems and decent service availability has been observed [5]. Using a common middleware, MQTT has been found to experience lower message delays than CoAP for lower values of packet loss [28]. Since the underlying ZigBee network can support retransmissions up to four times in case data is lost, low data loss can be assumed, hence, MQTT will have lower delay for this particular architecture than CoAP. However, MQTT requires an underlying network,
such as TCP/IP that provides an ordered connection capability. We decided to use MQTT-SN as the communication protocol for our network. MQTT-SN is a version of MQTT more adapted to wireless sensor networks, and is highly compatible with ZigBee networks [29]. In our system, the end points are MQTT-SN clients, the coordinator is an MQTT-SN gateway and functions as translator between MQTT-SN and MQTT. Finally, the server has the role of MQTT broker. The systems communication protocol stack is depicted in Figure 11.

VII. CONCLUSION

We proposed a highly accurate IoT system for smart irrigation of greenhouses using artificial neural networks, using 4 soil sensors. We collected the data required for training the system, and proposed a transfer learning approach that can address the following issues: i) low number of training samples ii) low processing power at the edge devices iii) flexibility to adding new environmental sensors. Our transfer learning is based on using a pre-trained model to a different soil or set of sensory inputs for training a new model. We showed the alternative state of the art SVR models do not satisfy the required accuracy, even with transfer learning. We are commercializing and implementing our techniques in real world under Sepantab start up company. There are several new directions. More experimentation using new collected field datasets can improve the techniques here. The approach can be extended to other environments such as farm irrigation. Also, the number of sensors can be adaptively chosen by the type of soil and accuracy level required to minimize costs.

REFERENCES

[1] S. Ghosh, S. Sayyed, K. Wani, M. Mhatre, and H. A. Hingoliwala, “Smart irrigation: A smart drip irrigation system using cloud, android and data mining,” in 2016 IEEE International Conference on Advances in Electronics, Communication and Computer Technology (ICAECCT), pp. 236–239, IEEE, 2016.
[2] S. Vaishali, S. Suraj, G. Vignesh, S. Dhivya, and S. Udhayakumar, “Mobile integrated smart irrigation management and monitoring system using iot,” in 2017 International Conference on Communication and Signal Processing (ICCSP), pp. 2164–2167, IEEE, 2017.
[3] S. B. Saraf and D. H. Gawali, “Iot based smart irrigation monitoring and controlling system,” in 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), pp. 815–819, IEEE, 2017.
[4] F. Capraro, D. Patino, S. Tosetti, and C. Schugurensky, “Neural network-based irrigation control for precision agriculture,” in 2008 IEEE International Conference on Networking, Sensing and Control, pp. 357–362, IEEE, 2008.
[5] A. Murthy, C. Green, R. Stoleru, S. Bhunia, C. Swanson, and T. Chaspari, “Machine learning-based irrigation control optimization,” in Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation, pp. 213–222, 2019.
[6] Y. Chang, T. Huang, and N. Huang, “A machine learning based smart irrigation system with lora p2p networks,” in 2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS), pp. 1–4, 2019.
[7] K. K. Namala, K. K. Prabhu A, V. A. Math, A. Kumari, and S. Kulkarni, “Smart irrigation with embedded system,” in 2016 IEEE Bombay Section Symposium (IBSS), pp. 1–5, 2016.

[8] A. Jalili. https://www.kaggle.com/amirmohammadjalili/soil-moisture-dataset.

[9] J. Genovia, E. Jaramillo-castro, E. Sedurifa, F. J. Sequetc, D. Capilitan, and S. J. Malayao, “Do-it-yourself (diy) science instrument water absorption of soil apparatus,” 05 2016.

[10] A. J. Smola and B. Schölkopf, “A tutorial on support vector regression,” Statistics and computing, vol. 14, no. 3, pp. 199–222, 2004.

[11] A. Risheh. https://www.kaggle.com/alirisheh/support-vector-regression.

[12] S. Datta, S. Taghvaeian, and J. Stivers, “Understanding soil water content and thresholds for irrigation management,” 06 2017.

[13] M. Khashei and M. Bijari, “An artificial neural network (p, d, q) model for timeseries forecasting,” Expert Syst. Appl., vol. 37, pp. 479–489, 01 2010.

[14] P. Zhang, E. Patuwo, and M. Hu, “Forecasting with artificial neural networks: The state of the art,” International Journal of Forecasting, vol. 14, pp. 35–62, 03 1998.

[15] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” Journal of Machine Learning Research, vol. 15, no. 56, pp. 1929–1958, 2014.

[16] X. Yin, J. Han, J. Yang, and P. S. Yu, “Efficient classification across multiple database relations: a crossmine approach,” IEEE Transactions on Knowledge and Data Engineering, vol. 18, no. 6, pp. 770–783, 2006.

[17] L. I. Kuncheva and J. J. Rodriguez, “Classifier ensembles with a random linear oracle,” IEEE Transactions on Knowledge and Data Engineering, vol. 19, no. 4, pp. 500–508, 2007.

[18] E. Baralis, S. Chiusano, and P. Garza, “A lazy approach to associative classification,” IEEE Transactions on Knowledge and Data Engineering, vol. 20, no. 2, pp. 156–171, 2008.

[19] X. J. Zhu, “Semi-supervised learning literature survey,” tech. rep., University of Wisconsin-Madison Department of Computer Sciences, 2005.

[20] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell, “Text classification from labeled and unlabeled documents using em,” Machine Learning, vol. 39, pp. 103–134, 05 2000.

[21] A. Blum and T. Mitchell, “Combining labeled and unlabeled data with co-training,” Proceedings of the Annual ACM Conference on Computational Learning Theory, 10 2000.

[22] T. Joachims, “Transductive inference for text classification using support vector machines,” ICML, 08 2001.

[23] Xingquan Zhu and Xindong Wu, “Class noise handling for effective cost-sensitive learning by cost-guided iterative classification filtering,” IEEE Transactions on Knowledge and Data Engineering, vol. 18, no. 10, pp. 1435–1440, 2006.

[24] Qiang Yang, C. Ling, X. Chai, and Rong Pan, “Test-cost sensitive classification on data with missing values,” IEEE Transactions on Knowledge and Data Engineering, vol. 18, no. 5, pp. 626–638, 2006.

[25] S. J. Pan and Q. Yang, “A survey on transfer learning,” IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 10, pp. 1345–1359, 2010.

[26] E. Torquebiau, C. Rosenweig, A. M. Chatrchyran, N. Andrieu, and R. Khosla, “Identifying climate-smart agriculture research needs,” 2018.

[27] M. McCready, M. D. Dukes, and G. Miller, “Water conservation potential of smart irrigation controllers on st. augustinagrass,” Agricultural water management, vol. 96, no. 11, pp. 1623–1632, 2009.

[28] D. Thangavel, X. Ma, A. Valera, H.-X. Tan, and C. K.-Y. Tan, “Performance evaluation of mqtt and coap via a common middleware,” in 2014 IEEE Ninth international conference on intelligent sensors, sensor networks and information processing (ISSNIP), pp. 1–6, IEEE, 2014.

[29] A. Stanford-Clark and H. L. Truong, “Mqtt for sensor networks (mqtt-sn) protocol specification,” International business machines (IBM) Corporation version, vol. 1, p. 2, 2013.