Model Reuse Paradigm in Edge Computing

Xenia Skotti

School of Computing Science
Sir Alwyn Williams Building
University of Glasgow
G12 8QQ

Masters project proposal

Decemeber 2021

Contents

1 Introduction

Lee et al. [5] define compute reuse as "the partial or full utilization of already executed computational task results by multiple users to complete a new task while avoiding computation redundancy". Systems that adopt compute reuse benefit from significant performance gains motivating model reuse in machine learning (ML). Model reuse [13] attempts to construct a model from other pre-existing and pretrained models for other tasks, in order to avoid building a model from scratch. Exploitation of pre-existing models can set a good basis for the training of a new model which translates into a reduced time cost, data amount and expertise required to train a new model. Moreover, model reuse has been used to tackle concept drift [12] and building ad-hoc analytic models [4].

Model reusability is compelling and therefore both theoretical [13] and empirical [4][11] frameworks have been proposed to take advantage of it. Many of the approaches proposed, involve a two-phased framework of a preprocessing and runtime phase. In the preprocessing phase, the model and its data are shared in a pool from which in the runtime phase the relevant ML models are identified. Consider the case of edge computing, where given a number of nodes and their corresponding datasets we want to decide for which nodes to train a distinct model and for which to reuse one. In this context the reuse comes from the fact that we don't train a model for all the nodes but instead reuse one of the existing ones. A framework for model reuse in edge computing requires that it is online, hence these steps are merged and to the best of our knowledge no such framework has been proposed.

2 Statement of Problem

Our project aims to contribute a novel online framework for model reuse in edge computing, which given a set of nodes and their corresponding datasets can determine for which nodes to train distinct models and for which nodes to reuse one. Therefore, we need to be able to determine both if nodes' datasets are similar but also the direction of reusability i.e. if one node's model would be better suited than the other. Avoiding the need to train models for all of the nodes in the network reduces the amount of computational resources that would be required if no reuse was applied. Consequently, the network can benefit from overall improved performance. Additionally, compared to other model reuse frameworks where the two distinct steps are executed at different points in time, our framework carries out both steps at a single point in time i.e. it is online.

3 Literature Survey

Compute reuse has been investigated in the context of edge computing by [5] to quantify its gain. Executing experiments on three applications: matrix multiplication, face detection, and chess, they found that systems that adopt compute reuse, compared to systems that

don't, can finish the same task up to five times faster. In addition to the benefits of compute reuse they also highlight some challenges including task representation and privacy considerations. Model tasks need to have a clear specification detailing their purpose and speciality in order to identify when they can be re-used while also preserving user privacy when they are shared. Motivated by similar concerns a theoretical paradigm named learnware was proposed by Zhou [13]. More specifically, a learnware is a machine learning model that is pretrained and achieves good performance paired with a detailed specification. The vision behind the paradigm was that learnware models can be shared in a pool without their raw data, allowing data scientists to identify pretrained models that satisfy their requirements without concerns over privacy violations. Therefore, the author identified three characteristics: reusable, evolvable and comprehensible as fundamental for a model to be considered a learnware.

Based on this paradigm, the reduced kernel mean embedding (RKME) [11] was presented, a two phased framework consisting of the upload and deployment phase. During the upload phase, each model is paired with its kernel mean embedding (KME) of the dataset and added to the pool of models. Roughly speaking, a kernel mean embedding is a point in the reproducing Hilbert space (RKHS) which "summarises" the probability distribution. Then in the deployment phase either a single or a combination of models is chosen based on the RKHS distance between the testing (target) mean embedding and reduced (source) embedding of pool models. Therefore, there is no need to access the raw data since KME acts a proxy for them. The RKME method is similar to the Maximum Mean Discrepancy (MMD) statistic [2], which is the largest difference between the mean embedding of two populations (source and target) and its aim is to determine if the two populations were drawn from the same distribution. Essentially, this is what the deployment phase of the framework does, it wants to find the model which minimises the difference and thus ensures that the target distribution is the same or as close as possible to the one of the source. The framework was tested in a series of experiments including a real-world project where it outperformed reuse baselines in terms of the root-mean-square error.

