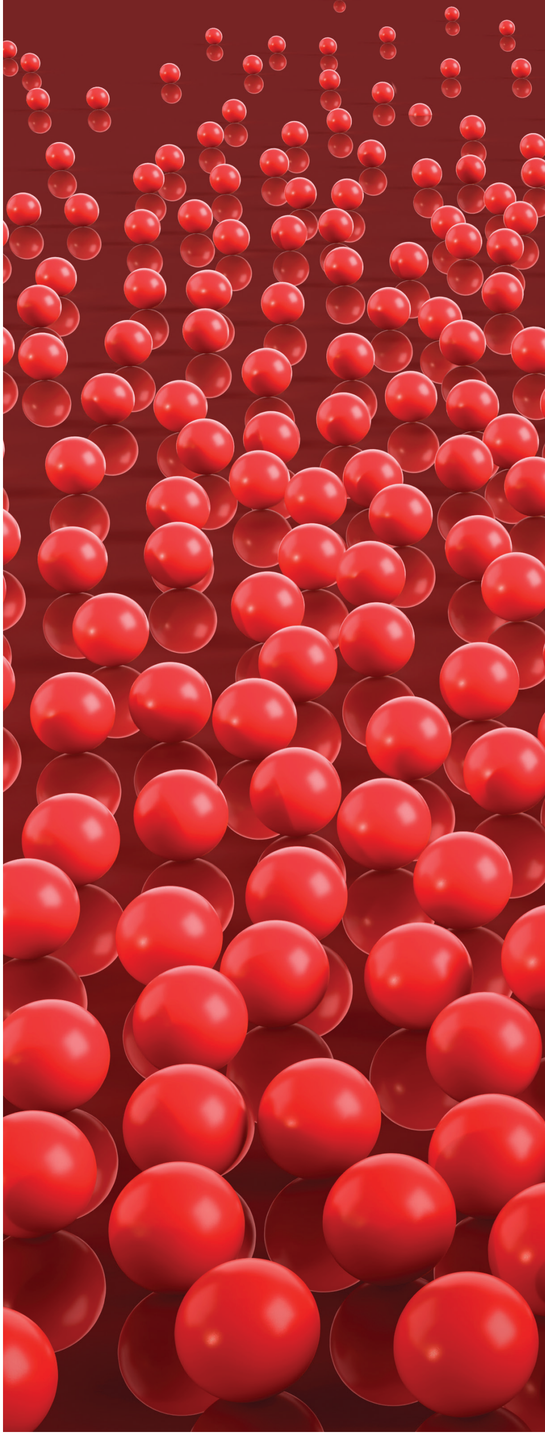
DOI:10.1145/3551635

A taxonomy of the methods used to obtain quality datasets enhances existing resources.

BY CHEN SHANI, JONATHAN ZARECKI, AND DAFNA SHAHAF

The Lean Data Scientist:

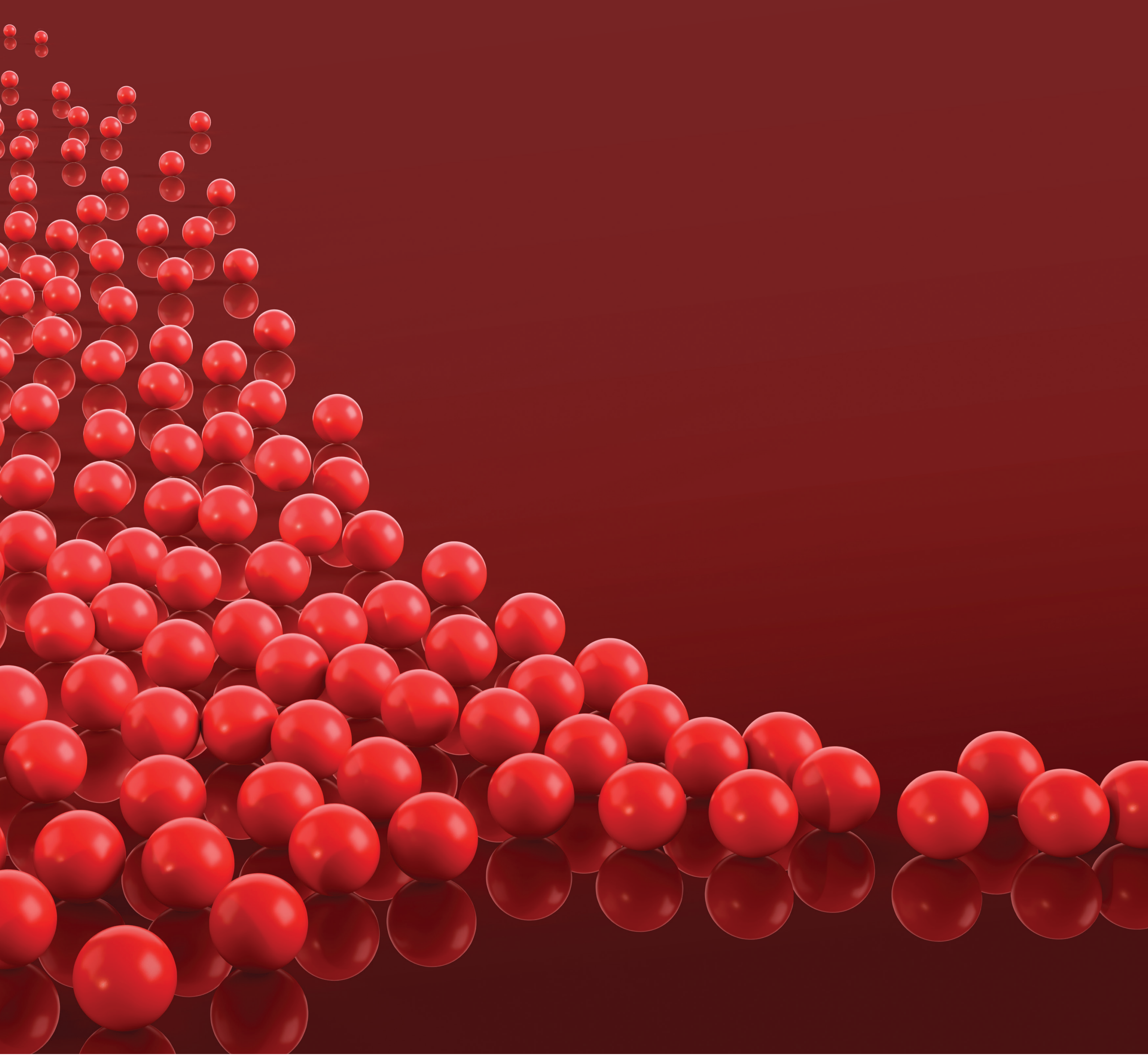
Recent Advances toward Overcoming the Data Bottleneck



OBTAINING DATA HAS become the key bottleneck in many machine-learning (ML) applications. The rise of deep learning has further exacerbated this issue. Although high-quality ML models are finally making the transition from expensive-to-develop, highly specialized code to something more like a commodity, these models involve millions (or even billions) of parameters and require massive amounts of data to train. Thus, the dominant paradigm in ML today is to create a new (large) dataset whenever facing a novel task. In fact, there are now entire conferences dedicated to the creation of new data resources (for example, the International Conference on Language Resources and Evaluation or resource papers at CIKM).

