UNIVERSITY OF CALIFORNIA
Los Angeles

Real-Time Cost-Aware Machine Learning at the Edge

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Orpaz Goldstein

2021

ABSTRACT OF THE DISSERTATION

Real-Time Cost-Aware Machine Learning at the Edge

by

Orpaz Goldstein
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2021
Professor Majid Sarrafzadeh, Chair

Exponential growth in the need for low latency offloading of computation was answered by the introduction of edge networks. Since these networks are essentially isolated islands of computing, current prevalent centralized approaches to training learning agents should be adapted to account for the decentralized nature of this new network structure. Since these networks are designed for low latency, cost-awareness must be built into machine learning models when dealing with data streams. Additionally, in order to debias or expand on the locally available data while maintaining edge benefits, multi-agent systems should be constructed to allow for limited coordination outside of a local node.

To address these issues, we suggest a novel end-to-end solution that supports the lifetime of a learning agent on the network. We reevaluate how learning agents receive information on an edge network and explore ways for them to communicate and coordinate with other agents efficiently while maintaining context. This thesis will dive into cost-awareness as it pertains to data acquired sequentially and messages exchanged on a network. Additionally, we will showcase our solution for knowledge transfer between remote agents that preserves all the benefits of running in a decentralized network environment.

The dissertation of Orpaz Goldstein is approved.

Ramin Ramezan

Baharan Mirzasoleiman

Ravi Netravali

Guy Van den Broeck

Majid Sarrafzadeh, Committee Chair

University of California, Los Angeles

2021

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First, I would like to thank Professor Majid Sarrafzadeh for his guidance and insights throughout the last few years. His support and encouragement not only made this time spent navigating my research pleasant but also taught me qualities I believe will guide my life beyond my Ph.D. Secondly, I would like to thank Professor Guy Van den Broeck, who patiently explained how to report research results properly and helped me write my first research paper. Thirdly, I would like to express my deep appreciation to the doctoral committee, Professor Ravi Netravali, Professor Ramin Ramezani, and Professor Baharan Mirzasoleiman, for their constructive feedback and invaluable advice. Finally, I want to thank my dear friends and members of the UCLA eHealth and Data Analytics Research Lab who helped me during this time. I greatly value these friendships and collaborations, and I believe that our connection will extend beyond this period.

VITA

2012–2016 B.Sc. of Software Engineering, Shenkar College of Engineering and Design

2016–2017 M.Sc. of Computer Science, University of California, Los Angeles

PUBLICATIONS

Goldstein, O., Kachuee, M., Shiell, D. and Sarrafzadeh, M., 2020. Real-Time Decentralized knowledge Transfer at the Edge. *arXiv preprint arXiv:2011.05961.*

Ovalle, A., **Goldstein, O.**, Kachuee, M., Wu, E., Hong, C., Holloway, I. W., & Sarrafzadeh, M. (2021). Leveraging Social Media Activity and Machine Learning for HIV and Substance Abuse Risk Assessment: Development and Validation Study. *Journal of medical Internet research, 23(4)*

Goldstein, O., Shah, A., Shiell, D., Rad, M.A., Pressly, W. and Sarrafzadeh, M., *2020, September. Edge Architecture for Dynamic Data Stream Analysis and Manipulation. In International Conference on Edge Computing (pp. 33-49). Springer, Cham.*

Kachuee, M., Karkkainen, K., **Goldstein, O.**, Darabi, S. and Sarrafzadeh, M., *2020. Generative Imputation and Stochastic Prediction. IEEE Transactions on Pattern Analysis and Machine Intelligence.*

Goldstein, O., Kachuee, M., Karkkainen, K. and Sarrafzadeh, M., 2020. Target-Focused Feature Selection Using Uncertainty Measurements in *Healthcare Data. ACM Transactions on Computing for Healthcare, 1(3), pp.1-17.*

Kärkkäinen, K., Kachuee, M., **Goldstein, O.** and Sarrafzadeh, M., 2019. Cost-Sensitive Feature-Value Acquisition Using Feature Relevance. *arXiv preprint arXiv:1912.08281.*

Goldstein, O., Kachuee, M., Karkkainen, K. and Sarrafzadeh, M., 2019. Target-Focused Feature Selection Using a Bayesian Approach. *arXiv preprint arXiv:1909.06772.*

Kachuee, M., Karkkainen, K., **Goldstein, O.**, Zamanzadeh, D. and Sarrafzadeh, M., 2019. Cost-sensitive diagnosis and learning leveraging public health data. *arXiv preprint arXiv:1902.07102.*

Kachuee, M., **Goldstein, O.**, Kärkkäinen, K., Darabi, S. and Sarrafzadeh, M., 2018, September. Opportunistic Learning: Budgeted Cost-Sensitive Learning from Data Streams. In *International Conference on Learning Representations.*

Kachuee, M., Karkkainen, K., **Goldstein, O.**, Zamanzadeh, D. and Sarrafzadeh, M., 2019. Nutrition and health data for cost-sensitive learning. *arXiv preprint arXiv:1902.07102.*

Lipsitt, J., Batteate, C., **Goldstein, O**. and Jerrett, M., 2018, November. Physical Activity through Sustainable Transport Activities (PASTA) in Los Angeles. In *APHA's 2018 Annual Meeting & Expo (Nov. 10-Nov. 14). American Public Health Association.*

Goldstein, O., 2018. Zero-Shot relation extraction from word embeddings (*Masters dissertation, UCLA).*

Hojaiji, H., **Goldstein, O.**, King, C.E., Sarrafzadeh, M. and Jerrett, M., 2017, October. Design and calibration of a wearable and wireless research grade air quality monitoring system for real-time data collection. In *2017 IEEE Global Humanitarian Technology Conference (GHTC) (pp. 1-10). IEEE.*

CHAPTER 1

Introduction

As machine learning ubiquitously enhances what can be done with computers and devices, the influx of network adjacent models will continue to increase. Many of these models will essentially be a reincarnation of robustly pre-trained public models or models trained on very similar data, solving highly comparable tasks. Optimally, collaborative learning could take place, enhancing and debiasing all equivalent models, to better benefit all such machine learning tasks [SWL02, SZC20]. In reality, however, many models will require private data in order to become highly localized finely tuned solutions, and disinclination towards sharing private data will prevent this idealistic approach. Transferring knowledge in a decentralized setting should allow models to retain their local insights, in turn allowing for local flavors of a machine learning model. This approach suits the decentralized architecture of edge networks, as a local edge node will serve a community of learning agents that will likely encounter similar data.

Moving information between models has been addressed in transfer-based methods [TS10, RMK05, YZH18]. While these methods predominantly deal with training a target model once a source model has finished its training process, we are interested in the case where actively learning models can still benefit each other in real-time. Moreover, knowledge transfer methods are largely evaluated on their applicability to learning a comparable but distinct task. In our problem setup, we wish to accurately inherit the source model's ability to classify or predict the data.

Addressing the decentralized component of collaborative learning, federating the process of training learning agents has been suggested [KMY16, HAA20, MMR17, HTT19, WYS20]. While federation captures the real-time model construction we are interested in, it is applied

towards building a centralized model aggregated from distributed data sources, thus losing the local variant that we are after.

Methods for efficient message passing between learning agents over a decentralized network have been proposed in Peer-to-Peer (P2P) models [Sha09, BGP06]. P2P message passing allows for efficient communication over a network and simplifies the decentralized learning model by defining a single knowledge transfer function that applies to all learning agents. Developing an adaptive knowledge transfer term for potentially nonlinear and deep learning models, we want to learn a different granularity of the data in each model layer, making a global knowledge transfer term is too vague for our case.

We propose a dynamic information acquisition and exchange network allowing learners to take advantage of information actively being generated or learned elsewhere. Our network is optimized for cost-aware knowledge transfer between agents, defined as models moving data horizontally between layers of source and target models. Our method allows for source agents that are still learning or are continuously updating their model. Our knowledge transfer prioritizes local data, preserves local insights, and adds on remote information. Accommodating for running agents on a real-world edge network, local agents can define their source contributors dynamically. Consideration could be based on their costs, the need for debiasing their local data, or improving accuracy on a sparsely or never-before-seen type of data. Knowledge transfer is done pairwise between a source and a target model layers, thus allowing us surgical precision to transfer information between shallow models as well as deep learning models.

The rest of this manuscript is organized as follows. Chapter 2 reviews the current relevant literature. Chapter 3 introduces an architecture for the support of learning agents on decentralized networks. In this chapter, we explore efficient communication methods for agent collaboration and coordination. Chapter 4 suggests a cost-aware, highly frugal approach to feature selection for data incrementally available. Chapter 5 presents an approach based on knowledge distillation for real-time knowledge transfer on edge networks. Finally, Chapter 6 concludes the thesis.

CHAPTER 2

Related Work

2.1 Dynamic Data Stream Edge Architecture

2.1.1 Computation Delegation

A lot of recent work that suggests improvement to current edge architecture is centered around reducing latency and increasing efficiency. This reduction could be made by combining the availability and low latency quality of the edge while inheriting the advantages of the data-center-based service delivery [CHM14], or by moving away from a centralized cloud approach to a more decentralized one [GME15]. On top of that, the Quality of Service (QoS) issue and understanding the benefits of offloading computation within an edge network become essential when scaling up service to more learning agents. An approach to periodically distribute incoming tasks is described in [SYY17], showing that internally distributing tasks can result in a more significant number of jobs processed. [YIJ17] extends the notion of offloading computation and computes a delay metric to find the best neighbor node in a network to offload. Multi-access edge computing (MEC) is surveyed in [MB17] as a promising target for improving the performance of delegated compute to an edge network. It compares different MEC concepts in terms of computation and control placement.

2.1.2 Computation Orchestration

Running models that require state retention on an edge network could be challenging to orchestrate. In the centralized case, federated models were proposed to compute an aggregate of all model updates and broadcast them back to the sub-models [KMY16]. However, this

centralized approach will not work well on streaming data or generalize to all possible state retaining applications. Some work addresses the need for internal communication and passing of information between models in the decentralized system. [WTS19] explores the benefits of message passing to compute the same federated aggregation and efficiently compute a decentralized, federated model. [MAS18] discusses the treatment of data streams on an edge network for the consumption of learning models. The locality of computation offloading and the minimization of raw data routed to a centralized location are highlighted as necessary for the overall performance of IoT supporting edge networks. Our work in this thesis presents a design that adheres to the same decentralized approach, focused on maximizing efficiency in handling data streams from a multitude of clients.

2.2 Cost Aware Feature Selection

2.2.1 Uncertainty Measurements

Uncertainty measurement in a machine learning model flows from applying a probabilistic approach to learning, also known as Bayesian learning. Sampling a trained probabilistic model for latent variables allows us to capture the inherent uncertainty in the model. Identifying the uncertainty within a model is important for giving accurate guarantees for what a model can efficiently predict [KA13]. For example: when trying to predict between two types of heart disease, one type could be more common and make the vast majority of samples available. Drilling into the results could show that the less common but more fatal disease shows the high variance in the classification predicted by the model for that class, indicating uncertainty in the more serious fatal type of heart disease. Developing models that are minimally uncertain of their predictions is the key to incorporating machine learning in accuracy-oriented domains such as healthcare. The usage of Gaussian weight distributions to estimate the uncertainty was first discussed in [DL91]. Later work includes [Bis06, Mur12, Gha15] and many more. The benefits of uncertainty measurement in the healthcare domain flow from the statistical rigor of the outputs provided by a model. In [KA13], the importance of minimizing uncertainty in biology is discussed, and [HO13] call

biologists to shift to Bayesian statistics and provides reasoning for the need for uncertainty measurements in this field.

Application of uncertainty to feature selection robustness appears in Same Decision Probability (SDP) [CXD12a], which measures the effect of feature acquisition on the shift in the decision-making process of a model. SDP measures the uncertainty in the model while acquiring features and reasons on stopping criteria based on a threshold of confidence and budget. More recently, an expected SDP query and an optimal feature selection algorithm based on SDP were proposed [CDB17]. SDP queries are generally PP^{PP}-*complete*, which makes it costly for many high-dimensional real-world applications.

2.2.2 Feature Selection

Classic approaches to feature selection focus on maximizing information gain and inferring feature relevance [BHP97, GE03]. Cost awareness for feature selection is prevalent where machine learning is applied to concrete problems such as in Health informatics. There, feature selection methods take into account real-world costs associated with the acquisition of features and the need to maintain a budget. Costs of tests, physician time, patient discomfort should all be taken into account when reasoning on which feature is to be acquired using cost-sensitive decision methods or active sensing [FCB07, YKR09]. In addition to costs, changes in data availability might call for iterative feature aggregation in training time, requiring an online cost-sensitive budgeted approach [KGK19b]. When managing a budget for features acquired iteratively, a deep reinforcement learning approach for optimizing a trade-off between the expected classification error and the feature cost was recently suggested [JPL19]. While it performs well for feature acquisition tasks, reinforcement learning usually runs for long periods before outputting a decision, making it unsuitable for our tasks.

2.2.3 Data Imbalance

Real-world data tends to be imbalanced, especially in fields such as healthcare; some conditions or variants of a disease are more common. Some targets carry more significance or

are more relevant to a specific diagnosis. The acquisition of relevant data is made possible using an active learning approach [NDR18], or by reducing redundancy in acquired features while maintaining relevance, [JLO19]. Contributing to the imbalance is also the sparseness of data. Due to the high dimensionality of the data, not all data points will have all features. For medical domain feature selection and prediction, ensemble methods have been used to reduce the effects of imbalanced data, and inherent missingness [HYJ16, LLY06], and more recently with a robust feature selection framework [ZZZ18]. While addressing the imbalance in data is closely related to our work, the acquisition of pertinent features to a specific target of focus is not addressed. The selective budgeted acquisition of a subset of features out of a massive amount of available data in the medical domain was also explored as a way to sift through vast amounts of information [LY17].

2.3 Decentralized Information Absorption

2.3.1 Knowledge transfer

Knowledge transfer in machine learning is commonly used to leverage a model trained on a source task in order to improve training a model for a corresponding target task. One prevalent method is Transfer Learning [TS10, RMK05, YZH18], which allows for reusing knowledge learned on a source model to a target model by recycling learned parameters and limiting further training to the lower layers of the source model. Transfer Learning can improve the time it takes to learn the new task in the same domain as the source task and the final performance of the model. Distilling knowledge from an ensemble of source models [HVD15, LKS20] is another approach for knowledge transfer where the original set or a subset of the data used to train the source models is leveraged. Distilling knowledge from source to target is done by defining a cross-entropy loss between outputs of source and target softmax layers. Recently, a method for zero-shot knowledge distillation [NMS19, MS19] was proposed, where a transfer set is extracted directly from the source model by sampling the Dirichlet distribution learned for each class for softmax probabilities. Sampled probabilities are then used to construct Data Impressions that correspond to model output per class,

replacing conventional input data. Transferring knowledge can also be achieved by using neuron activation [HLY19, RZK19]. Here, instead of extracting knowledge by considering the magnitude of neuron responses to data, minimizing the difference in neuron activation between source and target is used to train our target model. When transferring knowledge between models, some information learned on the source model might be insignificant or even harmful to the target model. Additionally, exploring where data should be injected into the target model can benefit our knowledge transfer. Therefore, defining meta models to decide what data should be transferred and to what location in the target model could positively impact knowledge transfer [JLH19].

2.3.2 Federated Learning

Federated Learning [KMY16, HAA20, MMR17, HTT19, WYS20], is a method for updating a centralized model using a training set that is distributed among multiple users. Federation allows local data to remain private by collecting local updates to a base model and aggregating them in a centralized location. After a centralized model has been updated, it is shared back with agents. Building on Federated Learning, Federated Multi-Task Learning [SCS17, CB19, YLS20] considers the known shared structure between pairs of models, improving the effectiveness of samples extracted from each of the local models. In order to mitigate the overhead required for decentralized updates to a centralized model over a network, a federated averaging method [MMR16] is suggested. Here, in each iteration, a random set of local models is selected to run a single step of gradient descent using local data and transmit back the results, which are then averaged. Federation works well when all we care about is the global accuracy term. However, when we wish to prioritize local information, adopted knowledge should be more carefully integrated.

2.3.3 Peer-to-Peer

Peer-to-Peer (P2P) agent communication over a defined network structure is described by Gossip Algorithms [Sha09, BGP06]. Considering a network with a known structure that

might change over time, pairwise communication of agents is proposed to replace a centralized network structure. Decentralized Collaborative learning based on Gossip Algorithms [VBT17] leverages a known network structure where neighbors are agents learning similar models. This collaborative approach uses an asynchronous update phase, where parameters are collected from neighbors and used to update a local model. This approach allows for some individuality in models but is restricted to networks where a structure is known. Smoothing terms in optimization and knowledge propagation that is agnostic of model structure prevent this model from applying to our problem.

CHAPTER 3

Edge Architecture for Dynamic Data Stream Analysis and Manipulation [1]

In this chapter, we will start developing our architecture for supporting learning agents in the decentralized world. We focus on the following points.

i Maintaining edge-level low latency and availability to physically close users while extending the availability of produced data streams globally with low latency, without going through a centralized location.

ii Extending the definition of computation on an edge network to be more dynamic in nature. Delegated computation or usage of a function or a model that is not on a user's local node should be handed off in-network to potentially multiple locations for added efficiency instead of reaching a centralized data center. Delegated computation should communicate meta-data back and forth to coordinate.

iii Providing an architecture where produced data streams, or the output of a model that takes that stream as input, is available to be consumed by multiple consumers globally. Similarly, input to an edge function that depends on multiple data streams produced in various geographical locations is available instantly. Consequently, to support this kind of global availability, a modular approach to computation delegation is considered. Supporting modularity of the edge, manifested in the chaining of edge functions and

[1] **This chapter is based on "Goldstein, O., Shah, A., Shiell, D., Rad, M.A., Pressly, W. and Sarrafzadeh, M., *2020, September. Edge Architecture for Dynamic Data Stream Analysis and Manipulation. In International Conference on Edge Computing (pp. 33-49). Springer, Cham.***

decentralized learning models on an edge network, requires adding context retention to the edge.

3.1 Problem Definition

In the serverless world, the idea of functions as a service (FaaS) is rapidly becoming the preferred solution for IoT use-cases. Typically, serverless functions are of limited expressiveness and are designed to scale, preventing state information from being stored between executions [BCC17]. Datastream-related computation delegation is, in turn, thought of as a rigidly defined task delegation. Unlike standard FaaS usage, we are interested in determining computation paths for pipe-lining execution of edge models and functions and provide a mechanism to orchestrate this execution and exchange of meta-data between functions and models. Data streams related tasks, such as video augmentation and analysis, might benefit from function chaining while retaining context, with multiple forking tasks based on slightly different final product requirements of different consumers. Similarly, a consumer who relies on data produced by multiple producers will benefit from function chaining. Another example is context-dependent models on the edge. In order to train and test machine learning models delegated to an edge network, context must be retained and potentially shared between locations.

For example: Consider a network camera that uploads a live video feed. One consumer with access to the raw data wants to run a facial recognition model on the feed, and another wants to augment the video and add bounding boxes to elements in the feed. Each of these consumers can define a function/model that directly subscribes to the availability of that video feed frames on the network. Once they are available, each function picks them up and computes a result that is in turn published as available on the network for the original and additional consumers to pick up.

We then wish to retain some contextual information while data is handed off from function to function and eventually returned to a consumer. Recent work suggests the addition of ephemeral storage for short-lived edge compute tasks to achieve near real-time performance

[KWS18]. This fine-grained scalability appears to be vital in developing future serverless applications that could process multiple data streams in parallel and achieve real-time performance. Whether that refers to facial recognition on mobile devices, flying drones, or driving intelligent cars, support for this computation with low latency is crucial [SCZ16]. Additionally, since users of an edge network will be geographically distributed, the low latency availability of an edge function should be unbound to a specific location or edge node. Similarly, data produced in one location should be simultaneously available as input to functions and models across all edge nodes.

For example: Suppose a user is training a facial recognition model on the edge using a video feed from a mobile camera that he carries with him. In that case, that model should be available with low latency regardless of a user's physical location. If multiple users are training the same type of facial recognition model, it might make sense to share the data stream with all users globally. Conversely, if the data stream is private, we should make each of the mini models available globally and utilize what they learned to minimize training time across the board.

Figure 3.1 Plots our data stream use-cases over a desired edge network architecture, where data produced are globally available. Any edge function/model can utilize output from other functions and continue computation while retaining context.

3.2 Proposed Edge Architecture

3.2.1 Background

A natural candidate to provide the foundation of an edge network is a content delivery network (CDN). A CDN can be seen as a specialized use case of an edge network, as it is a low latency distributed network in close physical proximity to consumers. A CDN is concerned with caching content as close as possible to end-users so multiple consumers could consume it with the least possible latency. Unlike the multi-purpose edge network, a CDN does not provide clients with an access point into its network. A CDN does not outsource

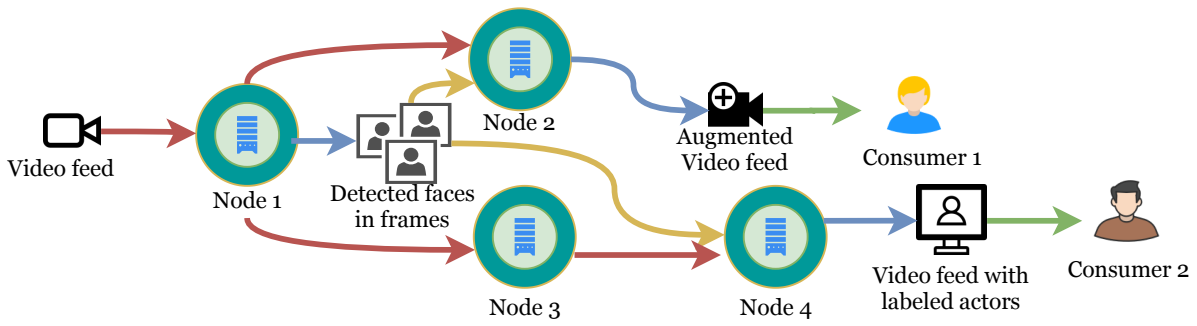


Figure 3.1: Red arrows represent the original video input available to edge nodes. Yellow arrows are intermediate context being fed along with data. Blue arrows are outputs from edge functions/models, and green arrows represent data consumed by an end consumer. This figure shows a video feed uploaded to an edge network, where different nodes ingest it; each potentially outputs a result that in turn could be again consumed.

computation to users as a service or allow them to upload any code to the CDN network. To utilize a CDN as an edge network, low latency edge nodes that enable users to access the larger network are needed, combined with support for requesting compute resources.

3.2.2 CDN as a platform for EdgeCompute

We propose an edge network implemented over an existing sizeable commercial content delivery network (CDN). By leveraging an existing global network of points of presence (PoPs) that are deployed in large metro areas around the world, we get physically close to a large portion of the population on the planet. We can then construct a globally available edge presence with exceptionally low latency from outside of the network to edge nodes and internally between our PoPs, from edge node to edge node.

We leverage existing CDN features when extending the network. The CDN is made to handle the load balancing of traffic while considering latency. A CDN has built-in support for routing incoming traffic to the nearest PoP with the capacity to process the request efficiently. Traffic routing and management and fail-overs from PoP to PoP is then taken care of by CDN logic. Since the CDN has a global presence, that translates to low latency hops globally. For an edge network user, that means that while he only maintains a connection with a

local edge node, he can still benefit from a low latency global computation delegation. A CDN network has valuable security features in place, such as web application firewall (WAF) and authentication to our network. Further benefits include rate-limiting of traffic and the ability to use the CDN cache when necessary. Inherently, edge compute traffic enjoys the same benefits. Lastly, we make use of a load-aware auto-scaling mechanism. On a CDN, when a piece of data becomes popular and frequently requested, it makes sense to replicate that piece of data to more cache servers so it could be served more efficiently from more servers without hurting the performance of the network. The auto-scaling mechanism is used when scaling up our edge compute tasks, and as we describe later, auto-scaling will be utilized when we augment the network with a new kind of data store.

3.2.3 Extending a CDN

3.2.3.1 Virtualization

To support edge computing on our network and generalize CDN services, we allow users to upload code to be run on our network in a virtualized environment. Allowing each machine to support multiple users operating in isolation on the same hardware resources, we bound models/functions to a user-space container on a device. The container approach for OS-level virtualization of resources is highly scalable and can be further improved by container orchestration software, automating global management and scaling of containers. Containers are fast and easy to deploy using provided packaging and deployment tools while allowing for individualized system configuration at deployment time. Containers require a small amount of resources to maintain, and their footprint on a system is minimal. We use Docker as our container platform and support uploading Docker images containing functions or models to be run on our edge network.

3.2.3.2 Data store and context retention

Unlike the typical edge network implementation, the edge functions uploaded by our users do not need to integrate with an HTTP request logic library in order to obtain data as input.

