Source Selection in Transfer Learning for Improved Service Performance Predictions

Hannes Larsson*, Jalil Taghia*, Farnaz Moradi*, and Andreas Johnsson*†
* Ericsson Research, Sweden, Email: {*hannes.larsson,jalil.taghia,farnaz.moradi,andreas.a.johnsson*}*@ericsson.com*
† Uppsala University, Department of Information Technology, Sweden, Email: *andreas.johnsson@it.uu.se*

***Abstract*—Learning performance models for network and cloud services is challenging due to the dynamics of the operational environment stemming from network changes, and scaling and migration decisions in the cloud. This requires exchange or adaptation of the models in order to maintain prediction accuracy over time. Approaches that incorporate previously acquired knowledge using transfer learning is a viable technique for timely and robust model adaptation, especially when the training data is limited.**

In this paper, we study the challenge of source selection in transfer learning for improved service performance prediction. We quantify the impact of different source domains on the accuracy of a target model in another domain. The evaluation is performed using data traces obtained from a testbed that runs a Video-on-Demand service and a Key-Value Store under various load conditions. We find that the choice of source domain can yield a transfer gain, and sometimes a substantial transfer penalty. To mitigate this, we propose and evaluate two source-selection approaches with the aim of selecting a source domain with *relevant* knowledge for the target domain. A key result is that such source selection should encourage source-domain diversity rather than domain similarity in scenarios with few samples in the target domain.

***Index Terms*—Network Automation and Management, Performance Modeling, Machine Learning, Transfer Learning.**

I. Introduction

Telecommunication operators and providers deliver many of their services under strict Service-Level Agreements (SLA), and it is well-known that automation and management of such systems is challenging and demanding. A promising approach enabling intelligent network and service management is the use of data-driven performance models that can predict and forecast the service quality at the client side, as well as other performance indicators, based on available observations in the network infrastructure. The ability to learn performance models from observations simplifies management tasks such as service on-boarding, network slice dimensioning, proactive service assurance, and root-cause analysis.

A key challenge in data-driven modelling for dynamic environments is the difficulty to maintain the relevance of the model over time. For example, services executed in a cloud environment generally rely on a virtualization layer that allows service components to migrate between physical execution environments, and the resources assigned to the service can dynamically be scaled up or down based on operator policies or user requirements. This is exemplified in Figure 1 where the service execution environment changes from one configuration to another (source and target domains). Further, the network path and assigned resources may also change as illustrated in Figure 1. Such changes reduce the accuracy of a performance model (or even makes it obsolete) which has been trained for a specific configuration and environmental condition. As a consequence, management functions that rely on a performance model are negatively affected, unless the model is appropriately updated. Further, extensive measurements and data collection is usually required for training machine-learning models. The data collection process takes time and the overhead associated with measurements and data collection can adversely affect the service itself and potentially co-located services as well. For certain services, such as short-lived Virtual Network Functions (VNFs), there is often not enough time to gather the data required for training a model.

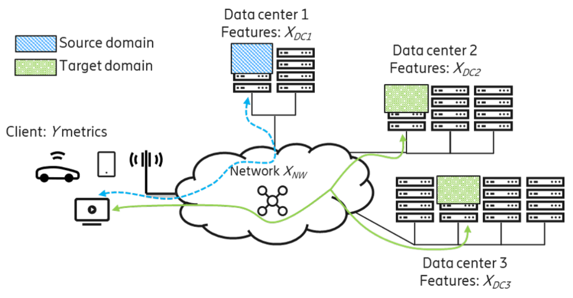

Fig. 1. Prediction of service metrics Y at the client given observations of network and cloud infrastructure state and utilization $\{X_{NW}, X_{DC1..3}\}$. The execution environment changes from source to target domain.

Transfer learning has been proposed for several types of problems as an approach to overcome the above challenges [1], where the knowledge embedded in a model and learned for one environment (source domain) is transferred to facilitate timely predictions in a changed execution environment (target domain). Transfer learning is also a key technology for reusing models where the predictive task, such as the service quality metric, changes over time.

Since transfer learning is an approach that aims at incorporating knowledge gained in a source domain into the target domain, *the transferred knowledge from the sources must be relevant to the target domain* in order to avoid negative transfer [2]. Figure 2 illustrates the concept of source selection, where the aim is to improve target model accuracy.

ISBN 978-3-903176-39-3© 2021 IFIP

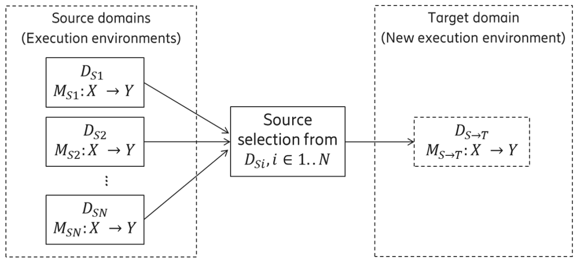

Fig. 2. Timely and accurate deployment of a target model $M_{S\to T}$ requires transfer of relevant source-domain knowledge embedded in $M_{S1..N}$.

This paper addresses the challenges of source selection in transfer learning for network and cloud service scenarios, and makes the following contributions. First, we provide insights from experimental results quantifying the impact of different source domains on the target domain, with a varying number of available samples. A source domain can either yield improved accuracy in the target domain, or result in *negative transfer*. Secondly, we elaborate on two strategies for source selection, namely diversity of the source domain, and similarity between the source and target domains. Our results indicate that relying on the similarity between the target and the source domains as the sole criterion for source selection in transfer learning may not be reliable, in particular when we have a limited understanding of the target domain (i.e., few samples). Rather, methods that encourage diversity provide better source selection for the studied scenarios.

The rest of the paper is organized as follows. Section II describes the problem setting. Section III describes the evaluation methodology and the source-selection approach. Section IV describes the testbed and traces, whereas Section V contains evaluation results and discussions. Section VI reviews related work, and conclusions are located in Section VII.

