Evaluating multi-task learning for Internet measurement classification

Nick Henderson
University of Oregon
nhenders@cs.uoregon.edu

Ramakrishnan Durairajan
University of Oregon
ram@cs.uoregon.edu

ABSTRACT

In this paper, we demonstrate how multi-task machine learning models compare to several single-task models trained to perform the same function in an Internet measurements use case. We employ recent innovations in weak supervision to provide noisy labels to a measurement dataset, and then train multiple models to perform network property classification. This performance is compared to that of a single multi-task model which uses the same noisy labels but shares information between tasks in an effort to improve accuracy. The accuracies and training times of both approaches are compared, and we show that multi-task learning outperforms the single-task approach by 5-10% per property while requiring nearly a third of the time to train.

1 INTRODUCTION

Being able to turn vast swaths of Internet measurements into meaningful information is a key challenge facing Internet measurements researchers. Because the amount of data is so significant, Internet measurements has become a prime domain for the application of machine learning due to its ability to scrutinize data and infer knowledge. Machine learning models have been used to classify traffic [1], optimize network infrastructure [2], and remove noise from measurements [6]. These applications are uniquely effective at their tasks due to the colossal amount of information considered by a model when it is deriving insight as well as the high generalizability of the end model, meaning the same model is useful even when applied to unseen datasets.

However, the Internet's quantity of data does not guarantee its quality, and a common phrase in the field of machine learning is that *a model is only as good as the data it is fed*. As such, much of the work in machine learning is put into preparing datasets to train a model. This involves cleaning the data to remove distracting, useless, or otherwise unrepresentative data points, engineering the dataset to select meaningful features, and — in supervised or weakly-supervised models — labeling a subset of the data. These tasks require significant amounts of time, resources, and knowledge in the relevant domain, yet upon their completion the model still has to be trained, evaluated, tuned, and retrained several times until it can be applied and insight can be derived. In an effort to get results sooner, it is a common goal to shorten the length of time between data collection and a high-quality trained model, despite the unpredictable amount of time required by the training process. This effort is especially important when there are multiple models in question, further extending the length of training time.

We identify multi-task learning as a potential method of reducing the time required to produce a trained model that performs several functions. Multi-task learning is a unique approach to machine learning in that it enables information sharing between several tasks so that a single model can be trained effectively for several purposes [3]. While single-task models can perform well given their specific function, they often discard information that is not relevant despite potential relevance to other models being trained on the same dataset. This drawback of single-task models suggests that multi-task learning can result in a single model that will outperform several models trained to do each task separately. More importantly, the single-task requires the training process be repeated for each task. Since this results in the multiplication of training overhead by the number of tasks [5], multi-task has the potential to greatly reduce training time while maintaining or even improving overall performance.

In this paper, we seek to compare the training time and performance of several single-task models against a multi-task model. We begin by generating a `ping` dataset featuring synthetic features meant to mimic those appearing in actual network traffic but with increased frequency and magnitude to enable straightforward identification. The features selected were noises, outages, and swells (periodic blocks of increases in latencies), given these features' prevalence in machine learning use cases, their relationships with each other, and for their relative simplicity. With the generated data, we write a suite of labeling functions and employ modern weak supervision methods [8] to identify a single property and provide weak labels for that property to our dataset. A generative model is then trained on these weak labels to provide more accurate predictions for the property in question. This generative model is the single-task model that will be compared to the multi-task model, which we build next. This labeling and training pipeline is repeated for all three properties.

For the multi-task model, the labeling functions written for the single-task models are again used to provide weak labels to our dataset, but now for all three properties. Instead of training a generative model on the labels, a multitask classifier is built and trained using these labels to identify the desired properties. Note that both the single-task models and the multi-task model equally employ weak supervision; they use the same labeling functions and neither is provided strong labels.

We measure the time taken to train each model as well as the model's predictive performance and provide comparison between the two approaches quantitatively and in terms of their differences in implementation. These evaluations showed that though the implementation process for a multi-task model was more intensive, it yielded considerable improvements in model performance while requiring substantially less time to train the single-task models.

2 BACKGROUND AND RELATED WORK

While machine learning has generally seen widespread use in Internet measurement research, multi-task learning has seen little to no application in the field [12]. A couple of reasons may be behind this deficiency. The integration of multi-task has mostly been limited to applications where single-task learning is not sufficient and multi-task is necessary in order to derive quality insight [10]. This is not the case among the most common uses for machine learning in Internet measurements, delaying the introduction of the multi-task method to the field for a lack of necessity. However, the most likely reason behind the lack of adoption is that much of the established research involving multi-task involves homegrown implementations which are prohibitively difficult to implement for groups that do not have explicit machine learning education or experience. That is, despite over twenty years since multi-task's definition there have been very few tools to enable its implementation.