While this approach resulted in significant advances, it suffers from a major caveat, as collecting large, high-quality datasets is often very demanding in terms of time and human resources. For several tasks, such as rare disease detection, large datasets are nearly infeasible to construct.

While there has been much effort suggesting workarounds to this *data-bottleneck* problem, they are scattered across many different subfields, often unaware of one another. There exist many method-specific and domain-specific surveys, but broader, big-picture surveys are difficult to find. The closest in spirit to our work is Roh et al.,[33] which focuses more



on the data management point of view and the early stages of the pipeline.

In this article, we aim to bring order to this area. Our main contribution is a simple yet comprehensive taxonomy of ways to tackle the data bottleneck. We survey major research directions and organize them into a taxonomy in a way designed to be useful for practitioners choosing between different approaches. The emphasis here is not on covering methods in depth; rather, we discuss the main ideas behind various methods, the assumptions they make and their underlying concepts. For each topic, we mention several important or interesting works, and refer the interested reader to surveys where possible.

We wish to first *raise awareness* of the methods that already exist, to encourage more efficient use of data. In addition (and perhaps more importantly), we hope the organization of the taxonomy would also reveal gaps in current techniques and suggest novel directions of research that could inspire the creation of new, less data-hungry learning methods.

Taxonomy

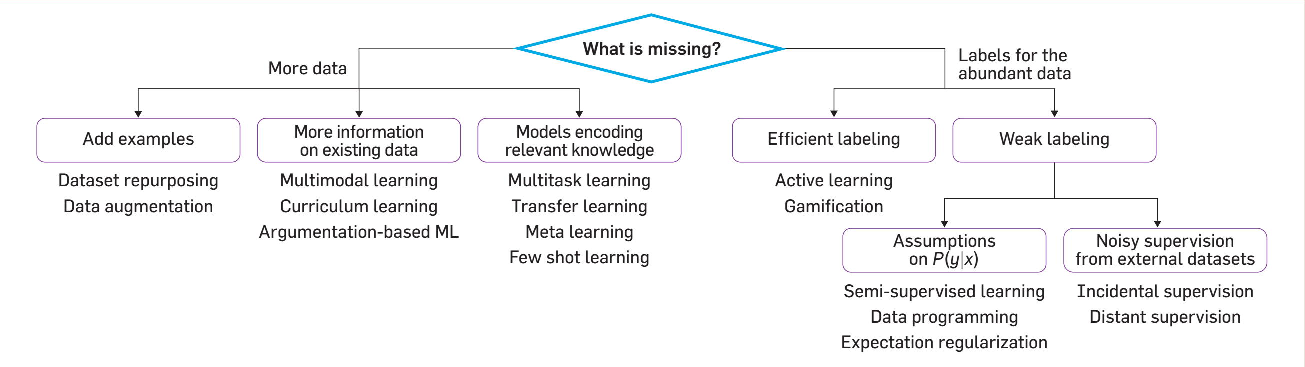
A note on scope. The data-bottleneck problem is widespread across the field of machine learning. It is especially crucial in supervised learning but applies

IMAGE BY ALICIA KUBISTA/ANDRIJ BORYS ASSOCIATES

» key insights

- **Recent machine learning algorithms are increasingly data-hungry. A widespread approach is to construct large, task-specific datasets, which is inefficient and sometimes infeasible.**
- **Many ways to tackle this data bottleneck problem have been proposed, but they are scattered across different subfields.**
- **We present a practitioner-centric taxomony of these methods. We distill each method's main assumptions and explain when it is useful, in hopes of encouraging more efficient use of resources as well as uncovering novel research directions.**

Figure 1. Flowchart of the taxonomy for ways to tackle the data bottleneck.



to unsupervised paradigms as well. In this work we focus on the supervised, unsupervised, and semi-supervised settings. Reinforcement learning is generally beyond the scope of this article, although some of the methods we present are applicable to it.

We start with a high-level view of our taxonomy, depicted in Figure 1. We first make the distinction between cases where *data* (X) is hard to collect, and cases where *labels* (Y) pose the difficulty. For example, collecting a dataset of patients with rare diseases is challenging due to the condition's rarity. In contrast, it is relatively easy to collect a large dataset of unlabeled images for an image segmentation task, but annotation is slow and costly.

If obtaining data is the main obstacle, we identify three major approaches:

▸ **Add examples:** Generate more examples from available data (for example, through data augmentation).

▸ **Use additional information on existing data:** Increase the dimensionality of X in a manner that can assist the learner (for example, curriculum learning).

▸ **Use models encoding relevant knowledge:** Instead of learning from scratch, take advantage of models trained in a different yet relevant setup (for example, transfer learning).

If unlabeled data is abundant but labels are difficult to obtain, we identify two main approaches:

▸ **Acquire labels efficiently:** Label examples that should heavily contribute to the learning process.

▸ **Weak labeling:** Using proxy labels, either making assumptions about label distribution (for example, semi-supervised learning) or about the labeling process (for example, data programming), or using external (noisy) supervision signals correlated with the true labels (incidental supervision).

We note these approaches may also be combined. For example, one might add more examples and increase the dimensionality of the data. Here, we follow the taxonomy and elaborate on the different approaches and best practices.

Obstacle: Missing Data

Quite often, data is hard (or impossible) to obtain. In the following, we survey some of the main methods from the left branch in Figure 1: obtaining more examples efficiently, adding informative dimensions to existing data, or taking advantage of related tasks.

Add examples. This category focuses on methods for obtaining more examples.

Dataset repurposing

Use a preexisting dataset for a new purpose.

Dataset repurposing is perhaps the most obvious method to add data and is mentioned here for the sake of completeness. The idea is to use a preexisting dataset for a different task than it was originally constructed for.

For example, ImageNet was originally made and used for classification, but later was reused for image generation.[45] Similarly, the MS-COCO image captioning dataset was reused for training visually grounded word embeddings.[20]

Data repurposing also includes *transformations* on existing datasets. For example, consider *inpainting*, the process of restoring lost parts of an image based on the surrounding information. Inpainting is done using various preexisting datasets such as CelebA, Place2, and ImageNet,[39] where the same image splits into both X and Y (sometimes in more than one way).

Of course, it is also possible to repurpose a dataset created with no machine-learning task in mind at all: for example, Bertero and Fung[6] used a dataset of TV sitcoms for a supervised humor detection task, with recorded laughter serving as labels.

Data augmentation

Perform transformations on X to enlarge the dataset.

Data augmentation is a common approach for generating more data; it artificially inflates the training set by applying modifications. This method's initial goal was to prevent overfitting.

Data augmentation often employs *vicinal risk minimization* (VRM).[48] In VRM, human knowledge is needed to define a neighborhood around each example in the training data, and virtual examples are drawn from this vicinity distribution. It is easiest to demonstrate this idea in the field of computer vision; there, common augmentations are geometric transformations such as flipping, cropping, scaling, and rotating (see Figure 2). The idea is to make the classifier invariant to change in position and orientation. Similarly, photometric transformations amend the color channels to make the classifier invariant to change in lighting and color.