II. Problem Setting

Figure 1 outlines the system under consideration, where clients are interacting, over a network, with services that are being executed in a data center. For the purpose of this paper, we consider data traces originating from testbed experiments where clients access two network services executing in one data center; a Video-on-Demand (VoD) service and a Key-Value Store (KVS) service.

In this work, we predict the service-level metrics Y_t at time t on the clients accessing VoD and KVS services, based on knowing the infrastructure metrics X_t. Using supervised machine learning, we train and evaluate models $M : X_t \to \hat{Y}_t$, such that $\hat{Y}_t$ closely approximate Y_t for a given X_t.

In [3], it was shown that the accuracy of prediction models decreases over time if the execution environment changes, for example, due to the scaling of resources, service migration, changed hardware platform, or other infrastructure dynamics. To mitigate the problem, a transfer learning approach was proposed and evaluated. However, this work did not investigate the impact of a source domain on the target domain, nor did they propose source-selection approaches.

In this paper, we use the definition of transfer learning from [1] to formalize the challenges of source selection in transfer learning for dynamic clouds. A domain $D = \{X, P(X)\}$ consists of two components: (1) a feature space X, and (2) a marginal probability distribution $P(X)$, where X corresponds to the infrastructure metrics. Further, a task $T = \{Y, M\}$ consists of two components: a target space Y corresponding to service-level metrics, and an objective predictive model M.

Transfer learning is then defined as follows. Given a source domain D_S and learning task T_S, a target domain D_T and learning task T_T, transfer learning aims at reducing the cost of learning the predictive model M in D_T using the knowledge in D_S and T_S, where $D_S \neq D_T$, or $T_S \neq T_T$. A model that is transferred from D_S to D_T is denoted $M_{S\to T}$, to separate it from a model M_S or M_T that is trained in isolation either in the source or the target domain.

This paper studies how the performance of the target model $M_{S\to T}$ varies with the choice of source domain D_S, and the number of available samples N_t at time t in D_T. Recall that the number of samples in the target domain may be limited due to overhead or time constraints, as discussed in Section I, which limits our understanding of the target domain. Transfer learning aims at addressing the problem by incorporating knowledge gained from other source domains into the target domain. Hence, the transferred knowledge from other sources should be *relevant* to the target domain which implies the need for *source selection* in transfer learning. The concept is illustrated in Figure 2.

Conceptually, a relevant source to a target domain is the one that ”improves” the information content of the target domain and thus improves our understanding of it. To improve the information content of the target domain, the information content of the candidate source must be complementary. Hence, on the one hand, one would need to encourage the diversity factor by selecting a source with high enough information content. On the other hand, one would need to select the source whose information content agrees just enough with the target domain, or more formally, the source domain whose underlying distribution is of sufficient similarity to that of the target domain. Thus, in order to select the relevant source D_S to the target D_T, we need to consider the two components of diversity of the source domains and similarity between the domains at the same time. Diversity is a marginalized quantity, meaning that we can measure the information content of a source domain independently from the target domain. However, similarity is a conditional quantity, meaning that we can only measure the similarity in underlying distributions of the target domain and a source domain if we have sufficient representative data from each respective domain. Our hypothesis is that the success of the transfer learning in dynamic network environments ultimately depends on how well we can meet the balance between these two competing objectives.

Thus, a challenge targeted by this paper is where data in the target domain is scarce and poorly representative of the target domain. In such scenarios, we may not be able to reliably measure the similarity in between underlying distributions of

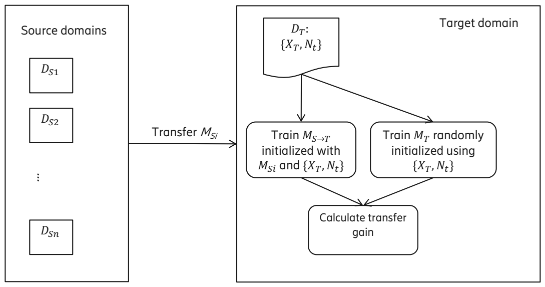
Source domains
D_{S1}
D_{S2}
⋮
D_{Sn}
Transfer M_{Si}
Target domain
D_T: $\{X_T, N_t\}$
Train $M_{S\to T}$ initialized with M_{Si} and $\{X_T, N_t\}$
Train M_T randomly initialized using $\{X_T, N_t\}$
Calculate transfer gain

Fig. 3. Methodology for determining the transfer gain, that is the impact of a source domain D_{Si} on the target domain D_T, for varying number of target-domain samples N_t.

the source domains. Given the inherited limitations in the availability of data, having its origin in the nature of dynamic networks and infrastructures, and difficulties in robustly measuring the similarity component, we argue in favor of methods for source selection which encourage the diversity component.

III. Approach

In this paper, the transfer learning approach builds upon re-training of a feed-forward neural-network model M, that is transferred from a source domain, inspired by the work in [4]. Such a model consists of an input layer, corresponding to samples X_t, $n-1$ hidden layers $L_1, ..., L_{n-1}$, weights $w_1, ..., w_n$, and an output layer L_n corresponding to Y_t.

The weights w_i for a neural-network model M_S, called the source model, are trained using backpropagation [5] with samples from the source domain D_S. In this paper, the knowledge transfer corresponds to training a target model where the weights of layers $L_1, ..., L_{n-1}$ are initialized with the weights from the source model M_S, whereas the weights of L_n are randomly initialized to allow the target model to quickly adapt to the new domain. The target model is then trained using available samples from the target domain D_T. After knowledge transfer, the resulting neural network is denoted $M_{S\to T}$ and is thus used for service-level metric prediction in D_T. In this paper we specifically study the challenging transfer scenarios where $T_S \neq T_T$. Note also that the neural-network architecture of M_S and $M_{S\to T}$ are identical in this study.

