MITSloan
Management Review

How to Build Good AI Solutions When Data Is Scarce

WINTER 2023
ISSUE

Data-efficient AI techniques are emerging — and that means you don't always need large volumes of labeled data to train AI systems based on neural networks.

Rama Ramakrishnan

Vol. 64, No. 2 • Reprint #64202 • sloanreview.mit.edu

How to Build Good AI Solutions When Data Is Scarce

Data-efficient AI techniques are emerging — and that means you don't always need large volumes of labeled data to train AI systems based on neural networks.

By Rama Ramakrishnan

CONVENTIONAL WISDOM HOLDS THAT you need large volumes of labeled training data to unlock value from powerful AI models. For the consumer internet companies where many of today's AI models originated, this hasn't been difficult to obtain. But for companies in other sectors — such as industrial companies, manufacturers, health care organizations, and educational institutions — curating labeled data in sufficient volume can be significantly more challenging.

There's good news on this front, however. Over the past few years, AI practitioners and researchers have developed several techniques to significantly reduce the volume of labeled data needed to build accurate AI models. Using these approaches, it's often possible to build a good AI model with a fraction of the labeled data that might otherwise be needed.

Assembling lots of labeled data is expensive and difficult. Imagine that you're the CEO of a manufacturer of home-office furniture. Your customers post reviews of your products on e-commerce sites and social media, and some of these reviews provide valuable insights into potential product defects and improvements.

Since your business is growing rapidly, the volume of review content has grown to a level where it is impossible to manually read through each piece and glean its potential for product improvement. You decide that you need to build an AI model that can "read" each review and assess whether it contains a defect, an improvement idea, or neither. With such a model in place, you can route the relevant reviews to the right teams for follow-up.

The traditional AI approach to solving this problem consists of these steps: (1) Assemble a data set of reviews; (2) design a process for labeling each review with "improvement," "defect," or "neither"; (3) recruit a team of labelers and train them to label data accurately; (4) label thousands (if not tens of thousands) of reviews; and (5) with this review-and-label data set, build a series of AI models in a trial-and-error process until you arrive at one that can classify reviews with acceptable accuracy.

Steps 3 and 4 can be more difficult and expensive than it might appear on the surface. Unlike looking at an image and deciding whether it's a dog or a cat, determining whether a review has a product improvement idea could be quite difficult. Different labelers might disagree on the correct label for a review, and you'll need a process to adjudicate disagreements.[1]

Pete Ryan / theispot.com

These issues can be even worse in certain contexts where additional labels may be impossible to obtain even if you're willing to invest time and capital. In health care, for example, consider sudden cardiac death (SCD). Standard data sets with millions of patient records typically have only a small number of records "labeled" as SCDs. With such a small number of labeled examples, it might not be possible to build an accurate model to predict the risk of SCD on the basis of other variables. And we can't choose to increase the number of labeled examples either, if the prevalence of SCD is very low in the patient population.[2] To take another example, if we want to build an AI system to detect defects on phones coming off a manufacturing line, we'll be faced with a similar problem if the factory has only ever made 50 defective phones.[3]

Against this backdrop, practitioners and researchers have recognized the value of and devised techniques for reducing the number of labeled examples needed to build accurate models. These approaches encompass ways to transfer models across related problems and to pretrain models with unlabeled data. They also include emerging best practices around data-centric AI, which we discuss below.

These methodologies show significant promise. For example, by using one of these techniques, researchers at the Broad Institute (affiliated with MIT, Harvard, and Massachusetts General Hospital) were able to build a model to predict the risk of atrial fibrillation with only about 6% of the labeled examples that would have been needed to build an equally good model from scratch.

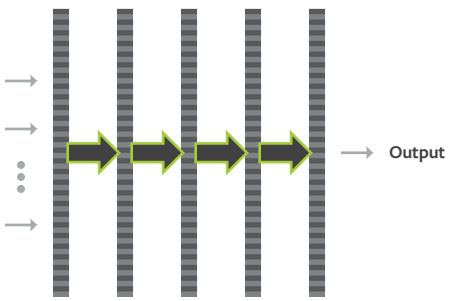
To understand how these techniques work, let's examine what happens inside a neural network — on which the vast majority of AI solutions are based — when it is trained to solve a prediction problem.

Neural Networks Are Representation Learners

Neural networks are typically composed of layers arranged in a sequence: an input layer, multiple layers in which the data is transformed, and an output layer. (See "How Neural Networks Learn.") Input data enters the network at one end, flows through the layers, and exits the other end as the prediction.