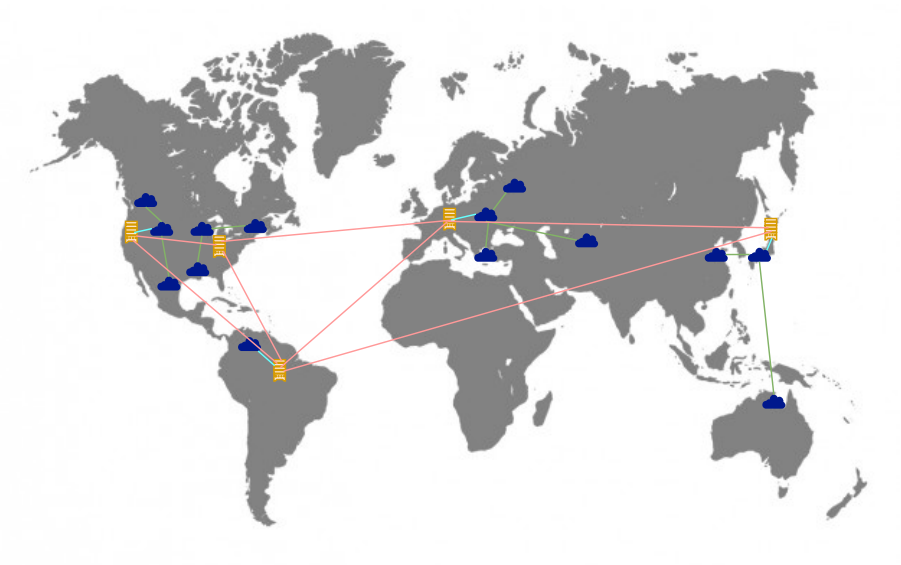


Figure 3.2: Edge CDN Architecture. Red connections are strongly consistent database connections. Green connections are edge node to edge node connectivity. Blue connections are edge node to database connections.

Instead, we implement a distributed globally available edge key-value store (EKV). Using a persistent, globally distributed data store provides an edge network with the ability to retain context between function executions or an online learning model updated from multiple nodes around the network. An EKV provides a producer with a low latency access point to upload data streams, after which the data propagates through the edge network quickly to become available globally. Equivalently, consumers can access data streams produced remotely on their local edge node instantly.

Figure 3.2 shows our CDN-based architecture; Global decentralization and low latency availability are critical in a network designed for massive-scale data stream input. Figure 3.3 shows our layering scheme and the path of a user request interacting with our network. Requests from the outer layer close to a user propagate internally using CDN mechanics augmented with a globally available storage system.

3.2.3.3 Computation channels

Once data is generated on the producer side, it is pushed to a local edge instance of our EKV. Once uploaded, we wish to notify functions and models who are dependant on this data that a new piece of data is available to be consumed. For edge functions or models to become aware of the new data availability, we implement computation channels that are essentially named communication channels that functions or models can subscribe to and receive data from. In practice, to let subscribers know when to pull data from the EKV store, a user implements a publish call when a producer has finished uploading data to EKV. This call allows functions and models that are subscribed to the computation channel dedicated to the data produced by a specific producer to get data from EKV and start working. Similarly, an implemented consumer function, model, or end-user subscribes to the channel that matches the data they wish to consume.

To create computation channels, we are using the pub/sub paradigm. This approach provides us the scalability and modularity required by our implementation. Although pub/sub has some inherent rigidity related to modifying published data, our system allows for flexibility in defining EKV keys published via our channels. A user might publish multiple data chunks via a single key if he is not concerned about consistency or publish a new key on every new upload if he cares about consistency. Data that is augmented by a function is considered new data and is (re)published separately. Using these channels is not limited to passing EKV keys. Computation delegation across different nodes that do not require EKV store might still use pub/sub channels. Channels will be used to exchange meta-data between executions, pass function return values that do not require storage, and coordinate runs across different locations.

To implement computation channels, we selected the MQTT messaging protocol as our message broker. MQTT shares the IoT approach where any device is a potential client and is flexible in using quality of service (QOS) assurances that tie nicely with a data stream approach. As our MQTT server, we use a Mosquitto broker on our edge nodes. Mosquitto is robust enough to run on our heavy-duty servers supporting high volumes of messages, as

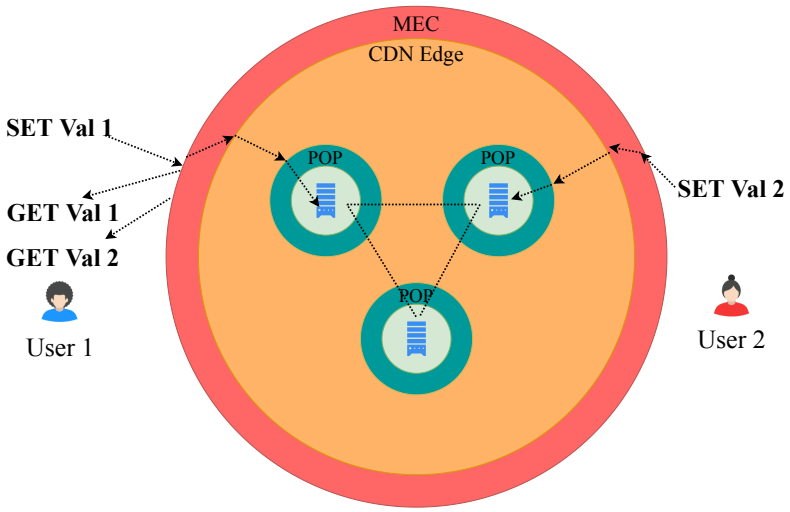


Figure 3.3: Edge Architecture layers. The plot shows extending current CDN architecture using a persistent global database. MEC outer layer allows low latency edge node computation and access to the larger CDN edge network. PoPs and EKV instances are interconnected. A value computed across the network is still available for local consumption.

well as lightweight sufficient for potentially running on dedicated low power edge hardware.

Figure 3.4 shows a demo of computational channels for data produced outside the network, and Figure 3.5 offers a demo of computational channels for data produced inside the network architecture. The different propagation paths of the data uploaded to the EKV store and the MQTT pub/sub calls are denoted using color arrows. This representation is meant to capture the concurrency of our network and the emphasis on global availability.

3.2.4 Resulting set up and real-world example

The described setup allows a highly dynamic computation pipeline on the edge. The subscription to computational channels could be as hierarchically complicated as needed, using multiple layers deep of edge computation subscriptions. This layered approach allows for fine-grained customization of computation that could be individualized up to a per-user case. Additionally, this allows for the invocation of highly localized edge functions or models that are physically far away but are on the same network and have access to the same EKV. A motivating example would be sharing a trained model without having access to the private

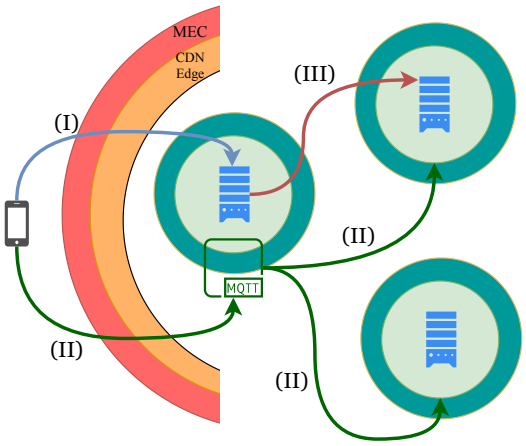


Figure 3.4: Computational pathways incoming demo. (i)Smartphone uploads text data (blue arrow) and publishes availability via MQTT broker (green arrow). (ii)MQTT broker forwards the publication (green arrows), and EKV store makes data available to all edge nodes. (iii)Once the subscribed edge function receives MQTT publications, it pulls the text data (red arrow) and runs compute.

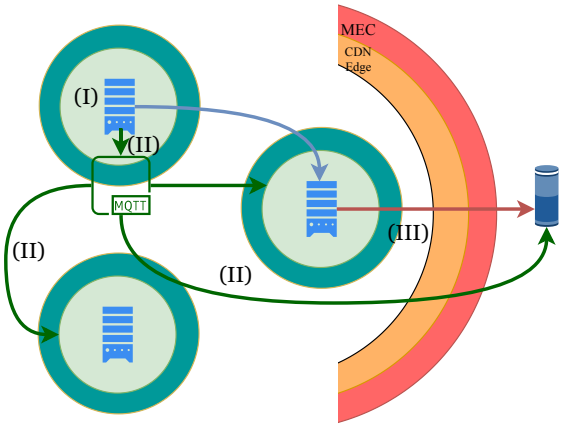


Figure 3.5: Computational pathways outgoing demo. (i)An edge function finishes computation, it pushes outputs data to EKV (local so no arrows), followed by an MQTT publish call (green arrows). (ii)Finally, a subscribed user receives the publication and requests the data from EKV, which retrieves it from source (blue arrow), and then allows the user to pull (red arrow).

data it was trained on. For instance: if we want to train a model on the edge, we might benefit from utilizing models on EKV that did something similar in different geo-locations. Collaborating across nodes can be seen as a debiasing stage that is both private, and edge contained. Federating models prevent bias from locally collected training data, and sharing models on the edge network instead of data keeps that data private. Another advantage of our network is the ability to retain context and make it available globally. Since we allow subscribing to computation results of another edge function or model, we sometimes need to maintain the proper context in addition to the output on EKV.

For example: Say we are feeding a video to the edge, and a function is subscribed to detect faces in frames of the video feed. The output of that function is the coordinates of a bounding box for a face in the frames. Now, suppose a function is subscribed to the results of that face detection function and is planning to use the face detection results and continue to augment the faces on these frames. In that case, it will need both the coordinates produced by the face detection function in addition to the original frames on EKV. To support such a use case, we need to understand what a subscription to a computational channel depends on. In this case, that subscription to the output of face detection is dependant on the original frames being available in EKV. We solve this by having the augmenting function subscribe to two computational channels and starting work when both a frame and its corresponding coordinates are known. Lastly, all computation is done on the edge, whether local or remote. There is no delegation to a centralized cloud. Instead, any non-local edge node might potentially participate in the computation if such delegation is needed.

Figure 3.6 and 3.7 show a demo of text-to-speech task execution on our architecture. The different propagation paths of the data uploaded to the EKV store and the MQTT pub/sub calls are denoted using color arrows. The latency observed on MQTT publication, and data transfers from outside the network to a PoP with an EKV instance and EKV to another PoP with our Text-to-Speech function is denoted in the plot. This representation captures our real-world experimentation with our architecture and the latency observed.

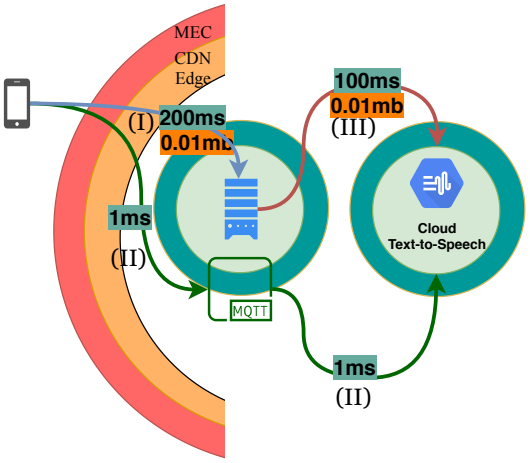


Figure 3.6: Text2speech incoming. (i)Smartphone uploads text data (blue arrow) and publishes availability via MQTT broker (green arrow). (ii)MQTT broker forwards the publication (green arrow), and EKV store makes data available to all edge nodes. (iii)Once the subscribed edge function receives MQTT publications, it pulls the text data (red arrow) and runs compute.

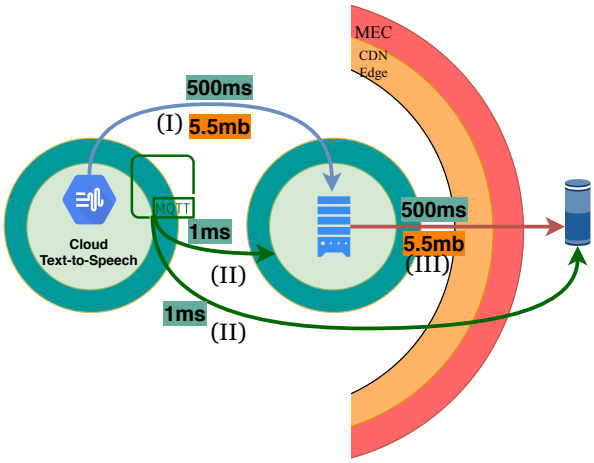


Figure 3.7: Text2speech outgoing. (i)Text-to-speech model finishes running. It pushes outputs data to EKV (local so no arrows), followed by an MQTT publish call (green arrows). (ii)Finally, a subscribed user receives the publication and requests the data from EKV, which retrieves it from source (blue arrow), and then allows the user to pull (red arrow).

3.3 Experiments and Measurements

Our architecture is built to support large amounts of data stream traffic and computation paths within the network. We show in our experiments the advantages of our network in a few fundamental edge-related tasks. We show how close we can get to real-time delivery of results from edge functions and models working on analyzing data streams. We evaluate the efficiency of chaining different functions while retaining meta-data between executions. And we evaluate how close we can get to real-time generation of data from a machine learning model, based on ques from a user outside our network. The tasks are as follows:

1. Run an emotion detection (ED) model as an edge function on recorded voice samples from an IoT device and show detected emotion in real-time.

2. Run a text to speech (T2S) function on the edge that accepts text from a user and outputs generated human voices that a second IoT device will then consume.

3. Pipeline 3 image-related machine learning model that will accept as input a video stream and output an augmented version of it (DF).

For each of these, we report all metrics relating to latency and connectivity throughout the path of execution. All experiments were run on our 32 core Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz, with the last two tasks utilizing a single 8GB NVIDIA Tesla M60 GPU. The edge node we are using in these experiments is located in close physical proximity (Los Angeles) and within 5ms to the client. The EKV instance is located remotely (Chicago) and within 7ms to the client. We show a few machine learning use-cases that take advantage of our proposed edge architecture. Models or functions running on our edge can seamlessly integrate into a user's edge function chain.

Table 3.1 shows a comparison between the different experiments and their key points we are interested in evaluating on our edge architecture.

Table 3.1: Experiment Comparison

	ED	T2S	DF
Pub/Sub	✓	✓	✓
EKV Data Store	×	✓	✓
Function chaining	×	×	✓
Small files incoming	✓	✓	×
Small files outgoing	✓	✓	×
Large files incoming	×	×	✓
Large files outgoing	×	✓	✓
Model is target specific	×	×	✓

3.3.1 Emotion detection from voice samples

With the expanding array of intelligent speakers consumers interact with, voice analysis becomes a common task for extracting commands and features from voice unrelated to the spoken text. We evaluate the usage of a machine learning model trained to classify positive and negative emotions on the edge and return a response in real-time. This task shows low latency availability of our emotion detection model and utilization of computation channels to exchange small amounts of data and coordinate execution.

Using a database of labeled actor voices emulating emotions, we pre-train our model and package it into a Docker container deployed on the edge. The dataset we used is described in [LR18]. We create two MQTT channels—one for incoming data and one for outgoing classification. We then create and subscribe an edge function that listens to incoming publications and branch an instance of the emotion detection model for each incoming request. On the client-side, we create a web page for recording voice samples using JavaScript and use paho-mqtt in the browser to subscribe to the channels for this process. The voice samples are recorded through the browser, serialized, and sent through MQTT as the payload of the MQTT publish call. The edge function receives the serialized file, passes it to the emotion

detection model, and publishes the classification results via MQTT. The browser then receives the publication and displays an emoticon on-screen [2]. The voice sample size depends on the recording length, but we have found that a file of about 100kb could be published via MQTT and received by the listener function in less than 3ms. After the function loads the pre-trained model and passes the data, the model takes about 4.5 seconds to run using only the CPUs on the edge node. The model output is then published back received by the client after another 2ms, and finally, the client displays an emoticon after about 4.5 seconds from the end of the recording process.

As a comparison, we consider a standard cloud architecture with serverless functions that accept data via HTTP requests. The ping to an average cloud instance takes an average of 224ms [3], and response time per message using HTTP is 200ms higher than that of MQTT when connection is reused [4]. Including the time it will take to move the data makes our task potentially 1-3 seconds longer on public cloud. A significant user impact.

3.3.2 Text to speech

Offloading the resource intense process of data generation using a relatively small amount of data is another area where edge networks shine. Asking an edge node to generate images for us using a description or Using a model to generate human voice from the raw text are only two examples. We will examine the latter example here. This task shows a producer and consumer operating independently in different locations using different edge nodes for EKV and edge function. The edge nodes used are in Chicago and California and are 1ms apart when using ping.

Using a model based on Deep Voice 3 [PPG17], we create an edge function that can receive a blob of raw text and parameters indicating what kind of speaker the model should generate and outputs a recording of a human voice that speaks the text that was received.

[2] A video showing the emotion detection task can be seen here

[3] Measured using 'curl' to 30 public cloud instances from different companies and averaged

[4] Detailed comparison can be seen here

We create two MQTT channels, one for publishing blobs of text and the other for publishing human voices. Our edge function subscribes to the text channel and waits for a publication that a new blob of text is available on EKV along with the parameters of what voice should be generated. An instance of our function starts for each blob/speaker pair. We create a producer of text that pushes the text to EKV and publishes on the text channel, and a separate consumer that subscribes to both channels and consumes both the text and the corresponding human voice that our edge function outputs. This setup simulates a situation where the producer is not the final destination for processing the produced data. We run the producer and consumer in separate locations and time the task of a producer pushing a blob of text and asks for ten different human voices generated for each blob. The producer pushes approximately 200 bytes of text to EKV and receives confirmation within 200ms; once the data is on EKV, the producer publishes via MQTT that a new blob is ready for consumption. The edge function pulls the text from EKV and starts ten instances of text-to-speech translation after about 500ms. The consumer receives the publication and prints out the text after 250ms of process start. The edge text to speech function outputs 10. WAV files weighing a total of 5.8Mb after working for 4 seconds. It then uploads them to EKV simultaneously and receives a final confirmation from EKV after a total of 5.5 seconds from process start. The function then publishes to MQTT the availability of results. Once the consumer receives the publication of newly available data, it pulls the data from EKV and saves them locally after about 6.3 seconds from process start [5].

To compare, we look at the average latency between nodes of popular cloud services [6]. In addition to the 224ms average ping time from client to cloud service and 200ms longer response time per message, the average latency between public cloud nodes is approximately 160ms. Depending on the implementation of storage and upload/download of data, our task will take at best seconds longer on the average public cloud.

[5] A video showing the text to speech task can be seen here

[6] latency average was computed based on information in: https://www.cloudping.co/

3.3.3 Video stream manipulation

Leveraging the edge as a live video manipulation tool opens the doors for many exciting use-cases such as dynamically augmented video streams. Combining that with machine learning models such as Deepfake lets us imagine a future where we consume personally tailored video streams, replacing actors in a movie we are watching on the fly. We will examine how we can use our architecture and create a pipeline of functions to create an augmented version of a video stream as close to real-time as possible. This task evaluates the chaining of functions and models to augment a video feed live. Producer and consumer are operating independently in different locations using different edge nodes for EKV and edge function. The edge nodes used are in Chicago and California and are 1ms apart when using ping.

We define four computation channels. We have an edge function and two edge models chained, each subscribed to the output of the previous functions, and an extra channel publishing the availability of the original frames on EKV that all are subscribed to, thus creating a pipeline of computation for the video frames to go through. We also define a helper edge function that extracts individual frames from a video clip using FFMPEG library [7].

3.3.3.1 (i) Face detection

First, we have a function based on OpenCV face detection that accepts video frames as input and outputs location of faces in frames. Once a frame is passed to the function, it is passed to OpenCV, where frames are rotated and scaled multiple times as the OpenCV detector function scans for faces. Coordinates for detected faces are saved on EKV.

3.3.3.2 (ii) Face classification

Coordinates for faces that are identified by the previous function as well as the original frames are ingested by a model based on VGG face classification [PVZ15] pre-trained for face classification using 2.6M images from 2622 identities. Our model is used to identify

[7] A video showing the augmentation task can be seen here

a specific face of interest that we wish to augment. It accepts a frame with a face and a reference image and outputs whether the face in the frame matches our person of interest back to EKV.

3.3.3.3 (iii) Face augmentation

Frames that the classification model marked are then picked up by our Deepfake model based on the work of [PGH16]. The model uses the stored coordinates for each frame to extract a cropped face to convert. The model then performs conversion of the frame, reconstructed to fake the source face in the original frame into the desired target face, and output the augmented frames back to EKV.

Data used for training our model has been scraped from YouTube videos of the source face and the target face and created around 5000 images of each. We train the model for one week using a single NVIDIA Tesla M60 GPU before compiled as an edge function.

3.3.3.4 (iv) Consumer

A consumer who is subscribed to the original stream and the output of the DeepFake function can pick up the video feed with augmented frames from EKV and view the feed locally.

3.3.3.5 Timing

The producer streams 20-second chunks of video to EKV weighing an average of 1.9Mb. A single chunk takes about 300ms to upload to EKV and receive confirmation. The producer then publishes via MQTT that the chunk is ready for consumption. Face detection then picks up the chunk and starts the process of detecting faces. Coordinates of each face detected are immediately pushed to EKV and published as available. The first video chunk takes about 36 seconds to process due to model loading, and each following chunk will be processed within 300ms of producer pushing to EKV. Face classification receives publication and within less than 20ms classifies the face, which pushes EKV and publishes availability. The face augmentation phase then has received publications from all channels it is subscribed to and

starts working on changing the faces on a chunk of video. Converting the entire 20-second chunk of video takes 18 seconds. The Deepfake model then pushes the changed frame back to EKV and publishes availability within a few milliseconds. Lastly, the consumer receives the publication of the availability of frames. Once there is a 20-second chunk of frames available, it uses the helper function to convert them back into a video file, downloads and plays them. From the time faked frames are published, it takes the consumer approximately 500ms to convert and download the 20-second chunk of video. Overall, the first chunk takes about 55 seconds to be augmented and viewed on the consumer end. After the first chunk, it will take under 20 seconds to finish working on a 20-second chunk for the entire pipeline. We then can keep our augmented video feed about 1 minute behind the live video.

We compare to the latency on a public cloud service. In addition to the 224ms average ping time from client to cloud service and 200ms longer response time per message, and an average of 160ms latency between distant nodes, we add the accumulating latency of making the intermediate results of each function globally available for consumption. Assuming data passes via HTTP/HTTPS, our task could not be augmented fast enough to be viewed in pseudo-live time.

3.4 Comparison with different architectures

In addition to the benefits accrued by our overarching edge architecture, there is room to break down individual components and compare them to other possible design choices. MQTT is one protocol chosen from the few emerging protocols of choice for the IoT world. While we evaluated both MQTT and CoAP and found both comparable, we chose MQTT for our pub/sub protocol as it had better library availability and broker selection. We compare our choice of MQTT with an HTTP-based signaling mechanism to support our architecture. In our architecture, we use MQTT as a signaling channel between subscribed clients waiting on streams of data and between edge nodes coordinating the execution of models on data. The key observation here is that our MQTT connection is seldom closed, and in most cases, it is reused often between the time they are established and close. The comparison made in

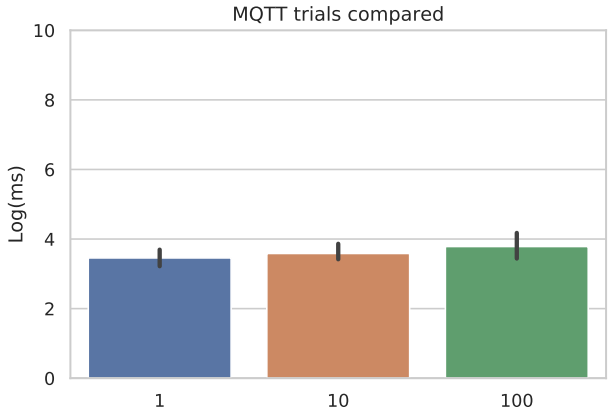


Figure 3.8: MQTT speed per number of requests compared at log scale of ms.

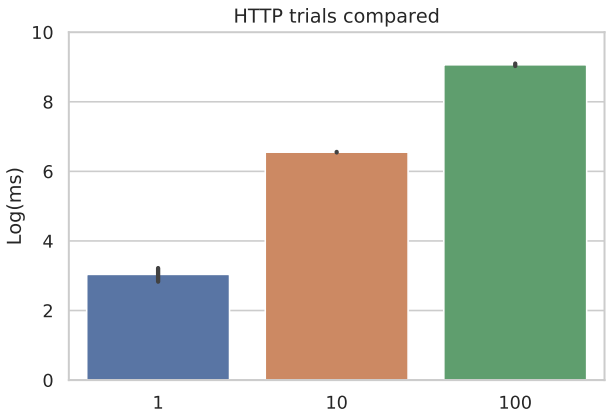


Figure 3.9: HTTP speed per number of requests compared at log scale of ms.