In the following, we describe the methodology for quantifying the source-domain impact on $M_{S\to T}$ accuracy, and two heuristic strategies for selecting a source domain.

A. Quantifying source-domain impact

We assess the choice of source domain D_S and its impact on model accuracy in D_T by training target models $M_{S\to T}$ for each source model M_S. For each combination of D_S and D_T, we evaluate the model accuracy given the number of available samples N_t at time t in D_T. See an illustration of the evaluation process in Figure 3.

As a baseline, we train a neural network from scratch, with randomly initialized weights, solely using samples from D_T that are available at time t, and this model is denoted by M_T.

We use the concept of *transfer gain* $\mathcal{G}$ to quantify the impact of D_S on $M_{S\to T}$, which we define as:

$$\mathcal{G} = e_T - e_{S\to T}, \tag{1}$$

where e_T and $e_{S\to T}$ are the model errors for M_T and $M_{S\to T}$, respectively. The transfer gain is inspired by the negative transfer gap defined in [2]. Transfer gain occurs when the accuracy of the transferred model $M_{S\to T}$ is higher than M_T, whereas a negative transfer gap means the opposite.

B. Source selection

In this paper, we consider two different heuristic source-selection methods which consider the *similarity between domains* and the *diversity for source domains*. The intention is to determine which source domains are most fitted despite the limited number of available samples N_t in the target domain D_T at time t, or even no samples at all.

In order to quantify the *diversity* we utilize the concept of differential Shannon entropy $h(X_S)$ for the source domain D_S, which is calculated as:

$$h(X_S) = -\int p_S(x) \log p_S(x) dx, \tag{2}$$

where $p(x)$ is a probability density function for the source domain trace X_S. We plan to investigate other diversity metrics in future work.

In order to measure *similarity* between the observed target domain and a source domain, we use the symmetrized Kullback-Leibler divergence D_{sym} [6]. It is calculated as:

$$D_{sym}(P_S, P_T) = \frac{1}{2}\left(D_{KL}(P_S||P_T) + D_{KL}(P_T||P_S)\right), \tag{3}$$

where P_S and P_T correspond to the probability density functions for the source and target domains, respectively, and

$$D_{KL}(P_S||P_T) = \int p_S(x) \log \frac{p_S(x)}{p_T(x)} dx.$$

In Section VI, we discuss different similarity metrics. As for diversity, we plan to investigate other similarity metrics in future work.

IV. Testbed and data traces

A. Testbed and services

The work presented in this paper is based on realistic traces obtained from a testbed environment. This section provides an overview of the experimental infrastructure, and the structure of the data traces. We also describe the services that run on the infrastructure; a Video-on-Demand (VoD) service and a Key-Value Store (KVS) service. Further, we describe the load patterns we use and experiments we run to obtain the traces on which this paper relies. More details are available in [7].

The testbed consists of a server cluster and a set of clients. The server cluster is deployed on a rack with ten high-performance machines interconnected by a Gigabit Ethernet. Nine machines are Dell PowerEdge R715 2U servers, each

TABLE I
TRACES USED FOR EVALUATION. RAVG = READ AVERAGE, WAVG = WRITE AVERAGE, FR = FRAME RATE, AND ABR = AUDIO BUFFER RATE.

Trace ID	Service(s)	Load pattern	Target Y	# samples
K1P	KVS	Periodic	*RAvg, WAvg*	28962
K1F	KVS	Flashcrowd	*RAvg, WAvg*	19444
K2P	KVS + VoD	Periodic	*RAvg, WAvg*	26488
K2F	KVS + VoD	Flashcrowd	*RAvg, WAvg*	24225
V1P	VoD	Periodic	*FR, ABR*	37036
V1F	VoD	Flashcrowd	*FR, ABR*	36633
V2P	VoD + KVS	Periodic	*FR, ABR*	27699
V2F	VoD + KVS	Flashcrowd	*FR, ABR*	29151

with 64 GB RAM, two 12-core AMD Opteron processors, a 500 GB hard disk, and four 1 Gb network interfaces. The tenth machine is a Dell PowerEdge R630 2U with 256 GB RAM, two 12-core Intel Xeon E5-2680 processors, two 1.2 TB hard disks, and twelve 1 Gb network interfaces. All machines run NPT-synchronized Ubuntu Servers (14.04 64 bits).

The *VoD service* uses a modified VLC media player software [8], which provides single-representation streaming with a varying frame rate. The VoD service is executed on six PowerEdge R715 machines.

The *KVS service* uses the Voldemort software [9]. It is installed on the same machines as the VoD service, and can execute in parallel. Six of the machines act as KVS nodes in a peer-to-peer fashion and the rest act as load generators emulating client populations.

B. Collected data and traces

This subsection provides a description of the data collected on the testbed, namely the input feature set X as well as the specific service-level metrics Y_{VoD} and Y_{KVS}.

Features X are extracted from the Linux kernels that run on the machines. To access the kernel data, we use System Activity Report (SAR), a popular open-source Linux library [10], which provides approximately 1700 features per server. Examples of such statistics are CPU utilization per core, memory utilization, and disk I/O.

The Y_{VoD} service-level metrics are measured on the client. In the experiments the following two metrics were captured: the number of displayed video frames per second (*FrameRate, FR*) and the number of audio buffers per second (*Audio Buffer Rate, ABR*).

The Y_{KVS} service-level metrics are also measured on the clients. During an experiment two main metrics are captured, namely the Read Response Time as the average read latency for obtaining responses over a set of operations performed per second (*RAvg*), and a corresponding metric for the Write Response Time (*WAvg*). The metric is computed using a customized benchmark tool of Voldemort.

A trace is generated by executing testbed experiments with different configurations where statistics are collected every second; specifically it includes features X, and service-level metrics Y_{VoD} and Y_{KVS}.