HOW NEURAL NETWORKS LEARN

In a neural network, data inputs are processed by a series of layers and are transformed at each stage into a new representation. They exit as a prediction, such as the likelihood of a credit applicant repaying a loan.



As data flows through a layer, it's transformed. It may enter the first layer as, for example, a list of 10 numbers and, following a series of mathematical operations, exit as a list of 20 numbers. This list of 20 numbers enters the second layer and may exit as a list of five numbers, and so on. Finally, the output layer will produce just a single number if we're trying to predict a single number (such as the probability that a loan applicant will repay their loan, the probability that an electrocardiogram indicates the presence of heart disease, or the likely value of a home) or multiple numbers (such as the probability that a product review cites a defect/improvement/neither, or the latitude and longitude of a ride-sharing vehicle). These transformed versions of the input produced by each of the intermediate layers are called *representations*. We can also think of a representation as an *encoded* version of the raw input, and the layers that produce these encoded versions can be thought of as *encoders*.

When a neural network is trained to solve a particular prediction problem, it ends up learning two things: how to transform or encode the input data into good representations, and how to predict the desired output from these representations.

These representations/encoders turn out to be very useful for the data-efficient construction of AI models.

New approaches let AI developers transfer models across related problems and pretrain models with unlabeled data.

Representations learned by a neural network for one problem can be used for other problems, because they capture intrinsic aspects of the type of data that is fed to the network.

For example, when a deep neural network is trained to detect whether an image contains one of a thousand categories of different real-world objects (such as people, animals, flowers, furniture, vehicles), the representations learned by the first layer turn out to correspond to simple features like straight lines and gradations of color; the second layer captures edges, corners, and circles; and the third layer builds on the previous layer to represent even more complex shapes (such as a honeycomb or the outline of a human torso), and so on.[4]

This raises an interesting possibility: Perhaps we can reuse representations to solve other problems that depend on the same type of input data. For example, could the representations from the neural network described above be used to efficiently build a neural network for any image classification problem where the inputs are images of everyday objects?

The answer turns out to be a resounding yes.

If we can access a network trained on the same type of input data as the one that we are working with, we can extract representations from this network and build a simple model with those representations as inputs and our labels as outputs. We won't need precious labeled data to learn good representations anymore, since we're getting good representations "for free." We can use all the labeled data we have just to learn the simple model. And since simple models aren't as data-hungry as more complex models, we can potentially get the job done with much less labeled data.

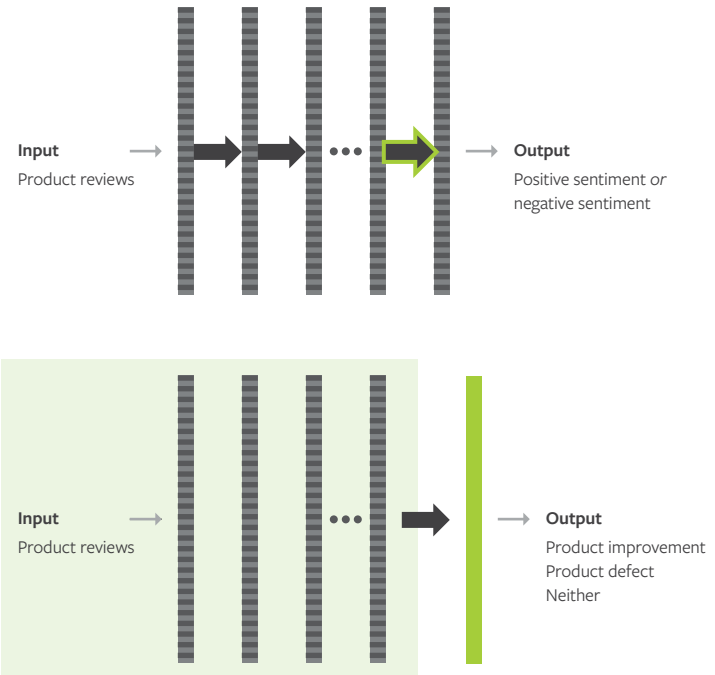
TRANSFER LEARNING: Finding and Repurposing Representations

A good starting point to find useful representations is to see whether other neural network models have been built with the same type of input data. Returning to the home-office furniture example, perhaps another team in the company has built a neural network to predict the sentiment expressed in a product review from the review text. Note that the output labels this neural network was trained to predict are different from your problem: positive sentiment/negative sentiment/neutral rather than improvement/defect/neither.