[Wan18] clearly shows the benefits of utilizing an open MQTT connection with exponential benefits over the same use case implemented using HTTP. Similarly to [Wan18], we investigate the difference between 1, 10, and 100 messages, each weighing 10 bytes, transmitted over MQTT and HTTP, over ten trials. This small message simulates transferring simple instructions and EKV data locations in our computation channels. For MQTT, we connect once and reuse the same connection to communicate all subsequent messages. For HTTP, we use POST requests. All communication was evaluated between an edge node and a local client, emulating a real-world scenario. Figure 3.8 and 3.9 show the log scale results for speed in ms, as observed in our test. Since HTTP grows as a factor of messages passed, we see the benefit of opening a single MQTT connection to be used over multiple messages.

Another aspect worthy of comparison is the speed gain of using our architecture compared to the same job implemented as a FAAS workflow. Results must be returned to a user before the next function in a pipeline is started. We compare a simple NumPy matrix multiplication task, called via our MQTT computation channels 1,10 and 100 times, where results are pushed to a MinIO storage instance. This computation is compared to the case where a function runs and returns a result directly to a client. If we run our function more than once, we compute the next result based on the result of the previous function. In the FAAS-like use case, the client sends back the result to the function, and in our architecture,

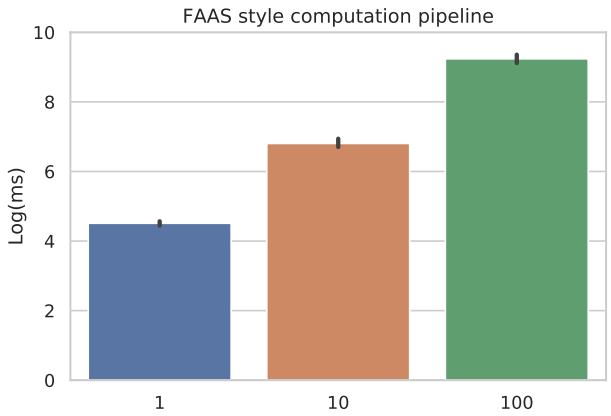


Figure 3.10: Speed of calling our function via HTTP POST requests, and sending back the result for all cases where the function was called more than once. Compared at log scale of ms.

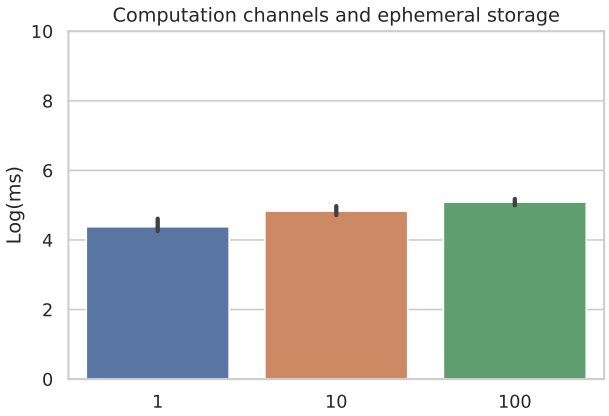


Figure 3.11: Speed of calling our function via an MQTT computation channel, send the result to an ephemeral storage and compute results based on the previous function run for all cases where we call the function more than once. Compared at log scale of ms.

the previous result is picked up from our MinIO instance. Figures 3.10 and 3.11 show the comparison between the two approaches. It can be seen that the impact on sending the little amount of data we use back and forth using HTTP POST requests essentially does not change the POST requests time for execution. In comparison, the time increases using MQTT computation channels and ephemeral storage, where an extra call to the MinIO server is needed. However, even with this increase, it can be seen that as the number of concurrent requests grows, the penalty incurred by POST requests is far more inhibiting than the extra hop to MinIO. As we have previously shown in our experiments, MQTT can be used for small-scale data and speed up computation even more in cases where not much data is moved in the network.

CHAPTER 4

Cost-Aware Target-Focused Feature Selection[1]

In this chapter, we will suggest a cost-aware method for frugal feature selection to support real-time iterative feature selection.

4.1 Preliminaries

4.1.1 Notation

The following notation is used throughout the chapter: $\mathbb{X}$ represents all data available to us at a given moment, $\mathbf{X}$ represents an instantiated subset of all available observations data; made up of data points of size r and feature set $\mathbf{F} = \{F_1...F_c\}$ of size c such that $\mathbf{X} \in \mathbb{R}^{r,c}$. λ represents the free parameters; $\mathbf{W}$ and $\mathbf{b}$ (weight and bias) that are learned by the model trying to estimate the latent variables. The true posterior distribution for our latent variables is represented by $\mathbf{z}$, i.e., the distribution of true targets that is unknown to us. The estimated posterior is represented by $\mathbf{y}$. θ represents a specific target class out of all potential targets. $\mathbf{t}_\theta$ denotes the true target vector for a target class θ and $\mathbf{y}_\theta$ a subset of our predictions corresponding to the true targets in $\mathbf{t}_\theta$.

4.1.2 Background

In order to capture uncertainty in a model, we need to learn a representation of a latent distribution over a set of parameters defining that distribution and sample the learned pa-

[1]**This chapter is based on "Goldstein, O., Kachuee, M., Karkkainen, K. and Sarrafzadeh, M., 2020. Target-Focused Feature Selection Using Uncertainty Measurements in *Healthcare Data. ACM Transactions on Computing for Healthcare, 1(3), pp.1-17.*"**

rameters to associate the captured uncertainty with test time examples of the data and targets. Our optimization function, therefore, will be taking a probabilistic approach.

Using Variational Inference, we will estimate $\lambda^\star$ using Kullback-Leibler (KL) divergence such that:

$$\lambda^\star = arg\ min_\lambda KL(q(\mathbf{z};\lambda)\ ||\ p(\mathbf{z}|\mathbf{X})), \tag{4.1}$$

where $q(\mathbf{z};\lambda)$ is the estimation of posterior distribution $p(\mathbf{z}|\mathbf{X})$ optimized over parameters λ.

Since the posterior $p(\mathbf{z}|\mathbf{X})$ is unknown to us, we will resort to maximizing the Evidence Lower Bound (ELBO) as an optimization function:

$$ELBO(\lambda) = \mathrm{E}_{q(\mathbf{z};\lambda)}[log\ p(\mathbf{X},\mathbf{z}) - log\ q(\mathbf{z};\lambda)], \tag{4.2}$$

which is equivalent to minimizing KL divergence [JGJ99, Bis06, KW13a].

Gradient optimization of ELBO is done via the reparameterization trick [KW13a].

$$\nabla_\lambda ELBO(\lambda) \approx \frac{1}{s}\sum_{s=1}^{s}[\nabla_\lambda(log\ p(\mathbf{X},\mathbf{z}(\epsilon;\lambda)) - log\ q(\mathbf{z}(\epsilon;\lambda);\lambda))], \tag{4.3}$$

where s is the number of samples drawn.

4.2 Target-Focused Feature Selection

4.2.1 Problem Set-Up

Our goal is to achieve reasonable confidence for a specific class using minimal features, as described in our objective function.

$$argmax_{FS}\ (confidence_\theta - \sum_{f_i \in FS}\frac{1}{v_i}), \tag{4.4}$$

$$Subject\ to:\ |FS| < \beta.$$

Such that FS is the set of acquired features we wish to minimize, $|FS|$ is the cardinality of the set, v_i is the value associated with each feature. The objective is to frugally acquire the most valuable features while achieving maximum confidence in a specific class θ, without exhausting our budget for features β.

4.2.2 Feature Value Measurement and Acquisitions

Evaluation of features per target considers the contribution of each feature towards minimizing uncertainty for our target of interest, jointly evaluated with the features already acquired. In addition to confidence scores, feature vectors are scored for their cosine similarity and their Hamming weight scores to gauge potential information gain from a candidate feature.

In order to use *ELBO* as our optimization function, we model the linear regression case in which our λ contains the input $\mathbf{X}$, a single layer of weights $\mathbf{W}$ and a bias $\mathbf{b}$ such that $\lambda = (\mathbf{W}, \mathbf{b}, \mathbf{X})$. Here $\mathbf{X} \in \mathbb{R}^{c,r}, w \in \mathbb{R}^{c,d}, b \in \mathbb{R}^{1,d}$. $\mathbf{X}$ has r data points and c features, and the model will learn the distribution over d targets. Assuming independence given our parameters:

$$p(\mathbf{z}|\mathbf{W}, \mathbf{b}, \mathbf{X}) = \prod_{n=1}^{r} p(\mathbf{z}_n|\mathbf{X}_n^\top \mathbf{W} + \mathbf{b}, \sigma_z^2), \tag{4.5}$$

where z is the *ELBO* optimized posterior estimation. We define the priors on both parameters to be the standard normal distribution.

4.2.2.1 Measuring Per-Target Uncertainty

Our available data is split into a training set $\mathbf{X}_{train}$ and a testing set $\mathbf{X}_{test}$. To obtain our input $\mathbf{X}$ we sample the training data $\mathbf{X}_{train}$ in a balanced way. For example, if we are trying to predict three targets, then $\mathbf{X}$ will have 33% of the data points correspond to each of our targets, regardless of the original distribution. In order to generate a validation input dataset $\mathbf{X}'$, we sample $\mathbf{X}_{train}$ according to its original distribution (no balancing).

At each iteration, a subset of all available feature $f_i \cup FS$ is trained to learn $\lambda = (\mathbf{W}, \mathbf{b}, \mathbf{X})$. Once trained, we score the feature subset on the validation set $\mathbf{X}'$ by measuring the effect acquired features had on a per-target uncertainty. Using our learned distribution, we sample each of our parameters such that $\mathbf{W}' \sim \mathbf{W}, \mathbf{b}' \sim \mathbf{b}$ and calculate the probability vector:

$$prob = softmax(\mathbf{X}'^\top \mathbf{W}' + \mathbf{b}'), \tag{4.6}$$

where $prob \in \mathbb{R}^{r,d}$ has the probability of each data point belonging to each possible target.

We then get the prediction vector by calculating softmax for each $prob_i$:

$$\mathbf{y}_i = argmax\ (prob_i) = argmax\ (\frac{exp(prob_i)}{\sum_d exp(prob_{i,d})}). \tag{4.7}$$

Next we evaluate precision, represented by the fraction of times that $\mathbf{y}_\theta$ corresponding to target θ, was equal the correct target for position i. Note that $\mathbf{y}_\theta \in \mathbf{y}$, and is of subset size $|\mathbf{y}_\theta|$:

$$precision_\theta = \frac{1}{|\mathbf{y}_\theta|} \sum_{i=1}^{|\mathbf{y}_\theta|} 1(\mathbf{y}_{\theta,i} = \theta), \tag{4.8}$$

where $1(\mathbf{y}_{\theta,i} = \theta)$ equals 1 if data point $\mathbf{y}_{\theta,i}$ has the target value θ, and 0 otherwise.

Repeating 4.6 - 4.8 for l iterations, sampling the distribution of our parameters each time, our confidence score becomes the averaged precision over multiple iterations. Therefore the confidence for a specific target:

$$confidence_\theta = \frac{1}{l} \sum_{j=1}^{l} (precision_{\theta,j}). \tag{4.9}$$

Here l is the number of times we sample our learned distributions. The trade-off using l is between a more accurate representation of the model confidence and a faster model. We have found that 300 iterations were accurate enough in reporting confidence in our case.

4.2.2.2 Adding Vector Similarity Scores

In addition to the confidence scores, we wish to capture the potential information gain of the current candidate feature f_i given the existing features in FS. We use the computed similarity scores: co-variance distance score and cosine similarity score. We sum the inverse scores for all such pairwise comparisons and then normalize them to the range [0,1].

$$CovScore = N^{0,1} \sum_{g_i \in FS} 1 - cov(g_i, f_i), \tag{4.10}$$

$$CosScore = N^{0,1} \sum_{g_i \in FS} 1 - cos(g_i, f_i). \tag{4.11}$$

CovScore and CosScore are the summed inverse co-variance distances and cosine similarities, transferred to the [0,1] range applying the normalization $N^{0,1}$.

Our final feature value for the current feature f_i is then

$$v_i = \omega_1 * \mathit{confidence}_\theta + \omega_2 * \mathit{CovScore} + \omega_3 * \mathit{CosScore} \tag{4.12}$$

, where $\omega_1, \omega_2, \omega_3$ are hyperparameters.

Once all features $F_i \notin FS$ have been scored and evaluated for their contribution towards class θ as part of set FS, we append the single feature that maximized v_i to the set FS

4.3 Time Complexity Analysis

Let FS be the set collecting all selected features, N be the number of available features, D the number of data points in our traning set, and assuming some constant budget β for features. For a single feature $f_i \notin FS$ we train a new model estimating the linear function $p(\mathbf{z}_n|\mathbf{X}_n^\top\mathbf{W} + \mathbf{b}, \sigma_z^2)$ (equation 4.5 in the chapter). The model is trained using a constant number of iterations and confidence is computed using a constant number of samples from the estimated distribution. *CovScore* and *CosScore* are both computed on the features already in FS in time $2 * \beta^2$. Since β is constant, so is the time to compute *CovScore* and *CosScore*. The final v_i value is the product of another constant time multiplication.

Once all features are scored, we append a single feature to the set FS, and the process starts again for $N-1$ features. Therefore, for N features, the process will run $N + N - 1 + N - 2... + 1 = N^2$ times. Training a model using D data points each time will result in DN^2 time complexity. We give a step-by-step description of our evaluation process in Algorithm 1.

4.4 Evaluation

Here we provide an empirical evaluation of our target-focused method (TF) compared with prevalent linear feature selection techniques.

Mutual Information (MI) is estimating statistical dependency for feature selection [KSG04], and is widely used as a non-parametric approach to evaluating data dependencies.

Algorithm 1: Target-Focused Feature Selection

Input: β : *Budget*; $FPT,\ FNT,\ CT$:

Thresholds for false positive, false negative, and confidence respectively;

$\mathbf{X}$: *Train set*; $\mathbf{X}^{'}$: *Validation set*; $\mathbf{X}_{test}$: *Test set*;

$\mathbf{y}$: *Targets for* $\mathbf{X}, \mathbf{X}^{'}$ *and* $\mathbf{X}_{test}$;

$\mathbf{F}$: $\{f_1, f_2, ..., f_n\}$, *set of available features*

Parameter: $FS \leftarrow \{\}$: *features selected*;

$M \leftarrow$ *model optimizing* $ELBO(\lambda)$

$F \leftarrow$ *function for computing* v_i : *value for feature* i;

FP_θ, FN_θ, $confidence_\theta$: *current false positive,*

false negative and confidence for specific target θ

Output: $SF \subseteq \mathbf{F}$ *within budget* β

1: **while** $|SF| < \beta, FP_\theta > FPT, FN_\theta > FNT,$
$confidence_\theta < CT$ **do**
2: **for** $f_i \in \mathbf{F}$ **do**
3: *Train* $M(\mathbf{X},\ SF \cup f_i,\ y)$
4: $v_i \leftarrow F(\ M(\mathbf{X}^{'},\ SF \cup f_i,\ y),\ FS, f_i)$
5: **end for**
6: $FS \leftarrow FS \cup f_i \mid argmax_i\ v_i \in V$
7: FP_θ, FN_θ, $confidence_\theta \leftarrow\ M(\ \mathbf{X}_{test},\ FS,\ y)$
8: **end while**
9: **return** solution

The MI approach works by estimating the correlation level based on entropy from k-nearest neighbor distances.

Max-relevance min-redundancy (mRMR) [PLD05] and **Advanced mRMR** (AmRMR) [JLO19] are first-order incremental feature selection methods based on Mutual Information (mRMR) or Pearson's correlation (AmRMR) that eliminates redundancy in features while selecting relevant ones.

Least absolute shrinkage and selection operator (Lasso) model [Tib96] is an L1-based feature selection approach. Lasso is an "automatic" approach to feature selection, performing some regularization and filtering unwanted features.

Extremely randomized trees (Extra Trees) [GEW06] is a tree-based model performing feature selection based on the importance values computed by the model.

A budgeted cost sensitive learning approach (CSL) [KGK19b] is a deep Q-networks based approach for cost-sensitive feature acquisition

A deep reinforcement learning-based approach (DRL) [JPL19] is a deep reinforcement learning approach to formalizing the problem as a Markov decision process (MDP) and solve it with linearly approximated Q-learning.

4.4.1 Datasets

We evaluate our model on an image classification task, as well as a breast cancer detection task. Both datasets chosen from the UCI machine learning repository [DK17a]. In addition, we evaluate various disease prediction tasks assembled using the Centers for Disease Control and Prevention's (CDC) National Health and Nutrition Examination Survey (NHANES) [nha18] data.

We select a target of particular interest for each of our sets, that we would like our model to focus on when deciding which features to acquire. Projecting this to the real world, the focus target will be a specific health issue in a dataset of symptoms and possible tests or images, pointing to more than one potential target class.

The data is as follows: From the UCI machine learning repository, we use SatLog data [2]. A dataset of evaluating image data and identifying a particular type of soil in satellite images. Again from UCI, we use the Breast Cancer Wisconsin dataset [3]. Providing features that are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass.

[2]Available here: UCI Statlog (Landsat Satellite)

[3]Available here: UCI Breast Cancer Wisconsin

From NHANES, we construct two datasets ourselves based on the approach described by [KKG19]. One for evaluating diabetes, and one for evaluating heart diseases. To construct our datasets, we join all possible NHANES tables that are correlated with our targets. For example, we merge all tables containing correlated features to any of five heart conditions for the heart disease dataset. This merge causes the resulting sets to have a vast amount of possible features.

Dataset statistics and the target chosen for each dataset are listed in Table 4.1. For the NHANES datasets, targets are renamed from the original data for convenience. Blood glucose refers to the feature LBXGLU, the amount of glucose in the blood when fasting, used here to indicate whether or not an individual has diabetes. Congestive heart failure (CHF) refers to the feature MCQ160B, and it is one of 5 heart conditions we construct the dataset for (MCQ160E, MCQ160F, MCQ160C, MCQ160B, MCQ180B).

4.4.2 Evaluation Methodology

Assuming a constant budget for features, we run all feature selection approaches on the same training subset of the data and iteratively evaluate each feature we add. We select a single target to act as the focus of our method. We emphasize the model confidence for that specific target value that we wish to maximize overall targets. The target chosen for each dataset is listed in Table 4.1.

The compared models were constructed with the following parameters:

(i) Mutual information (MI) between our training data and the training target was calculated using a different number of neighbors. Balancing the estimation variance and bias, we evaluated number of neighbors $k \in [1, 2, 3, 5, 10]$. The $k = 3$ instance, giving the best average result in all cases was selected.

(ii) mRMR/AmRMR was evaluated on "MIQ", "MID", and "Rvalue" feature evaluation methods.

(iii) Lasso with cross-validation was used in this experiment. In order to find the best α value for the regularization process, we considered $\alpha \in [1, 0.1, 0.001, 0.0005, 0.0001]$. The

best setup of Lasso for the average case was as follows: a maximum number of iterations was set to 1000, tolerance was set to 0.1, and the number of cross-validation folds set to 10.

(iv) Extra Trees classifier was used in our experiments. The number of estimators in this model was set to 1000, with no maximum depth defined. The minimum number of samples to split a node was set to 2, and the quality of split was measured by Gini impurity.

(v) CSL was evaluated using a Baysian L1 utility function. We ran the model for up to 40000 epochs to allow the model time to learn. We stopped the model when it converged.

(vi) DRL approach was evaluated using many models with different cost-accuracy trade-off hyper-parameter values. We ran the model for up to 40000 epochs to allow the model time to learn. We stopped the model when it converged.

(vii) Our Target-Focused (TF) feature selection was trained using $\omega_1 = 0.4, \omega_2 = \omega_3 = 0.3$.

The machine used for evaluation had specifications: Intel 12 core i9-7920x (2.90GHz) CPU, 128 GB RAM, and 4 GeForce RTX 2080TI GPUs.

4.4.3 Run Time Comparison

Finding a globally optimal feature subset out of a collection of available features is an NP-Hard problem. Instead of spending an exponential amount of time evaluating subsets, feature selection methods use heuristics to select an approximation of an optimal set. In our experiments, we compare with polynomial-time statistical methods (MI, mRMR/AmRMR, Lasso, Extra Trees), and deep learning approaches (CSL, DRL). Our proposed model trains a shallow machine learning model, using a single layer estimating the latent distribution, falling in the middle of these two groups in terms of complexity. In practice, taking NHANES heart dataset as our baseline, we see Extra Trees and mutual information running fastest, finishing within 30 minutes - 1 hour, followed by MI and mRMR/AmrMR taking about 1 hour - 2 hours. Our method and the deep learning methods take advantage of the GPUs on our machine since they heavily rely on linear computation. With a single GPU, our method takes between 2-3 hours to finish. CSL takes approximately three days, and DRL takes 4-5

Dataset	Size	Features	Targets	Focus target	Missingness
UCI Breast cancer	569	32	2	Malignant	0%
UCI Satlog	4435	37	6	Damp grey soil	0%
NHANES Diabetes	25474	581	2	Blood Glucose	25%
NHANES Heart	49346	555	5	CHF	25%

Table 4.1: Datasets statistics

days to finish and return results.

4.5 Results

In this section, we report confidence, false positive, and false negative scores and F1 scores of our model at intervals as features are acquired. We separate our comparison to statistical methods for feature selection and deep learning approaches for sequential feature acquisition. When plotting model trends of the metrics mentioned above, we denote the variance scores of model confidence as the line margin on confidence plots, as shown in the plots below.

4.5.1 Comparing with statistical methods

On datasets with low feature count and little missingness, our method achieved better overall scores, faster than the comparable methods For a specific target value. As can be seen in Table 4.2.

Figure 4.1 shows a confidence trend for acquiring 30 features on the Breast Cancer Wisconsin dataset using our method, comparing specific target confidence in the four compared models. In this case, our model can be seen on par with the confidence achieved by the Mutual information and Extra trees methods. As shown in Table 4.2, our model presents slightly better F1 scores, indicating a faster false positive and false negative reduction. The breast cancer dataset proved to be a relatively simple prediction problem; it can be seen that all methods performed relatively well, achieving reasonable model confidence and F1 scores.

F1 scores										
	Breast Cancer					Satlog				
f	**MI**	**mRMR**	**Lasso**	**Extra Trees**	**TF**	**MI**	**mRMR**	**Lasso**	**Extra Trees**	**TF**
5	0.93	0.94	0.80	0.91	0.95	0.70	0.27	0.35	0.08	0.86
10	0.94	0.93	0.83	0.94	0.93	0.64	0.68	0.72	0.78	0.90
15	0.93	0.94	0.88	0.95	0.94	0.80	0.70	0.85	0.78	0.90
20	0.93	0.93	0.94	0.95	0.93	0.83	0.71	0.85	0.81	0.90
25	0.93	0.93	0.94	0.95	0.94	0.83	0.77	0.87	0.81	0.91

Table 4.2: Comparing F1 scores for feature selection on low feature count sets. f indicates the number of features acquired.

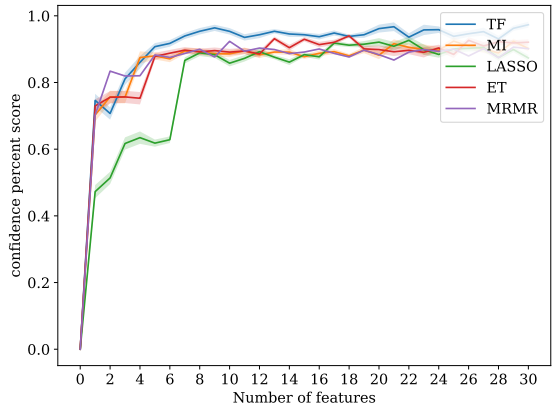


Figure 4.1: Comparing model confidence in predicting malignant breast cancer. Line thickness indicates variance.

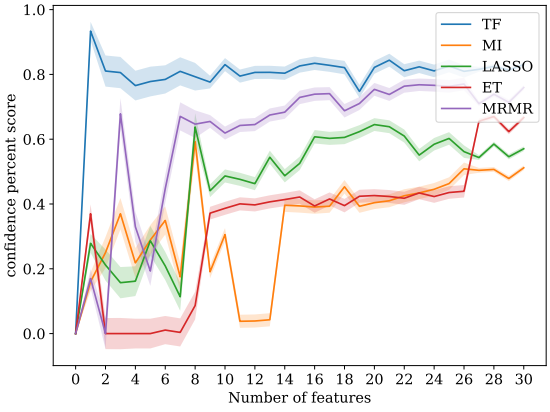


Figure 4.2: Comparing model confidence in predicting one class out of the Satlog dataset. Line thickness indicates variance.