However, the introduction and release of the Snorkel library [4] has greatly increased the availability of multi-task learning for all domains and includes various weak supervision tools that can also be leveraged alongside. In this paper, we employ various functions within Snorkel for both the single-task and multi-task approaches, including labeling functions [7], weakly-supervised generative models, and multi-task classification models [9]. The Snorkel library as a project didn't include multi-task learning until recently, when its side project MeTaL was merged with the Snorkel base, which previously only included labeling tools. As a result, there is a lack of up-to-date documentation for the multi-task elements which makes their use somewhat difficult. Remedying this problem would make multi-task even more accessible and could further bring multi-task learning to the forefront of machine learning-based research.

3 DESCRIPTION AND METHODOLOGY

The approach taken in this paper involves separately training three single-task models to identify one property of the dataset each, and comparing their performance to that of a multi-task model trained to recognize the same functions of the same dataset. Beyond all being easy to identify, the properties in question (noise, outages, and swells), were each selected for different reasons. Noise is a common problem in Internet measurements and the identification of it can be useful, yet difficult to its unpredictable nature. Outages are meant to be the easiest to identify, since all measurements where the latency is zero is considered an outage. This simplicity allows us to easily understand the performance of the model at-a-glance, since we should expect perfect classification of this property. Swells act as a difficult but realistic use case — identifying swells can aid in tasks like server selection or anomaly detection — but can also be combined with noise so that models have to label certain elements with two positive labels in order to be correct. Both the single-task and multi-task models are trained on recognizing the same properties, as the unifying goal in the experiment design is to compare these two approaches as equally as possible. This is accomplished in large part by using the same labeling functions and providing the same level of supervision.

3.1 Dataset Creation

In order to be able to reliably test the performance and efficiency of machine learning models, it is important to have a dataset that is rich with the properties we are seeking to identify. Since direct, natural Internet measurements may not feature these properties with as much frequency or magnitude as we desire, we instead add these properties synthetically. We begin with a base set of 15,000 natural `ping` measurements, collected by pinging Google's public DNS IP address from a home network. This data is cleaned to be as simple as possible, so that any interesting properties are limited to the ones we add deliberately (exempting unreturned pings, which were labelled as outages appropriately). A unique advantage of generating the dataset rather than collecting natural measurements is that we can simultaneously produce a perfect set of ground truth labels for model evaluation, and we want to preserve this advantage.

The addition of properties is done with randomness in mind, in order to retain as much unpredictability as possible so the models are challenged as they would be with direct measurements. Beginning with noise, we choose a probability that any given point is noisy. A uniformly random number is generated for each measurement, and if the number is less than the given probability, the measurement is manipulated to be noisy. Specifically, this meant adding a randomly large increase in latency to the given data point. The increase is

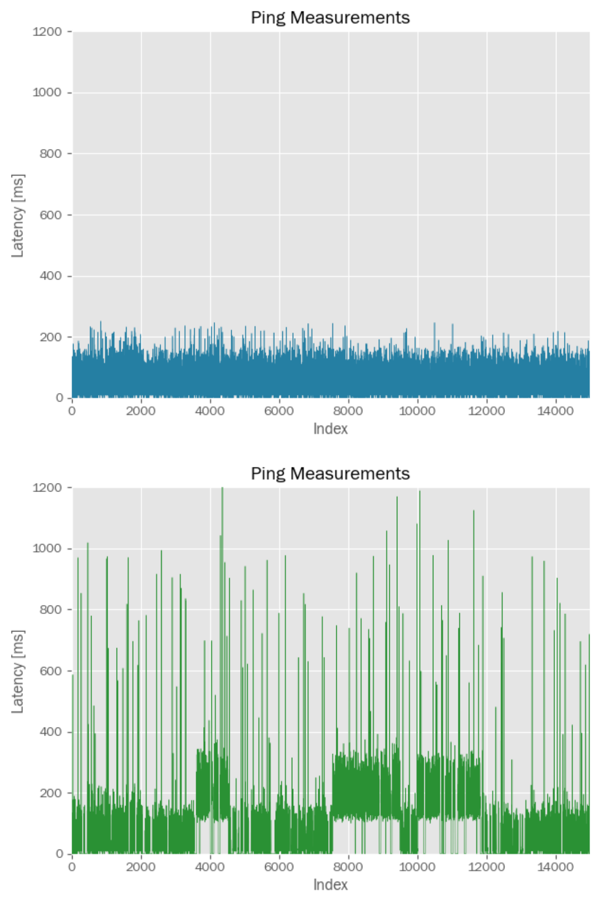


Figure 1: Collected base measurements (top) and the same measurements after noise, outages, and swells have been added (bottom).

sufficiently large such that all selected measurements are guaranteed to resemble noise.