To perform accurately, this sentiment-detection neural network had to have learned useful patterns present in language and the specific sort of text that occurs in product reviews. This learning will be reflected in the representations generated by its layers when a review is put into this network. In particular, the representation coming out of the penultimate layer is likely to be a useful representation of the input review. (See "Repurposing Neural Networks With Transfer Learning.")

REPURPOSING NEURAL NETWORKS WITH TRANSFER LEARNING

Developers can use a previously built neural network trained with the same type of input data (but different output labels) as the starting point for a new neural network. They can take the original neural network through the second-to-last layer, attach a new output layer, and then train the new network with a different set of output labels.



The neural network from the input layer through the penultimate layer can be viewed as a review encoder. We can extract this encoder from the neural network, attach a new output layer to it to form a new neural network, and train this network with the output labels for our problem.

When training this network, we can learn just the parameters of the output layer we attached (keeping the parameters of the encoder fixed), or we can update *all* of the parameters. This parameter-updating step is called *fine-tuning*, and the overall process we have described is called *transfer learning*.

Transfer learning can enable the building of good AI models with significantly less labeled data. In a problem involving the classification of images into one of 10 categories, extracting an encoder from another neural network and fine-tuning it on *just 50 labeled examples* yielded a model that was more accurate than a network trained from scratch on 5,000 examples.[5]

SELF-SUPERVISED LEARNING: Learning Representations From Unlabeled Data

While transfer learning can be very effective, its starting point is a model that has been trained on similar inputs. But what if such a pretrained model isn't available? Can we build an encoder (that generates good representations) by ourselves, using only unlabeled data?

Indeed, this is possible using a technique called *self-supervised learning*. The key idea behind self-supervised learning is to create an "artificial" data set of inputs and labels from unlabeled data and then train a neural network model on this artificial data set.

While researchers have experimented with many approaches for creating artificial input-label data sets, one of the most effective and widely used techniques is a surprisingly simple two-step process.

In the first step, we take each unlabeled example and randomly remove a small number of its individual variables (in other words, randomly blank out a part of it). What's left is the artificial input. The parts that were removed become the artificial labels.

To see how this approach can be applied to text inputs, consider this unlabeled data point: "The mission of the MIT Sloan School of Management is to develop principled, innovative leaders who improve the world and to generate ideas that advance management practice."

We'll randomly choose three words from this sentence for removal: *mission*, *leaders*, and *generate*. These words become the artificial labels, and what remains of the original sentence is the artificial input: "The ____ of the MIT Sloan School of Management is to develop principled, innovative ____ who improve the world and to _____ ideas that advance management practice." Now imagine doing this on all of your available text data, creating thousands of input-label(s) pairs.

After an artificial data set has been generated as described, in the second step we train a neural network to predict the artificial labels from the artificial inputs. (See "Training Neural Networks With Self-Supervised Learning," p. 52.)

In the process of learning to predict the blanked-out elements of the input from the remaining elements, the neural network learns how these elements are related to one another and thereby learns a good representation of the input data. An encoder that generates these representations can be easily extracted from this neural network and then fine-tuned with precious labeled data (just like in transfer learning).

The process described for creating artificial input-label pairs works well when the input data is natural language text or structured data. For image inputs, a combination of techniques has been found to be very effective: *data augmentation*, along with a self-supervised approach known as *contrastive learning*, in which a neural network is trained so that slightly altered copies of an image have similar representations while distinct images do not.[6]

Building an encoder with self-supervised/contrastive learning followed by fine-tuning can enable the building of good AI models with significantly less labeled data. In a study involving the well-known ImageNet image data set, building an encoder with unlabeled images using contrastive learning and fine-tuning it on just 1% of available labeled examples yielded a model that was more accurate than a model trained on *all* labeled examples.[7]

We've recommended a two-step process to build models with less labeled data: Download a pretrained encoder, or build your own using unlabeled data, and then fine-tune the encoder with your scarce labeled data.

Finding an appropriate encoder or building your own has been made significantly easier by the emergence of model hubs (aka *model zoos*). Model hubs are online venues where AI researchers and practitioners worldwide make available their trained neural network models and code. Three popular model hubs are TensorFlow, PyTorch, and Hugging Face. By leveraging model hubs, it's often possible to build data-efficient AI models with significantly less effort.

Finding an appropriate encoder is now easier thanks to online model hubs where AI researchers and practitioners share trained neural network models and code.

DATA-CENTRIC AI: Iterate on the Data, Not Just on the Model

We have seen that starting with a pretrained model can reduce the need for labeled data. We now turn to the

question of what to do when our pretrained model has been fine-tuned with precious labeled data but its performance is still not good enough for our needs.