Figure 4.2 shows a confidence trend for acquiring 30 features on the Satlog dataset using our and compared methods. Here we can see our target-focused method achieves a more confident model faster, as well as a better overall F1 score. In this case, the target of focus chosen appeared to be the hardest target to model out of the available targets, since all compared models struggled to find features that best model the data, in addition to it being

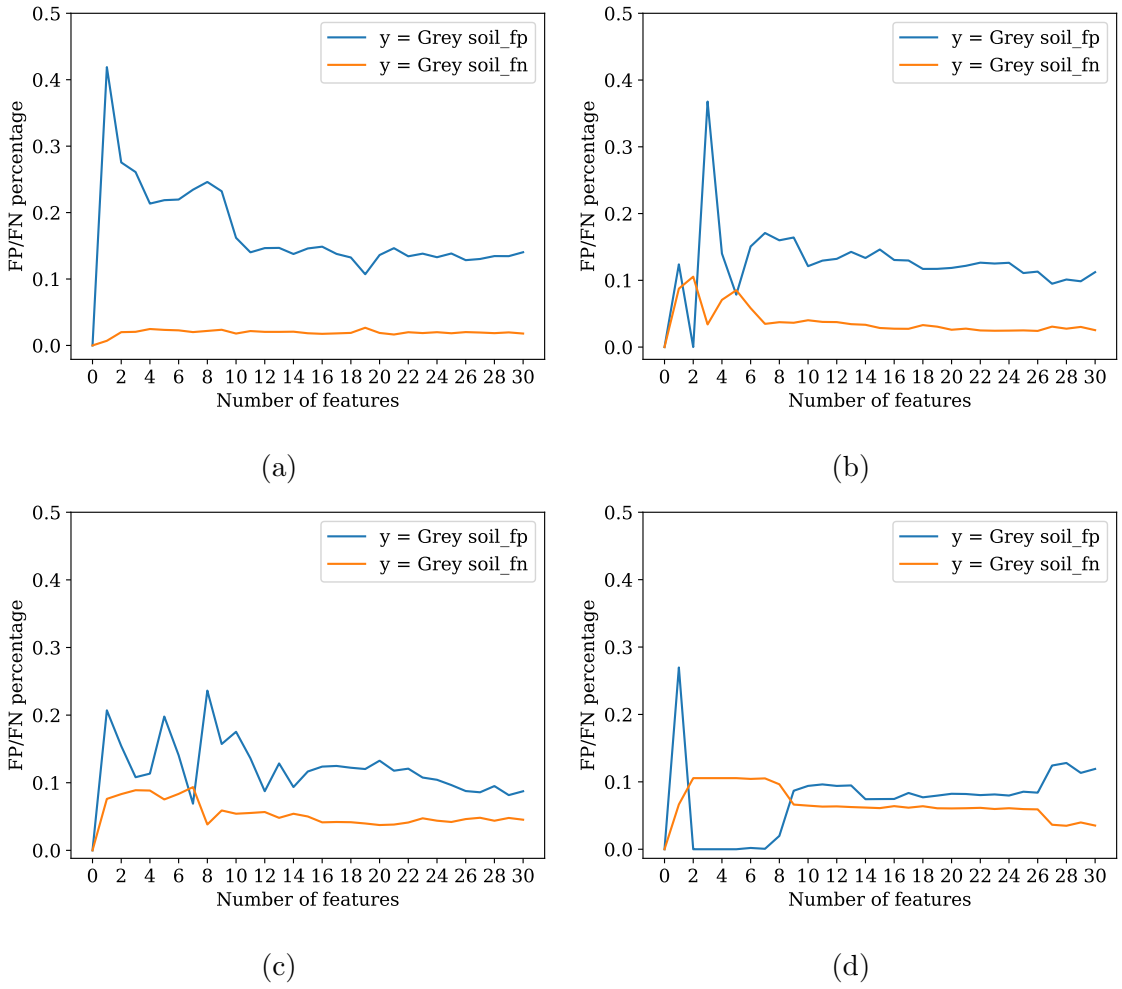


Figure 4.3: Analysis of UCI Satlog dataset comparing FP/FN rates as features are acquired. 4.3a shows evaluation using our approach, 4.3b shows evaluation using mRMR "MIQ" method, 4.3c shows evaluation using Lasso method, 4.3d shows evaluation using Extra trees

F1 scores										
	NHANES Diabetes					NHANES Heart				
f	**MI**	**mRMR**	**Lasso**	**Extra Trees**	**TF**	**MI**	**mRMR**	**Lasso**	**Extra Trees**	**TF**
5	0.76	0.79	0.61	0.79	0.92	0.66	0.55	0.28	0.39	0.72
10	0.78	0.77	0.73	0.79	0.92	0.31	0.48	0.59	0.57	0.78
15	0.80	0.79	0.77	0.77	0.92	0.78	0.69	0.56	0.32	0.87
20	0.78	0.80	0.76	0.74	0.92	0.68	0.63	0.61	0.68	0.87
25	0.76	0.80	0.64	0.77	0.92	0.85	0.76	0.66	0.47	0.86

Table 4.3: Comparing F1 scores for feature selection on high feature count sets. f indicates the number of features acquired.

one of the minority classes. Despite that, our model has gained the most confidence while using a low number of features. Figures 4.3a to 4.3d show the FP/FN evaluation over the Satlog dataset. We see our method shows a consistent non-volatile decline in FP rates while maintaining a low FN rate throughout.

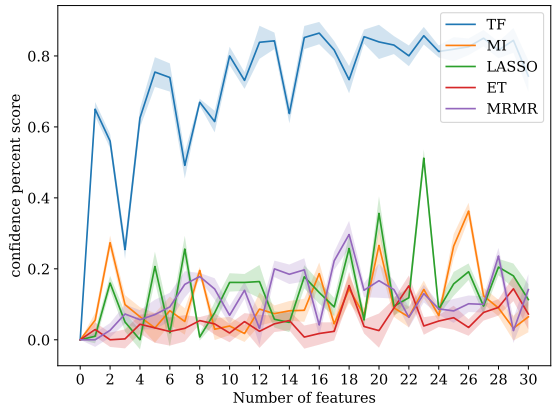


Figure 4.4: Comparing model confidence in predicting congestive heart failure. Line thickness indicates variance.

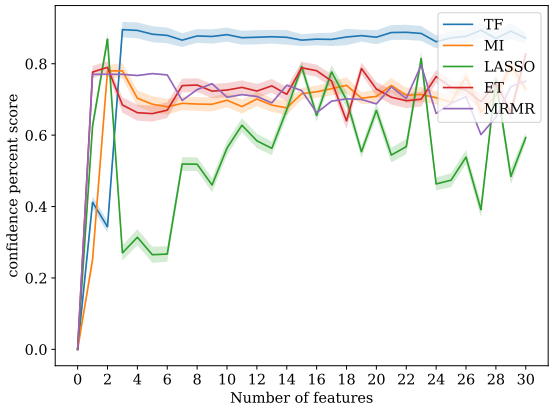


Figure 4.5: Comparing model confidence in predicting diabetes. Line thickness indicates variance.

Table 4.3 shows the F1 scores of the different feature selection methods compared on high feature count datasets, with missing values and lots of features. Here our model heuristic

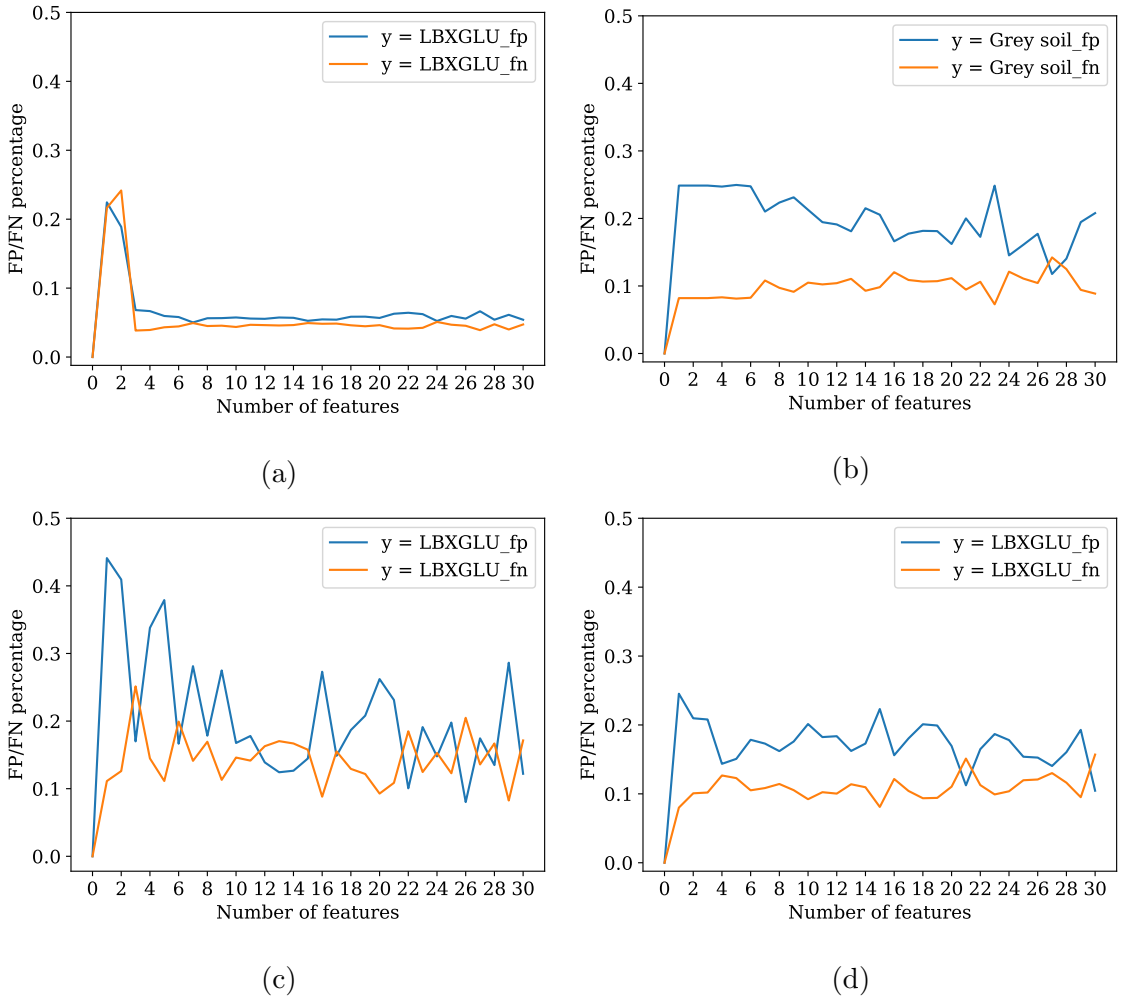


Figure 4.6: Analysis of NHANES Diabetes constructed dataset comparing FP/FN rates as features are acquired. 4.6a shows evaluation using our approach, 4.6b shows evaluation using mRMR "MIQ" method, 4.6c shows evaluation using Lasso method, 4.6d shows evaluation using Extra trees.

F1 scores						
	Breast Cancer			**Satlog**		
f	**CSL**	**DRL**	**TF**	**CSL**	**DRL**	**TF**
5	0.91	0.76	0.95	0.00	0.23	0.86
10	0.93	0.88	0.93	0.22	0.51	0.90
15	0.92	0.86	0.94	0.49	0.35	0.90
20	0.92	0.92	0.93	0.62	0.45	0.90
25	0.95	0.86	0.94	0.70	0.10	0.91

Table 4.4: Comparing F1 scores for feature selection on low feature count sets. f indicates the number of features acquired.

is evaluated on publicly available real-world healthcare data. Our method, being specific target aware, can consistently pick out a good subset of the features, consequently using fewer features that contribute most to maximizing the selected target class in focus.

As can be seen in the diabetes confidence evaluation in Figure 4.5, and in the FP/FN evaluation in Figures 4.6a to 4.6d, our method outperformed the compared methods in minimizing FP and FN scores quickly, in addition to achieving a consistent amount of confidence in the target of interest relatively fast. It can be seen in Figure 4.5 that all comparable methods achieve a high amount of confidence quicker than our target-focused method. However, comparing Figures 4.6a to 4.6d we can see our model minimizes false positive and false negative scores quicker and therefore receives higher F1 scores.

Confidence evaluation for the heart disease dataset can be seen in Figure 4.4. Our model is gaining confidence using fewer features as before and keeps a relatively increasing trend of confidence. Other models failed to increase their confidence significantly as this was the most complex task, with multiple targets and high dimensionality.

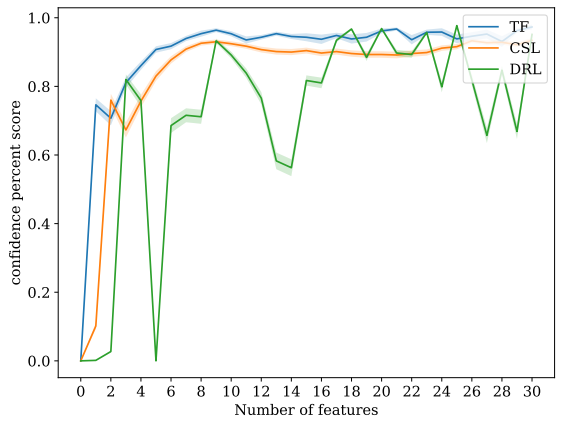


Figure 4.7: Comparing model confidence in predicting malignant breast cancer. Line thickness indicates variance.

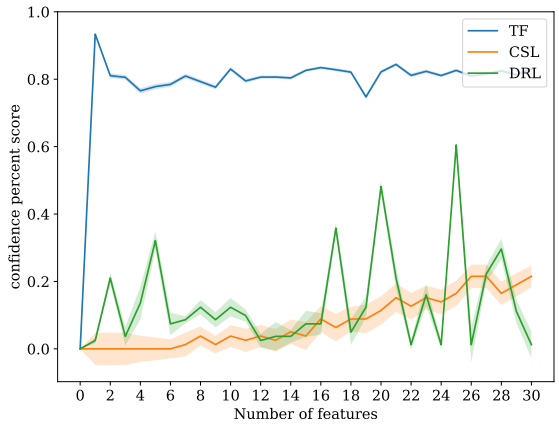


Figure 4.8: Comparing model confidence in predicting one class out of the Satlog dataset. Line thickness indicates variance.

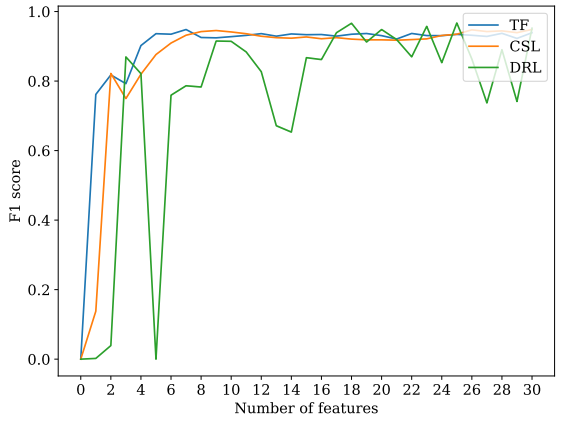


Figure 4.9: Comparing model F1 curve in predicting malignant breast cancer. Line thickness indicates variance.

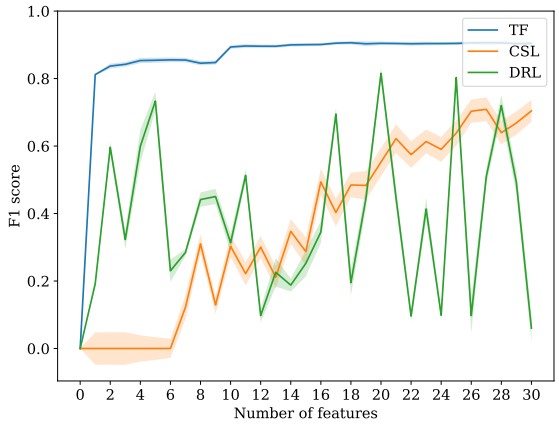


Figure 4.10: Comparing model F1 curve in predicting one class out of the Satlog dataset. Line thickness indicates variance.

4.5.2 Comparing with deep learning methods

As shown in Table 4.4, we achieve faster convergence on all datasets with little missingness and low feature count.

Similar to the statistical methods comparison, our model can be seen gaining confidence faster than compared methods. Figure 4.7 shows a confidence trend for acquiring 30 features on the Breast Cancer Wisconsin dataset using our target focused (TF) method, comparing specific target confidence in the three compared models. Our model can be seen gaining confidence slightly faster than the CSL method. As shown in Figure 4.9 the F1 score curve climb suggests faster learning of our target of interest using fewer features in both our method and CSL. As our method gains confidence faster, it has a slight advantage on CSL in terms of observed performance on predicting our target.

Figure 4.8 shows a confidence trend for acquiring 30 features on the Satlog dataset using compared methods. For the task with high class count and low feature count, we are able to show much higher confidence convergence. Since the target chosen was the least frequent one, it appears that both deep learning approaches struggle to learn the target given the amount of data available. Despite data shortage, our model has gained the most confidence while using a low number of features. Figure 4.10 shows the F1 score curve evaluation over the Satlog dataset. We see our method shows a steady non-volatile gain, consistent with learning the target of interest with low FP, FN rates.

In Table 4.5 we can see the F1 scores of the deep learning feature selection methods along with our approach compared on high feature count datasets, with missing values and lots of features. Here our model heuristic is evaluated on publicly available real-world healthcare data. Our method is able to consistently outperform the deep learning methods in terms of F1 scores.

Figure 4.11 shows a steady increase in confidence in our model in what is essentially the most challenging type of task with multiple classes of heart conditions and a vast amount of features. Our target-focused approach is able to gain confidence efficiently towards the target of interest. Looking at the F1 curve in Figure 4.13 suggests that as both CSL and DRL

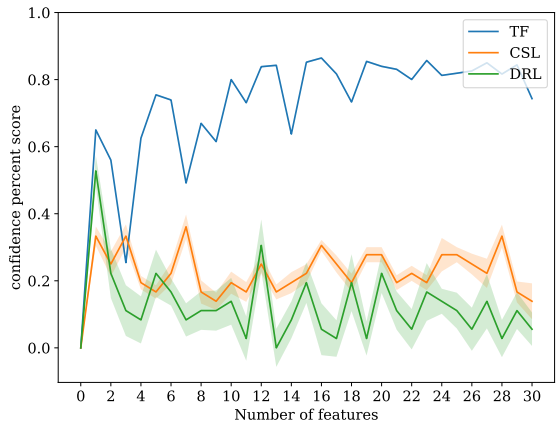


Figure 4.11: Comparing model confidence in predicting congestive heart failure. Line thickness indicates variance.

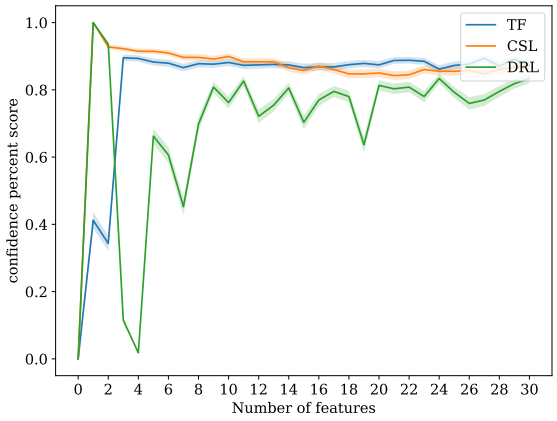


Figure 4.12: Comparing model confidence in predicting diabetes. Line thickness indicates variance.

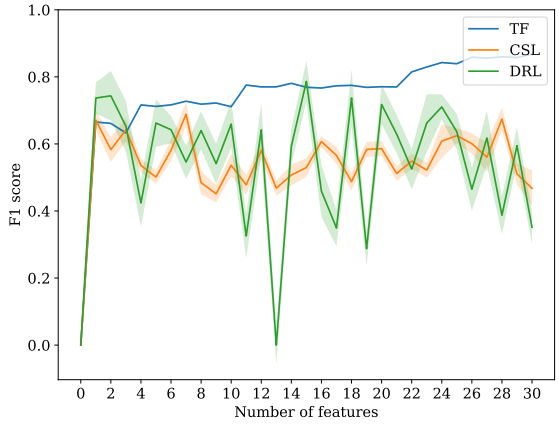


Figure 4.13: Comparing model F1 curve in predicting congestive heart failure. Line thickness indicates variance.

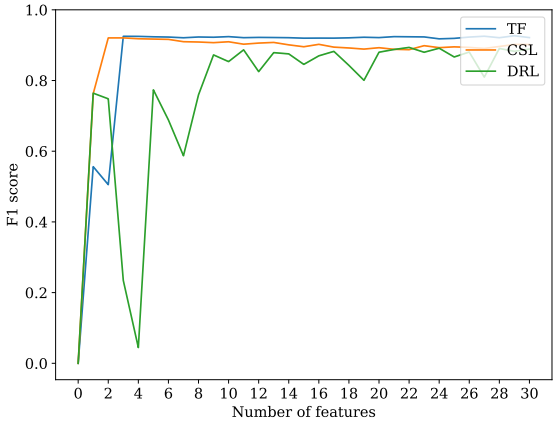


Figure 4.14: Comparing model F1 curve in predicting diabetes. Line thickness indicates variance.

F1 scores						
	NHANES Diabetes			NHANES Heart		
f	**CSL**	**DRL**	**TF**	**CSL**	**DRL**	**TF**
5	0.92	0.69	0.92	0.58	0.64	0.72
10	0.90	0.89	0.92	0.48	0.33	0.78
15	0.90	0.87	0.92	0.61	0.46	0.87
20	0.89	0.89	0.92	0.51	0.63	0.87
25	0.89	0.88	0.92	0.60	0.46	0.86

Table 4.5: Comparing F1 scores for feature selection on low feature count sets. f indicates the number of features acquired.

continue to acquire features, they are not improving the overall performance on our target. Meanwhile, our model can continuously acquire features that will enhance performance on the target of interest and receive higher F1 scores throughout.

Diabetes task confidence can be seen in Figure 4.12, and its corresponding F1 curve in Figure 4.14. While DRL feature selection starts with the most confidence, it can be seen in the F1 plot that the overall performance did not correspond with confidence scores, suggesting high FP rates for our target of interest. Conversely, our approach and the CSL model both did well in gaining confidence towards our target of focus, with our model yielding slightly higher F1 scores.

4.5.3 Final note

Since the other feature selection models are unaware of the single target uncertainty in the model, the results obtained by the compared models could be largely dependent on the distribution of targets. In effect, the selected target of focus might get better results if it is also the majority target. All models compared were able to find features to construct an efficient, frugal model on at least one of the sets, but our method has shown higher consistency across all sets. While real-world health data usually is sparse and feature-rich, we can see that even on smaller datasets with fewer features, our method provides a good

heuristic as to the value of features when acquired towards a single target.

4.6 Impact on Healthcare Discussion

Development of learning models that will maximize confidence and acquire features while considering their costs could be the catalyst for adopting these models by health professionals. For certain diagnostics, the ability to accurately measure the FP and FN rates of a model significantly affects its usability in a healthcare setting. The ability to assist a healthcare provider's diagnostic skills and decision-making process using a learning model is dependent on the provider's ability to trust such a model. Trust could be established if a model accurately advertises its expected statistical guarantees given the data that was fed into it. For example, as a physician diagnoses a patient, feeding low-cost data (such as a standard medical questionnaire) into a machine learning model that is trained to diagnose a specific disease will keep track of a vast amount of possible tests that could be performed next on a patient. Additionally, the target-specific model will provide the expected gain towards a diagnosis or ruling out a hypothesis while considering the cost of all such tests.

Using a confidence-based feature selection method then allows for an informed decision-making process using actionable data from a machine learning model. That model can utilize vastly more data than is typically taken into account in a diagnosis process performed by a human. Training a target-specific model on as much expert knowledge as possible will allow this information to propagate and be used quickly in a machine learning context, where esoteric cases can be efficiently considered by selecting the most critical features to acquire next.

CHAPTER 5

Real-Time Decentralized knowledge Transfer at the Edge[1]

In this chapter, we propose an algorithm for knowledge transfer between agents.

We define a real-time knowledge transfer architecture consisting of horizontal knowledge transfer models in charge of moving information between source layers and target layers, as shown in Figure A.1. Additionally, We evaluate our knowledge transfer method and show it can successfully model data seen by other agents while retaining local insights without exposing any data used by the source models.