The 8 traces used in this paper are summarized in Table I. The trace ID is encoded according to service under investigation (KVS or VoD), number of concurrent services (1 or 2), and load pattern (periodic or flashcrowd load). For example, K2F corresponds to a trace where KVS is under investigation, that both services are executing in parallel, and that the load generators are operating in flashcrowd mode.

C. Generating load on the testbed

Two load generators are running in parallel in the testbed, one for the VoD application and another for the KVS application. The VoD load generator dynamically controls the number of active VoD sessions, spawning and terminating VLC clients. The KVS load generator controls the rate of KVS operations issued per second. Both generators produce load according to two distinct load patterns described below:

1) Periodic-load: the load generator produces requests following a Poisson process whose arrival rate is modulated by a sinusoidal function with a starting load level, amplitude, and period of 60 minutes.
2) Flashcrowd-load: the load generator produces requests following a Poisson process whose arrival rate is modulated by a flashcrowd model [11]. The arrival rate starts at a low load level and peaks at flash events.

All traces we used in the considered scenarios have been created using stochastic models in an attempt to approximate real scenarios.

V. EVALUATION AND RESULTS

We apply the evaluation methodology and source-selection approaches described in Section III to a set of transfer scenarios for VoD and KVS, respectively. The evaluation studies the need for selecting a good source domain, and the feasibility of two heuristic approaches for selecting a source domain prior transfer to the target domain.

A source domain D_S, and its corresponding model M_S, is created for each trace of Table I. The 8 different source domains are listed in Table II. A target domain corresponds to a change in the prediction task, i.e., $T_S \neq T_T$. That is, in the KVS scenario the read average time is predicted in the source domain, whereas in the target domain the assumption is that the service provider would like to predict the write response time instead. Corresponding change is studied for VoD. Further, depending on which source is selected in the end, additional scenario-dependent changes include the number of services executing on the platform, and/or the distribution of samples $P(X)$, that depends on the load pattern for a specific trace. The target domains are defined in Table III. Note that in Tables II and III we use the terms *Periodic* and *FlashCrowd* to describe $P(X)$ to indicate the origin of the distribution of the samples.

Based on the source and target domains we define 8 transfer scenarios, summarized in Table IV. That is, for each target domain D_T there are 4 different source domains to select from.

Further, the *Normalized Mean Absolute Error* is used as the metric for quantifying model performance, and is defined as:

$$e = \frac{1}{\bar{y}}(\frac{1}{m}\sum_{t=1}^{m}|y_t - \hat{y}_t|), \quad (4)$$

where $\hat{y}_t$ is the model prediction for the measured performance metric y_t, and $\bar{y}$ is the average of the samples y_t of the test set of size m.

We manually reduced the number of features to 18 using domain knowledge following the approach in [12]. Manual feature selection across traces ensures an identical feature space, enabling homogeneous transfer [13], for source and target domains.

For the purpose of this evaluation, we have selected a feed-forward neural-network model architecture with an input layer with 18 nodes corresponding to the features, 3 hidden layers with 256 nodes each, and one output layer. The same architecture is applied for both VoD and KVS. Note that the ambition is not to find the optimal neural-network architecture, but rather to evaluate the impact of source domains.

To obtain the source models $M_{Si}, i \in [1,..,8]$ for each source domain, the neural networks are initialized with random weights and are trained on samples (X_t, Y_t) from the source domains. For training, the source domains are reduced to the same size of 16000 samples randomly split into training (80%) and validation (20%) sets. The model and its weights are then transferred for tuning and predictions in the target domain. Similarly, in the target domain the samples are split into training, validation, and test sets. The training set is varied between 10 and 100 samples to emulate the shortage of samples in the target domain, and is used to either update the transferred source model $M_{S\to T}$ or to train a new target model M_T. The test set always corresponds to 20% of the samples in the trace, and the remaining samples are used for the validation set. Note that the validation set is used for early stopping and the test set is used for evaluation.

For the implementation of the neural networks, Keras library [14] running on top of TensorFlow [15] was used. We used the rectified linear unit (ReLU) activation function for all the layers, Adam optimizer [16] with a learning rate starting at 0.001 using exponential decay with decay rate 85, 1000 decay steps and staircase function, and mean absolute error (MAE) as the loss function. Each experiment was run for a maximum of 200 epochs, with early stopping using a patience of 10 epochs and weight restoring, with a batch size of 32.

A. Quantifying the impact of source domains

We quantify the impact of source domains on the error of $M_{S\to T}$ for each transfer scenario described in Table IV, following the methodology illustrated in Figure 3. For each scenario, we select 10, 25, 50, 75 and 100 samples from the target domain. Using the selected target samples, we train a baseline model M_T and at the same time update the transferred models $M_{S_i\to T}$ for each source D_{S_i}. The data from the source domain is normalized, and the target domain data used for the fine tuning is scaled using the same transformation. For training the baseline model M_T, the selected target samples are normalized according to the full target data set. Once the models are trained, the transfer gain $\mathcal{G}$ can be computed by evaluating $M_{S_i\to T}$ and M_T on the same test set. Note that the test set is scaled differently for evaluating $M_{S_i\to T}$ and M_T due to the origin of training data. This evaluation process is repeated 25 times with different randomly chosen target samples for each target sample size.

The evaluation results are shown in Figure 4, where each graph plots the mean of the transfer gain and its 95% confidence interval versus the number of available samples in the target domain, for different source domains.

The results for transfer scenarios 1 - 4, corresponding to the task of predicting *Write Response Time* (*WAvg*), for KVS, are shown in the graphs of the upper row in Figure 4. In all 4 scenarios, transfer learning gives a transfer gain $\mathcal{G} > 0$. The

TABLE II
THE SOURCE DOMAINS DEFINED USING THE TRACES IN TABLE I.