Outages and swells are generated in a fashion similar to one another. Since single-measurement outages were retained in the base dataset, we only needed to add longer outages where there are many zero values in a row. This is similar to swells in that both are blocks of multiple altered measurements, rather than the single points altered in noise generation. Therefore, we choose a probability that a given point is the *beginning* of an outage or a swell. We again generate a uniformly random number, and test it against that property's probability. Then, we generate a random length n and alter the next n measurements to fit the desired property. For outages, this means simply setting the values to zero. For swells, we add a random increase in latency much like we did when generating noise. We also generate a random-length gap between swells to prevent multiple swells being generated on top of each other.

Throughout the generation process we take note of which points we alter, yielding a ground truth label set for each property. Since a measurement cannot be both noisy and an outage, we filter through these labels and correct any of these conflicts. However, a point can be both noisy and part of a swell, so we maintain both properties in this case.

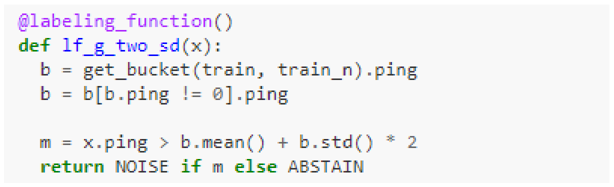
The probabilities selected for the final dataset were .01 for noise and .005 for outages. The probability for swells varied such that we were guaranteed at least three swells. These probabilities are different than they would be for normal measurements, but by increasing the frequency of these properties we ensure that any split of the dataset will contain ample features to train on and identify. For every use of the dataset in this study, we divide it into three subsets, 60% training, 20% validation, and 20% testing.

3.2 Labeling

With data generation complete, the next step is to generate noisy labels to serve as weak supervision when training the single- and multi-task models. Due to the configuration of Snorkel's labeling operation, labels are best generated one property at a time. For each property, a suite of labeling functions are written, each of which produces a vote when applied to a single measurement: `NORMAL`, `<PROPERTY>`, or `ABSTAIN`. A unique advantage of using this method of generating weak labels is that Snorkel's labeling model is designed for imperfect, overlapping labeling functions. Functions do not need to be relevant to each data point and they don't need to be correct every time — functions can vote `ABSTAIN` in some cases which effectively removes that function from consideration when labeling that point. We employ `ABSTAIN` more frequently in the functions that we are less confident in the accuracy of.

However, the way labeling is implemented in Snorkel makes it difficult to consider the context of a measurement when determining how to label it. For properties like noise and swells, this context is critical in making a decision. To enable this, we use an original bucketing function that yields the context surrounding a given measurement. The amount of context provided was tweaked to maximize performance, and differs per property. If the bucket function is called on a measurement at index i with bucket size S, it would return the measurements between indices $i - (S/2)$ and $i + (S/2)$.

With the ability to get the context of a measurement, writing labeling functions is straightforward. For example, a noise labeling function would return `NOISE` if the measurement is greater than the mean plus two times the standard deviation of its bucket. One swell function looks at the average "height", or the distance between its values and zero, of a measurement's bucket and votes `SWELL` if it is much higher



```
@labeling_function()
def lf_g_two_sd(x):
  b = get_bucket(train, train_n).ping
  b = b[b.ping != 0].ping

  m = x.ping > b.mean() + b.std() * 2
  return NOISE if m else ABSTAIN
```

Figure 2: Example of a labeling function for the noise property.

than the height of prior buckets. An outage labeling function simply votes OUTAGE if the measurement is zero.

These labels are applied to each measurement in the dataset using Snorkel's labeling function dataframe applier. Because labeling functions found to be more accurate or have a wider coverage are weighted more heavily than those that may not be as precise, it is not sufficient to take simply use the sum of the votes to make a label decision. Instead, we train a model to consider these votes and make a final decision.

3.3 Single-Task Models

Using the weak labels provided by the label generation process, we can separately train a classification model for all three tasks. Since each task is trained separately from one another, this model training process can simply be appended to the labeling process. Once labels are generated, we train a generative classification model using them. The generative model does not have access to gold truth labels, but instead learns from how different labeling functions agree or disagree with each other. The model considers labeling function accuracies, reweights them, and combines them appropriately to improve predictive accuracy.