5.0.1 Problem Setup

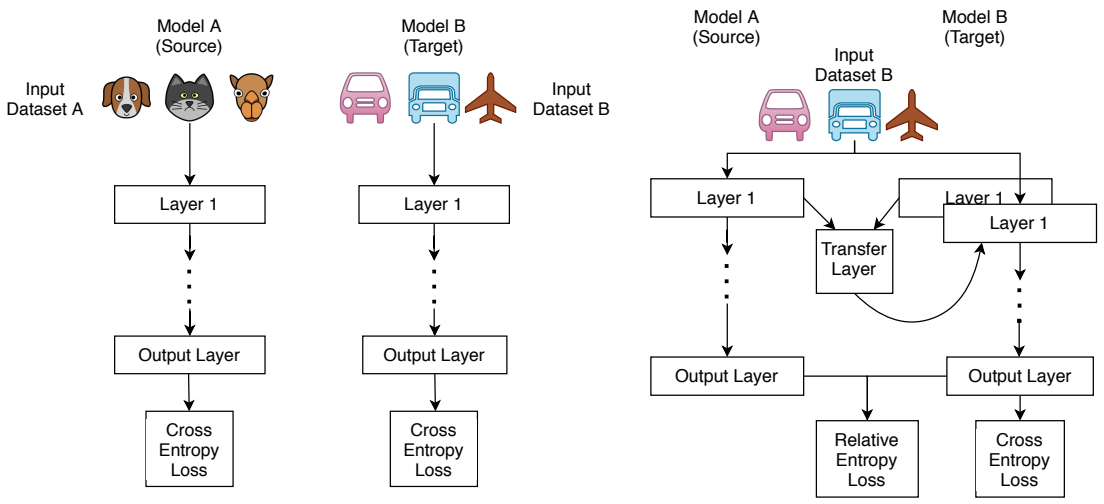
Translating knowledge from source and target parameters to new parameters that contain information from both source and target $M : \overset{2n\times n}{\mathcal{R}} \rightarrow \overset{n\times n}{\mathcal{R}}$, we strive to minimize the following objective function $\mathcal{L}$.

$$\mathcal{L} = |(W^*X + b - W_aX + b)| + |(W^*X + b - W_bX + b)|, \tag{5.1}$$

$$W^* = ((W_a \frown W_b)M), \tag{5.2}$$

where $(\overset{2n\times n}{W_a \frown W_b})$ represents the combined parameters from a source $\overset{n\times n}{W_a}$ and target $\overset{n\times n}{W_b}$, transformed into a new $n \times n$ matrix W^* by the transformation term M. The first term

[1]**This chapter is based on "Goldstein, O., Kachuee, M., Shiell, D. and Sarrafzadeh, M., 2020. Real-Time Decentralized knowledge Transfer at the Edge.** ***arXiv preprint arXiv:2011.05961.***



(a) Local Train Step

(b) Knowledge Transfer Step

Figure 5.1: High-level intuitive illustration of our proposed knowledge transfer method. The transfer layer uses both the source and target model layers parameters and produces a new layer for the target model. The horizontal model then evaluates knowledge transfer by producing a combined loss term, which in turn is used to optimize the transfer process.

is the difference between the outputs using the transformed matrix, and the outputs using the untransformed source. The second term is the difference between the outputs using the transformed matrix and the outputs using the untransformed target.

5.1 Decentralized Learning at the Edge

Our goal is to facilitate knowledge transfer between edge models that are total strangers. The only connection between source and target models is being part of the same network, sharing and consuming metadata about other models in the network. Models are able to choose source models based on their defined task and meta parameters, such as geographical location (for debiasing and data differentiation expectations) and data already acquired. Once a source to target pipeline is created, the target model will converge faster on local data and learn how to predict or classify targets that were seldom or never before seen by the target model. Working on an edge network, real-world costs are taken into account. For

example, the benefit of adding a source should be evaluated against the added model runtime incurred by that specific source. To show this architecture works well in practice, we need to establish real-world guarantees when deploying this architecture on a latency-minded network.

5.1.1 Horizontal Models for Knowledge Transfer

For a target model, transferring information into layer i, we construct a transfer pipeline flowing from a layer in the source model $s_i \in \mathbb{L}_S$, to a corresponding layer in the target model $t_i \in \mathbb{L}_T$. This pipeline consists of a shallow model $m_i \in M$ that is matched to the type of source and target layers (dense, convolution, normalization, etc.). Once a pipeline is in place, the agent learning the target model can control the orchestration of learning steps involving source models. For example, an agent might have a transfer step from each of its source models after every single local step, or it might use an entire epoch for each of the remote and local learning sources.

5.1.1.1 Horizontal model structure

For layers with more than two dimensions (convolutions), we attach a model consisting of a single convolution with an input of $2n$ channels, the concatenated size of the source and target layers, and an output of n channels. For layers with less than two dimensions, we attach a model consisting of a single dense layer with a concatenated input of $2n \times n$ and an output $n \times n$. Therefore source and target layers s_i, t_i must be of the same type. Each transfer model m_i produces a new target layer $t_i^* \in \mathbb{L}_T^*$. Each transfer model for the same source runs at the same iteration, replacing all previous layers $\mathbb{L}_T$ with new layers $\mathbb{L}_T^*$. Figure 5.2 shows an example single model defined between a source and a target layer.

5.1.1.2 Pipeline construction

Constructing the knowledge transfer pipeline involves iterating over source and target model layers and matching a model to pairs of layers. For a source and target using the same

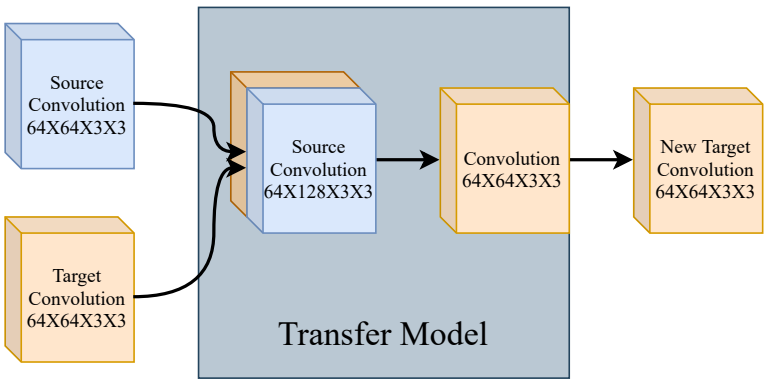


Figure 5.2: example of a convolution knowledge transfer model, producing a single new target layer.

model architecture, there is a 1 to 1 correlation, and a model is created for selected pairs of matching layers. For dissimilar models, there is some freedom with pairing up layers. Since investigating the best possible layer pairing is out of the scope of this work, we focus our examples on similar models, where pairing is done symmetrically. In addition to the unique model, each transfer pipeline receives a personal optimizer. Figure 5.3 shows a demo of our entire pipeline structure.

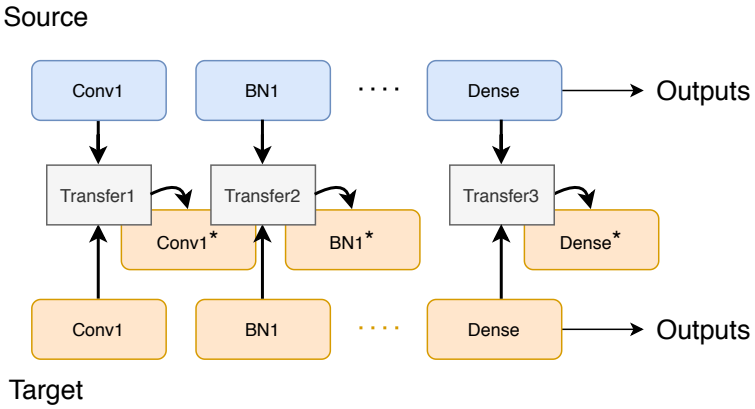


Figure 5.3: Horizontal pipeline structure, producing our replacement target layers.

5.1.1.3 Knowledge transfer objectives

Since our knowledge transfer happens while the target model is training on local data, we have to make sure parameters do not lose information learned locally while optimizing the

knowledge transfer pipeline. We do this by combining two objective functions: one for each local and transfer parts. First, we use a local objective based on the model requirements. In this work, we will use the cross-entropy loss used in classification,

$$loss_1 = \sum_{i=1}^{N} -log\frac{e_i^x}{\sum_j e_j^x}. \tag{5.3}$$

Our second term operates on the new target layers $\mathbb{L}_T^*$ that have been produced by all pipelines and facilitates moving information into the target model. To achieve that, we feed both the source and target models the same local inputs, and measure the difference in the distribution of outputs. Our second loss term is defined as the Kullback-Leibler (KL) divergence Loss between the two vectors P and Q,

$$loss_2 = D_{KL}(P||Q) = \sum_{i=1}^{N} p(x_i) \cdot (log\ p(x_i) - log\ q(x_i)). \tag{5.4}$$

Our learning process is then divided into two interchanging parts. In the first, we train our model as usually done on local data. Using $loss_1$, we update our local target model parameters by inputting local data and calculating the local loss term. In the second, we combine knowledge from source and target layers s_i, t_i and run through the pipeline that produces a new target layer t_i^*. Once all such new layers $\mathbb{L}_T^*$ have been produced, we use the same inputs used in the first local step to evaluate and optimize the transfer pipeline. The loss term for pipeline optimization while preserving local learning is then

$$\mathcal{L} = \frac{\alpha * loss_1 + \beta * loss_2}{2}, \tag{5.5}$$

Where α and β are coefficients controlling the loss magnitude of each of the parts who's sum equals 1.

Once pipeline optimization stagnates, we reduce the learning rate by a constant factor. This allows for a transfer layer to stabilize over time and then remain essentially constant through the remainder of our training time. Each type of layer is allowed a different time to stabilize, i.e., we have found that a convolution transfer layer should be allowed approximately 3 times more epochs with no decreased loss than a dense transfer layer before the learning rate is slashed. Additionally, we have found the factor applied to the learning rate should be five times higher for a dense transfer layer than a convolution transfer layer.

5.2 Collaborating in a Multi-Agent Network

In order to encourage models to collaborate with stranger models, the benefits should outweigh the drawbacks by a good margin. Agents should be able to estimate what collaboration with a potential source would yield. To do that, agents need to share some basic information about their local model and data. Since all agents are part of the same network, making their model available to another agent is a relatively simple task. Any agent on our network will be able to view remote models, explore their architecture, and learned parameters. An agent will also advertise what type of data is fed to their local model. For example, if a model classifies ten different types of vehicles, it should be advertised along with how much data from each class has been fed into the model so far. This information allows a potential collaborator to select models that contain data that they do not see as much of.

For our problem formulation, we assume all agents on our network report statistics regarding their data distribution as well as model statistics and that those models are available for other agents to inspect. In making a selection of sources, an agent will consider the following in its potential source models.

Source task Given a set of possible sources of information $\mathbb{A}$ and a target b, a choice of good sources will take into account the intersection of classes $\mathcal{X}_a \cap \mathcal{X}_b$ observed by a source $a \in \mathbb{A}$ and target b, and the difference between them, $\mathcal{X}_a \setminus \mathcal{X}_b$. A target might want sources with a large difference set, a large intersection, or both.

Source model Given a set of possible sources of information $\mathbb{A}$ and a target b, a choice of good sources will consider the structure of the source model and account for the intersection of layers between all layers of a source $\mathbb{L}_a$ where $a \in \mathbb{A}$ and layers of a target $\mathbb{L}_b$, $\mathbb{L}_a \cap \mathbb{L}_b$. A target might only be interested in sources with the same structure as his own for a 1 to 1 relationship or would consider models with a different structure where there might not be a direct match for each layer on either side.

Source data stats For each of the classes seen by a source model $x_{ai} \in \mathcal{X}_a$, where $a \in \mathbb{A}$, the distribution of data seen for each of the classes will be considered.

5.2.1 Algorithm

Algorithm 2: Train target model using local data and remote knowledge

Input:

b is the target model, l_{bi} is the ith layer of the model

A is a list of remote sources, $\mathbb{L}_{ai}$ are the ith layers of the source models

M is an array of models corresponding to each layer of interest.

x is the local data sample

repeat

$outputs_1 \leftarrow b(x)$

$loss_1 \leftarrow$ **LocalLossTerm**$(outputs_1, labels_1)$

$loss_2 \leftarrow 0$

$b \leftarrow$ **LocalOptimizer**$(loss_1, b)$

for a **in** A **do**

$outputs_2 \leftarrow a(x)$

$loss_2 = loss_2 +$

RemoteLossTerm$(outputs_1, outputs_2)$

end for

$loss_2 = loss_2 /$ **SizeOf**(A)

$loss = (loss_1 + loss_2)/2$

for m_i **in** M **do**

$m_i \leftarrow$ **PipelineOptimizer**$(loss_2, m_i)$

$l_{bi} \leftarrow m_i(l_{bi}, \mathbb{L}_{ai})$

end for

until end training

Our Algorithm 2 describes the sequence of operations taken on the local agent's side and

abstracts away some of the procedures taken when optimizing and fetching remote source knowledge. Additionally, since this is the local view of the knowledge transfer process, the remote sources training process is not explicitly shown here but is assumed to be happening simultaneously. Each of the models $b, m_i \in M$ has its optimizer that scales as knowledge gain plateaus locally or from a remote source.

5.2.2 Runtime

5.2.2.1 Hardware used

Running our experiments, we used a single NVIDIA® Tesla® M60 GPU per target model. Our machine was equipped with 36 core Intel® Xeon® Silver 4210 and 128Gb of memory. Approximately 32Mb of disk space was used to keep information about remote source models used in the training process.

5.2.2.2 Analysis

Using our pairwise transfer requires multiple steps for each epoch in addition to the local training step. Clearly, the larger the number of remote collaborating sources, the more time each epoch lasts. Extensive experiments using two remote sources showed an increase of approximately a 1.28 factor to the target model's run time. However, an encouraging measurement shows that learning time, as measured in epoch over accuracy gained, has decreased slightly more than the increase in learning time at 0.7 of the time required to achieve similar accuracy in the non-knowledge-transfer version of our model.

Going through a single epoch of CIFAR10 data using a batch size of 128 takes approximately 30 seconds for a single target model and two remote models. Training our model for 30 epochs, as reported in the chapter, takes approximately 15 minutes. The amount of messages exchanged on the network were a total of 60.

5.3 Experiments

We experiment with our knowledge transfer architecture and validate it performs well in several key areas. We make a comparison both with similar methods, as well as different configurations of a knowledge transfer network. We test a learning set-up where models spend one epoch learning local data, followed by one epoch of transferring knowledge from remote sources. Datasets used in our comparisons are CIFAR-10 [KH09], CelebA [LLW15] and FMNIST [XRV17]. Data is distributed non-i.i.d. between three agents. For CIFAR-10 and FMNIST, Local data comprises four out of the ten targets available in the datasets, and each of the remote source models is trained on three of the targets available. For CelebA, we focus on detecting smiling faces and distribute the data between the three agents without overlap. To emulate a situation where a local model has minimal exposure to some targets and needs to acquire knowledge about these targets from a remote model, we add 5% random data to the local data pool. This addition creates a slight overlap with training data to the remote models. For all our experiments and comparisons, we use Resnet-50 as our base model.

5.3.1 Comparing with related models

Our first comparison is with a modified knowledge distillation [HVD15] method, where we adapt knowledge distillation to a real-time, actively learning scenario. Since pairwise knowledge transfer methods such as transfer learning and knowledge distillation are not designed for real-time knowledge transfer, we compare with a variation of these that is a valid option for real-time execution. The basic form of knowledge distillation can be applied to a real-time knowledge transfer case by minimizing the logit distance of the source and target models as they are learning new data. Our second comparison is based on [VBT17], where we implement an ADMM [ZYZ18] Gossip algorithms based neighbor-to-neighbor knowledge transfer with a random component. Lastly, we compare to a decentralized, federated learning approach for non-i.i.d. data as described in [ZLL18, MMR16]. We consider the consolidated result of this approach to be comparable to our local model when only a single agent's

Table 5.1: Comparison of average accuracy after 25 epochs across our 3 compared models. Accuracy was averaged over 25 runs.

Average Accuracy - CIFAR-10

	Ours	KD	Gossip	Federated
Local data	0.95	0.96	0.63	0.70
Remote data	0.60	0.50	0.52	0.66
Combined	0.77	0.71	0.56	0.68

Average Accuracy - CelebA

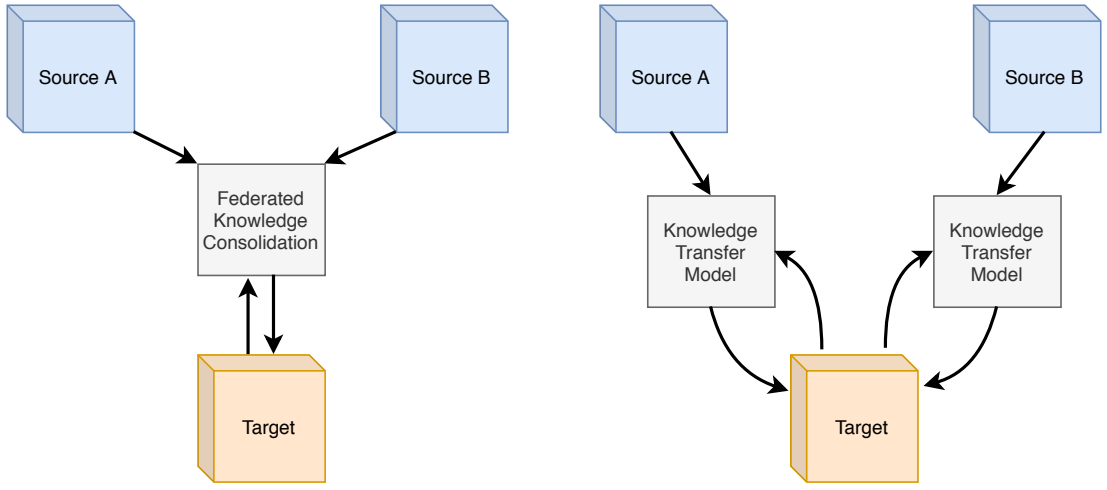
	Ours	KD	Gossip	Federated
Local data	0.97	0.94	0.70	0.93
Remote data	0.91	0.93	0.54	0.91
Combined	0.93	0.93	0.60	0.92

Average Accuracy - FMNIST

	Ours	KD	Gossip	Federated
Local data	0.93	0.92	0.64	0.76
Remote data	0.67	0.64	0.66	0.76
Combined	0.77	0.75	0.65	0.76

improvement is measured, where that agent corresponds to our termed local agent.

Figure 5.4 depicts the difference between the pairwise knowledge transfer approach (ours, knowledge distillation, and Gossip) and the federated approach (Federated Learning).



(a) Federated knowledge transfer

(b) Pairwise knowledge transfer

Figure 5.4: Comparing the two knowledge transfer configurations used in our comparison.

Table A.1 compares accuracy on the our sets. We report the average accuracy over 25

runs for each local, remote, and combined target. As shown in the table, our method can best retain local insights while increasing accuracy on all targets. In Figure 5.5 we can see the learning curve of all compared models, in addition to a baseline comparison with a Resnet-50 model trained on local data with no knowledge transfer. As can be seen in the plot, our method can increase accuracy faster by facilitating knowledge transfer and is stabler while in the process of learning.

Tables 5.2, 5.3, 5.4, and 5.5 show the distribution of predictions for each of our compared models as they compare to true labels in the CIFAR-10 dataset. Each line represents a single true label. The diagonal shows true positive predictions, and other entries on the row represent false negative predictions.

Exploring our purposed knowledge transfer, we investigate Tables 5.2, and 5.3. As knowledge is transferred to our model we transition from Table 5.2 to Table 5.3. We can observe the redistribution of predictive knowledge on the different labels and the increase in true positives for the data seldom observed before (labels plane to dog incl.). For example, we can see that the knowledge transferred to our local model improved the number of correct predictions of deer from 452 to 678. In addition, deer were previously classified as frog 174 times, but only 70 times after knowledge transfer. On the other hand, horse, which was observed abundantly by our model, was mistaken for deer only 8 times and for a frog, 19 times before knowledge transferred but was mistaken for deer 28 times after and for frog only 9 times. We believe this is evidence of true observable intuitive knowledge transferring between models. After all, a horse *is* closer to deer in appearance, and therefore our model adjusted its mistakes in a way that makes intuitive sense. Moreover, our model corrected its understanding of labels that were sparsely seen before, i.e., fewer false negative predictions were made on these labels. Those were redistributed to labels that are closer in appearance.

Comparing with related models, we investigate Tables 5.4, and 5.5. It can be seen that the distribution of correct predictions is more evenly spread out across all targets, indicating adopting more remote knowledge while abandoning some of the local insights learned by the model. For example, the federated and gossip-based models predicted that horses are a bird 33 and 46 times, respectively. These mistakes happened even though local knowledge

contained an abundance of horse examples that should have been prioritized over increasing global accuracy. Another interesting distinction is observing the subset of targets making a large percentage of the accuracy gained in the federated and gossip-based models. For example, our model made most of its mistakes by incorrectly classifying cars as trucks. On the other hand, since trucks were abundantly observed, very few mistakes were made in classifying trucks. In the federated and gossip models, the number of cars falsely classified as trucks decreased significantly, while incorrectly classifying trucks increased proportionally.

This example brings to light the trade-off that we need to take into account when knowledge transferring. In our attempt to preserve local insights while increasing global accuracy from remote knowledge, we give up on our model gaining a more uniformly distributed understanding of the other targets. For our use case, where a model's priority should be the local data, global accuracy can still be increased by having knowledge distributed less evenly within our model.

(a) CIFAR-10 (b) FMNIST

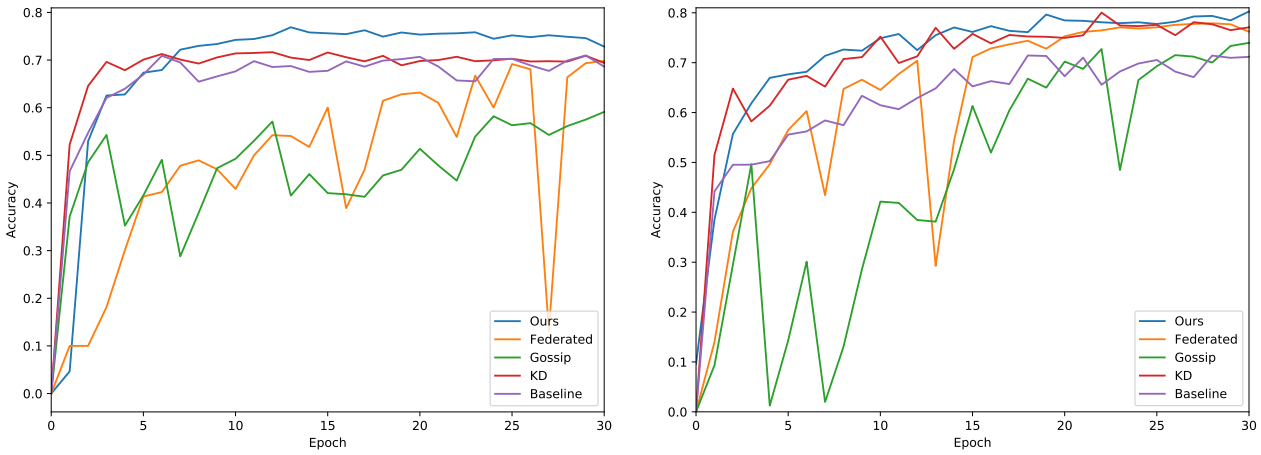


Figure 5.5: Comparing knowledge distillation, Gossip algorithms based method, decentralized federated learning, and our method.

Tables 5.6, 5.7, 5.8, and 5.9 show the distribution of predictions for each of our compared models as they compare to true labels in the FMNIST dataset. Each line represents a single true label. The diagonal shows true positive predictions, and other entries on the row represent false negative predictions.

Comparing true labels vs. predicted labels for our Resnet-50 on local data, with no knowledge transfer on CIFAR-10. T denotes true labels, P denotes predicted label.

Table 5.2: No knowledge transfer

	P plane	*P car*	*P bird*	*P cat*	*P deer*	*P dog*	*P frog*	*P horse*	*P ship*	*P truck*
T plane	537	1	35	2	13	5	36	45	193	133
T car	5	510	0	2	3	7	34	14	103	322
T bird	52	1	405	25	50	30	236	139	48	14
T cat	15	0	22	312	26	88	286	171	52	28
T deer	13	1	36	32	452	7	174	235	43	7
T dog	6	1	19	103	18	475	135	212	18	13
T frog	2	0	4	7	5	6	961	9	4	2
T horse	1	0	2	5	8	7	19	942	7	9
T ship	4	3	0	1	1	0	9	2	953	27
T truck	3	3	0	5	0	0	9	14	16	950

Table 5.3: Our Pairwise knowledge transfer

	P plane	*P car*	*P bird*	*P cat*	*P deer*	*P dog*	*P frog*	*P horse*	*P ship*	*P truck*
T plane	705	4	45	4	15	1	17	18	107	84
T car	16	693	3	0	0	4	12	3	46	223
T bird	70	0	589	25	86	40	90	52	24	24
T cat	21	1	53	459	53	139	124	83	25	42
T deer	23	1	57	32	678	22	70	94	20	3
T dog	9	2	42	146	27	599	46	99	9	21
T frog	3	2	16	7	6	6	946	4	4	6
T horse	4	0	4	13	28	13	9	916	1	12
T ship	21	4	3	1	1	0	6	2	952	10
T truck	8	21	1	3	0	1	3	3	21	939

Table 5.4: Federated learning

	P plane	*P car*	*P bird*	*P cat*	*P deer*	*P dog*	*P frog*	*P horse*	*P ship*	*P truck*
T plane	588	43	61	44	42	15	14	16	111	66
T car	15	791	7	10	5	13	9	14	36	100
T bird	58	5	467	112	105	90	61	55	36	11
T cat	27	13	74	435	88	182	71	62	20	28
T deer	22	7	117	102	507	69	51	90	23	12
T dog	19	5	60	253	60	475	28	75	13	12
T frog	5	13	72	99	63	51	659	13	11	14
T horse	21	6	33	76	76	80	11	663	7	27
T ship	61	49	15	29	18	12	7	9	763	37
T truck	30	116	10	40	12	13	15	18	54	692

Table 5.5: ADMM Gossip

	P plane	*P car*	*P bird*	*P cat*	*P deer*	*P dog*	*P frog*	*P horse*	*P ship*	*P truck*
T plane	726	11	76	31	22	4	4	9	91	26
T car	18	841	7	10	2	4	13	1	41	63
T bird	59	5	608	70	104	45	52	21	23	13
T cat	26	8	94	514	83	142	52	39	24	18
T deer	18	2	81	64	697	33	40	47	15	3
T dog	12	4	60	195	63	568	21	48	18	11
T frog	8	5	53	84	50	14	769	6	9	2
T horse	23	3	46	42	88	56	8	707	7	20
T ship	38	23	11	16	2	9	3	2	883	13
T truck	40	111	7	21	4	0	4	12	45	756

Comparing tables 5.6 and 5.7 we can see the increase in global accuracy when knowledge transfer is used. While in the non knowledge transfer matrix there is a slightly better accuracy in some of the local targets (shirt to Ankle boot incl.) it generally performs worst when classifying remote data and makes more mistakes on these. After knowledge transfer we can see some local targets benefit, for example sneaker. We believe this is due to gaining knowledge of what sandals look like as the errors made misclassifying these decreased dramatically.