Source domain	Trace	Service	$P(X_S)$	T_S
D_{S1}	K1P	KVS	Periodic	*RAvg*
D_{S2}	K1F	KVS	FlashCrowd	*RAvg*
D_{S3}	K2P	KVS + VoD	Periodic	*RAvg*
D_{S4}	K2F	KVS + VoD	FlashCrowd	*RAvg*
D_{S5}	V1P	VoD	Periodic	*FR*
D_{S6}	V1F	VoD	FlashCrowd	*FR*
D_{S7}	V2P	VoD + KVS	Periodic	*FR*
D_{S8}	V2F	VoD + KVS	FlashCrowd	*FR*

TABLE III
THE TARGET DOMAINS DEFINED USING THE TRACES IN TABLE I.

Target domain	Trace	Service	$P(X_T)$	T_T
D_{T1}	K1P	KVS	Periodic	*WAvg*
D_{T2}	K1F	KVS	FlashCrowd	*WAvg*
D_{T3}	K2P	KVS + VoD	Periodic	*WAvg*
D_{T4}	K2F	KVS + VoD	FlashCrowd	*WAvg*
D_{T5}	V1P	VoD	Periodic	*ABR*
D_{T6}	V1F	VoD	FlashCrowd	*ABR*
D_{T7}	V2P	VoD + KVS	Periodic	*ABR*
D_{T8}	V2F	VoD + KVS	FlashCrowd	*ABR*

TABLE IV
THE TRANSFER SCENARIOS STUDIED IN THIS WORK. FOR EACH D_T THERE ARE 4 SOURCE DOMAINS D_{Si}, WHERE $T_S \neq T_T$.

Scenario	Target domain	Source domains
1	D_{T1}	$D_S, i \in [1, ..., 4]$
2	D_{T2}	$D_S, i \in [1, ..., 4]$
3	D_{T3}	$D_S, i \in [1, ..., 4]$
4	D_{T4}	$D_S, i \in [1, ..., 4]$
5	D_{T5}	$D_S, i \in [5, ..., 8]$
6	D_{T6}	$D_S, i \in [5, ..., 8]$
7	D_{T7}	$D_S, i \in [5, ..., 8]$
8	D_{T8}	$D_S, i \in [5, ..., 8]$

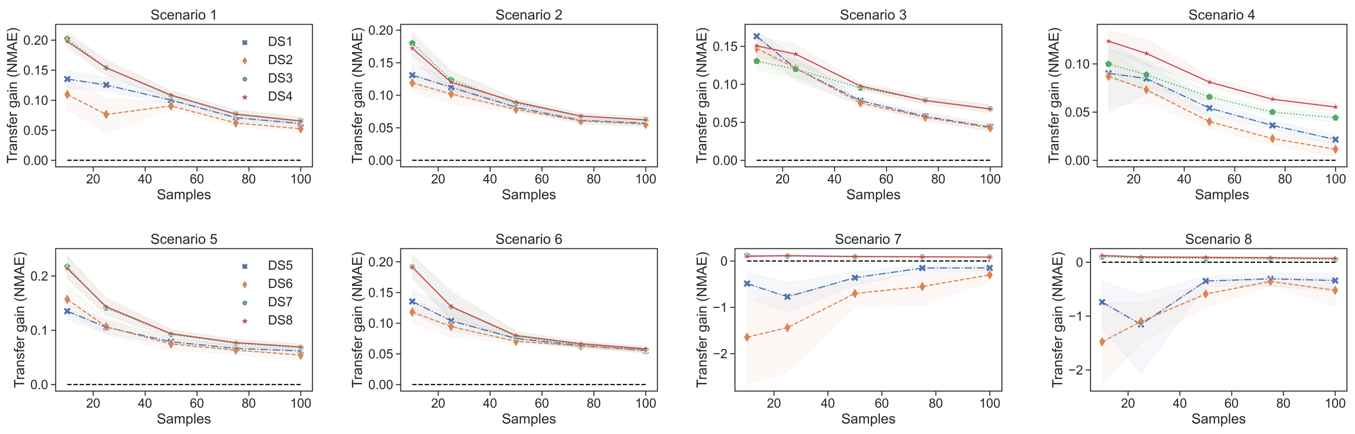

Fig. 4. Transfer gain $\mathcal{G}$ given source domains versus sample size in D_T for the 8 different target domains. See Table II and III for details regarding the domains, and Table IV for transfer scenarios. The legend in Scenario 1 applies for scenarios 1-4, whereas the legend in Scenario 5 applies for scenarios 5-8.

transfer gain is reduced when the number of target-domain samples increases, which corroborates the observations made in [3]. It is also clear that the choice of source domain is important, especially when few samples are available in the target domain. Selecting a source domain with co-located services, that is D_{S3} and D_{S4}, gives a higher transfer gain $\mathcal{G}$ compared to selecting a source domain that only had one service executing. The difference in transfer gain in the worst case corresponds to an NMAE of approximately 0.1 as observed in scenario 1. In scenario 3 the separation of the results for the source domains is not as prominent, especially for the case of only 10 samples in the target domain, as in scenarios 1, 2 and 4. We believe this is due to random effects when selecting samples from the target domain.

The evaluation results for transfer scenarios 5 - 8, corresponding to the task of predicting *Audio Buffer Rate* (*ABR*), for VoD, are shown in the graphs of the lower row in Figure 4. In scenario 5 and 6, corresponding to single-service target domains, transfer learning always gives a transfer gain $\mathcal{G} > 0$ regardless of the source domain, similarly to the KVS scenarios. Again, the source domains with co-located services (i.e. D_{S7}, D_{S8}) provide a higher transfer gain. In scenarios 7 and 8, where the target domain corresponds to a co-located service execution environment, the choice of source domain becomes more important. The penalty of selecting the wrong source for the target domain with co-located services is striking. Selecting a source domain with co-located services, that is D_{S7} or D_{S8}, gives $\mathcal{G} > 0$, whereas a negative transfer gain is observed for the other two sources. The difference in transfer gain in the worst case corresponds to an NMAE of approximately 1.8 in scenario 8. For source domains yielding $\mathcal{G} < 0$, we note that an increase in target-domain samples reduces the negative transfer gain but it does not reach $\mathcal{G} > 0$.