At this point, Snorkel's article on labeling [8] suggests the use of a discriminative model to improve generalizability. However, we chose not to employ such a model and chose to study only the generative model. This decision was made largely due to the lack of a notable performance increase with the discriminative model. While we could have trained a discriminative model regardless of need, we deemed it important to keep the model as slim as possible in order to provide a fair comparison to multi-task learning.

3.4 Multi-Task Models

For the multi-task model, we use a PyTorch classifier through Snorkel's multi-task framework. While several other options were considered, the PyTorch base is especially flexible in adding new tasks and metrics, a common need when working with multi-task models. We then define tasks using the output from the labeling functions in section *3.2*. This enables the most fair comparison possible between the single-

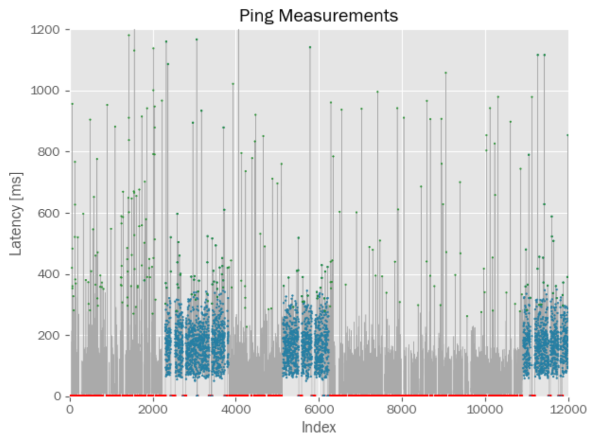


Figure 3: Graph annotated with the labels provided by the single-task models. Green dots indicate noise, blue dots indicate swells, and red dots indicate outages.

and multi-task models by providing the same amount of supervision at an identical level of accuracy.

The most significant difference between this classifier and a single-task classifier is that there exists a shared layer through which each task can share information. When we define the tasks, we also provide instructions for how to utilize this shared layer. Though the shared layer is shared by all three tasks in this study, a more complex application of multi-task may choose to share information between certain layers and not others, or to have multiple shared layers to support different sharing configurations. Once the tasks and these layers are defined and instantiated, we train the model by passing in the list of tasks to the multi-task classifier.

3.5 Quantitative Evaluation

To evaluate the models, we consider two quantitative metrics: training time in seconds and accuracy. Each model was trained 10 times and the time was averaged for a final training time, though each training effort took about the same amount of time. For the single-task models, noise classification took 104 seconds (114it/s), outage classification took 21 seconds (554it/s), and swell classification took 48 seconds (250it/s). The multi-task model required 64 seconds to train on the same dataset, significantly less than half the single-task models' sum of 173 seconds.

The accuracy of the single-task model trained to recognize noise is .88, and the accuracy of the model trained to recognize swells is .9. The accuracy of the outage classification model was, predictably, 1.0. While these scores could be considered excellent for this application, the multi-task model outperformed the single-task models significantly. For

	Noise	Outage	Swell	Time
Single-Task Acc.	0.88	1.0	0.9	173s
Multi-Task Acc.	0.98	1.0	0.95	64s

Table 1: Table of single-task accuracies and multi-task accuracies alongside the total training time each approach required.

noise, swells, and outages, accuracies from multi-task were .98, .95, and 1.0 respectively.

These results show that for applications similar to the one in this study, multi-task learning can outperform single-task models trained for the same task while requiring significantly less time to train.

3.6 Qualitative Evaluation

While we focus on quantitative metrics like training time and accuracy as a primary means of evaluating the two different approaches, it is also necessary to have an understanding of the differences in the implementation process to make an informed decision. While the multi-task model outperformed the single-task model in both metrics, multi-task was considerably more intensive to implement. Because simple single-task classification models are found in so many established machine learning libraries, their implementation is "plug-and-play" for a majority of applications. Over two decades since multi-task learning was established as a concrete concept, similar tools for multi-task are only just now being made available. Researchers can choose to avoid needing to build a multi-task model from scratch and instead use these frameworks, but will instead be faced by a lack of documentation or difficulty tailoring the framework's model for their application.

Teams without a member with experience in machine learning may find it difficult to produce a high-quality multi-task model without devoting substantial time and resources into it. Depending on the application and the timeline, it may be useful for these teams to first implement a simpler single-task equivalent and evaluate these models before investing resources in multi-task. While all machine learning models are somewhat opaque, multi-task is especially so and in the case of poor performance it can be difficult to differentiate problems with model configuration from problems with the data. Beginning with a single-task implementation can be straightforward given the tools used in this study and can help researchers gain experience tuning a model to their data before they attempt to implement multi-task learning. As they were used in this study, single-task models can be useful as an intermediate step before implementing the less-forgiving multi-task for an increase in performance. Even for applications where there is little performance increase, it may still be beneficial to implement multi-task if the training time needs to be absolutely minimized.