Data augmentation leads to improved generalization, especially with small datasets[3] or when the dataset is unbalanced (instead of under sampling, which is data-inefficient).

Augmentation methods have seen

a recent surge of interest. Recent advances include methods that jointly train a model for generating augmentations,[28] and methods that learn which augmentations best fit the data.[7] For example, AutoAugment[7] randomly chooses a sub-policy of batch transformation and searches for the one that yields the highest validation accuracy.

Beyond human-defined transformations, recent methods suggested using pretrained generative adversarial networks (GANs) to create new examples. Interestingly, the generated data points do not have to be *interpretable* by humans. For example, Mixup[59] trains a neural network on convex combinations of pairs of examples and their interpolated labels, treating it as "noisy" training data.

More information on existing data. Instead of adding new data points, this set of methods focuses on adding dimensions to existing points.

Multimodal learning

Integrate associated information on X from multiple modalities.

Multimodal learning attempts to enrich the input to the learning algorithm, giving the learner access to more than one modality of X; for example, an image accompanied by its caption. Multimodal learning is intuitive and like how infants learn (that is, children see new objects is often accompanied by additional semantic information). The main drawbacks of multimodal learning are obtaining rich input and effectively integrating it into the model.

Although the term "multimodal learning" is recent, many works combined information from different modalities.[11,22,41] These works, and more recent ones, show the promise of this method as an effective way to reduce data requirements and improve generalization.

Moreover, multimodal learning is also often used when the number of data points is extremely small, and in particular, few-, one-, and zero-shot learning (when only a few target-specific labeled examples exist for the learning process; thus, the learner must understand new concepts using only a handful of examples). For example, Visotsky et al.[51] used multimodal learning for few-shot learning by integrating additional per-sample information—in this case, a list of objects appearing in the input image (see Figure 3). Schwartz et al.[37] demonstrated that it is possible to outperform previous state of the art results on the popular miniImageNet and CUB few-shot learning benchmarks by combining images with multiple and richer semantics (category labels, attributes, and natural language descriptions).

Curriculum learning

Present examples to the learner according to a predetermined order, usually based on difficulty.

In *curriculum learning*, the learner is exposed to examples using a predetermined curriculum, where examples are usually sorted in increasing order of difficulty. Meta-data on X is needed to determine its place in the learning process.

The motivation behind curriculum learning comes from humans, as teachers tend to start by teaching simpler concepts (for example, learning to ride a bicycle with training wheels first). Thus, curriculum learning attempts to augment training examples with a difficulty score, often corresponding to *typicality*.

Given the difficulty score, the algorithm starts with a set of simple data points and gradually increases the dif-

Figure 2. Examples for common data augmentation manipulations of images as presented by Taylor and Nitschke.[40]

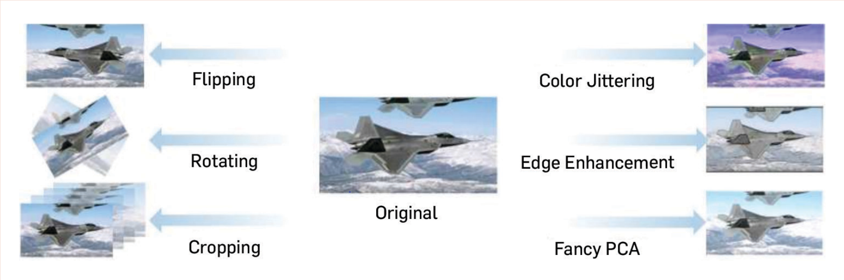


Figure 3. An illustration of the learning setup used by Visotsky et al.[51]

Labeled examples are accompanied with rich information that provides hints or explains classification. These labels are created during the training phase, where annotators write a list of objects, they observe in a visual scene using free text. Irrelevant background objects are often ignored. During the test phase, only the image is provided.

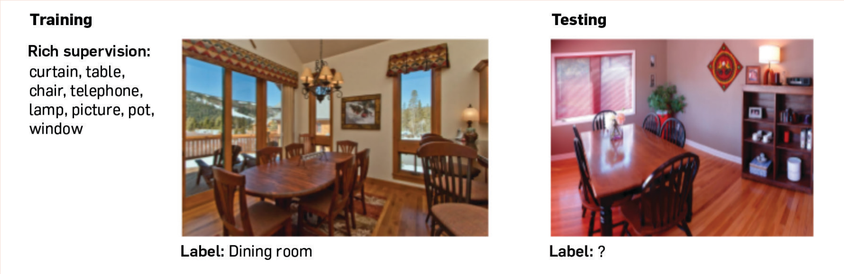
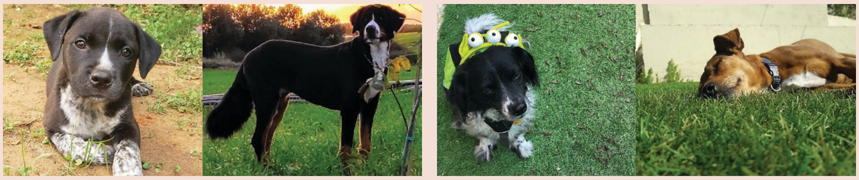


Figure 4: Typical versus non typical images of dogs are considered to be easy versus hard, respectively, in a dogs versus cats classification task.



(a)
Typical image of dogs,
easier to classify.

(b)
Non-typical images of dogs,
more difficult to classify.

ficulty of training examples throughout the learning process. This progression enables the model to learn the broad concept on a few easy examples and later refine the concept with more difficult ones. Figure 4 shows photos of dogs in the top row are more typical and should be easier for a classifier to recognize.

Curriculum learning has been shown to improve performance while decreasing the number of examples needed for convergence.[17] For example, Zaremba and Sutskever[58] showed how curriculum improves learning for the task of predicting the output of Python code without executing it.

A major caveat of curriculum learning is the inherent need for a difficulty-label estimator. Human labeling of difficulty can be very demanding, perhaps even more than standard annotation. In practice, the difficulty of each example is often learned by a teacher model, which may have access to related training data.[17]

A related concept is *self-paced learning* (SPL).[19] Intuitively, the curriculum in SPL is determined by the student's abilities, rather than being fixed by the teacher. Instead of heuristically designing a difficulty measure, SPL introduces a regularizor into the learning objective, with the goal of optimizing a curriculum for the model itself. This makes SPL broadly applicable.

Argumentation-based machine learning

Use experts' local knowledge to restrict the search space.

Argumentation-based machine learning (ABML) is a method to constrain the search space using *experts' local knowledge*.[26] In a nutshell, in ABML the learner attempts to find if-then rules to explain argumented examples in a rule induction process. The learner starts by finding a rule, adding it to a set of rules and removing all training data points that are covered by that rule. This process is repeated until all examples are removed. ABML's main advantage is the use of expert knowledge to justify *specific* examples, which is often easier than explaining global phenomena.

For example, Možina et al.[26] used ABML for medical records of deceased patients, where they used a physician's reasoning for the cause of death to limit the search space.

The data-bottleneck problem is widespread across the field of machine learning.

ABML is perhaps less popular than the other methods in this section. Nevertheless, if expert local knowledge is available, ABML is a powerful way to integrate partial prior knowledge. Moreover, the induced hypothesis should make more sense to an expert, as it must be consistent with the input arguments.