When faced with this situation, AI/machine learning teams typically *iterate on the model* — a trial-and-error process that involves generating new input variables by combining existing variables (known as *feature engineering*), trying various alterations to the model's structure and other design parameters, and evaluating the resulting performance on test data.

While this is a reasonable strategy, there's a growing realization that modern neural network architectures are already quite powerful and there's not much headroom to improve on them. It might be more effective to get a good pretrained model (as described earlier) and iterate on the data instead — an emerging practice called *data-centric AI*.

The practice of data-centric AI starts by focusing on inconsistent labels. In many real-world problems, the right label for an example might be subject to interpretation. To take a simple example, imagine that you're building a model to classify images of bananas into three categories: unripe, ripe, or rotten. Since there's no unambiguous way to define ripeness, the same image could be classified as unripe or ripe by different labelers.[8] For a more serious example, consider labeling an image of an industrial part as having a defect or not. This can be challenging for "borderline" cases, and even expert labelers might disagree on some examples.[9]

Inconsistent labels are problematic because today's highly flexible AI models may go out of their way to accommodate them and thereby overfit the training data. As a result, such models tend to do poorly when they're deployed in production. How can this issue be mitigated? Getting more labeled data can help, since more data can average out the damage caused by inconsistent labels. But many more labeled examples might be needed.

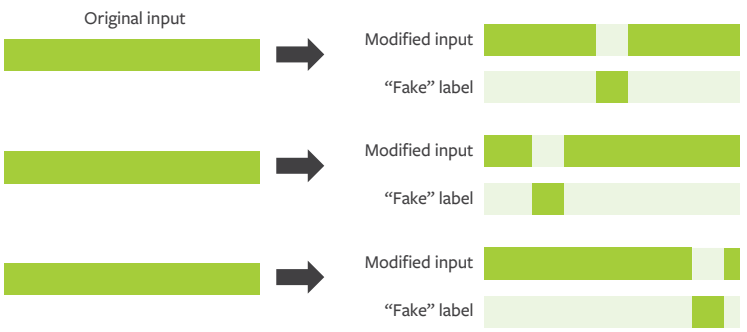
There's a better option: Fix the labels! This can be much more cost-effective than collecting more labeled data. In one scenario where about 12% of 500 training examples had inconsistent labels, the accuracy was below 50%. Fixing the labels without changing the model increased the accuracy to 60%. But to get to this accuracy level without fixing the labels, another 1,000 examples would have been needed.[10]

Fixing inconsistent labels and retraining the model will typically boost performance, but if we want to improve the model even further, collecting more labeled examples will be necessary. However, the number of additional examples that need to be collected can be reduced by being very deliberate in what we choose to collect and label.

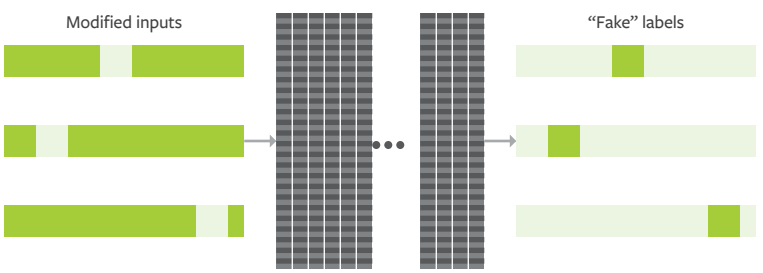
TRAINING NEURAL NETWORKS WITH SELF-SUPERVISED LEARNING

The first step of self-supervised learning is to modify the original input data to create fake input-label pairs by randomly selecting a part of the input and making it the label. In the second step, the neural network is trained to predict the fake labels from the modified inputs — that is, to fill in the blanks.

1. We modify the original input data to create "fake" (input, label) pairs by selecting a part of the input and making it the label.



2. We then train a deep neural network to predict the "fake" labels from the modified inputs, i.e., to fill in the blanks.



Typically, models don't commit errors uniformly across the entire range of input data. Their error rates might be a lot higher for certain regions of the input data compared with other regions. By collecting more labeled data from those error regions rather than from across the board, and retraining the model with this additional data, improvements in model performance can be achieved more efficiently.