4 SUMMARY

In this paper, we evaluate the performance of several single-task models against one multi-task model trained to classify the same network properties in the same `ping` timeseries. Multi-task learning uses sharing between tasks to improve overall performance, and features decreased training overhead compared to many single-task models. As new libraries improve the accessibility of multi-task learning, more researchers will consider its application in their studies. For Internet measurement research, where datasets and tasks can very well resemble the ones used in this study, our results help illuminate the decision between single- and multi-task.

To evaluate these methods, we begin by generating a dataset rich with network noise, outages, and swells to simulate an especially active network for our models to train and test on. We then develop a labeling pipeline leveraging Snorkel's labeling function framework to generate noisy labels to be used for weak supervision. Next, we separately train three single-task models to each identify one of our selected properties using a generative classification model. For multi-task, we define tasks using the weak labels produced by the labeling functions as well as additional layer that enables information sharing between all tasks. We evaluate the two approaches by examining their accuracy and the time required to train them, and determine that the multi-task model outperforms the single-task model in accuracy by 5-10% on each task and requires almost a third of the time to train.

We briefly discuss the qualitative differences in implementation for the two approaches, which may prohibit smaller or less technical teams from implementing a high-quality multi-task learning in a reasonable time frame. However, for applications with a high expectation for performance and training time, we posit that a multi-task model is certainly superior to several single-task models.

5 CONCLUSION AND FUTURE WORK

Because of the colossal amounts of data collected Internet measurements, the Internet measurements community often employs emerging technologies to gain valuable insight. We believe that multi-task learning could have significant ramifications for any machine learning application in Internet measurements, like providing aid in server selection for CDNs or detecting network anomalies. Future work could explore these functions in a real-world application of multi-task learning. Future studies could also employ the use of libraries like *tsfresh* [11] to have multiple features to train on.

Multi-task models are especially capable with datasets that include many different features, and the Snorkel labeling components also perform better in cases where there are many factors to write functions for [8].

With the increasing accessibility of multi-task learning, researchers should consider utilizing this approach in future studies. By implementing both single-task and multi-task models in an identical research function, we show that the use of multi-task learning can lead to a meaningful improvement in performance while demanding less time to train.

REFERENCES

[1] T. Bakhshi and B. Ghita. 2016. On Internet Traffic Classification: A Two-Phased Machine Learning Approach. *Journal of Computer Networks and Communications* (2016).

[2] R. Boutaba, M.A. Salahuddin, and N. Limam. 2018. A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. *J Internet Serv Appl* 9, 16 (2018).

[3] R. Caruana. 1997. Multitask Learning. *Machine Learning* 28 (1997), 41–75.

[4] Snorkel: The System for Programmatically Building and Managing Training Data. 2019. https://tsfresh.readthedocs.io/en/latest/. (2019).

[5] T. Liu, S. Alibhai, J. Wang, Q. Liu, X. He, and C. Wu. 2019. Exploring Transfer Learning to Reduce Training Overhead of HPC Data in Machine Learning. *IEEE International Conference on Networking, Architecture and Storage (NAS)* 2019 (2019), 1–7.

[6] A. Muthukumar and R. Durairajan. 2019. Denoising Internet Delay Measurements using Weak Supervision. *ICMLA* 2019 (2019).

[7] A. Ratner, S. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré. 2017. Snorkel: Rapid Training Data Creation with Weak Supervision. *Proceedings of the VLDB Endowment* 11, 3 (2017), 269–282.

[8] A. Ratner, S. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré. 2019. Snorkel: Rapid Training Data Creation with Weak Supervision. *IEEE International Conference on Networking, Architecture and Storage (NAS)* 2019 (2019), 1–7.

[9] A. Ratner, B. Hancock, J. Dunnmon, F. Sala, S. Pandey, and C. Ré. 2019. Training Complex Models with Multi-Task Weak Supervision. *Proceedings of the AAAI Conference on Artificial Intelligence* 33 (2019), 4763–4771.

[10] Sebastian Ruder. 2017. An Overview of Multi-Task Learning in Deep Neural Networks. *ArXiv* abs/1706.05098 (2017).

[11] tsfresh. 2019. https://www.snorkel.org/. (2019).

[12] Y. Zhang and Q Yang. 2018. A Survey on Multi-Task Learning. *ArXiv* (2018).