Models encoding relevant knowledge. Here, we go beyond the classical pipeline of training a model for a task; we present models that can take advantage of other, related tasks.

Multi-task learning

Co-learn multiple tasks simultaneously to enhance cross task similarities for better generalization.

Multi-task learning (MTL) is a prominent area of research where one attempts to train on multiple different (yet related) tasks simultaneously. These multiple tasks are solved concurrently, exploiting commonalities and differences across them.

It has been shown that challenging the learner to solve multiple problems at the same time results in better generalization and better performance on each individual task.[36] Indeed, MTL is successfully used in both vision and NLP. The key factors for this success in the absence of a large dataset are: It is an implicit data augmentation method, based on cross-task commonalities; it enables unraveling cross tasks and feature correlations; and encouraging a classifier to also perform well on a slightly different task is a better regularization than uninformed regularizers (for example, enforce weights to be small, which is the typical $L2$-regularization).

As an example, consider the case of spam-filtering. Quite often, data from an individual user is insufficient for training a model. Intuitively, different people have different distributions of features that distinguish spam from legitimate email. For example, email messages in Russian are probably spam for English speakers, but not for Russian speakers. However, inter-user commonalities can be utilized to solve this problem (for example, text related to money transfer is probably spam). To build upon these similarities, Attenberg et al.[4] created an MTL-based spam-filter, treating each individual user as one distinct but related classification task and training a model across the different users.

A more recent example of MTL learning is the T5 model (see Figure 5).[29] This model achieves state-of-the-art results on many NLP benchmarks while being flexible enough to be fine-tuned to a variety of downstream tasks. T5 receives as input the task at hand and thus allows the use of the same model, loss function, and hyperparameters for any NLP task.

MTL implementations can be divided into two main categories – hard versus soft parameter sharing of the hidden layers, where hard parameter sharing is more commonly used. In the hard type, the hidden layers are shared between all tasks while keeping several task-specific output layers. Baxter[5] showed that hard parameter sharing reduces the risk of overfitting to order N (the number of tasks), which is smaller than the risk of overfitting the task-specific parameters (the output layers). In soft parameter sharing, each task has its own model and parameters. The distance between model's parameters is then regularized to encourage them to be similar (enhance cross tasks' similarity), as done by Duong et al.[10]

Transfer learning

Transfer knowledge gained while solving one problem to a different yet related problem.

Transfer learning is a widely used, highly effective way to integrate prior knowledge, like humans, who never approach a new problem tabula rasa, but rather with rich experience of somewhat similar problems and their solutions.[42]

The idea is to use preexisting models trained on related tasks. These pretrained models are usually used as an initialization for finetuning using a small dataset for the task in hand. Thus, significantly less task-specific examples are needed for convergence.

Another beneficial side effect is the use of the model's initial wide domain knowledge, compared to initialization with random weights. In other words, the model starts the fine-tuning phase with some relevant world knowledge.

For example, models trained on ImageNet have been transferred to medical imaging tasks, including inspecting chest x-rays[54] and retinal fundus images.[8] The idea is that a network trained on a large and diverse dataset of images captures universal visual features such as curves and edges in its early layers (similar to the primary visual cortex of humans and many other mammals, a Nobel prize winning discovery[a]). Despite the difference between the images in ImageNet and those in the downstream tasks, these features are relevant for many vision tasks. Therefore, this approach significantly decreases the size of labeled task-specific data needed.

In NLP, the commonly used pretrained model BERT achieves state-of-the-art results in various tasks.[9] Pretraining such models is often done in a *self-supervised* manner, where different parts of the input are masked, and the learner's goal is to predict the masked parts. For example, given a sentence, it is possible to iterate over it, masking a different word each time, to create various examples.

Fine-tuning in deep networks is usually done either by adding an untrained last layer and training the new model on the small task-specific dataset or by taking the output embeddings of the next to last layer. Another possible fine-tuning technique is to train the whole network with a relatively small learning rate; that is, perform small changes on the already-decent weights (as a heuristic, about 10 times smaller than the learning rate used for pretraining). Fine-tuning can also be done by freezing the weights of the first few layers of the pretrained model. The motivation behind this technique is that the first layers capture universal features that would probably also be relevant to the new task. Thus, freezing them during fine-tuning should keep the captured information that is relevant for both the original and the new tasks.

To conclude, transfer learning is a powerful tool for both reducing the amount of task-specific data needed and improving models' performance.

Meta learning

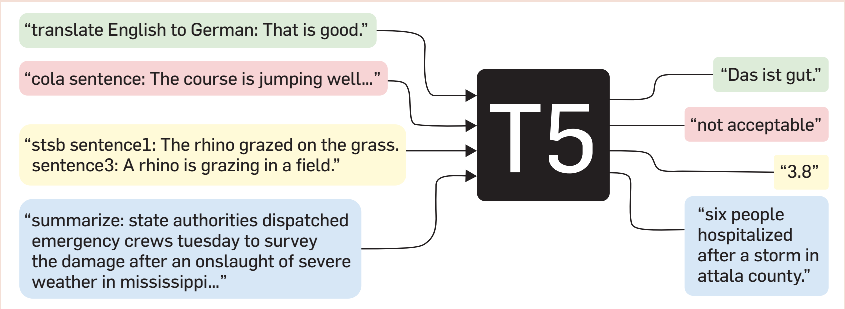
Improve the learning algorithm by generalizing based on experience from multiple learning episodes.

Meta-learning (also known as "learning to learn") is a recent subfield of machine learning,[12] focusing on designing models that can learn new tasks or adapt to new environments rapidly, with only a few training examples. It is based on creating a meta-learner that has wide prior knowledge regarding the relevant topic(s). Meta learning is also inspired by human learning. For example, people who know how to ride a bicycle are more likely to quickly learn to ride a motorcycle.

Note that while meta learning can often be meaningfully combined with MTL systems, their objectives are different. While MTL aims to solve all training tasks, meta learning aims to use the training tasks for solving new tasks with small data. Thus, meta learning is about creating models with *prior experience* that can quickly adapt to new tasks. Specifically, the meta-learner gradually learns meta-knowledge across tasks, which can be

Figure 5. Multi-task paradigm as presented by Raffel et al.[29]

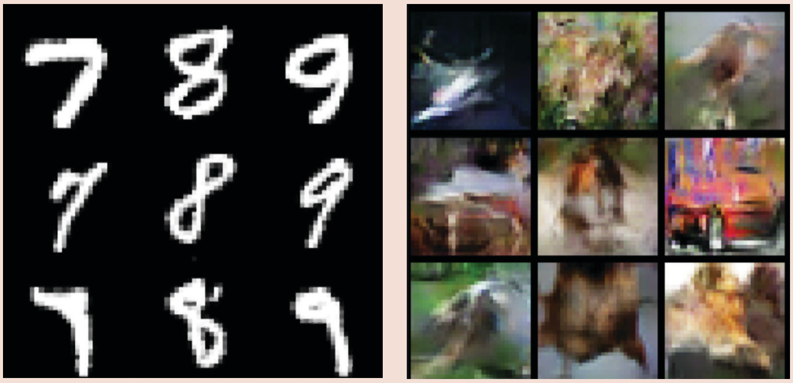
The objective is to train a model to perform several tasks that are closely related. The input contains the current task, which allows the use of same model, loss function and hyperparameters across various tasks.