Comparing with other decentralized knowledge transfer models, tables 5.8 and 5.9 show a more uniform distribution of the knowledge. Knowledge assimilated in these models seem to come in expense of our local knowledge. For example, in our model, the remote targets (T-shirt to Sandal incl.) are heavily miscllasified as a shirt. In both the Federated and Gossip approaches these mistakes decrease. However, this comes on the expense of classifying our local data, shirt, correctly. As can be seen in the tables, the true positives of shirt decreased in both Federated Learning and Gossip. This happens because these model gain information that contributed to global accuracy but damages the model's understanding of local data.

5.3.2 Different mesh configurations

Benchmarking our model in different network configurations captures the various possibilities of configured knowledge transfer networks, where remotely defined components might be out of our control. For example, a remote source might or might not have an active pipeline with another unrelated agent, changing the overall knowledge transfer graph. Conducting these experiments is meant, on the one hand, to show the benefits of our approach as compared to a centralized method, and on the other, to capture the added benefit that could be gained by having agents cooperating on the network in a pairwise fashion. When allowing agents to construct pipelines in a pairwise approach, network configurations could be more elaborate than a centralized knowledge consolidation. Here we compare our method in four such scenarios.

1. The local agent has defined pipelines with two remote agents, but they **did not** ex-

Comparing true labels vs. predicted labels for our Resnet-50 on local data, with no knowledge transfer on FMNIST. T denotes true labels, P denotes predicted label.

Table 5.6: No knowledge transfer

	P T-shirt/top	*P Trouser*	*P Pullover*	*P Dress*	*P Coat*	*P Sandal*	*P Shirt*	*P Sneaker*	*P Bag*	*P Ankle boot*
T T-shirt/top	351	1	17	12	0	0	605	0	13	1
T Trouser	0	947	1	5	2	1	40	0	4	0
T Pullover	1	1	551	1	74	0	366	0	6	0
T Dress	8	9	8	525	39	0	392	0	15	4
T Coat	0	0	70	5	485	0	431	0	8	1
T Sandal	0	0	8	0	0	731	3	143	78	37
T Shirt	14	1	19	3	17	0	927	0	19	0
T Sneaker	0	0	0	0	0	4	0	836	31	129
T Bag	2	0	1	0	0	0	15	1	980	1
T Ankle boot	0	0	0	0	0	1	1	8	0	990

Table 5.7: Our Pairwise knowledge transfer

	P T-shirt/top	*P Trouser*	*P Pullover*	*P Dress*	*P Coat*	*P Sandal*	*P Shirt*	*P Sneaker*	*P Bag*	*P Ankle boot*
T T-shirt/top	551	3	38	20	1	1	366	0	20	0
T Trouser	0	949	18	6	5	0	18	0	4	0
T Pullover	8	1	657	4	48	0	253	0	29	0
T Dress	8	9	70	617	20	2	263	0	11	0
T Coat	0	2	139	14	495	0	318	0	32	0
T Sandal	0	0	0	0	0	872	0	60	22	46
T Shirt	55	2	77	9	30	2	801	0	24	0
T Sneaker	0	0	0	0	0	36	0	842	10	112
T Bag	0	0	2	2	0	0	14	0	979	3
T Ankle boot	0	0	0	0	0	1	0	9	1	989

Comparing true labels vs. predicted labels for our Resnet-50 on local data, with no knowledge transfer on FMNIST. T denotes true labels, P denotes predicted label.

Table 5.8: Federated learning

	P T-shirt/top	*P Trouser*	*P Pullover*	*P Dress*	*P Coat*	*P Sandal*	*P Shirt*	*P Sneaker*	*P Bag*	*P Ankle boot*
T T-shirt/top	752	6	40	76	12	1	87	3	23	0
T Trouser	4	951	6	27	6	0	3	0	3	0
T Pullover	12	0	729	21	1139	0	90	0	9	0
T Dress	21	8	17	875	40	0	32	0	7	0
T Coat	1	1	126	42	705	0	116	0	9	0
T Sandal	0	0	0	3	0	933	0	44	2	18
T Shirt	125	6	154	66	102	0	509	0	38	0
T Sneaker	0	0	0	0	0	42	0	918	1	39
T Bag	1	1	9	4	2	6	10	8	959	0
T Ankle boot	0	0	0	1	0	7	1	71	1	919

Table 5.9: ADMM Gossip

	P T-shirt/top	*P Trouser*	*P Pullover*	*P Dress*	*P Coat*	*P Sandal*	*P Shirt*	*P Sneaker*	*P Bag*	*P Ankle boot*
T T-shirt/top	860	3	20	43	7	3	50	0	14	0
T Trouser	2	962	3	22	4	0	5	0	2	0
T Pullover	14	1	785	13	97	0	84	0	6	0
T Dress	35	10	26	865	31	0	29	1	2	1
T Coat	1	0	124	50	722	0	96	0	7	0
T Sandal	0	0	0	1	0	944	0	40	3	12
T Shirt	240	3	140	43	87	0	460	0	27	0
T Sneaker	0	0	0	0	0	23	0	944	0	33
T Bag	4	0	6	4	6	1	7	5	966	1
T Ankle boot	0	0	0	0	0	10	1	49	0	940

Table 5.10: Comparison of average accuracy after 25 epochs across our 3 compared mesh configurations. Accuracy was averaged over 25 runs.

Average Accuracy - CIFAR-10

	Our Half Mesh	Our Full Mesh	Our Transitive	Federated
Local data	0.95	0.93	0.97	0.70
Remote data	0.60	0.61	0.56	0.66
Combined	0.77	0.74	0.72	0.68

change information between them. This corresponds to a half mesh configuration.

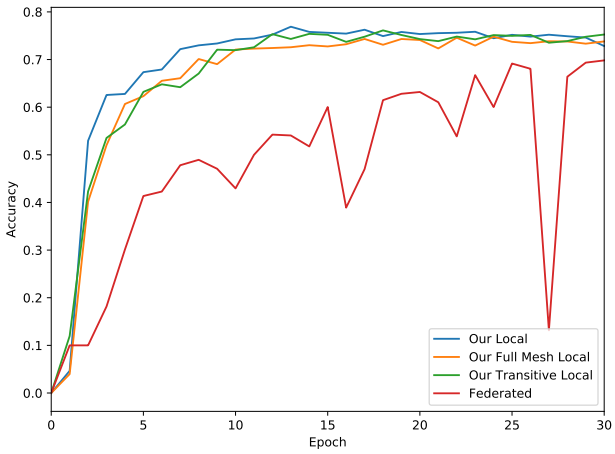
2. The local agent has defined pipelines with two remote agents, and they **did** exchange information between them. This corresponds to a full mesh configuration.

3. The local agent has defined a pipeline with a single remote agent who has himself defined a pipeline with another single remote agent. This corresponds to a transitive knowledge transfer.

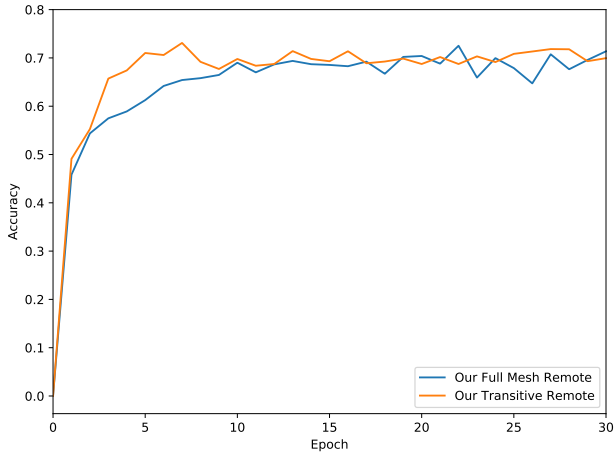
4. In addition to the 3 scenarios above, we compare with federated learning implantation of our model, where all models are consolidated to a single model.

Figures 5.4 and 5.7 depicts the different configurations a network of three learning agents might be arranged in. The average accuracy of our different mesh configuration experiments can be seen in Table 5.10. It can be seen that our model works similarly regardless of the mesh configuration, as long as the local agent has a direct pipeline to the remote models. When considering the transitive configuration confusion matrix (in the appendix), it can be seen that second-hand knowledge transfer is a bit less accurate on remote data and more accurate on local data. This measurement indicates our model did not learn as many remote insights as in the other two configurations. Still, it appears second-hand knowledge is propagated in the model, as the confusion matrix will show.

In Figure 5.6a we can see how our local model learns in the different configurations. While slight differences exist, it appears that knowledge efficiently flows from the two source agents to our target model despite the difference in the network configuration. Additionally,



(a) Comparing different knowledge transfer network configurations of our method, as seen from the local agent's perspective.



(b) Comparing different knowledge transfer network configurations of our method, as seen from a remote agent's perspective.

Figure 5.6: CIFAR-10

we can see that the transitive pipeline causes a slight volatility increase in the learning graph. This volatility could be attributed to our local agent's "second-hand" knowledge from source model A. Compared to the federated method, volatility is considerably less in either network configuration in our method.

In Figure 5.6b we can see a comparison between the same configurations previously inspected, only this time accuracy is measured over the way model B is learning (defined in Figure 5.7). Similar to our local agent, remote model B learns comparably despite the configuration of the network. We remind the reader that model B started with less data (3 classes out of 10), and therefore a slightly lower accuracy could be expected. Despite that, the remote model's accuracy is comparable to the local model's. It seems our method creates an efficient knowledge transfer model based on local insights. Therefore, knowledge gained remotely is absorbed in the local model in similar learning stages, thereby affecting the learning graph similarly.

Tables 5.11 and 5.12 show the distribution of predictions as they compare to true labels for the two new cases of a full mesh knowledge graph and transitive knowledge graph. As we would expect, the full mesh network moves knowledge to the local model similar to the half

Table 5.11: Comparing true labels vs. predicted labels for our local model, with a full mesh knowledge transfer network. T denotes true labels, P denotes predicted label.

	P plane	P car	P bird	P cat	P deer	P dog	P frog	P horse	P ship	P truck
T plane	695	17	67	6	9	0	9	38	93	66
T car	8	773	0	1	0	2	13	4	20	179
T bird	70	6	611	29	39	44	89	81	16	15
T cat	17	14	63	414	32	148	153	94	24	41
T deer	9	1	126	43	567	16	73	138	21	6
T dog	4	15	70	127	17	579	52	106	10	20
T frog	4	9	38	4	3	3	922	6	4	7
T horse	2	2	9	12	10	20	4	928	4	9
T ship	17	13	3	1	2	0	4	5	939	16
T truck	8	23	0	1	0	1	2	9	11	945

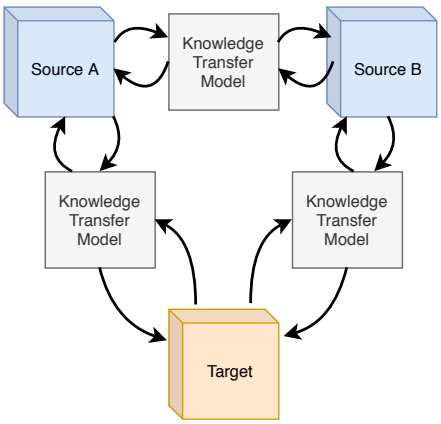
Table 5.12: Comparing true labels vs. predicted labels for our local model, in a transitive knowledge transfer network. T denotes true labels, P denotes predicted label.

	P plane	P car	P bird	P cat	P deer	P dog	P frog	P horse	P ship	P truck
T plane	631	7	45	3	12	0	26	40	146	90
T car	7	715	0	1	0	3	14	6	36	218
T bird	40	4	495	25	62	33	194	102	28	17
T cat	12	1	34	422	27	97	204	118	34	51
T deer	15	1	26	29	613	11	122	157	22	4
T dog	6	0	22	138	18	485	114	178	12	27
T frog	0	0	5	5	4	0	977	5	2	2
T horse	2	0	0	6	7	5	8	958	5	9
T ship	6	6	0	1	1	0	5	2	961	18
T truck	2	9	0	1	0	0	1	8	15	964

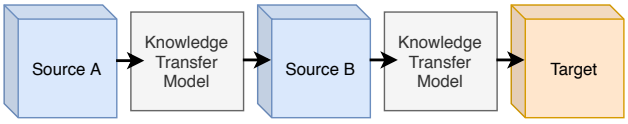
mesh network. We can see true positives increase and false negatives fall in a very similar way. Some dissimilarities do exist; for example, one relatively big difference we can see is that car true positives rose significantly higher on the full mesh network, suggesting an increased aptitude to learning that label in the full mesh structure. In the transitive network, all true positives similarly increased. One interesting observation is that the frequently observed labels (label frog to truck incl.) increased in true positive observations, even above the no knowledge transfer case. This result suggests that exchanging information with remote models could improve our ability to learn local data.

5.4 Discussion

In an edge-like networking environment, local data will affect how a local agent operates. Inability to share local data limits how agents could collaborate and benefit from valuable local insights unavailable elsewhere. Constructing pipelines to transfer knowledge without exposing private data is one way agents can collaborate and source learned insights into data unavailable to them locally. Prioritizing local knowledge in the transfer process is critical in the way agents operate on edge networks. Since data is distributed non-i.i.d. geographically



(a) Full mesh knowledge transfer



(b) Transitive knowledge transfer

Figure 5.7: Comparing three knowledge transfer configuration over a network of learning agents not willing to share their local data.

(across edge nodes), it is essential to train transfer models that are "familiar" with the specific distribution of a source. Additionally, minimizing communication costs by having as few transfer steps as possible is important. For this reason, having a separate trained component to facilitate the transfer helps reduce the training time of the target model.

This chapter has proposed such a set-up and has shown the benefits to models that collaborate with remote sources learning different data. Exploring different possible configurations, we have shown that although a knowledge transfer network structure depends on agents we do not control, Our local model benefits in the same way from the ability to source insights from remote models, regardless of the structure.

As seen in our experiments, a modified knowledge distillation method can achieve close results to our proposed method. In some cases, this will result in better performance as there is no extra model to train. However, in most collaborative cases, having a transfer model that is co-evolved with a source will create a less volatile and more accurate pipeline, as shown in our experiment results. While we reported results for our cutoff time of 25-30 epochs, it should be noted that in the case of FMNIST federated learning will achieve better accuracy than our model given enough time. We believe that for cases where many sources will be considered, time to converge and volatility are non-issues, and local accuracy is not

prioritized, federation might be a better choice. However, in the case where we wish to preserve local insights and limit the communication of a handful of agents on a network, our method will be a better choice.

CHAPTER 6

Conclusion

This thesis has examined the role decentralized networks play in near future connectivity requirements and has proposed an architecture and algorithms to close that gap. We have demonstrated the benefits of having both computational paths that can operate as meta-data exchange channels for coordination or limited message passing. And the benefit of having a globally connected data store for context retention and global low latency availability. We have investigated the cost-aware approach of acquiring features based on a specific target of interest out of two or more targets. We see a frugal approach as an essential addition to the process of feature selection on data streams as data availability grows dramatically and utilization of data remains somewhat inefficient, particularly in the domain of healthcare. We have discussed the application of our target-focused approach to both well-known sources of machine learning datasets and real-world public healthcare data converted into datasets. We have demonstrated the value of having a target-aware method to feature selection compared to feature selection methods that are target-agnostic. We have introduced a Bayesian confidence-based scoring mechanism that we proceeded to show is robust in both scalability and consistency on different types of datasets. Practically, we were able to minimize uncertainty in a specific target of interest with a minimal budget while minimizing the general uncertainty, false positive, and false-negative rates. Finally, we proposed an algorithm for decentralized knowledge transfer tailored to edge networks. Our method can efficiently source information from remote models based on a target model's needs and prioritize locally learned data while enhancing understanding of data that was not available locally.

APPENDIX A

Decentralized Knowledge Transfer for Cancer Detection from Images

Medical data observed in patients local to a provider are by definition a biased sample, as different populations will experience different variants of a medical condition depending on local conditions and genetic variants. In the case of skin cancer, it would be fair to assume that in sunny places where exposure to UV radiation is expected, skin cancer will offer us more variants of this disease than in a cold place with less UV exposure. In an optimal scenario, the ability to exchange local medical data collected by providers on demand would have lead to better performing predictive and classification models, especially on cases that would be otherwise rarely seen locally [SWL02, SZC20]. For example, a model in Sweden could ideally source information it lacks from the Australian model about skin cancer variations when it encounters a sample it is unsure of. However, since medical data are highly protected, exchanging raw patient data in this way would not be possible. We would then require a different mechanism for exchanging model knowledge without exposing patient data.

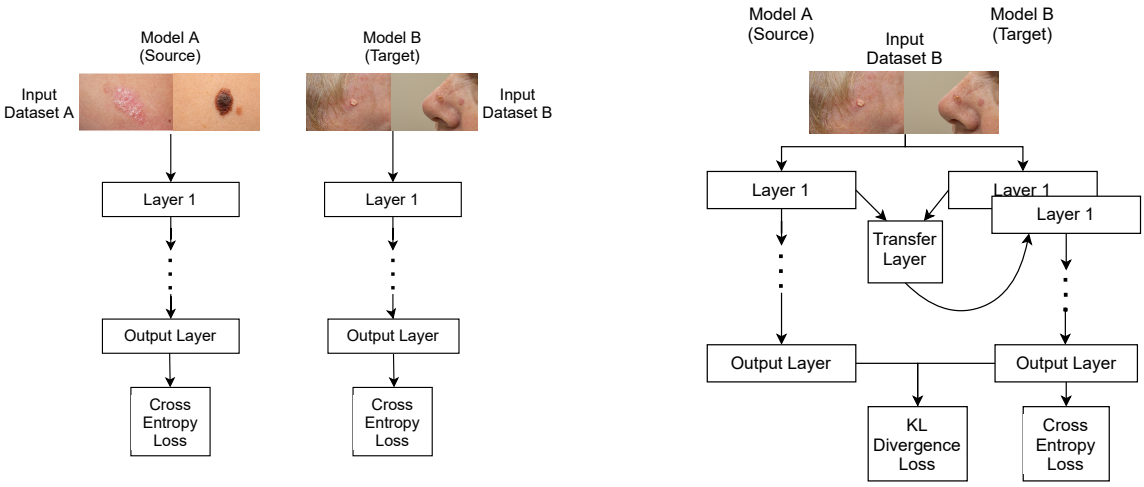
Using a method for real-time knowledge transfer tailored to the decentralized nature of edge networks [GKS20], we propose a dynamic information exchange network for medical data distributed unevenly based on location. Specifically, we exchange information learned regarding cancer classification allowing learners to take advantage of information learned by another remote agent, as shown in Figure A.1. This knowledge transfer pipeline allows for surgical precision in adding knowledge and is done privately, with no need to share data used to construct the model. Using this knowledge transfer mechanism enables accurate answers to model queries on data sparsely or never before seen by a local model.

Transfer Learning [TS10, RMK05, YZH18] is one knowledge transfer mechanism commonly used machine in learning. This technique leverages a pre-trained source model to improve training a target model for a corresponding task. Recycling learned parameters, a source model is used as a basis for a learning process where it is adapted to new information by limiting further training to the lower layers of the source model. Transfer Learning improves the time it takes to learn the new task in the same domain as the source task as well as the final performance of the model. Knowledge distillation from a group of source models [HVD15, LKS20] is another approach for knowledge transfer where the original set or a subset of the data used to train the source models is leveraged. Distilling knowledge from source to target is done by defining a cross-entropy loss between outputs of source and target softmax layers. Detecting breast cancer in images, transfer learning has been used to adapt a deep learning model pre-trained on generic image classification [DSK18]. This approach involves using two well-known image classification models as source targets and allowing retraining of the last two layers, convolution and a dense layer, to absorb the new breast cancer image data. Knowledge distillation can be used to improve detection of chest X-Ray Abnormalities [HG20]. Using a robust teacher-student selection and optimization, this method can enhance the distillation process by suggesting a new saliency technique and using self-training. Transfer Learning deals with training a target model using an already trained source model. In our case, source and target could both still be in the process of learning a model and should still be able to benefit each other. Moreover, transfer learning and distillation are normally evaluated on a distinct set of labels. We are interested in both inheriting the classification power of the source model as well as add local knowledge.

Federated Learning [KMY16, HAA20, MMR17, HTT19, WYS20] aggregates distributed agents updates in real-time into a single centralized model that is better fitted to the overall data. Solving a decentralized modeling of data, the federation is real-time and privacy-preserving, making it especially suited for the data-driven healthcare requirements [RHL20]. Predicting preterm birth from distributed EHR records [BJV19], federated learning is able to keep patient data private while exploiting the knowledge available on local nodes in a robust centralized model. While federated models capture the decentralized knowledge flow

we are interested in, they aggregate it in a centralized manner. A resulting model will trend toward a uniform distribution of confidence overall target classes. In our case, we are after a model that prioritizes local information while still sourcing remote data.

To solve this, we utilize our method for decentralized pairwise knowledge transfer described in chapter 5.



(a) Local Train Step

(b) Knowledge Transfer Step

Figure A.1: High-level intuitive illustration of our proposed knowledge transfer method for cancer data. The transfer layer uses both the source and target model layers parameters and produces a new layer for the target model. The horizontal model then evaluates knowledge transfer by producing a combined loss term, which in turn is used to optimize the transfer process.

A.1 Experiments

We show our method performs well for our experiments when medical data is distributed in a non-i.i.d way across three learning agents. Since a local model arbitrarily selects source models from which to absorb knowledge, we show the case where information is exchanged in only one way, from source to target. We chose a pre-trained Resnet-50 as our base model as it performs well on image classification tasks out of the box. Since the images in

Table A.1: Comparison of average accuracy across our 3 compared models. Accuracy was averaged over 25 runs.

Average Accuracy

	Brain tumour		Skin cancer	
	transfer	no transfer	transfer	no transfer
Local data	0.96	0.99	0.86	0.88
Remote data	0.92	0.86	0.79	0.52
Combined	0.94	0.92	0.75	0.70

Table A.2: Comparing true labels vs. predicted labels for our local model predicting skin cancer, with no knowledge transfer. T denotes true labels, P denotes predicted label, pink denotes the local classes

	P akiec	*P bcc*	*P bkl*	*P df*	*P mel*	*P vasc*
T akiec	37	0	5	0	3	0
T bcc	10	28	8	5	2	0
T bkl	24	9	71	7	21	0
T df	7	0	0	5	0	0
T mel	15	3	14	1	35	0
T vasc	0	0	0	0	0	19

our cancer datasets are highly similar, we chose to facilitate knowledge transfer only across the two bottom-most layers (the last dense layer and the previous convolution). This way, the high-level image understanding is retained, and information is added to the granular differentiation.