Analyzing the model's errors can yield valuable clues to what the error regions might be. This involves identifying commonalities across data points where the model makes errors. For example, an AI team working on a speech recognition model realized that the model's error rate was a lot higher for speech input where car noises were present. The team collected more speech data with car noises and retrained the model, and its performance improved.[11]

Similarly, researchers working on a model to detect metastatic breast cancer from pathology slides realized

that a significant fraction of errors were the result of false-positive classification based on cancer look-alikes in difficult negative regions in the input slides. By generating additional examples from those regions, they were able to improve the model.[12]

Practitioners have reported promising results from adopting a data-centric approach. In a study of three problems that arose in an industrial setting, a data-centric approach was able to achieve gains where a purely model-centric approach could not.[13]

Another benefit of the data-centric approach is that it opens the door for domain experts who are not well versed in AI/machine learning to contribute to a project. They are likely to be very familiar with the data and can play a central role in setting up a robust labeling process, in fixing inconsistent labels, and in identifying "error regions" to source more labeled data from. After the model is deployed, they can help monitor the model to ensure that it continues to be effective and can collect new training data to retrain the model as the surrounding environment changes. These activities can be streamlined and made efficient with special-purpose software and workflows optimized for data-familiar end users who are not machine learning experts.

Bringing Data-Efficient AI Practice Into the Organization

Armed with a high-level understanding of possible solutions, managers can begin exploring them with their AI project leads, guided by the following questions:

- What's the starting point for model development? Is it a pretrained model, or are we building one from scratch?
- If it's the former, how did we go about selecting the pretrained model? How do we know it's appropriate for the types of input data in this project?
- If it's the latter, what are the reasons for not starting with a pretrained model? Does it stem from the unique or proprietary nature of the input data? If so, did we try to build our own encoder using self-supervised or contrastive learning? If not, why not?
- What's the source of the labels in the training and test data? How are we detecting and mitigating the presence of errors in the labels?
- What's our process for analyzing modeling errors and identifying the next most useful set of data to collect labels for?
- For label error detection and model error analysis, is the work being done manually? Is there an opportunity to make this workflow more efficient by introducing software tools?

THE DATA-EFFICIENT AI TECHNIQUES AND best practices I have described can help senior leaders who are championing the development and use of AI applications in their organizations but are running up against the challenge of insufficient labeled data. By ensuring that AI teams are exploring all options, leaders can increase the likelihood of success, improve return on effort, and shorten time to value for their strategic AI projects. ■

Rama Ramakrishnan *is Professor of the Practice at the MIT Sloan School of Management.*

REFERENCES

1. M. Bernstein, "Labeling and Crowdsourcing," Data-Centric AI, accessed June 13, 2022, https://datacentricai.org.
2. N. Diamant, E. Reinertsen, S. Song, et al., "Patient Contrastive Learning: A Performant, Expressive, and Practical Approach to Electrocardiogram Modeling," PLOS Computational Biology 18, no. 2 (Feb. 14, 2022): 1-16.
3. S. Brown, "Why It's Time for 'Data-Centric Artificial Intelligence,'" MIT Sloan School of Management, June 7, 2022, https://mitsloan.mit.edu.
4. M.D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," in "Computer Vision — ECCV 2014," eds. D. Fleet, T. Pajdla, B. Schiele, et al. (Zurich: Springer, 2014), 824.
5. A. Kolesnikov, L. Beyer, X. Zhai, et al., "Big Transfer (BiT): General Visual Representation Learning," arXiv, May 5, 2020, https://arxiv.org.
6. T. Chen, S. Kornblith, M. Norouzi, et al., "A Simple Framework for Contrastive Learning of Visual Representations," arXiv, Feb. 13, 2020, http://arxiv.org.
7. Ibid.
8. F. Chollet, "Deep Learning With Python," 2nd ed. (Shelter Island, New York: Manning Publications, 2021).
9. A. Ng, "MLOps: From Model-Centric to Data-Centric AI," PDF file (Palo Alto, California: DeepLearning.AI, June 2021), www.deeplearning.ai.
10. Chollet, "Deep Learning With Python."
11. DeepLearning.AI, "A Chat With Andrew: MLOps: From Model-Centric to Data-Centric AI," YouTube video, 1:00:10, March 4, 2021, www.youtube.com.
12. D. Wang, A. Khosla, R. Gargeya, et al., "Deep Learning for Identifying Metastatic Breast Cancer," arXiv, June 18, 2016, https://arxiv.org.
13. Chollet, "Deep Learning With Python."

Reprint 64202. For ordering information, see page 4.

MITSloan
Management Review

PDFs • Reprints • Permission to Copy • Back Issues

MIT Sloan Management Review articles, permissions, and back issues can be purchased on our website, **shop.sloanreview.mit.edu**, or you may order through our Business Service Center (9 a.m. - 5 p.m. ET) at the phone number listed below.