The author of the learnware paradigm [13] recognises transfer learning as a preliminary attempt to reusability. The aim of transfer learning is to transfer the knowledge of a pretrained model to a new model that is used for a different but related problem. In transfer learning there are three key research issues as identified in [8]: when, how and what to transfer. This corresponds to identifying a source domain that would benefit the target domain, then using an algorithm the transferable knowledge across domains is discovered. A two-stage framework dubbed as Learning to Transfer (L2T) was presented [10], which exploits previous transfer learning experiences to optimize what and how to transfer between domains. In the first stage each transfer learning experience is encoded into three parts: a pair of source and target domains, the transferred knowledge between them represented by latent factors and the performance improvement ratio. Using these transfer learning experiences, L2T learns a reflection function, which approximates the performance improvement ratio and thus encrypts transfer learning skills of deciding what and how to transfer. The improvement ratio in this framework is the difference between domains calculated by MMD further highlighting the similarity to RKME [11]. In addition to the MMD between domains, the variance is also calculated since a small MMD paired

with an extremely high variance still indicates little overlap. A potential drawback of the RKME [11] framework, and by extension the learnware paradigm, is that the variance between pairs cannot be calculated since the raw data are not available during the testing phase. During the second stage, whenever a new pair of domains arrives, L2T optimizes the knowledge to be transferred by maximising the value of the learned reflection function.

Concerns over intellectual property (IP) infringement and vulnerability propagation of deep learning models (DNN) motivated the proposal of ModelDiff [7], a testing-based approach to DNN model similarity comparison. They compare the decision logic of models on the test inputs represented by a decision distance vector (DDV), a newly defined data structure in which each value is the distance between the outputs of the model produced by two inputs. These inputs are pairs of normal and corresponding adversarial samples and thus when used to calculate the DDV, the decision boundary is captured. In contrast to RKME [11] which is a compute reuse framework, ModelDiff is a model reuse detector.

Model reuse has also been used to handle concept drift, a situation where the distribution of the data (usually stream data) changes. The assumption that previous data contain some useful information, indicates that the models corresponding to the data can be leveraged. Condor was proposed [12] as an approach which can handle concept drift through model reuse. Condor consists of two modules, ModelUpdate and WeightUpdate which leverage previous knowledge to build new model, hence updating the model pool and adapting the weights of previous models to reflect current reusability performance respectively. The effectiveness of the approach was validated using both synthetic and real-world datasets.

Hasani et al. [4] proposed a two-phased approach, to build faster models for a popular class of analytic queries by leveraging model reuse. Similar to other approaches such as RKME [11], there is a preprocessing and a runtime phase. During the first phase the models, their statistics and some meta-data are stored, while in the second phase relevant models are identified from which an approximate model is constructed. Moreover, they propose two methods for generating approximate models, one which is extremely fast but does not provide a fine-tuning option and another which does at the cost of efficiency. Their approach can achieve speed-ups of several orders on magnitude on very large datasets, however it is only geared towards exploratory analysis purposes and the approach is potentially less robust under concept drift.

Lee et al. [5] also discuss alternative approaches and corresponding challenges of compute reuse including in networks. They identify that reuse can be achieved either in a distributed or centralized manner. The distributed approach involves forwarding tasks to the compute reuse node that is responsible for the operation. This adds additional complexity to the forwarding operations of routers resulting in a potential downgrade in performance. Reuse of results in a network setting undoubtedly improves performance, however speeding up the estimation of parameters can also be beneficial in that regard. Nodes in a network can collaborate to estimate parameters as discussed in [6]. More specifically, their method takes advantage of the joint sparsity of vectors used for computations enhancing estimation performance. Joint sparsity simply means that the indexes of nonzero entries for all nodes are the same, but their values differ. The authors also adopt an intertask cooperation

strategy to consider intertask similarities. Their method assumes that both the vectors of interest and their associated noise follow a zero-mean Gaussian distribution which is a strong assumption for the data to hold.