A.1.1 Datasets

We test our method on two cancer detection tasks. The first is a melanoma detection task using skin lesion images [GCC16] which contains over 10,000 images for six classes: Bowen's

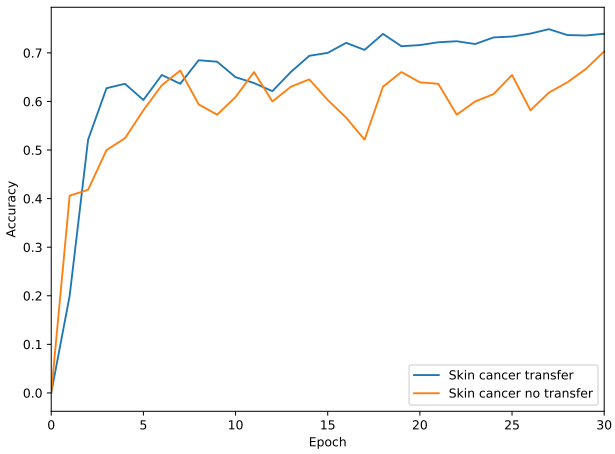
Table A.3: Comparing true labels vs. predicted labels for our local model predicting skin cancer, in a pairwise knowledge transfer environment. T denotes true labels, P denotes predicted label, pink denotes the local classes

	P akiec	*P bcc*	*P bkl*	*P df*	*P mel*	*P vasc*
T akiec	37	0	6	0	2	0
T bcc	7	36	4	3	2	1
T bkl	11	7	86	6	21	1
T df	0	0	2	10	0	0
T mel	9	2	18	0	38	1
T vasc	0	0	0	0	1	18

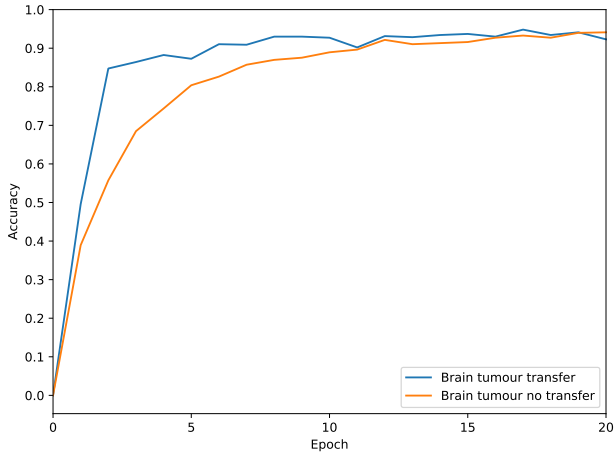
disease (akiec), basal cell carcinoma (bcc), benign keratosis-like lesions (bkl), dermatofibroma (df), melanoma (mel), melanocytic nevi (nv) and vascular lesions (vasc). The second set is brain tumor detection task with over 3,700 images [KBP13], where image classes are a binary "1" for the existence of a tumor and "0" for no tumor. In our tests, we distribute the data between three agents such that no overlap exists between agents. Then we add 5% random data from the entire set to our local agent to simulate a situation where a small amount of data exists locally but not enough to meaningfully learn those classes, consequently promoting the need to source more data in these classes. In the brain tumor case, since classes are binary, we test how exchanging information in a binary task improves decision making.

A.1.2 Results

Our results inspect the benefit of our pairwise knowledge transfer in a network of three learning agents with non-i.i.d distributed data between them. We evaluate the added accuracy from a local agent's perspective, as our pipeline can be defined arbitrarily without remote agent active participation. Table A.1 shows the shift in accuracy on local, remote, and combined data with and without knowledge transfer. The table shows that in all sets, some



(a) Skin cancer identification



(b) Brain tumour identification

Comparing learning with our knowledge transfer, and without our knowledge transfer.

local accuracy has been sacrificed to gain remote knowledge and improve global accuracy. However, since our pipeline prioritizing local information, local accuracy remains close to a non transfer case.

Tables A.2 and A.3 show the resulting confusion matrix over the skin cancer identification targets. Knowledge transfer results in better accuracies for most classes. It can be seen that some additional classification mistakes are made on our local targets. Equivalently less prediction mistakes are made on the remote targets.

Figures A.2a and A.2b show improved learning speed and accuracy when knowledge transfer is added. This improvement indicated that sourcing knowledge not only benefits model accuracy but also speed of convergence.

REFERENCES

[AAB16] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems." *arXiv preprint arXiv:1603.04467*, 2016.

[AGO13] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. "A Public Domain Dataset for Human Activity Recognition using Smartphones." In *ESANN*, 2013.

[AM05] Mussa Abdella and Tshilidzi Marwala. "The use of genetic algorithms and neural networks to approximate missing data in database." In *Computational Cybernetics, 2005. ICCC 2005. IEEE 3rd International Conference on*, pp. 207–212. IEEE, 2005.

[And57] Theodore W Anderson. "Maximum likelihood estimates for a multivariate normal distribution when some observations are missing." *Journal of the american Statistical Association*, **52**(278):200–203, 1957.

[ARZ18] Sanjeev Arora, Andrej Risteski, and Yi Zhang. "Do GANs learn the distribution? some theory and empirics." 2018.

[ASV11] Navid Amini, Majid Sarrafzadeh, Alireza Vahdatpour, and Wenyao Xu. "Accelerometer-based on-body sensor localization for health and medical monitoring applications." *Pervasive and mobile computing*, **7**(6):746–760, 2011.

[AWD20] Aliya Aleryani, Wenjia Wang, and Beatriz De La Iglesia. "Multiple Imputation Ensembles (MIE) for Dealing with Missing Data." *SN Computer Science*, **1**:1–20, 2020.

[BCC17] Ioana Baldini, Paul Castro, Kerry Chang, Perry Cheng, Stephen Fink, Vatche Ishakian, Nick Mitchell, Vinod Muthusamy, Rodric Rabbah, Aleksander Slominski, et al. "Serverless computing: Current trends and open problems." In *Research Advances in Cloud Computing*, pp. 1–20. Springer, 2017.

[BET08] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. "Speeded-up robust features (SURF)." *Computer vision and image understanding*, **110**(3):346–359, 2008.

[BG10] S van Buuren and Karin Groothuis-Oudshoorn. "mice: Multivariate imputation by chained equations in R." *Journal of statistical software*, pp. 1–68, 2010.

[BGP06] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. "Randomized gossip algorithms." *IEEE transactions on information theory*, **52**(6):2508–2530, 2006.

[BHP97] H Bentz, M Hagstroem, and G Palm. "Selection of relevant features and examples in machine learning." *Neural Networks*, **2**(4):289–293, 1997.

[Bis95] Christopher M Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[BJV19] Sabri Boughorbel, Fethi Jarray, Neethu Venugopal, Shabir Moosa, Haithum Elhadi, and Michel Makhlouf. "Federated uncertainty-aware learning for distributed hospital ehr data." *arXiv preprint arXiv:1910.12191*, 2019.

[BM98] Catherine L Blake and Christopher J Merz. "UCI Repository of machine learning databases [http://www. ics. uci. edu/˜ mlearn/MLRepository. html]. Irvine, CA: University of California." *Department of Information and Computer Science*, **55**, 1998.

[CB19] Luca Corinzia and Joachim M Buhmann. "Variational federated multi-task learning." *arXiv preprint arXiv:1906.06268*, 2019.

[CBG00] Diogo Ayres-de Campos, Joao Bernardes, Antonio Garrido, Joaquim Marques-de Sa, and Luis Pereira-Leite. "SisPorto 2.0: a program for automated analysis of cardiotocograms." *Journal of Maternal-Fetal Medicine*, **9**(5):311–318, 2000.

[CC11] Olivier Chapelle and Yi Chang. "Yahoo! learning to rank challenge overview." In *Proceedings of the Learning to Rank Challenge*, pp. 1–24, 2011.

[CCD13] Suming Jeremiah Chen, Arthur Choi, and Adnan Darwiche. "An Exact Algorithm for Computing the Same-Decision Probability." In *IJCAI*, pp. 2525–2531, 2013.

[CCD14] Suming Jeremiah Chen, Arthur Choi, and Adnan Darwiche. "Algorithms and applications for the same-decision probability." *Journal of Artificial Intelligence Research*, **49**:601–633, 2014.

[CCD15] Suming Jeremiah Chen, Arthur Choi, and Adnan Darwiche. "Value of Information Based on Decision Robustness." In *AAAI*, pp. 3503–3510, 2015.

[CDA16a] Gabriella Contardo, Ludovic Denoyer, and Thierry Artières. "Recurrent neural networks for adaptive feature acquisition." In *International Conference on Neural Information Processing*, pp. 591–599. Springer, 2016.

[CDA16b] Gabriella Contardo, Ludovic Denoyer, and Thierry Artieres. "Sequential Cost-Sensitive Feature Acquisition." In *International Symposium on Intelligent Data Analysis*, pp. 284–294. Springer, 2016.

[CDB17] YooJung Choi, Adnan Darwiche, and Guy Van den Broeck. "Optimal feature selection for decision robustness in Bayesian networks." In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, 2017.

[CHM14] Hyunseok Chang, Adiseshu Hari, Sarit Mukherjee, and TV Lakshman. "Bringing the cloud to the edge." In *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 346–351. IEEE, 2014.

[CLC19] Pengfei Chen, Benben Liao, Guangyong Chen, and Shengyu Zhang. "Understanding and Utilizing Deep Neural Networks Trained with Noisy Labels." *arXiv preprint arXiv:1905.05040*, 2019.

[com] multi-access-edge computing. "https://www.etsi.org/technologies/multi-access-edge-computing.".

[CXD12a] Arthur Choi, Yexiang Xue, and Adnan Darwiche. "Same-decision probability: A confidence measure for threshold-based decisions." *Int. J. Approx. Reasoning*, **53**:1415–1428, 2012.

[CXD12b] Arthur Choi, Yexiang Xue, and Adnan Darwiche. "Same-decision probability: A confidence measure for threshold-based decisions." *International Journal of Approximate Reasoning*, **53**(9):1415, 2012.

[CXW12] Minmin Chen, Zhixiang Xu, Kilian Weinberger, Olivier Chapelle, and Dor Kedem. "Classifier cascade for minimizing feature evaluation cost." In *Artificial Intelligence and Statistics*, pp. 218–226, 2012.

[CZC10] B Barla Cambazoglu, Hugo Zaragoza, Olivier Chapelle, Jiang Chen, Ciya Liao, Zhaohui Zheng, and Jon Degenhardt. "Early exit optimizations for additive machine learned ranking systems." In *Proceedings of the third ACM international conference on Web search and data mining*, pp. 411–420. ACM, 2010.

[CZZ13] Peng Cao, Dazhe Zhao, and Osmar Zaiane. "An optimized cost-sensitive SVM for imbalanced data learning." In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 280–292. Springer, 2013.

[Dar09] Adnan Darwiche. *Modeling and reasoning with Bayesian networks*. Cambridge university press, 2009.

[DBP16] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. "Adversarially learned inference." *arXiv preprint arXiv:1606.00704*, 2016.

[DG17] Dheeru Dua and Casey Graff. "UCI Machine Learning Repository.", 2017.

[DK17a] Dua Dheeru and Efi Karra Taniskidou. "UCI Machine Learning Repository.", 2017.

[DK17b] Dua Dheeru and Efi Karra Taniskidou. "UCI Machine Learning Repository.", 2017.

[DL91] J. S. Denker and Y. LeCun. "transforming neural-net output levels to probability distributions." In R. Lippmann, J. Moody, and D. Touretzky, editors, *Advances in Neural Information Processing Systems (NIPS 1990)*, volume 3, Denver, CO, April 1991. Morgan Kaufman.

[DLR77] Arthur P Dempster, Nan M Laird, and Donald B Rubin. "Maximum likelihood from incomplete data via the EM algorithm." *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1977.

[DSK16] Hamid Dadkhahi, Nazir Saleheen, Santosh Kumar, and Benjamin Marlin. "Learning Shallow Detection Cascades for Wearable Sensor-Based Mobile Health Applications." *arXiv preprint arXiv:1607.03730*, 2016.

[DSK18] Erkan Deniz, Abdulkadir Şengür, Zehra Kadiroğlu, Yanhui Guo, Varun Bajaj, and Ümit Budak. "Transfer learning based histopathologic image classification for breast cancer detection." *Health information science and systems*, **6**(1):1–7, 2018.

[EBC10] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. "Why does unsupervised pre-training help deep learning?" *Journal of Machine Learning Research*, **11**(Feb):625–660, 2010.

[EFM16] Kirstin Early, Stephen E Fienberg, and Jennifer Mankoff. "Test time feature ordering with FOCUS: interactive predictions with minimal user burden." In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 992–1003. ACM, 2016.

[EHJ04] Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. "Least angle regression." *The Annals of statistics*, **32**(2):407–499, 2004.

[Ell17] Alex Ellis. "Introducing Functions as a Service (Open-FaaS)." *URL: https://blog. alexellis. io/introducing-functions-as-a-service*, 2017.

[EMF16] Kirstin Early, Jennifer Mankoff, and Stephen E Fienberg. "Dynamic Question Ordering in Online Surveys." *arXiv preprint arXiv:1607.04209*, 2016.

[EVL20] Gudrun Eisele, Hugo Vachon, Ginette Lafit, Peter Kuppens, Marlies Houben, Inez Myin-Germeys, and Wolfgang Viechtbauer. "The effects of sampling frequency and questionnaire length on perceived burden, compliance, and careless responding in experience sampling data in a student population." *PsyArXiv preprint PsyArXiv:10.31234*, 2020.

[FCB07] Alberto Freitas, Altamiro Costa-Pereira, and Pavel Brazdil. "Cost-sensitive decision trees applied to medical data." In *International Conference on Data Warehousing and Knowledge Discovery*, pp. 303–312. Springer, 2007.

[FGJ09] Armando Fox, Rean Griffith, Anthony Joseph, Randy Katz, Andrew Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, and Ion Stoica. "Above the clouds: A

berkeley view of cloud computing." *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, **28**(13):2009, 2009.

[GAS15] Hassan Ghasemzadeh, Navid Amini, Ramyar Saeedi, and Majid Sarrafzadeh. "Power-aware computing in wearable sensor networks: An optimal feature selection." *IEEE Transactions on Mobile Computing*, **14**(4):800–812, 2015.

[GCC16] D Gutman, N Codella, Emre Celebi, Brian Helba, Michael Marchetti, Nabin Mishra, and Allan Halpern. "Skin lesion analysis toward melanoma detection." In *International Symposium on Biomedical Imaging (ISBI),(International Skin Imaging Collaboration (ISIC), 2016)*, 2016.

[GD18] David Güera and Edward J Delp. "Deepfake video detection using recurrent neural networks." In *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 1–6. IEEE, 2018.

[GE03] Isabelle Guyon and André Elisseeff. "An introduction to variable and feature selection." *Journal of machine learning research*, **3**(Mar):1157–1182, 2003.

[GEW06] Pierre Geurts, Damien Ernst, and Louis Wehenkel. "Extremely randomized trees." *Machine learning*, **63**(1):3–42, 2006.

[GG16] Yarin Gal and Zoubin Ghahramani. "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning." In *international conference on machine learning*, pp. 1050–1059, 2016.

[GGI17] Zoubin Ghahramani, Y Gal, and R Islam. "Deep Bayesian Active Learning with Image Data." 2017.

[GGR02a] Russell Greiner, Adam J Grove, and Dan Roth. "Learning cost-sensitive active classifiers." *Artificial Intelligence*, **139**(2):137–174, 2002.

[GGR02b] Russell Greiner, Adam J. Grove, and Dan Roth. "Learning Cost-sensitive Active Classifiers." *Artif. Intell.*, **139**(2):137–174, August 2002.

[Gha15] Zoubin Ghahramani. "Probabilistic machine learning and artificial intelligence." *Nature*, **521**:452 EP –, 05 2015.

[GK11] Tianshi Gao and Daphne Koller. "Active Classification based on Value of Classifier." In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pp. 1062–1070. Curran Associates, Inc., 2011.

[GKK13] Peter Groves, Basel Kayyali, David Knott, and Steve Van Kuiken. "The 'big data'revolution in healthcare." *McKinsey Quarterly*, **2**(3), 2013.

[GKS20] Orpaz Goldstein, Mohammad Kachuee, Dereck Shiell, and Majid Sarrafzadeh. "Real-Time Decentralized knowledge Transfer at the Edge." *arXiv preprint arXiv:2011.05961*, 2020.

[GME15] Pedro Garcia Lopez, Alberto Montresor, Dick Epema, Anwitaman Datta, Teruo Higashino, Adriana Iamnitchi, Marinho Barcellos, Pascal Felber, and Etienne Riviere. "Edge-centric computing: Vision and challenges.", 2015.

[GPM14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets." In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[GPS17] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. "On calibration of modern neural networks." *arXiv preprint arXiv:1706.04599*, 2017.

[HAA20] Chaoyang He, Murali Annavaram, and Salman Avestimehr. "Group Knowledge Transfer: Federated Learning of Large CNNs at the Edge." *Advances in Neural Information Processing Systems*, **33**, 2020.

[Ham94] DM Hamby. "A review of techniques for parameter sensitivity analysis of environmental models." *Environmental monitoring and assessment*, **32**(2):135–154, 1994.

[Has92] Sherif Hashem. "Sensitivity analysis for feedforward artificial neural networks with differentiable activation functions." In *Neural Networks, 1992. IJCNN., International Joint Conference on*, volume 1, pp. 419–424. IEEE, 1992.

[HBB17] Kelsey Hightower, Brendan Burns, and Joe Beda. *Kubernetes: Up and Running Dive into the Future of Infrastructure*. O'Reilly Media, Inc., 1st edition, 2017.

[HDE12] He He, Hal Daumé III, and Jason Eisner. "Cost-sensitive dynamic feature selection." In *ICML Inferning Workshop*, 2012.

[HG20] Thi Kieu Khanh Ho and Jeonghwan Gwak. "Utilizing Knowledge Distillation in Deep Learning for Classification of Chest X-Ray Abnormalities." *IEEE Access*, **8**:160749–160761, 2020.

[HLY19] Byeongho Heo, Minsik Lee, Sangdoo Yun, and Jin Young Choi. "Knowledge transfer via distillation of activation boundaries formed by hidden neurons." In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3779–3787, 2019.

[HMK16] He He, Paul Mineiro, and Nikos Karampatziakis. "Active information acquisition." *arXiv preprint arXiv:1602.02181*, 2016.

[HO13] Stefan Herzog and Dirk Ostwald. "Sometimes Bayesian statistics are better." *Nature*, **494**(7435):35–35, 2013.

[HRU17] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. "Gans trained by a two time-scale update rule converge to a local nash equilibrium." In *Advances in Neural Information Processing Systems*, pp. 6626–6637, 2017.

[HS06] Geoffrey E Hinton and Ruslan R Salakhutdinov. "Reducing the dimensionality of data with neural networks." *science*, **313**(5786):504–507, 2006.

[HTS99] Trevor Hastie, Robert Tibshirani, Gavin Sherlock, Michael Eisen, Patrick Brown, and David Botstein. "Imputing missing data for gene expression arrays.", 1999.

[HTT19] Chaoyang He, Conghui Tan, Hanlin Tang, Shuang Qiu, and Ji Liu. "Central server free federated learning over single-sided trust social networks." *arXiv preprint arXiv:1910.04956*, 2019.

[HVD15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network." *arXiv preprint arXiv:1503.02531*, 2015.

[HYJ16] Shamsul Huda, John Yearwood, Herbert F Jelinek, Mohammad Mehedi Hassan, Giancarlo Fortino, and Michael Buckland. "A hybrid feature selection with ensemble classification for imbalanced healthcare data: A case study for brain tumor diagnosis." *IEEE access*, **4**:9145–9154, 2016.

[HZ94] Geoffrey E Hinton and Richard S Zemel. "Autoencoders, minimum description length and Helmholtz free energy." In *Advances in neural information processing systems*, pp. 3–10, 1994.

[HZR16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[JC07] Shihao Ji and Lawrence Carin. "Cost-sensitive feature acquisition and classification." *Pattern Recognition*, **40**(5):1474–1485, 2007.

[JGJ99] Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. "An Introduction to Variational Methods for Graphical Models." *Machine Learning*, **37**(2):183–233, Nov 1999.

[JGP16] Eric Jang, Shixiang Gu, and Ben Poole. "Categorical reparameterization with gumbel-softmax." *arXiv preprint arXiv:1611.01144*, 2016.

[JK02] Kalervo Järvelin and Jaana Kekäläinen. "Cumulated gain-based evaluation of IR techniques." *ACM Transactions on Information Systems (TOIS)*, **20**(4):422–446, 2002.

[JLH19] Yunhun Jang, Hankook Lee, Sung Ju Hwang, and Jinwoo Shin. "Learning what and where to transfer." *arXiv preprint arXiv:1905.05901*, 2019.

[JLO19] Insik Jo, Sangbum Lee, and Sejong Oh. "Improved Measures of Redundancy and Relevance for mRMR Feature Selection." *Computers*, **8**(2):42, 2019.

[JPL17] Jaromír Janisch, Tomáš Pevnỳ, and Viliam Lisỳ. "Classification with Costly Features using Deep Reinforcement Learning." *arXiv preprint arXiv:1711.07364*, 2017.

[JPL19] Jaromír Janisch, Tomáš Pevnỳ, and Viliam Lisỳ. "Classification with costly features using deep reinforcement learning." In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3959–3966, 2019.

[KA13] Martin Krzywinski and Naomi Altman. "Points of significance: Importance of being uncertain.", 2013.

[KB14] Diederik Kingma and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980*, 2014.

[KBF12] Sergey Karayev, Tobias Baumgartner, Mario Fritz, and Trevor Darrell. "Timely object recognition." In *Advances in Neural Information Processing Systems*, pp. 890–898, 2012.

[KBP13] Michael Kistler, Serena Bonaretti, Marcel Pfahrer, Roman Niklaus, and Philippe Büchler. "The Virtual Skeleton Database: An Open Access Repository for Biomedical Research and Collaboration." *J Med Internet Res*, **15**(11):e245, Nov 2013.

[KCL19] Pasha Khosravi, YooJung Choi, Yitao Liang, Antonio Vergari, and Guy Van den Broeck. "On tractable computation of expected predictions." In *Advances in Neural Information Processing Systems*, pp. 11169–11180, 2019.

[KCZ14] Matt J Kusner, Wenlin Chen, Quan Zhou, Zhixiang Eddie Xu, Kilian Q Weinberger, and Yixin Chen. "Feature-Cost Sensitive Learning with Submodular Trees of Classifiers." In *AAAI*, pp. 1939–1945, 2014.

[KDM18] Mohammad Kachuee, Sajad Darabi, Babak Moatamed, and Majid Sarrafzadeh. "Dynamic Feature Acquisition Using Denoising Autoencoders." *IEEE transactions on neural networks and learning systems*, 2018.

[KGK19a] Mohammad Kachuee, Orpaz Goldstein, Kimmo Kärkkäinen, Sajad Darabi, and Majid Sarrafzadeh. "Opportunistic Learning: Budgeted Cost-Sensitive Learning from Data Streams." 2019.

[KGK19b] Mohammad Kachuee, Orpaz Goldstein, Kimmo Kärkkäinen, and Majid Sarrafzadeh. "Opportunistic Learning: Budgeted Cost-Sensitive Learning from Data Streams." In *International Conference on Learning Representations*, 2019.

[KH09] Alex Krizhevsky and Geoffrey Hinton. "Learning multiple layers of features from tiny images." Technical report, Citeseer, 2009.

[KHM17] Mohammad Kachuee, Anahita Hosseini, Babak Moatamed, Sajad Darabi, and Majid Sarrafzadeh. "Context-aware feature query to improve the prediction performance." In *Signal and Information Processing (GlobalSIP), 2017 IEEE Global Conference on*, pp. 838–842. IEEE, 2017.

[KKG19] Mohammad Kachuee, Kimmo Karkkainen, Orpaz Goldstein, Davina Zamanzadeh, and Majid Sarrafzadeh. "Nutrition and Health Data for Cost-Sensitive Learning." *arXiv preprint arXiv:1902.07102*, 2019.

[KKM17] Mohammad Kachuee, Mohammad Mahdi Kiani, Hoda Mohammadzade, and Mahdi Shabany. "Cuffless Blood Pressure Estimation Algorithms for Continuous Health-Care Monitoring." *IEEE Transactions on Biomedical Engineering*, **64**(4):859–869, 2017.

[KMH17] Mohammad Kachuee, Lisa D Moore, Tali Homsey, Hamidreza Ghasemi Damavandi, Babak Moatamed, Anahita Hosseini, Ruyi Huang, James Leiter, Daniel Lu, and Majid Sarrafzadeh. "An Active Learning Based Prediction of Epidural Stimulation Outcome in Spinal Cord Injury Patients Using Dynamic Sample Weighting." In *Healthcare Informatics (ICHI), 2017 IEEE International Conference on*, pp. 478–483. IEEE, 2017.

[KMY16] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. "Federated learning: Strategies for improving communication efficiency." *arXiv preprint arXiv:1610.05492*, 2016.

[KSD17] Iryna Korshunova, Wenzhe Shi, Joni Dambre, and Lucas Theis. "Fast face-swap using convolutional neural networks." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3677–3685, 2017.

[KSG04] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. "Estimating mutual information." *Physical review E*, **69**(6):066138, 2004.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks." In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[KVC20] Pasha Khosravi, Antonio Vergari, YooJung Choi, Yitao Liang, and Guy Van den Broeck. "Handling missing data in decision trees: A probabilistic approach." *arXiv preprint arXiv:2006.16341*, 2020.

[KW13a] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes." *CoRR*, **abs/1312.6114**, 2013.

[KW13b] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes." *arXiv preprint arXiv:1312.6114*, 2013.