"translate English to German: That is good."
"cola sentence: The course is jumping well…"
"stsb sentence1: The rhino grazed on the grass. sentence3: A rhino is grazing in a field."
"summarize: state authorities dispatched emergency crews tuesday to survey the damage after an onslaught of severe weather in mississippi…"
T5
"Das ist gut."
"not acceptable"
"3.8"
"six people hospitalized after a storm in attala county."

[a] https://www.nobelprize.org/uploads/2018/06/hubel-lecture.pdf

Figure 6. Images generated by the GAN transformation approach for "near-miss" examples.[43]

Images generated based on the MNIST dataset are interpretable to humans, while this is not the case for the CIFAR10 examples.



(a) MNIST

(b) CIFAR 10

generalized to a new task using little task-specific information.

There are three common approaches to meta-learning: metric-based (similar to nearest-neighbor algorithms), optimization-based (meta-gradients optimizing), and model-based (no assumptions about data distribution).

As an example of a metric-based approach, Vinyals et al.[50] proposed a framework that explicitly learns from a given support set to minimize a loss over a batch. The result is a model that learns to map a small, labeled support set and an unlabeled example to its label, obviating the need for fine-tuning to adapt to new class types. They then showed the superiority of this method in both vision and NLP tasks.

A well-known work in the optimization-based line of research is model-agnostic meta-learning (MAML), which is a general optimization algorithm, compatible with any gradient descent-based model.[12] It uses a meta-loss specifically designed to induce quick changes when fine-tuned on new tasks and is based on N-gradients (where N is the total number of tasks).

In the model-based line of research, Munkhdalai and Yu[27] presented MetaNet, a meta-learning model designed specifically for rapid generalization across tasks. The rapid generalization of MetaNet relies on "fast weights", which are parameters of the network with a smaller timescale for changes than the regular gradient-based weight changes. This Hebbian short-term plasticity maintains a dynamically changing short-term memory of the recent history of the units' activities in the network, as opposed to the standard slow recurrent connectivity. This model outperforms various other recurrent models across several tasks.

Obstacle: Missing Labels

We now turn our attention to the second major branch in Figure 1, where unlabeled data is abundant, but there are few labels (or no labels at all). This setting is common in practice because unlabeled data is often much easier to obtain than labeled data. In this section we cover two main approaches. The first deals with ways to acquire labels efficiently, and the other uses weak labels.

Active learning

Generate examples which are close to the decision boundary. These examples should contribute to the learning process more than random examples.

Acquiring labels efficiently. When more labels are needed but annotation is costly, an immediate question would be how to acquire new labeled data *efficiently*. The prime example of this is active learning, in which the learner can iteratively query an oracle (information source) to label new data points.[32] These queries can include unlabeled examples either from the dataset or new ex-nihilo data points, often ones that are close to the decision boundary. The rationale is that not all examples contribute equally to the learning process: diverse examples that are difficult for the learner to classify might be especially useful and could decrease the number of data points needed for learning.

There are many methods to determine which data points from the training set should be queried next. Common objectives include picking examples which will change the current model the most, examples which the current model is least certain about, or diverse examples that resemble the data distribution. For example, Hacohen et al.[16] recently showed that in the presence of little data it is most beneficial to present the model with typical examples (compared to scenarios with more data, in which it is best to use examples that are close to the decision boundary).

When generating *new* examples (rather than selecting unlabeled ones from the training set), it is important to remember that humans will be the ones labeling them. We wish to point out that while data augmentation modifies the input but *keeps* its label (as discussed earlier), active learning generates examples *without labels*. Thus, the generation algorithm should keep the new points interpretable, that is, ensure they have a clear label.[14] For example, Zarecki and Markovitch[57] automatically transformed sentences' sentiment by replacing key words that bring them closer to the classification boundary (while keeping their syntax).

Recent approaches use GANs to generate new examples, either from scratch (and label them),[60] or by modifying an existing example (while attempting to preserve the label).[43] Both scenarios update the learner and the GAN model simultaneously after labeling a new example.

Importantly, the GAN approaches are more expressive than transformation-based approaches, but the result is often less interpretable. Figure 6 shows an example of modified images from Tran et al.[43] Note that while the MNIST examples (handwritten digits) have relatively clear labels, the CIFAR10 examples (tiny images in ten classes such as airplane, dog, and ship) are not as easy to label.

A note on gamification. Active learning is the dominant paradigm for reducing the *number* of annotations needed. However, a different approach to label efficiently is to reduce the cost of annotations. A notable example is *gamification*—applying gaming mechanics to non-gaming environments, to make tasks more enjoyable and give annotators a non-monetary incentive to provide labels. The challenge in gamification is often to design the game to create the right incentive. This is far from trivial, and requires knowledge of game design, motivational psychology, and an understanding of the target group.[25] Ignorance of the complexity involved in gamification often results in modest outcomes.

The seminal work of Von Ahn and Dabbish[52] demonstrated a two-player game for image labeling, where the players gain points for describing an image using the exact same term. The researchers famously estimated that if users were to play the game at the same rate as other popular online games, most images on the Web could be labeled (for free) within only a few months. Another example is the unfun.me corpus used in humor research. This corpus was constructed via an online game where players change satirical headlines into serious ones with minimal edits.[55]

Weak labeling. If we cannot obtain labels efficiently, we could choose to obtain noisy labels as a proxy. In vision, this is sometimes referred to as "automatic image annotation." We cover two main types of noisy labels here.

Assumptions on $P(Y=y|X=x)$.

When generating *new* examples (rather than selecting unlabeled ones from the training set), it is important to remember that humans will be the ones labeling them.

Semi-supervised learning

Harness information regarding $P(X=x)$ to reduce labeling requirements by integrating labeled and non-labeled examples in the learning process.

Semi-supervised learning (SSL) is a very large and active area of research, and we do not profess to cover all it; for a recent survey on SSL, we refer the reader to van Engelen and Hoos.[46]

SSL estimates the distribution $P(X=x)$ using a large amount of *unlabeled*, to reduce the annotated data requirements. It makes strong assumptions about the relation between $P(X=x)$ and $P(Y=y|X=x)$ to reduce the number of labeled examples needed.[56] Typically, these assumptions take the following forms:

- **Smoothness:** Points that are close to each other are more likely to share a label. More formally, every two adjacent samples x, x' should have similar labels.
- **Cluster-ability:** Data tend to form discrete clusters where points belonging to the same cluster are more likely to share a label. Thus, the decision boundary can only pass through low-density areas in the feature space.
- **Manifold:** Data lies approximately on a manifold of a much lower dimension than the input space. Thus, when considering low-dimensional manifolds of the input space, any data points on the same manifold should have the same label.

All three assumptions can be seen as different definitions of interpoints similarity: The smoothness defines it as proximity in the input space, the cluster-ability assumes high-density areas contain similar data points, and the manifold states that points which lie on the same low-dimensional manifold are similar.