In summary, the benefit of choosing a good source is notable especially when few samples are available in the target domain. Further, the results show that the risk of obtaining a significant penalty in performance also decreases with a growing number of target-domain samples. We believe that the insights of source-domain impact can be generalized to performance modeling of other network and cloud services, and points towards the importance of source selection in order to balance the performance gains and the risk of penalty.

B. Evaluation of source selection approaches

In order to evaluate the two heuristic source-selection methods proposed in Section III, we consider the loss in transfer gain. If we correctly identify the best source, the loss would be zero and if we incorrectly identify the source, the loss would be a value greater than zero. The loss in transfer gain is computed as:

$$\ell_{\mathcal{G}} = \mathcal{G}_{i_{\text{ref}}} - \mathcal{G}_{i_*}, \tag{5}$$

where i_* is the index of the source selected by the source selection method, and

$$i_{\text{ref}} = \max_i \mathcal{G}_i, \quad \mathcal{G}_i = e_T - e_{S_i \to T}, \quad \forall i.$$

In the following, we evaluate the loss in transfer gain for two methods of source selection based on: (1) similarity between the domains measured in terms of symmetrized KL divergence $D_{sym}(X_S, X_T)$ as defined in Eq. 3; (2) and based on diversity of the source domains measured in terms of the differential entropy $h(X_S)$ as defined in Eq. 2.

For the computation of $D_{sym}(X_S, X_T)$ and $h(X_S)$, we need to estimate the underlying sample distributions of the source and target domains. This was done by first fitting a Gaussian mixture model, implemented in Scikit-Learn [17], to the data using variational Bayes inference [18, Chapter 10]. Prior to the modelling, data was standardized by removing the mean and scaling to the unit variance. For each transfer learning experiment, the source and the target data were standardized by the transformer fitted to the source data.

Gaussian mixture models were initialized with 15 components and fitted to the source datasets using variational inference [18, Chapter 10]. For the target data, the initial number of Gaussian components was 5 for the case where the

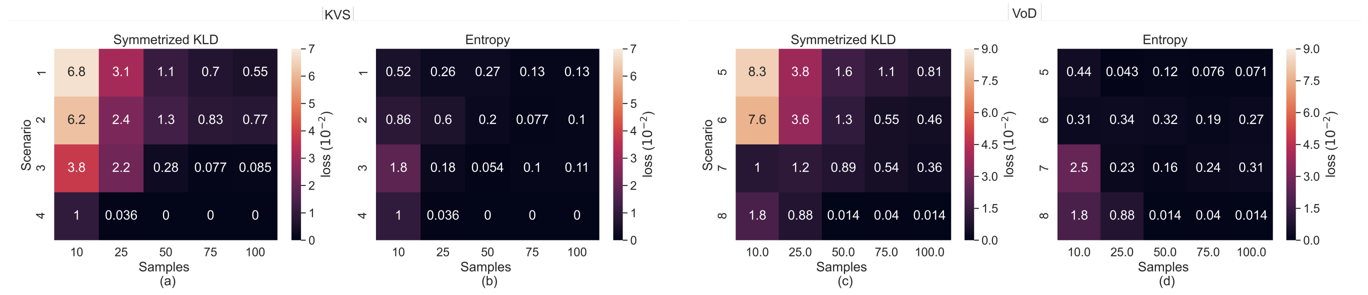

Fig. 5. Loss of transfer gain ℓ_G for source-selection strategies based on symmetrized KL divergence and differential entropy, respectively, for KVS and VoD. If a source is correctly identified by the source-selection method $\ell_G = 0$. Otherwise $\ell_G > 0$ and is dependent on the selected source.

target domain consisted of 10 samples, otherwise there was 15 components. This was done to reduce possible overfitting where there are limited number of samples in the target data. Given the estimated probability density functions by the Gaussian mixture models, $D_{sym}(X_S, X_T)$ and $h(X_S)$ were approximated using 10^6 Monte Carlo samples.

Figure 5 summarizes the source selection results in terms of loss in transfer gain ℓ_G for each transfer scenario in Table IV. Graphs (a) and (b) report on ℓ_G using symmetrized KL divergence and Entropy-based source selection for transfer scenarios 1 - 4 (KVS), whereas (c) and (d) report corresponding results for scenarios 5 - 8 (VoD).

It is notable that the entropy measure in majority of cases correctly identified the best source, that is $\ell_G \approx 0$, especially for scenarios 1, 2, 5 and 6 where the target domain is of low complexity corresponding to single-service traces (K1P, K1F, V1P, V1F). When the target domain is of high complexity, in the other scenarios, the most similar source is also a complex source, reducing the difference between the two approaches. Typically the loss is larger for a low number of samples and is reduced with additional samples. This can be explained by the fact that the lower number of samples we have in the target, the more the choice of a source impacts the final model, hence a non-optimal choice of source model will cause a larger performance penalty.

The performance of source selection using symmetrized KL divergence is more complex. In scenarios 1, 2, 5, and 6 corresponding to target domains defined using single service traces, the symmetrized KL divergence performed poorly in comparison to the entropy measure - the approach failed to correctly identify the best source in all cases, see Figure 5 (a) and (c). One explanation is that the single-service traces have a low complexity. This can be seen from Table V which shows the complexity of each source in terms of the differential entropy. A source selection based on the symmetrized KL divergence selects the source whose underlying distribution is most similar to its own. If the target inherently has a low complexity, transfering knowledge from a similar source can only marginally help as there is little added new knowledge. This becomes even more problematic when there only is a low number of samples in the target domain. In such cases, it becomes challenging to robustly estimate the symmetrized KL divergence between the domains.