In conclusion, reusing models results in significant reduction in compute usage resources. Both theoretical and empirical frameworks have been proposed to take advantage of the performance improvement of model reusability. Nevertheless, model reuse has also been used to tackle concept drift and building ad-hoc analytic models. While model reuse is undoubtedly beneficial many have raised concerns including user privacy and intellectual property considerations. These are legitimate concerns of model sharing, however our model reuse framework is novel and therefore user privacy is not a concern at this stage of development. In the future we could amend the framework to not expose any data outside of the node. At this stage we're interested in whether the framework is feasible. In contrast to previous research in which frameworks required two distinct steps, our framework is online, and they are therefore merged. Our framework includes determining which datasets are similar, but also the direction of reusability. Similarly, to the L2T [10] framework we use MMD to measure the similarity of two dataset domains. In previous research there was no requirement to determine the direction of reusability hence we propose a novel approach, using the One-class Support Vector Machine (OCSVM) model of each node to predict the other node's inliers and measuring the overlap.

4 Proposed Approach

In this section we elaborate on the implementation of our proposed approach and the corresponding a experimental setup to test out our framework along with a summary of the results of the experiments executed so far.

4.1 Detailed Framework Implementation

Our online model reuse framework needs to be able to determine two things given a pair of nodes. First and foremost, one of the fundamental requirements of any model reuse framework is to be able to choose the model that best fits the (test) data of the target domain. One of the ways this can be achieved is by finding the model whose source domain (training data) is drawn from the same distribution as the target domain. Therefore, the difference between domains needs to be quantified and minimised to find the best model. This is essentially what the Maximum Mean Discrepancy (MMD) [2] statistic does. In addition to measuring the similarity between two dataset domains, we need to determine the direction of reusability. In other frameworks where the reused model originated from a pool there was no such requirement because there was only one direction of reusability, the pool. In this setting though there are two directions per pair, and we need to define a method to do so. A simple solution to this, is to measure the overlap between the inlier points of two datasets. Any dataset is expected to have a few outliers and a simple filtering

Algorithm 1: Calculates the average similarity MMD (ASMMD) between the given nodes.

Data: ***samples***: dictionary associating each node (pi2-pi5) with a sample used for the MMD calculation, ***similar_nodes***: nodes which we have visually identified as similar to each other, ***other_nodes***: the rest of the nodes, ***kernel***, ***bandwidth***: the kernel type and scalar value to be used for the MMD calculation.

Result: ASMMD

```
begin
    // Calculating the baseline ASMMD
    similar_mmds ⟵ []
    for x, y in get_pair_combos(similar_nodes) do
        sx ⟵ samples[x], sy ⟵ samples[y]
        mmd ⟵ MMD(sx, sy, kernel, bandwidth)
        similar_mmds.append(mmd)
    end
    // Compare whether any of the other_nodes are similar to any of the
       similar_nodes using the baseline ASMMD
    for x in other_nodes do
        sx ⟵ samples[x]
        for y in similar_nodes do
            sy ⟵ samples[y]
            mmd ⟵ MMD(sx, sy, kernel, bandwidth)
            current_asmmd ⟵ mean(similar_mmds)
            if mmd < (current_asmmd + 1) * 0.05 then
                similar_mmds.append(mmd)
            end
        end
    end
    // Compare whether any of the other_nodes are similar to each other
       using the current ASMMD
    if len(other_nodes > 1) then
        for x, y in get_pair_combos(other_nodes) do
            sx ⟵ samples[x], sy ⟵ samples[y]
            mmd ⟵ MMD(sx, sy, kernel, bandwidth)
            current_asmmd ⟵ mean(similar_mmds)
            if mmd < (current_asmmd + 1) * 0.05 then
                similar_mmds.append(mmd)
            end
        end
    end
    asmmd = mean(similar_mmds)
end
```

technique would be to use One-class Support Vector Machines (OCSVM) [9] to determine which points are inliers. The method, first presented by Schölkopf et. al [9], utilizes a training data set with normal data to learn the boundaries of the normal data points. Therefore, data points which lie outside of the region are classified as outliers. Given two nodes and their corresponding OCSVM models, we can use each OCSVM model to predict the other node's inliers and then find the probability of detecting them, hence their overlap.

At this point it is worth mentioning that initially, we were investigating whether either MMD or OCSVM or both could be used to detect similar pairs. Therefore, OCSVM would detect both similar pairs and the direction of reusability. Nonetheless, preliminary experiments and analysis showed that the OCSVM similar pairs method was unstable in contrast to MMD. Since, the probability of detecting inliers was still a useful piece of information to determine direction of reusability, we decided instead to pivot the framework to it's current form. Hence, we've adapted our method to using MMD to detect similar pairs and OCSVM the direction of reusability.