Another important distinction in SSL is between inductive and transductive methods. The former yields a classification model to predict the label of a new example, like supervised learning ($f: X \rightarrow Y$). The latter do not yield such a model, but instead directly provide predictions. Transductive approaches are usually graph-based, while the inductive approaches can be further divided into *unsupervised preprocessing*, *intrinsically semi-supervised*, and *wrapper* methods.[46]

One popular way of using the unsupervised preprocessing approach is to use the knowledge on $P(X = x)$ to extract useful features in a lower dimension than the original dimension of X and thus reduce the learning complexity. This includes learning a representation using an auto-encoder model[49] or applying a dimensionality reduction method like PCA.[1]

Under the inductive approach, it is also possible to use an intrinsically semi-supervised model like semi-supervised SVM, which changes the optimization target to find a decision boundary with maximal margin from both labeled and unlabeled points (for example, using SVM).[47] This can also be applied to neural networks by adding a form of regularization over the unlabeled data.[30]

In wrapper methods, a model is initially trained from the available set of examples.[44] It then makes predictions on the unlabeled dataset. The model's pseudo-labels are added as labeled data for the next iteration of supervised learning. This process is repeated until convergence.

Data programming

Integrate multiple weak heuristics regarding the labeling process $f: X \rightarrow Y$ to create noisy labels.

Data programming is a paradigm for the programmatic creation of training sets. In data programming, users express weak supervision strategies or domain heuristics as labeling functions (LFs), which are programs that label subsets of the data. Importantly, LFs are imprecise and can contradict each other, resulting in noisy labels. By explicitly representing the labeling process $f: X \rightarrow Y$ as a generative model, data programming aims to "denoise" the generated training set.

For example, in spam-detection, potential LFs would return "spam" if the email contains a URL or a money transfer request, and "no-spam" if coming from someone in your contact list. These functions alone achieve poor performance; however, like ensemble methods (where a group of weak learners comes together to form a strong one with superior accuracy), the strength of data programming is in the combination of many weak heuristics.

A popular system for data programming is Snorkel.[31] It applies the (noisy) LFs to the data and estimates their accuracy and correlations, using only their agreements and disagreements. This information is then used to reweight and combine LF predictions to output probabilistic noise-aware training labels. This process is presented in Figure 7.

Expectation regularization

Using prior knowledge regarding the proportion of the different labels in sub-groups of the data to create noisy labels.

Prior knowledge regarding labels' *proportion* in various subgroups of the data, makes it possible to automatically create noisy labels in a process called *expectation regularization* (learn from label proportions).[53]

This estimation process relies on uniform convergence properties of the expectation operator. It uses empirical means of the sub-groups to approximate expectations with respect to a group's distribution. The latter is then used to compute expectations with respect to a given label, and finally, the conditional means on the label distribution are used to estimate the conditional group means.

A recent work in this area is *ballpark learning*, which relaxes the assumption of known label proportions, assuming instead soft constraints on proportions within and between groups of instances (for example, "the percentage of spam in emails mentioning a certain word is between k_{low} and k_{high}", or "emails containing a link have at least k% more spam than emails without links").[18] Ballpark learning learns a model that labels individual instances while satisfying these soft, noisy constraints.

Noisy Supervision from External Datasets. It is sometimes possible to take advantage of preexisting datasets to get a noisy supervision signal.

Distant supervision

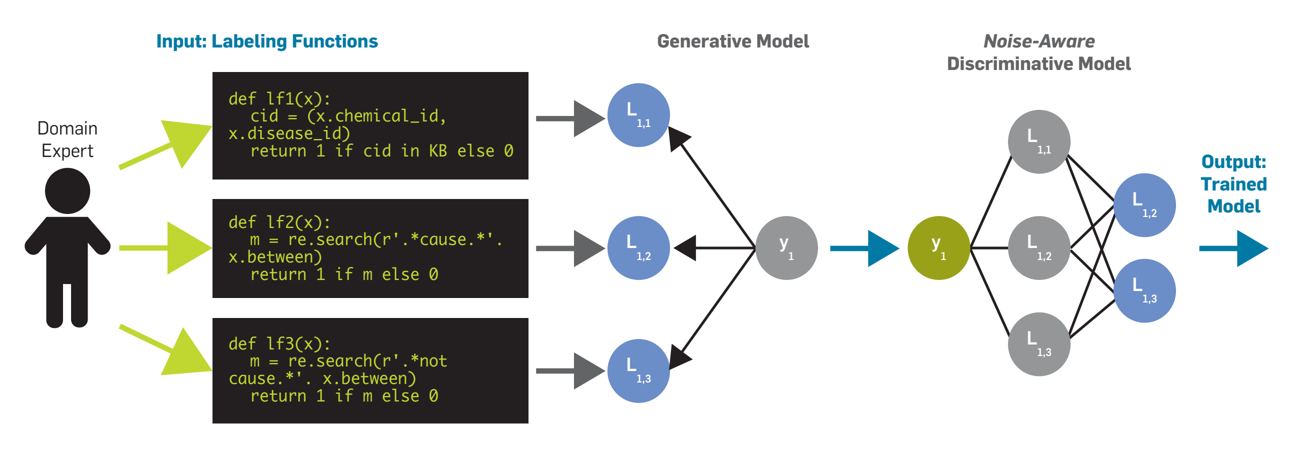
Use a preexisting database to collect examples for the desired relation. These examples are then used to automatically generate labeled training data.

Distant supervision is a popular method to use existing datasets. In distant supervision, a model is learned given a labeled training set, as in "standard" supervised ML, but the training data is weakly labeled (that is, labeled automatically, based on heuristics or rules).

For example, Mintz et al.[24] used Freebase, a large, unlabeled, semantic database, to provide distant supervision for relation extraction. The intuition is that any sentence that contains a pair of entities with a known Freebase relation is likely to express that relation in some way.[23] For example, each pair of "Barack Obama" and "Michelle Obama" that appear in the same sentence can be extracted as a positive example for the marriage relation. Due to the poten-

Figure 7. Illustration of Snorkel's pipeline.

The domain expert creates noisy labeling functions (LFs). A generative model learns to resolve and model the output of these LFs. The model's output is the input to a discriminative model. Image reproduced from https://towardsdatascience.com/snorkel-a-weak-supervision-system-a8943c9b639f



```
def lf1(x):
   cid = (x.chemical_id,
x.disease_id)
   return 1 if cid in KB else 0
```

```
def lf2(x):
   m = re.search(r'.*cause.*'.
x.between)
   return 1 if m else 0
```

```
def lf3(x):
   m = re.search(r'.*not
cause.*'. x.between)
   return 1 if m else 0
```

tially large number of sentences that contain a given entity pair, it is possible to extract and combine noisy features for the labeling process. Based on these semantic signals, Mintz et al.[24] was able to use 116 million unlabeled instances.

Incidental supervision

Exploit weak signals that exist in data independently of the task at hand.

The *incidental supervision* framework is based on the idea that informative cues for a task could exist in datasets that were not constructed with this task in mind. For example, suppose we want to infer gender from first names. One could use Wikipedia, which was not created for this task. The incidental signal would be pronouns and other gender indicators appearing in the first paragraph of Wikipedia pages about people with that first name. This signal is correlated to the task at hand and (together with other signals and inferences), could be used for supervision, reducing the need for annotations.