TABLE V
DIFFERENTIAL ENTROPY $h(X_S)$ FOR ALL SOURCE DOMAINS.

Trace ID	$h(X_S)$	Trace ID	$h(X_S)$
K1P	-38	V1P	-38
K1F	-44	V1F	-36
K2P	-8.7	V2P	-9.1
K2F	-6.6	V2F	-7.5

All together, our results point at the effectiveness of source selection based on entropy which in turn suggest the importance of encouraging diversity in transfer learning. We believe this is one of the key contributions of this work. Indeed entropy is one information theoretic approach towards measuring the diversity. A future study is needed for investigation of various diversity measures and their usefulness in transfer learning.

C. Discussion

The general understanding is that transfer learning is most effective when there are limited samples in the target domain. The result of our experiments agree with this understanding. We argued that similarity-based techniques for source selection, which seek for the source domain that is most similar to the target domain, have inherent limitations in particular when there are limited samples in the target domain. This is due to the difficulties in robustly computing the similarity measures from a limited pool of data samples and, perhaps most importantly, the fact that such techniques do not explicitly encourage the diversity aspect. We then argued in favor of constructing source selection methods based on the diversity measure. A diversity-based source selection technique seeks for the most diverse source (the most complex source e.g. in terms of entropy). An advantage of the approach is that it does not depend on the availability of samples in the target domain. Under the assumption that there are sufficient number of samples in source domains, one can robustly measure diversity of the source domains. In this work, we used the Shannon differential entropy as the measure of diversity, which is one approach amongst other approaches. The source selection method based on this measure of diversity showed encouraging results in our experimental evaluations. Our results suggest the importance of diversity in the design of source selection techniques for transfer learning. A direction for future work is to examine other information theoretic metrics as diversity measures and their effectiveness in transfer learning.

Based on the results and the discussion above it is evident that intelligent source selection in transfer learning is essential for *performance modeling* of services executing in a dynamic environment, as the approach not only improves model performance when few samples are available but also avoids negative transfer stemming from irrelevant knowledge. From a management perspective, source selection enables faster and more robust deployment of performance models, also when the environment upon which the service executes change due to re-orchestration or scaling of resources. Although it is not within the scope of this paper to define a management architecture for data-driven functions for network and service management, we argue based on the results that a *model store* containing source-domain representations and a diversity-based source-selection function, as outlined in Figure 2, are essential components. Each time a new model with high performance is learned it should be added to the model store, that is, it is populated by the network or service operator over time. It remains however additional work to further define the components and their interaction, as well as evaluating the performance of such model store.

Regarding computational complexity, the proposed diversity-based source selection technique (entropy of source domains) may be preferred to the similarity-based technique (based on measuring KLD divergence between the source and target domains) in dynamic network environments. The tasks, or the environment settings, undergo changes during their lifetime which is translated into the changes in the representative samples of the target domain. In such environments, when using a similarity-based technique, there would be a need for continuously re-selecting the sources that benefit the target domain. On the contrary, the diversity-based technique operates independent from the target domain.

VI. Related Work

Transfer learning has received considerable attention in areas such as image processing and natural language processing (NLP), and also computer systems and networks. This section provides a review of relevant literature.

One of the main challenges in transfer learning is automated selection of a good source domain for a given target domain, and several works rely on statistical similarity (or distance) between domains. The authors of [6] proposed a source-selection strategy based upon distance (χ^2-divergence, Maximum Mean Discrepancy, Wassertein distance and KL divergence) between source and target domains. Further, in [19] the authors presented a novel distance metric between domain tasks, the H-score, for determining the performance of transferred representations. In [20] a method for guided sampling is proposed which exploits knowledge from source domains that are determined similar to the target domain.

In [21] an information theoretic framework is presented that is used to understand the source Convolutional Neural Networks (CNNs) prior using them in a target domain. More specifically, the framework enables automatic ranking of source CNNs for a given target. In [4], the authors investigated the transferability of features in a neural network for image processing and show that the transferability decreases when the distance between source and target tasks increases. Further, in [22], the authors investigate how transferable the layers of a neural-network model in the field of NLP are and show that the semantic similarity of the source and target tasks impacts the transferability of the neural-network models.

This paper addresses the limitation of domain similarity approaches, that has its roots in limited knowledge and short lifetime of the execution environment that constitutes the target domain in dynamic clouds, by introducing a source-domain diversity metric.

Recently, transfer learning has also been used in other areas including performance predictions in the network and data center domains. A number of studies have looked into using transfer learning for identifying the best application configurations. For example, in [23], the authors present a transfer learning approach for performance prediction of configurable software across different hardware platforms. The source model is built using a regression tree from a random sample of configurations on the source hardware, then a linear regression model transfers the results into the target domain. Further, in [24], an empirical study was performed on four software systems, with varying software configurations and environmental conditions, to identify the key knowledge pieces that can be exploited for transfer learning. Insights from the paper include that for non-severe hardware changes, a linear transfer model can be deployed across environments. However, virtualization may hinder transfer learning. Further, even for some severe environmental changes when the performance distributions are similar there is a potential for learning a non-linear transfer function. In comparison, this paper adopts a more advanced transfer-learning approach, and a source-selection heuristic based on diversity, and also studies the challenges of transfer learning in a different use case. Further, in [25] the authors propose and evaluate a transfer-learning framework, BEETLE, for identifying and learning from the most relevant sources for performance optimization and configuration of software systems. However, the paper does not discuss the challenges related to limited target-domain knowledge as addressed in this paper.

In [26] the authors propose a deep-learning based approach for identifying software configurations for a high-performing application, when limited resources are available for data collection in the target domain by combining information from exhaustive observations collected at a smaller scale with limited observations collected at a larger target scale. The work has similarities with this paper, but the neural networks are trained given feature sets, outputs, and assumptions that do not generalize to the domain of this paper, nor do the authors study diversity heuristics for source selection.