Using the MMD two samples are drawn from the same distribution if it is zero. However, we're using an estimate and thus in this case, we need to set a threshold for what score would be deemed similar. We've dubbed this threshold to be the average similarity MMD (ASMMD) and we've devised an algorithm to allow us to calculate it (Algorithm 1). The algorithm requires that we categorise nodes into two sets, one where all nodes are similar to each other and the rest of them. We calculate a baseline ASMMD by calculating the MMD of all pair combinations of the similar nodes. Then, we use ASMMD (allowing for a 5% variation) to judge whether the rest of the nodes are similar to each other or to the similar nodes. If they are we calculate the new ASMMD and we use this to judge the next pair. Using the result of this process we can then judge which pairs are similar for a given experiment. After we've identified the similar pairs we find the direction of reusability using their OCSVM models. We can use one node's model to predict the other node's inliers and then find the probability of detecting them.

4.2 Experimental Setup

4.2.1 Training Data

In order to test the feasibility of our framework we first need a dataset. For the purposes of our framework we've experimented with the ***GNFUV Unmanned Surface Vehicles Sensor Data Set*** [3] which includes data from three experiments. In each experiment there are four sets of mobile sensor readings data recorded by the Raspberry Pi's corresponding to four Unmanned Surface Vehicles (USVs). Each node (USV) dataset contains the humidity and temperature recorded when they were floating over the sea surface in a GPS pre-defined trajectory in a coastal area of Athens (Greece). The data description and visualisation can be found in Table 1 and Figure 1 respectively.

