**Introduction**

Lee et al. [7] 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 [3] 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 [2] and building ad-hoc analytic models [8].

Model reusability is compelling and therefore both theoretical [3] and empirical [1][8] 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. Sharing models in a pool raises concerns including user privacy [3][7] and intellectual property (IP) considerations [4]. These are legitimate concerns and one of the ways this can be eliminated is by not sharing the data used to train the models and instead creating a proxy for them [1][4]. However, in some scenarios the need for data sharing is eliminated entirely. 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. Since we’re the owner there are no IP violations while no data are exposed since models are trained locally from scratch and they are the only thing shared. Therefore, a framework for model reuse in edge computing needs to be online and no such framework has been proposed.

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 [9] 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 technique would be to use OCSVM [10] to determine which points are inliers. Therefore, 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.

**Literature Review**

Lee et al. [7] 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"*. The authors of the paper executed a set of experiments on three edge computing applications, matrix multiplication, face detection and chess, to quantify its gain. They found that systems that adopt compute reuse, compared to systems that don't, can finish the same task up to five times faster. Consequently, model reusability is compelling due to significant performance gains.

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 [3]. 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) [1] 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 MMD statistic [9], 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 as 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 [3] 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 [11]: 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 [5], 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 [1]. 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 [1] 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 [4], 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 [1] 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 [2] as an approach to handling 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 adapt 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. [8] 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 [1], 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. [7] 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.

Many of the approaches discussed involve a two-phased framework of a preprocessing and runtime phase but no online framework has been proposed. In edge computing we may have a number of nodes which record data. How can we determine for which nodes to build a model and which model to reuse for the rest? In other words, how can we determine which datasets are similar, but also the direction of reusability. As discussed in the L2T [5] framework MMD can be used to measure the similarity of two dataset domains. A simple solution to the direction of reusability 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 technique would be to use OCSVM [10] to determine which points are inliers. Therefore, 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.

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 model reuse in edge computing for example can simply be about deciding for which nodes to train a distinct model and for which to reuse one. Consequently, an online framework is required which can determine both which pairs are similar and the direction of reusability.

**References**

[1] X. Z. Wu, W. Xu, S. Liu, and Z. H. Zhou, “Model Reuse with Reduced Kernel Mean Embedding Specification,” *IEEE Trans. Knowl. Data Eng.*, pp. 1–21, 2021, doi: 10.1109/TKDE.2021.3086619.

[2] P. Zhao, L. W. Cai, and Z. H. Zhou, “Handling concept drift via model reuse,” *Mach. Learn.*, vol. 109, no. 3, pp. 533–568, 2020, doi: 10.1007/s10994-019-05835-w.

[3] Z. H. Zhou, “Learnware: on the future of machine learning,” *Front. Comput. Sci.*, vol. 10, no. 4, pp. 589–590, 2016, doi: 10.1007/s11704-016-6906-3.

[4] Y. Li, Z. Zhang, B. Liu, Z. Yang, and Y. Liu, *ModelDiff: Testing-based DNN similarity comparison for model reuse detection*, vol. 1, no. 1. Association for Computing Machinery, 2021.

[5] Y. Wei, Y. Zhang, J. Huang, and Q. Yang, “Transfer learning via learning to transfer,” *35th Int. Conf. Mach. Learn. ICML 2018*, vol. 11, pp. 8059–8068, 2018.

[6] C. Li, S. Huang, Y. Liu, and Z. Zhang, “Distributed jointly sparse multitask learning over networks,” *IEEE Trans. Cybern.*, vol. 48, no. 1, pp. 151–164, 2018, doi: 10.1109/TCYB.2016.2626801.

[7] J. Lee, A. Mtibaa, and S. Mastorakis, “A case for compute reuse in future edge systems: An empirical study,” *2019 IEEE Globecom Work. GC Wkshps 2019 - Proc.*, pp. 1–6, 2019, doi: 10.1109/GCWkshps45667.2019.9024587.

[8] S. Hasani *et al.*, “ApproxML: Efficient approximate AdHoc ML models through materialization and reuse,” *Proc. VLDB Endow.*, vol. 12, no. 12, pp. 1906–1909, 2018, doi: 10.14778/3352063.3352096.

[9] A. Gretton, “A Kernel Two-Sample Test,” vol. 13, pp. 723–773, 2012.

[10] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and J. Piatt, “Support vector method for novelty detection,” *Adv. Neural Inf. Process. Syst.*, no. May 2014, pp. 582–588, 2000.

[11] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, 2010, doi: 10.1109/TKDE.2009.191.