Incidental supervision does not assume knowledge about the labeling process.[34] Moreover, incidental signals can be noisy, partial, or only weakly correlated with the target task, and still be used to provide supervision and facilitate learning. Note that the notion of supervision here is different from that of distant supervision: In distant supervision, the model learns in the standard supervised learning way, but the training set is labeled automatically, based on heuristics. In incidental supervision, a complete training set might never exist.

Context-sensitive spelling and grammar checking is a task that has been relying on incidental supervision for over 20 years now.[13] Under the assumption that most edited textual resources (books, newspapers, Wikipedia) do not contain many spelling and grammar errors, these methods generate contextual representations for words, punctuation marks and phenomena such as agreements. These representations are then used to identify mistakes and correct them in a context-sensitive manner.[35]

An unintentional example for the power of incidental signals comes from image processing, where the task of gender detection based on the iris texture was solved with great accuracy (over 80% for most papers and an impressive score of 99.5% reported by Alrashed and Berbar[2]). However, it was later discovered that most models did not detect a person's gender; rather, they detected the use of cosmetic mascara, which is a much easier task and is indeed correlated with the original assignment.[21] Thus, although unintentionally, this finding emphasized the potential of using incidental cues.

Identifying assumptions is essential for breaking them—and breaking assumptions is an established technique for encouraging creativity and innovation.

Conclusion

The dominant paradigm in ML today is creating large, task-specific datasets (often using crowdsourcing). In this review we devise a taxonomy for alternative ways to tackle the data bottleneck problem. The taxonomy aims to bring order to the various methods suggested across different subfields, as well as making it easier to identify underlying assumptions and potential new directions. Identifying assumptions is essential for breaking them—and breaking assumptions is an established technique for encouraging creativity and innovation.

For example, surveying the taxonomy, several common assumptions that stand out are that samples tend to be representative of the data, that we have information about X and Y conjointly, and that each example has exactly one correct label. This raises the prospect of new learning settings (for example, what if we only have knowledge about the distributions of data points $P(X = x)$ and labels $P(Y = y)$, separately?), and of new ways to aggregate multiple (correct but different) labels.

We note that our taxonomy covers widely diverse techniques, making very different assumptions. Ultimately, we expect that choosing a technique will often boil down to what the practitioner has access to (that is, which assumptions are met). For example, in multitask learning the practitioner not only possess labeled data for their task, but also for several related tasks; in data programming, they have no (or very few) labels for their task but possess some partial knowledge about the labeling process; in curriculum learning, they know something about the hardness of data points; and so on.

We further wish to point out that it is not always obvious whether a method's assumptions are met in practice, or to estimate which method is better

more online
A version of this article with a comprehensive list of references is available at https://dx.doi.org/10.1145.3551635

suited for a specific use case. The answer might depend on many factors, such as the inherent difficulty of the concept one wishes to learn, biases in the data, or the manual effort needed to obtain high-quality input for the different methods. For example, in methods using weak labeling, the tradeoff between implementation speed and accuracy for different weak labels is often not clear in advance.

In addition to the inherent difficulty of collecting large datasets, we note there are growing concerns about such datasets, including environmental costs, financial costs, opportunity costs, and more.[38] We also note that large datasets are still prone to fitting artifacts, and that several recent methods have attempted to address the recurring challenges of the annotation artifacts and human biases found in many existing datasets.[15]

In conclusion, ML has made tremendous progress using large datasets, but they are not a panacea for all problems. Our hope is that this paper will encourage re-thinking about current annotation-heavy approaches.

Acknowledgment. This work was supported by a grant from Israel Ministry of Science and Technology and by the European Research Council (ERC) under European Union's Horizon 2020 research and innovation program (grant no. 852686, SIAM).