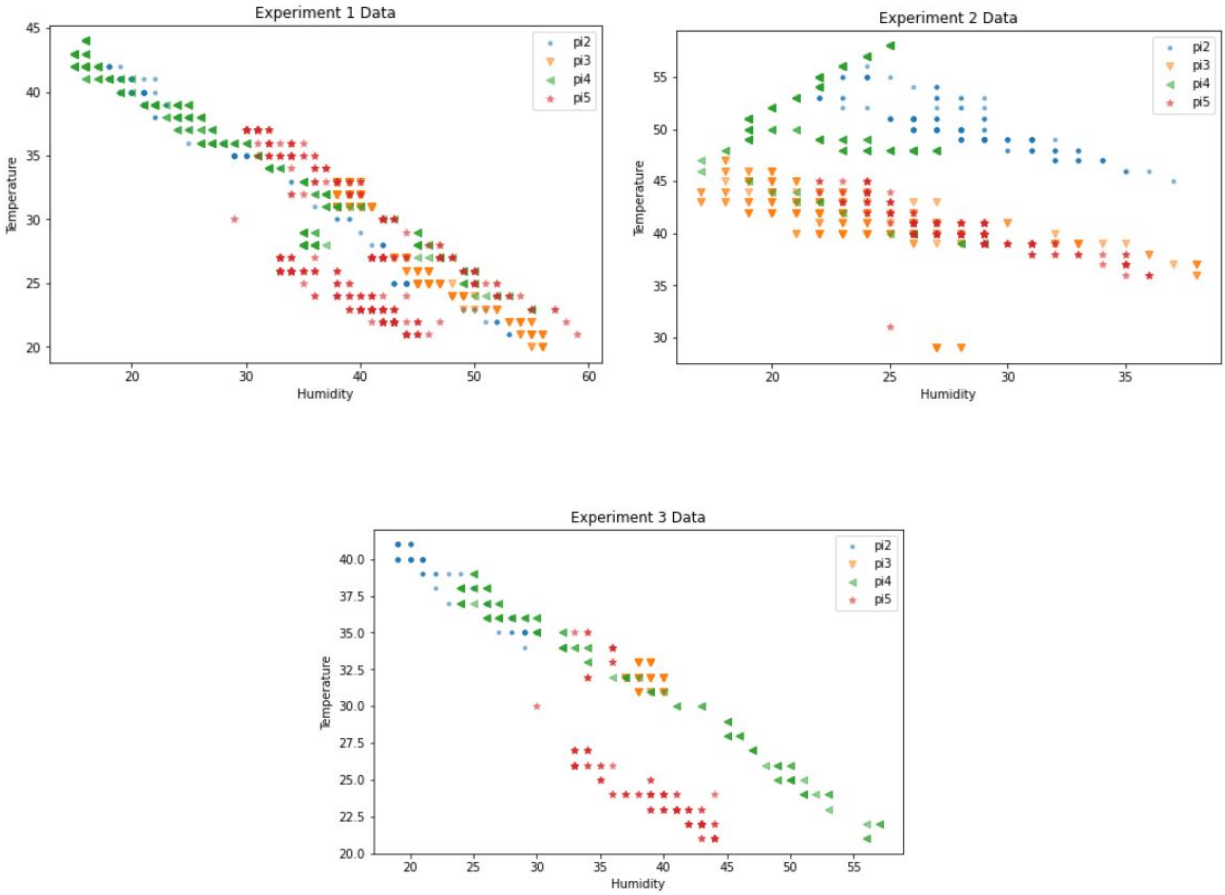

Figure 1: Data visualisation per experiment

Experiment 1									
Node	**No. of Entries**	**Humidity**				**Temperature**			
		Min	Max	Avg	Std	Min	Max	Avg	Std
pi2	1532	3	45	35.85	6.57	15	57	27.6	11.08
pi3	899	19	33	28.53	4.39	34	59	43.04	6.17
pi4	1766	18	45	35.47	6.55	0	64	28.2	11.71
pi5	2078	18	37	27.51	5.1	20	63	39.78	6.9
Experiment 2									
pi2	580	26	56	48.48	6.6	21	49	28.42	3.86
pi3	807	27	49	41.73	4.14	17	46	23.41	5.66
pi4	1021	30	59	50.56	6.4	16	36	22.76	2.76
pi5	1407	30	48	40.13	2.73	20	49	28.05	3.36
Experiment 3									
pi2	342	33	42	38.39	2.39	17	31	23.44	4.09
pi3	264	27	33	31.95	0.77	34	43	38.63	1.26
pi4	488	20	39	31.24	5.31	23	59	38.28	10.56
pi5	555	21	37	25.23	3.91	20	47	38.13	4.65

Table 1: Dataset Description

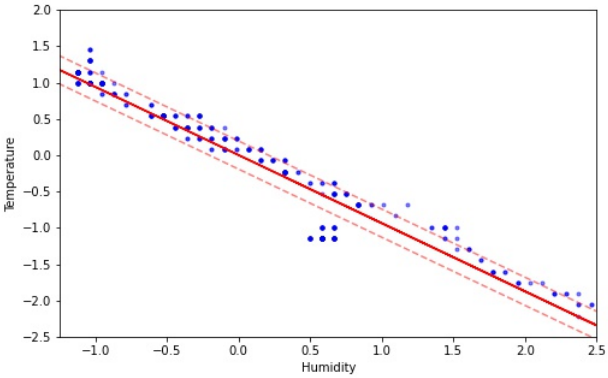

Figure 2: Visual Representation of an SVR

4.2.2 Classifier of Choice

Using the humidity and temperature attributes of the dataset we have trained regression models to capture the relationship between humidity and temperature. The classifier of choice is Support Vector Regression Machine's (SVRs). SVRs are a version of SVM for regression proposed by Vapnik et al. [1]. The adaptation is accomplished by introducing an ϵ-insensitive region around the function, called the ϵ-tube as shown in Figure 2. Similarly to other SVM methods, the hyperplane is represented in terms of support vectors, which are training samples that lie outside the boundary of the tube. The optimization problem objective is to find the tube that best approximates the continuous-valued function, while balancing model complexity and prediction error. Therefore, the goal is to first minimise an ϵ-insensitive loss function and find the flattest tube that contains most of the training instances. SVRs have a few variables of interest we want to optimise for each node model. It is worth nothing that we experiment with both the linear and rbf kernels in order to evaluate how different kernels interact with our framework. We optimise the regularization parameter and the epsilon in the epsilon-SVR model using grid search given a node's dataset.