In [27] the authors study the challenge of predicting server behavior and proposed a random-forest-based transfer learning approach. The challenge is that small data centers exhibit too few labeled training examples to build a proper model, since the distribution of problematic and normal server behavior is highly skewed. The approach is to combine training examples from several small data centers into one pool of training samples. A model for the target domain is then built based on samples from all small data centers that resembles the target domain good enough.

Finally, the authors of [28] explore an ML method for fault localization using IT infrastructure event data. Transfer learning is used for incrementally enriching the models, and hence provides a method for online fault localization. However, the results and insights cannot be generalized to the challenges targeted by this paper.

VII. Conclusions

In this paper, we proposed and evaluated two heuristic methods for automated source selection in transfer learning for improved performance modeling of network services. The source-selection methods are based on domain similarity (symmetrized Kullback-Leibler Divergence) and source-domain diversity (differential entropy).

We evaluated the impact of source selection in multiple scenarios with realistic data sets obtained from a testbed for two different network services under varying load, namely a Video-on-Demand and a Key-Value Store service. We provide empirical evidence showing that the transfer gain is highly dependent on the source domain. In scenarios where the target domain constitutes a complex shared service environment we observe a significant penalty from selecting the wrong source. The positive impact of transfer learning, and also the penalty, is reduced with an increased number of target-domain samples.

Moreover, as the key contribution of this paper, our empirical results suggest that a source-selection strategy, in environments with inherent limitations in data availability stemming from the network and cloud dynamics, should encourage source-domain diversity (entropy) rather than domain similarity (KL divergence). Additional work is however needed to provide general guidelines for source-domain selection for data-driven modeling supporting management and automation of networked services.

References

[1] 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, 2009.

[2] Z. Wang, Z. Dai, B. Póczos, and J. Carbonell, "Characterizing and avoiding negative transfer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 293–11 302.

[3] F. Moradi, R. Stadler, and A. Johnsson, "Performance prediction in dynamic clouds using transfer learning," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2019, pp. 242–250.

[4] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in neural information processing systems*, 2014, pp. 3320–3328.

[5] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics New York, 2001, vol. 1, no. 10.

[6] K. Bascol, R. Emonet, and Fromont, "Improving domain adaptation by source selection," pp. 3043–3047, Sep. 2019.

[7] R. Yanggratoke, J. Ahmed, J. Ardelius, C. Flinta, A. Johnsson, D. Gillblad, and R. Stadler, "A service-agnostic method for predicting service metrics in real time," *International Journal of Network Management*, vol. 28, no. 2, p. e1991, 2018.

[8] "Vlc, 2016. [online]. available: http://www.videolan.org/vlc/."

[9] "Voldemort, 2016. [online]. available: http://www.project-voldemort.com/voldemort/."

[10] "Sar, 2016. [online]. available: http://linux.die.net/man/1/sar."

[11] I. Ari, B. Hong, E. L. Miller, S. A. Brandt, and D. D. Long, "Managing flash crowds on the internet," in *11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems, 2003. MASCOTS 2003.* IEEE, 2003, pp. 246–249.

[12] J. Ahmed, T. Josefsson, A. Johnsson, C. Flinta, F. Moradi, R. Pasquini, and R. Stadler, "Automated diagnostic of virtualized service performance degradation," in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2018, pp. 1–9.

[13] C. Shi, Y. Li, J. Zhang, Y. Sun, and S. Y. Philip, "A survey of heterogeneous information network analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 1, pp. 17–37, 2016.

[14] "Keras, 2018. [online]. available: https://keras.io/."

[15] "Tensorflow, 2018. [online]. available: https://github.com/tensorflow/tensorflow."

[16] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[18] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[19] Y. Bao, Y. Li, S. Huang, L. Zhang, L. Zheng, A. Zamir, and L. Guibas, "An information-theoretic approach to transferability in task transfer learning," in *2019 IEEE International Conference on Image Processing (ICIP)*, Sep. 2019, pp. 2309–2313.

[20] P. Jamshidi, M. Velez, C. Kästner, and N. Siegmund, "Learning to sample: Exploiting similarities across environments to learn performance models for configurable systems," in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM, 2018, pp. 71–82.

[21] M. J. Afridi, A. Ross, and E. M. Shapiro, "On automated source selection for transfer learning in convolutional neural networks," *Pattern recognition*, vol. 73, pp. 65–75, 2018.

[22] L. Mou, Z. Meng, R. Yan, G. Li, Y. Xu, L. Zhang, and Z. Jin, "How transferable are neural networks in nlp applications?" *arXiv preprint arXiv:1603.06111*, 2016.

[23] P. Valov, J.-C. Petkovich, J. Guo, S. Fischmeister, and K. Czarnecki, "Transferring performance prediction models across different hardware platforms," in *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering*. ACM, 2017, pp. 39–50.

[24] P. Jamshidi, N. Siegmund, M. Velez, C. Kästner, A. Patel, and Y. Agarwal, "Transfer learning for performance modeling of configurable systems: An exploratory analysis," in *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*. IEEE Press, 2017, pp. 497–508.

[25] R. Krishna, V. Nair, P. Jamshidi, and T. Menzies, "Whence to learn? transferring knowledge in configurable systems using beetle," 2019.

[26] A. Marathe, R. Anirudh, N. Jain, A. Bhatele, J. Thiagarajan, B. Kailkhura, J.-S. Yeom, B. Rountree, and T. Gamblin, "Performance modeling under resource constraints using deep transfer learning," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2017, p. 31.

[27] J. Bogojeska and D. Wiesmann, "Transfer learning for server behavior classification in small it environments," in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2018, pp. 1–9.

[28] Y. Shehu and R. Harper, "Towards improved fault localization using transfer learning and language modeling," in *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium (NOMS)*. IEEE, 2020.