References

1. Alaíz, C., Fanuel, M., and Suykens, J. Convex formulation for kernel PCA and its use in semisupervised learning. *IEEE Trans. Neural Networks and Learning Systems 29*, 8 (2017), 3863–3869.
2. Alrashed, H. and Berbar, M. *Facial Gender Recognition Using Eyes Images* (2013).
3. Anaby-Tavor, A., et al. Do not have enough data? Deep learning to the rescue. In *Proceedings of the AAAI Conf. Artificial Intelligence 34* (2020), 7383–7390.
4. Attenberg, J., Weinberger, K., Dasgupta, A., Smola, A., and Zinkevich, M. Collaborative email-spam filtering with the hashing trick. CEAS (2009).
5. Baxter, J. A Bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine Learning 28*, 1 (1997), 7–39.
6. Bertero, D. and Fung, P. Predicting humor response in dialogues from TV sitcoms. In *2016 IEEE Intern. Conf. Acoustics, Speech, and Signal Processing*, 5780–5784.
7. Cubuk, E., Zoph, B., Mané, D., Vasudevan, V., and Le, Q. AutoAugment: Learning augmentation strategies from data. In *Proceedings of 2019 IEEE/CVF Conf. Computer Vision and Pattern Recognition*, 113–123.
8. De Fauw, J., et al. Clinically applicable deep learning for diagnosis and referral in retinal disease. *Nature Medicine 24*, 9 (2018), 1342–1350.
9. Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conf. the North American Chapter of the Assoc. Computational Linguistics: Human Language Technologies, 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, MN, 4171–4186. https://doi.org/10.18653/v1/N19-1423
10. Duong, L., Cohn, T., Bird, S., and Cook, P. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd Annual Meeting of the Assoc. Computational Linguistics* and the *7th Intern. Joint Conf. Natural Language Processing 2* (Short Papers, 2015, 845–850.
11. Farhadi, A., Endres, I., Hoiem, D., and Forsyth, D. Describing objects by their attributes. In *Proceedings of the 2009 IEEE Conf. Computer Vision and Pattern Recognition*, 1778–1785.
12. Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th Intern. Conf. Machine Learning-Volume 70*. JMLR. Org, 2017, 1126–1135.
13. Golding, A. and Roth, D. A winnow-based approach to context-sensitive spelling correction. *Machine Learning 34*, 1–3 (1999), 107–130.
14. Gurevich, N., Markovitch, S., and Rivlin, E. Active Learning with near Misses. AAAI Press, 2006, 362–367.
15. Gururangan, S., Swayamdipta, S., Levy, O., Schwartz, R., Bowman, S., and Smith, N. Annotation artifacts in natural language inference data. 2018; *arXiv:1803.02324*.
16. Hacohen, G., Dekel, A., and Weinshall, D. Active learning on a budget: Opposite strategies suit high and low budgets. 2022; *arXiv:2202.02794*.
17. Hacohen, G. and Weinshall, D. On the power of curriculum learning in training deep networks. 2019; arXiv:1904.03626.
18. Hope, T. and Shahaf, D. Ballpark learning: Estimating labels from rough group comparisons. In *Proceedings of the Joint European Conf. Machine Learning and Knowledge Discovery in Databases*. Springer, 2016, 299–314.
19. Jiang, L., Meng, D., Zhao, Q., Shan, S., and Hauptmann, A. Self-paced curriculum learning. In *Proceedings of the 29th AAAI Conf. Artificial Intelligence*, 2015.
20. Kottur, S., Vedantam, R., Moura, J., and Parikh, D. VisualWord2Vec (Vis-W2V): Learning visually grounded word embeddings using abstract scenes. In *Proceedings of the 2016 IEEE Conf. Computer Vision and Pattern Recognition*, 2015, 4985–4994.
21. Kuehlkamp, A., Becker, B., and Bowyer, K. Gender-from-iris or gender-from-mascara. In *Proceedings of the 2017 IEEE Winter Conf. Applications of Computer Vision*, 1151–1159.
22. Lampert, C., Nickisch, H., and Harmeling, S. Learning to detect unseen object classes by between-class attribute transfer. In *Proceedings of the 2009 IEEE Conf. Computer Vision and Pattern Recognition*, 951–958.
23. Lin, Y., Liu, Z., Sun, M., Liu, Y., and Zhu, X. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the 29th AAAI Conf. Artificial Intelligence*, 2015.
24. Mintz, M., Bills, S., Snow, R., and Jurafsky, D. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conf. 47th Annual Meeting of the ACL* and the *4th Intern. Joint Conf. Natural Language Processing of the AFNLP, 2*. Assoc. Computational Linguistics, 2009, 1003–1011.
25. Morschheuser, B. and Hamari, J. The gamification of work: Lessons from crowdsourcing. *J. Management Inquiry 28*, 2 (2019), 145– 148.
26. Možina, M., Žabkar, J., and Bratko, I. Argument based machine learning. *Artificial Intelligence 171*, 10-15 (2007), 922–937.
27. Munkhdalai, T. and Yu, H. Meta networks. In *Proceedings of the 34th Intern. Conf. Machine Learning 70*. JMLR. Org, 2017, 2554–2563.
28. Perez, L. and Wang, J. The effectiveness of data augmentation in image classification using deep learning. 2017; *arXiv:1712.04621*.
29. Raffel, C., et al. Exploring the limits of transfer learning with a unified text-to-text transformer. 2019; *arXiv:1910.10683*.
30. Rasmus, A., Berglund, M., Honkala, M., Valpola, H., and Raiko, T. Semi-supervised Learning with Ladder Networks. In *NIPS*, 2015.
31. Ratner, A., Bach, S., Ehrenberg, H., Fries, J., Wu, S., and Ré, C. Snorkel: Rapid training data creation with weak supervision. *The VLDB J.* (2019), 1–22.
32. Ren, P., et al. A survey of deep active learning. *ACM Computing Surveys 54*, 9 (2021), 1–40.
33. Roh, Y., Heo, G., and Whang, S. A survey on data collection for machine learning: a big data-ai integration perspective. *IEEE Trans. Knowledge and Data Engineering 33*, 4 (2019), 1328–1347.
34. Roth, D. Incidental supervision: Moving beyond supervised learning. In *Proceedings of the 31st AAAI Conf. Artificial Intelligence*, 2017.
35. Rozovskaya, A. and Roth, D. Building a state-of-the-art grammatical error correction system. *Tran. Assoc. Computational Linguistics 2* (2014), 419–434.
36. Ruder, S. An Overview of Multi-Task Learning in Deep Neural Networks. 2017; *arXiv abs/1706.05098* (2017).
37. Schwartz, E., Karlinsky, L., Feris, R., Giryes, R., and Bronstein, A. Baby steps towards few-shot learning with multiple semantics. arXiv preprint *arXiv:1906.01905* (2019).
38. Schwartz, R., Dodge, J., Smith, N., and Etzioni, O. Green AI. *arXiv preprint arXiv:1907.10597*, 2019.
39. Shin, Y., Sagong, M., Yeo, Y., Kim, S., and Ko, S. Pepsi++: fast and lightweight network for image inpainting. *IEEE Trans. Neural Networks and Learning Systems* (2020).
40. Taylor, L. and Nitschke, G. Improving deep learning using generic data augmentation. 2017; *arXiv:1708.06020*.
41. Tian, Y., Shi, J., Li, B., Duan, Z., and Xu, C. Audiovisual event localization in unconstrained videos. In *Proceedings of the 2018 European Conf. Computer Vision*, 247–263.
42. Torrey, L. and Shavlik, J. Transfer learning. *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*. IGI Global, 2010, 242–264.
43. Tran, T., Do, T., Reid, I., and Carneiro, G. Bayesian Generative Active Deep Learning. 2019; *arXiv abs/1904.11643*.
44. Triguero, I., García, S., and Herrera, F. Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Knowledge and Information Systems 42* (2013), 245–284.
45. van den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. Pixel Recurrent Neural Networks. In *ICML, 2016*.
46. van Engelen, J. and Hoos, H. A survey on semi-supervised learning. *Machine Learning* (2019), 1–68.
47. Vapnik, V. Statistical learning theory (1998).
48. Vapnik, V. The vicinal risk minimization principle and the SVMs. T*he Nature of Statistical Learning Theory*. Springer, 2000, 267–290.
49. Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P. Extracting and composing robust features with denoising autoencoders. *ICML '08*.
50. Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. Matching networks for one shot learning. *Advances in Neural Information Processing Systems 29* (2016), 3630–3638.
51. Visotsky, R., Atzmon, Y., and Chechik, G. Few-shot learning with per-sample rich supervision. 2019; *arXiv:1906.03859* (2019).
52. Von Ahn, L. and Dabbish, L. Labeling images with a computer game. In *Proceedings of the 2004 SIGCHI Conf. Human Factors in Computing Systems*, 319–326.
53. Wang, M. and. Manning, C. Cross-lingual projected expectation regularization for weakly supervised learning. *Trans. Assoc. Computational Linguistics 2* (2014), 55–66; https://doi.org/10.1162/ tacl_a_00165
54. Wang, X., Peng, Y., Lu, L., Lu, Z., Bagheri, M., and Summers, R. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *Proceedings of the 2017 IEEE Conf. Computer Vision and Pattern Recognition*, 2097–2106.
55. West, R. and Horvitz, E. Reverse-engineering satire, or 'paper on computational humor accepted despite making serious advances.' In *Proceedings of the 2019 AAAI Conf. Artificial Intelligence 33*, 7265–7272.
56. Xiaojin, Z. Semi-supervised learning literature survey. *Computer Sciences TR 1530* (2008).
57. Zarecki, J. and Markovitch, S. Textual membership queries. In *Proceedings of the 29th Intern. Conf. Intern. Joint Conf. Artificial Intelligence*, 2021, 2662–2668.
58. Zaremba, W. and Sutskever, I. Learning to execute. 2014; *arXiv:1410.4615*.
59. Zhang, H., Cissé, M., Dauphin, Y., and Lopez-Paz, D. Mixup: Beyond Empirical Risk Minimization. 2017; *arXiv abs/1710.09412*.
60. Zhu, J. and Bento, J. Generative Adversarial Active Learning. 2017; *arXiv abs/1702.07956*.

Chen Shani is a Ph.D. student at The Hebrew University of Jerusalem, Israel.

Jonathan Zarecki is R&D Group Lead for Israeli Military Intelligence, Tel Aviv, Israel.

Dafna Shahaf is an associate professor of data science at The Hebrew University of Jerusalem, Israel.

© 2023 ACM 0001-0782/23/2