4.2.3 Evaluation

The aim of the project is to test whether we can detect similar pairs and the direction of reusability using MMD and OCSVM respectively. In order to test this hypothesis we need to find the similar pairs of nodes in the experiments (e.g. (pi2, pi4)) and which node's model to replace which i.e. forward (pi2, pi4) or backward (pi4, pi2). Once we identify the pairs, assume that we have identified them to be (pi3, pi5) and (pi2, pi4), we will train and optimise a model for each node (using Grid Search) and then for each pair we will use one node's model on their counterpart i.e. pi3 model on pi5 and the pi5 model on pi3 to determine if the given direction of reusability given by OCSVM is correct. We can test the correctness of MMD using the discrepancy error, the absolute difference between the

predicted coefficient of determination (R2) (i.e. score given when using the counterpart model to make predictions) and the actual R2 (i.e. score when using the native model of the node). This can measure the drop in performance and hence whether this is a good pair for reusability. Now to test the correctness of the direction of reusability indicated by OCSVM we measure if using the node indicated by OCSVM will result in a better reusability model for both nodes compared to the opposite direction.

In our experiments we wanted to investigate the feasibility of our framework while also taking into account some variables of interest that can affect results. The sample size, for example can be varied to test the framework in different contexts and hence we experimented with 3 distinct networks and sample sizes. More specifically, for each of the 100 samples per experiment (and node) we draw, the size of the sample is half of the minimum sample size in the network i.e. the node with the minimum number of entries. There is an added condition that the sample size needs to be at least 500 unless the minimum number of entries of any node in the network is less in which case we choose the minimum sample size in the network. The reason behind this choice is that we've found that a large sample provides a much better consensus across samples and more stable results. Even though for some nodes we're not really creating a sample but simply taking all the points of the node, the remainder of the nodes will be samples and therefore we're testing a different relationship between nodes with each sample and hence we do not compromise the validity of our experiment. For the GNFUV dataset the sample size is always 500 unless this minimum requirement cannot be met. Nevertheless, we plan to test our framework with much larger dataset and hence this will be important in the future. Another variable of interest is the choice of kernel for SVR which impacts the sensitivity of the model and thus it's response to reusability. Additionally, we chose to optimise parameters for SVR models in order to get the best model for each node individually. Lastly, we've used both the original and standardised version of the data to investigate the effect of variance on the framework.

Since we're executing a 100 iterations of the hypothesis we also need metrics that summarise the results for the variables of interest. Essentially, we're interested in two things, first the precision of the method and second the gain in performance. In terms of the precision we can measure this individually for each component of the framework and combined. For the OCSVM direction of reusability, we can simply measure the number of times that the highest performing model (the one for which the model R2 minus the the discrepancy is highest), is also the one pointed by OCSVM. The precision for OCSVM was measured in two ways, first strictly as described above and the other by allowing that the candidate best performing models can vary by 0.05 from the highest performing model. For MMD, we're testing if it can find good pairs for reusability and a good pair would be one where the reused model's performance would be to some extent close to the performance of the true model. Accordingly, given a threshold of the ratio between the reused and true model's performance we can judge whether MMD has detected a good pair or not by measuring if the highest performing model exceeds the threshold. For the combined precision we can simply check the number of times that the OCSVM direction is correct but also to what extent (using the ratio). The speedup is only measured for the framework as a whole, by finding the nodes for which a model is not needed and measuring the ratio of saved time by these nodes compared to if we were to train a model for every node in the network. At this

point it worth noting that for both metrics we report the weighted average of the results per experiment. We weight the precision value of each experiment by the number of pairs identified in each experiment.

4.3 Current Results

From our initial experiments we have found the following. First, for all pairs (forward, backward) non-linear models have higher baseline scores while linear models on average result in lower discrepancy and better performance. However, when it comes to the kernel of the best model, this varies across experiments and data types (standardised and non-standardised). Additionally, pairs where both models have good baseline scores are good models for reusability on both sides. OCSVM correctly predicts the direction of reusability for the model that yields the best results with weighted precision of 0.82 and 0.6 while allowing a 0.05 error margin it increases to 0.97 and 0.62 for non-standardised and standardised data respectively. MMD identifies pairs whose model performance (R2 score) is at least 85% of the true model for a node, with weighted precision 0.7 for non-standardised data and 0.81 for non-standardised data. The combined precision for standardised data with threshold 0.8 is 0.6, and 0.74 if we allow a 0.05 error margin. For the same threshold standardised data achieve scores of 0.43 and 0.59 respectively. Finally, in terms of speedup the results across data types are almost equal and the framework results in a drop of training time by 26%.