[KWS18] Ana Klimovic, Yawen Wang, Patrick Stuedi, Animesh Trivedi, Jonas Pfefferle, and Christos Kozyrakis. "Pocket: Elastic ephemeral storage for serverless analytics." In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, pp. 427–444, 2018.

[KYR11] Balaji Krishnapuram, Shipeng Yu, and R Bharat Rao. *Cost-sensitive Machine Learning*. CRC Press, 2011.

[LBC17] Shuang Liu, Olivier Bousquet, and Kamalika Chaudhuri. "Approximation and convergence properties of generative adversarial learning." In *Advances in Neural Information Processing Systems*, pp. 5545–5553, 2017.

[LBO12] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. "Efficient backprop." In *Neural networks: Tricks of the trade*, pp. 9–48. Springer, 2012.

[LCB98] Yann LeCun, Corinna Cortes, and Christopher JC Burges. "The MNIST database of handwritten digits.", 1998.

[LCW18] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P Trevino, Jiliang Tang, and Huan Liu. "Feature selection: A data perspective." *ACM Computing Surveys (CSUR)*, **50**(6):94, 2018.

[LDW20] Linchao Li, Bowen Du, Yonggang Wang, Lingqiao Qin, and Huachun Tan. "Estimation of missing values in heterogeneous traffic data: Application of multimodal deep learning model." *Knowledge-Based Systems*, p. 105592, 2020.

[LEV09] Hugo Larochelle, Dumitru Erhan, and Pascal Vincent. "Deep Learning using Robust Interdependent Codes." In *AISTATS*, pp. 312–319, 2009.

[Lew97] David D Lewis. "Reuters-21578 text categorization test collection, distribution 1.0." 1997.

[LJM19] Steven Cheng-Xian Li, Bo Jiang, and Benjamin Marlin. "Learning from Incomplete Data with Generative Adversarial Networks." In *International Conference on Learning Representations*, 2019.

[LKS20] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. "Ensemble distillation for robust model fusion in federated learning." *arXiv preprint arXiv:2006.07242*, 2020.

[LLW15] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. "Deep Learning Face Attributes in the Wild." In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[LLY06] Peng Liu, Lei Lei, Junjie Yin, Wei Zhang, Wu Naijun, and Elia El-Darzi. "Healthcare data mining: Prediction inpatient length of stay." In *Intelligent Systems, 2006 3rd International IEEE Conference on*, pp. 832–837. IEEE, 2006.

[LPB16] B. Lakshminarayanan, A. Pritzel, and C. Blundell. "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles." *ArXiv e-prints*, December 2016.

[LR18] Steven R Livingstone and Frank A Russo. "The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English." *PloS one*, **13**(5):e0196391, 2018.

[LR19] Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*, volume 793. Wiley, 2019.

[LS95] Huan Liu and Rudy Setiono. "Chi2: Feature selection and discretization of numeric attributes." In *Tools with artificial intelligence, 1995. proceedings., seventh international conference on*, pp. 388–391. IEEE, 1995.

[LSA12] Mars Lan, Lauren Samy, Nabil Alshurafa, Myung-Kyung Suh, Hassan Ghasemzadeh, Aurelia Macabasco-O'Connell, and Majid Sarrafzadeh. "Wanda: An end-to-end remote health monitoring and analytics system for heart failure patients." In *Proceedings of the conference on Wireless Health*, p. 9. ACM, 2012.

[Luc16] Sam Lucero et al. "IoT platforms: enabling the Internet of Things." *White paper*, 2016.

[LWM20] Suwen Lin, Xian Wu, Gonzalo Martinez, and Nitesh V Chawla. "Filling Missing Values on Wearable-Sensory Time Series Data." In *Proceedings of the 2020 SIAM International Conference on Data Mining*, pp. 46–54. SIAM, 2020.

[LWS12] Dijun Luo, Fei Wang, Jimeng Sun, Marianthi Markatou, Jianying Hu, and Shahram Ebadollahi. "Sor: Scalable orthogonal regression for non-redundant feature selection and its healthcare applications." In *Proceedings of the 2012 SIAM International Conference on Data Mining*, pp. 576–587. SIAM, 2012.

[LXL17] Meng Liu, Chang Xu, Yong Luo, Chao Xu, Yonggang Wen, and Dacheng Tao. "Cost-Sensitive Feature Selection via F-Measure Optimization Reduction." In *AAAI*, pp. 2252–2258, 2017.

[LY17] Choong Ho Lee and Hyung-Jin Yoon. "Medical big data: promise and challenges." *Kidney research and clinical practice*, **36**(1):3, 2017.

[MAS18] Mehdi Mohammadi, Ala Al-Fuqaha, Sameh Sorour, and Mohsen Guizani. "Deep learning for IoT big data and streaming analytics: A survey." *IEEE Communications Surveys & Tutorials*, **20**(4):2923–2960, 2018.

[MB17] Pavel Mach and Zdenek Becvar. "Mobile edge computing: A survey on architecture and computation offloading." *IEEE Communications Surveys & Tutorials*, **19**(3):1628–1656, 2017.

[MF18] Pierre-Alexandre Mattei and Jes Frellsen. "missIWAE: Deep Generative Modelling and Imputation of Incomplete Data." *arXiv preprint arXiv:1812.02633*, 2018.

[MG99] Dunja Mladenic and Marko Grobelnik. "Feature selection for unbalanced class distribution and naive bayes." In *ICML*, volume 99, pp. 258–267, 1999.

[MGW06] Tammara Massey, Tia Gao, Matt Welsh, Jonathan H Sharp, and Majid Sarrafzadeh. "The design of a decentralized electronic triage system." In *AMIA annual symposium proceedings*, volume 2006, p. 544. American Medical Informatics Association, 2006.

[MHZ14] Fan Min, Qinghua Hu, and William Zhu. "Feature selection with test cost constraint." *International Journal of Approximate Reasoning*, **55**(1):167–179, 2014.

[MKK18] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. "Spectral normalization for generative adversarial networks." *arXiv preprint arXiv:1802.05957*, 2018.

[MKS13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. "Playing atari with deep reinforcement learning." *arXiv preprint arXiv:1312.5602*, 2013.

[MKS15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. "Human-level control through deep reinforcement learning." *Nature*, **518**(7540):529, 2015.

[MM01] George B Moody and Roger G Mark. "The impact of the MIT-BIH arrhythmia database." *IEEE Engineering in Medicine and Biology Magazine*, **20**(3):45–50, 2001.

[MMR16] H Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Aguera y Arcas. "Communication-Efficient Learning of Deep Networks from Decentralized Data.(2016).", 2016.

[MMR17] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. "Communication-efficient learning of deep networks from decentralized data." In *Artificial Intelligence and Statistics*, pp. 1273–1282. PMLR, 2017.

[MOC15] Toni Mastelic, Ariel Oleksiak, Holger Claussen, Ivona Brandic, Jean-Marc Pierson, and Athanasios V Vasilakos. "Cloud computing: Survey on energy efficiency." *Acm computing surveys (csur)*, **47**(2):33, 2015.

[MP18] Karthika Mohan and Judea Pearl. "Graphical models for processing missing data." *arXiv preprint arXiv:1801.03583*, 2018.

[MS19] Paul Micaelli and Amos J Storkey. "Zero-shot knowledge transfer via adversarial belief matching." In *Advances in Neural Information Processing Systems*, pp. 9551–9561, 2019.

[MSS19] Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh. "Agnostic Federated Learning.", 2019.

[Mur12] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[Mur18] Jared S Murray et al. "Multiple imputation: a review of practical and theoretical findings." *Statistical Science*, **33**(2):142–159, 2018.

[NDR13] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. "Learning with noisy labels." In *Advances in neural information processing systems*, pp. 1196–1204, 2013.

[NDR18] Sriraam Natarajan, Srijita Das, Nandini Ramanan, Gautam Kunapuli, and Predrag Radivojac. "On Whom Should I Perform this Lab Test Next? An Active Feature Elicitation Approach." In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pp. 3498–3505. International Joint Conferences on Artificial Intelligence Organization, 7 2018.

[Nea96] Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag, Berlin, Heidelberg, 1996.

[Nea04] Richard E Neapolitan et al. *Learning bayesian networks*, volume 38. Pearson Prentice Hall Upper Saddle River, NJ, 2004.

[NH10] Vinod Nair and Geoffrey E Hinton. "Rectified linear units improve restricted boltzmann machines." In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

[nha18] "National Health and Nutrition Examination Survey.", 2018.

[NJ09] Thomas Dyhre Nielsen and Finn Verner Jensen. *Bayesian networks and decision graphs*. Springer Science & Business Media, 2009.

[NMS19] Gaurav Kumar Nayak, Konda Reddy Mopuri, Vaisakh Shaj, R Venkatesh Babu, and Anirban Chakraborty. "Zero-shot knowledge distillation in deep networks." *arXiv preprint arXiv:1905.08114*, 2019.

[NOG18] Alfredo Nazabal, Pablo M Olmos, Zoubin Ghahramani, and Isabel Valera. "Handling incomplete heterogeneous data using VAEs." *arXiv preprint arXiv:1807.03653*, 2018.

[NS17] Feng Nan and Venkatesh Saligrama. "Adaptive Classification for Prediction Under a Budget." In *Advances in Neural Information Processing Systems*, pp. 4727–4737, 2017.

[OGK15] Seyyed Salar Latifi Oskouei, Hossein Golestani, Mohamad Kachuee, Matin Hashemi, Hoda Mohammadzade, and Soheil Ghiasi. "GPU-based Acceleration of Deep Convolutional Neural Networks on Mobile Platforms." *arXiv preprint arXiv:1511.07376*, 2015.

[ORE16] Michael K Ong, Patrick S Romano, Sarah Edgington, Harriet U Aronow, Andrew D Auerbach, Jeanne T Black, Teresa De Marco, Jose J Escarce, Lorraine S Evangelista, Barbara Hanna, et al. "Effectiveness of remote patient monitoring after discharge of hospitalized patients with heart failure: the Better Effectiveness After Transition–Heart Failure (BEAT-HF) randomized clinical trial." *JAMA internal medicine*, **176**(3):310–318, 2016.

[ORN15] Tadashi Okoshi, Julian Ramos, Hiroki Nozaki, Jin Nakazawa, Anind K Dey, and Hideyuki Tokuda. "Attelia: Reducing user's cognitive load due to interruptive notifications on smart phones." In *Pervasive Computing and Communications (PerCom), 2015 IEEE International Conference on*, pp. 96–104. IEEE, 2015.

[PGC17] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. "Automatic differentiation in PyTorch." In *NIPS-W*, 2017.

[PGH16] Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. "Variational autoencoder for deep learning of images, labels and captions." In *Advances in neural information processing systems*, pp. 2352–2360, 2016.

[PKU15] Erman Pattuk, Murat Kantarcioglu, Huseyin Ulusoy, and Bradley Malin. "Privacy-aware dynamic feature selection." In *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*, pp. 78–88. IEEE, 2015.

[PLD05] Hanchuan Peng, Fuhui Long, and Chris Ding. "Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy." *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (8):1226–1238, 2005.

[PPG17] Wei Ping, Kainan Peng, Andrew Gibiansky, Sercan O Arik, Ajay Kannan, Sharan Narang, Jonathan Raiman, and John Miller. "Deep voice 3: Scaling text-to-speech with convolutional sequence learning." *arXiv preprint arXiv:1710.07654*, 2017.

[PVZ15] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, et al. "Deep face recognition." In *bmvc*, volume 1, p. 6, 2015.

[Qui86] J. Ross Quinlan. "Induction of decision trees." *Machine learning*, **1**(1):81–106, 1986.

[Ran14] Rajiv Ranjan. "Streaming big data processing in datacenter clouds." *IEEE Cloud Computing*, **1**(1):78–83, 2014.

[RHL20] Nicola Rieke, Jonny Hancox, Wenqi Li, Fausto Milletari, Holger R Roth, Shadi Albarqouni, Spyridon Bakas, Mathieu N Galtier, Bennett A Landman, Klaus Maier-Hein, et al. "The future of digital health with federated learning." *NPJ digital medicine*, **3**(1):1–7, 2020.

[RKK20] Seunghyoung Ryu, Minsoo Kim, and Hongseok Kim. "Denoising Autoencoder-Based Missing Value Imputation for Smart Meters." *IEEE Access*, **8**:40656–40666, 2020.

[RLA14] Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. "Training deep neural networks on noisy labels with bootstrapping." *arXiv preprint arXiv:1412.6596*, 2014.

[RMK05] Michael T Rosenstein, Zvika Marx, Leslie Pack Kaelbling, and Thomas G Dietterich. "To transfer or not to transfer." In *NIPS 2005 workshop on transfer learning*, volume 898, pp. 1–4, 2005.

[Ros14] Brian C Ross. "Mutual information between discrete and continuous data sets." *PloS one*, **9**(2):e87357, 2014.

[ROS16] Jorge-L Reyes-Ortiz, Luca Oneto, Albert Sama, Xavier Parra, and Davide Anguita. "Transition-aware human activity recognition using smartphones." *Neurocomputing*, **171**:754–767, 2016.

[Rub76] Donald B Rubin. "Inference and missing data." *Biometrika*, **63**(3):581–592, 1976.

[Rub04] Donald B Rubin. *Multiple imputation for nonresponse in surveys*, volume 81. John Wiley & Sons, 2004.

[RZK19] Maithra Raghu, Chiyuan Zhang, Jon Kleinberg, and Samy Bengio. "Transfusion: Understanding transfer learning for medical imaging." In *Advances in neural information processing systems*, pp. 3347–3357, 2019.

[Sat17] Mahadev Satyanarayanan. "The emergence of edge computing." *Computer*, **50**(1):30–39, 2017.

[Sch81] Jeff Schlimmer. "Mushroom records drawn from The Audubon Society field guide to north American mushrooms." *GH Lincoff (Pres), New York*, 1981.

[SCS17] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet Talwalkar. "Federated Multi-Task Learning.", 2017.

[SCW11] Myung-kyung Suh, Chien-An Chen, Jonathan Woodbridge, Michael Kai Tu, Jung In Kim, Ani Nahapetian, Lorraine S Evangelista, and Majid Sarrafzadeh. "A remote patient monitoring system for congestive heart failure." *Journal of medical systems*, **35**(5):1165–1179, 2011.

[SCZ16] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. "Edge computing: Vision and challenges." *IEEE Internet of Things Journal*, **3**(5):637–646, 2016.

[SD16] Weisong Shi and Schahram Dustdar. "The promise of edge computing." *Computer*, **49**(5):78–81, 2016.

[Set12] Burr Settles. "Active learning." *Synthesis Lectures on Artificial Intelligence and Machine Learning*, **6**(1):1–114, 2012.

[SG02] Joseph L Schafer and John W Graham. "Missing data: our view of the state of the art." *Psychological methods*, **7**(2):147, 2002.

[Sha09] Devavrat Shah. *Gossip algorithms*. Now Publishers Inc, 2009.

[SHY17] Hajin Shim, Sung Ju Hwang, and Eunho Yang. "Why Pay More When You Can Pay Less: A Joint Learning Framework for Active Feature Acquisition and Classification." *arXiv preprint arXiv:1709.05964*, 2017.

[SKS20] Marek Śmieja, Maciej Kołomycki, Łukasz Struski, Mateusz Juda, and Mário A. T. Figueiredo. "Can auto-encoders help with filling missing data?" In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

[SLY15] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. "Learning structured output representation using deep conditional generative models." In *Advances in neural information processing systems*, pp. 3483–3491, 2015.

[SO98] Joseph L Schafer and Maren K Olsen. "Multiple imputation for multivariate missing-data problems: A data analyst's perspective." *Multivariate behavioral research*, **33**(4):545–571, 1998.

[SRA08] Andrea Saltelli, Marco Ratto, Terry Andres, Francesca Campolongo, Jessica Cariboni, Debora Gatelli, Michaela Saisana, and Stefano Tarantola. *Global sensitivity analysis: the primer*. John Wiley & Sons, 2008.

[SRG10] K Srinivas, B Kavihta Rani, and A Govrdhan. "Applications of data mining techniques in healthcare and prediction of heart attacks." *International Journal on Computer Science and Engineering (IJCSE)*, **2**(02):250–255, 2010.

[SS95] Peter K. Sharpe and RJ Solly. "Dealing with missing values in neural network-based diagnostic systems." *Neural Computing & Applications*, **3**(2):73–77, 1995.

[SWL02] Amy Soller, Janyce Wiebe, and Alan M Lesgold. "A machine learning approach to assessing knowledge sharing during collaborative learning activities." In *CSCL*, pp. 128–137, 2002.

[SYY17] Yaozhong Song, Stephen S Yau, Ruozhou Yu, Xiang Zhang, and Guoliang Xue. "An approach to QoS-based task distribution in edge computing networks for IoT applications." In *2017 IEEE international conference on edge computing (EDGE)*, pp. 32–39. IEEE, 2017.

[SZC20] Rachael Hwee Ling Sim, Yehong Zhang, Mun Choon Chan, and Bryan Kian Hsiang Low. "Collaborative machine learning with incentive-aware model rewards." In *International Conference on Machine Learning*, pp. 8927–8936. PMLR, 2020.

[SZS13] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. "Intriguing properties of neural networks." *arXiv preprint arXiv:1312.6199*, 2013.

[Tib96] Robert Tibshirani. "Regression shrinkage and selection via the lasso." *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[TIL04] Emmanuel Munguia Tapia, Stephen S Intille, and Kent Larson. "Activity recognition in the home using simple and ubiquitous sensors." In *International Conference on Pervasive Computing*, pp. 158–175. Springer, 2004.

[TKD16] Dustin Tran, Alp Kucukelbir, Adji B. Dieng, Maja Rudolph, Dawen Liang, and David M. Blei. "Edward: A library for probabilistic modeling, inference, and criticism." *arXiv preprint arXiv:1610.09787*, 2016.

[TLS89] Naftali Tishby, Esther Levin, and Sara A. Solla. "Consistent inference of probabilities in layered networks: Predictions and generalization." In Anon, editor, *IJCNN Int Jt Conf Neural Network*, pp. 403–409. Publ by IEEE, 12 1989.

[TMC02] Benjamin B Thompson, Robert J Marks, Jai J Choi, Mohamed A El-Sharkawi, Ming-Yuh Huang, and Carl Bunje. "Implicit learning in autoencoder novelty assessment." In *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on*, volume 3, pp. 2878–2883. IEEE, 2002.

[TS10] Lisa Torrey and Jude Shavlik. "Transfer learning." In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pp. 242–264. IGI global, 2010.

[TS13] Kirill Trapeznikov and Venkatesh Saligrama. "Supervised sequential classification under budget constraints." In *Artificial Intelligence and Statistics*, pp. 581–589, 2013.

[TSM17] Tarik Taleb, Konstantinos Samdanis, Badr Mada, Hannu Flinck, Sunny Dutta, and Dario Sabella. "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration." *IEEE Communications Surveys & Tutorials*, **19**(3):1657–1681, 2017.

[TZA17] Cao Truong Tran, Mengjie Zhang, Peter Andreae, and Bing Xue. "Multiple imputation and genetic programming for classification with incomplete data." In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 521–528, 2017.

[VBT17] Paul Vanhaesebrouck, Aurélien Bellet, and Marc Tommasi. "Decentralized collaborative learning of personalized models over networks." 2017.

[VJ04] Paul Viola and Michael J Jones. "Robust real-time face detection." *International journal of computer vision*, **57**(2):137–154, 2004.

[VLB08] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. "Extracting and composing robust features with denoising autoencoders." In *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103. ACM, 2008.

[VS10] Alireza Vahdatpour and Majid Sarrafzadeh. "Unsupervised discovery of abnormal activity occurrences in multi-dimensional time series, with applications in

wearable systems." In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pp. 641–652. Society for Industrial and Applied Mathematics, 2010.

[Wan18] Charlie Wang. "HTTP vs. MQTT: A tale of two IoT protocols.", 2018.

[Wil97] Christopher KI Williams. "Computing with infinite networks." In *Advances in neural information processing systems*, pp. 295–301, 1997.

[WKB18] Yichuan Wang, LeeAnn Kung, and Terry Anthony Byrd. "Big data analytics: Understanding its capabilities and potential benefits for healthcare organizations." *Technological Forecasting and Social Change*, **126**:3–13, 2018.

[WLZ18] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. "High-resolution image synthesis and semantic manipulation with conditional gans." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8798–8807, 2018.

[WRK17] Stephen F Weng, Jenna Reps, Joe Kai, Jonathan M Garibaldi, and Nadeem Qureshi. "Can machine-learning improve cardiovascular risk prediction using routine clinical data?" *PloS one*, **12**(4):e0174944, 2017.

[WTS19] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K Leung, Christian Makaya, Ting He, and Kevin Chan. "Adaptive federated learning in resource constrained edge computing systems." *IEEE Journal on Selected Areas in Communications*, **37**(6):1205–1221, 2019.

[WYS20] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. "Federated learning with matched averaging." *arXiv preprint arXiv:2002.06440*, 2020.

[WZH14] Jialei Wang, Peilin Zhao, Steven CH Hoi, and Rong Jin. "Online feature selection and its applications." *IEEE Transactions on Knowledge and Data Engineering*, **26**(3):698–710, 2014.

[XGS21] Jie Xu, Benjamin S Glicksberg, Chang Su, Peter Walker, Jiang Bian, and Fei Wang. "Federated learning for healthcare informatics." *Journal of Healthcare Informatics Research*, **5**(1):1–19, 2021.

[XKW14] Zhixiang Eddie Xu, Matt J Kusner, Kilian Q Weinberger, Minmin Chen, and Olivier Chapelle. "Classifier cascades and trees for minimizing feature evaluation cost." *Journal of Machine Learning Research*, **15**(1):2113–2144, 2014.

[XRV17] Han Xiao, Kashif Rasul, and Roland Vollgraf. "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms.", 2017.

[XWC12] Zhixiang Xu, Kilian Weinberger, and Olivier Chapelle. "The greedy miser: Learning under test-time budgets." *arXiv preprint arXiv:1206.6451*, 2012.

[YHZ17] Shanhe Yi, Zijiang Hao, Qingyang Zhang, Quan Zhang, Weisong Shi, and Qun Li. "Lavea: Latency-aware video analytics on edge computing platform." In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, p. 15. ACM, 2017.

[YIJ17] Ashkan Yousefpour, Genya Ishigaki, and Jason P Jue. "Fog computing: Towards minimizing delay in the internet of things." In *2017 IEEE international conference on edge computing (EDGE)*, pp. 17–24. IEEE, 2017.

[YJV18] Jinsung Yoon, James Jordon, and Mihaela Van Der Schaar. "Gain: Missing data imputation using generative adversarial nets." *arXiv preprint arXiv:1806.02920*, 2018.

[YKR09] Shipeng Yu, Balaji Krishnapuram, Romer Rosales, and R Bharat Rao. "Active sensing." In *Artificial Intelligence and Statistics*, pp. 639–646, 2009.

[YLK20] Joonyoung Yi, Juhyuk Lee, Kwang Joon Kim, Sung Ju Hwang, and Eunho Yang. "Why Not to Use Zero Imputation? Correcting Sparsity Bias in Training Neural Networks." In *International Conference on Learning Representations*, 2020.

[YLS20] Tianlong Yu, Tian Li, Yuqiong Sun, Susanta Nanda, Virginia Smith, Vyas Sekar, and Srinivasan Seshan. "Learning Context-Aware Policies from Multiple Smart Homes via Federated Multi-Task Learning." In *2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pp. 104–115. IEEE, 2020.

[YZH18] Wei Ying, Yu Zhang, Junzhou Huang, and Qiang Yang. "Transfer learning via learning to transfer." In *International Conference on Machine Learning*, pp. 5085–5094, 2018.

[ZBP14] Indre Zliobaite, Albert Bifet, Bernhard Pfahringer, and Geoffrey Holmes. "Active learning with drifting streaming data." *IEEE transactions on neural networks and learning systems*, **25**(1):27–39, 2014.

[ZGM18] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. "Self-attention generative adversarial networks." *arXiv preprint arXiv:1805.08318*, 2018.

[ZLL18] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. "Federated learning with non-iid data." *arXiv preprint arXiv:1806.00582*, 2018.

[ZQL05] Shichao Zhang, Zhenxing Qin, Charles X Ling, and Shengli Sheng. "" Missing is useful": missing values in cost-sensitive decision trees." *IEEE transactions on knowledge and data engineering*, **17**(12):1689–1693, 2005.

[ZYZ18] Tianyun Zhang, Shaokai Ye, Kaiqi Zhang, Jian Tang, Wujie Wen, Makan Fardad, and Yanzhi Wang. "A systematic dnn weight pruning framework using alternating direction method of multipliers." In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 184–199, 2018.

[ZZZ18] Wei Zheng, Xiaofeng Zhu, Yonghua Zhu, and Shichao Zhang. "Robust Feature Selection on Incomplete Data." In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pp. 3191–3197. International Joint Conferences on Artificial Intelligence Organization, 7 2018.