To summarise, we've provided a detailed description of our framework implementation along with the experimental set up and the experiments executed so far.

5 Work Plan

Moving forward we expect to re-execute our research hypothesis to a much larger dataset in order to compare the results and have a more broad understanding of how the framework responds to different contexts. For this reason the dataset we will use is the classification UCI Bank Marketing (BM) Dataset with a total number of 45211 instances. As a first step we would need to do some exploratory analysis on the dataset to get a better understanding of it. The dataset has many features and therefore we would need to perform feature selection using Principal Component Analysis. Since the dataset is not really split into nodes as the GNFUV one we would need to devise a strategy to both split the data meaningfully into nodes but also determine the similar pairs. Given the above, we can re-execute our hypothesis with the minor change of using Logistic Regression (LR) as our classifier of choice instead of SVR. We chose LR in particular as it is generally considered a good baseline for classification.

Consequently, we propose the following milestones and corresponding completion dates for the project:

- January 14th: Exploratory Analysis of BM Dataset (including feature selection), Devise dataset splitting strategy into nodes and identification of similar pairs
- January 28th: Refactoring of hypothesis testing related code and executing of hypothesis testing for the BM dataset, Refactoring of results presentation and metrics related code
- Feb 4th: Data Analysis of Results
- Feb 11th: Amendment of Literature Review to Related Work
- Feb 18th: Preliminaries of Paper
- Feb 25th: Detailed description of framework implementation and experimental set up
- Mar 11th: Presentation of and evaluation of results
- Mar 18th: Discussion
- Mar 25th: Conclusion, Introduction, Abstract
- Mar 28th: First Draft
- Apr 15th: Corrections and Final Draft

References

[1] Harris Drucker, Chris J. C. Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. Support vector regression machines. In *Proceedings of the 9th International Conference on Neural Information Processing Systems*, NIPS'96, page 155–161, Cambridge, MA, USA, 1996. MIT Press.

[2] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *J. Mach. Learn. Res.*, 13:723–773, March 2012.

[3] Natascha Harth and Christos Anagnostopoulos. Edge-centric efficient regression analytics. In *2018 IEEE International Conference on Edge Computing (EDGE)*, pages 93–100, 2018.

[4] Sona Hasani, Saravanan Thirumuruganathan, Abolfazl Asudeh, Nick Koudas, and Gautam Das. Efficient construction of approximate ad-hoc ml models through materialization and reuse. *Proc. VLDB Endow.*, 11(11):1468–1481, July 2018.

[5] Jonathan Lee, Abderrahmen Mtibaa, and Spyridon Mastorakis. A case for compute reuse in future edge systems: An empirical study. In *2019 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6, 2019.

[6] Chunguang Li, Songyan Huang, Ying Liu, and Zhaoyang Zhang. Distributed jointly sparse multitask learning over networks. *IEEE Transactions on Cybernetics*, 48(1):151–164, 2018.

[7] Yuanchun Li, Ziqi Zhang, Bingyan Liu, Ziyue Yang, and Yunxin Liu. Modeldiff: Testing-based dnn similarity comparison for model reuse detection. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA 2021, page 139–151, New York, NY, USA, 2021. Association for Computing Machinery.

[8] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[9] Bernhard Schölkopf, Robert Williamson, Alex Smola, John Shawe-Taylor, and John Platt. Support vector method for novelty detection. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, page 582–588, Cambridge, MA, USA, 1999. MIT Press.

[10] Ying WEI, Yu Zhang, Junzhou Huang, and Qiang Yang. Transfer learning via learning to transfer. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5085–5094. PMLR, 10–15 Jul 2018.

[11] Xi-Zhu Wu, Wenkai Xu, Song Liu, and Zhi-Hua Zhou. Model reuse with reduced kernel mean embedding specification. *CoRR*, abs/2001.07135, 2020.

[12] Peng Zhao, Le-Wen Cai, and Zhi-Hua Zhou. Handling concept drift via model reuse. *CoRR*, abs/1809.02804, 2018.

[13] Zhi-Hua Zhou. Learnware: on the future of machine learning. *Frontiers of Computer Science*, 10, 06 2016.