Flipper: A Systematic Approach to Debugging Training Sets

Paroma Varma, Dan Iter, Christopher De Sa, Christopher Ré
Stanford University
Stanford, California 94305
paroma,daniter,cdesa@stanford.edu,chrismre@cs.stanford.edu

ABSTRACT

As machine learning methods gain popularity across different fields, acquiring labeled training datasets has become the primary bottleneck in the machine learning pipeline. Recently, *generative models* have been used to create and label large amounts of training data, albeit noisily. The output of these generative models is then used to train a discriminative model of choice, such as logistic regression or a complex neural network. However, any errors in the generative model can propagate to the subsequent model being trained. Unfortunately, these generative models are not easily interpretable and are therefore difficult to debug for users. To address this, we present our vision for Flipper, a framework that presents users with high-level information about why their training set is inaccurate and informs their decisions as they improve their generative model manually. We present potential tools within the Flipper framework, inspired by observing biomedical experts working with generative models, which allow users to analyze the errors in their training data in a systematic fashion. Finally, we discuss a prototype of Flipper and report results of a user study where users create a training set for a classification task and improve the discriminative model's accuracy by 2.4 points in less than an hour with feedback from Flipper.

ACM Reference format:
Paroma Varma, Dan Iter, Christopher De Sa, Christopher Ré. 2017. Flipper: A Systematic Approach to Debugging Training Sets. In *Proceedings of HILDA'17, Chicago, IL, USA, May 14, 2017,* 5 pages.
DOI: http://dx.doi.org/10.1145/3077257.3077263

1 INTRODUCTION

Machine learning has become a ubiquitous tool across a variety of fields. This has partly been fueled by the success of methods based on complex neural networks such as convolutional neural networks (CNNs) [8] and long short-term memory networks (LSTMs) [5] that have alleviated the need for manual feature engineering. These methods in turn rely on the availability of large, labeled training sets, such as ImageNet [2], CallHome corpus [12] and MNIST [10]. However, only a few groups have the computational and human resources available to *generate* this magnitude of labeled data. For most real-world experiments, obtaining hand-labeled training data is an expensive process that requires time as well as domain knowledge.

To address this issue, recent approaches have utilized generative models to create and label large amounts of data efficiently. Generative adversarial networks (GANs) have been used to generate data samples indistinguishable from real data, especially for images [4]. Another approach uses a probabilistic graphical model that utilizes a collection of noisy labels and a few clean labels to train an end-to-end deep learning system for image classification [17]. Data programming is another method that has been applied to noisily label text data by modeling heuristics, which include weak and distant supervision sources, as a generative process [13]. The imperfect training sets that the generative models output can then be used to train a discriminative model of choice, allowing users to apply a wide range of machine learning models to their data.

However, debugging these generative models can be challenging for users who have to scan through individual training points that are mislabeled or not well-defined by their heuristics. To address this issue, we introduced Socratic learning [14], which automatically identifies information that could be used to improve the generative model and updates the model accordingly. We observed domain experts in the biomedical area who worked with Socratic learning and noticed that if users could interpret *why* the generative model was failing, they could make manual adjustments to the model that would improve its accuracy beyond the automated correction.

We present our vision for Flipper, a framework that focuses on improving noisy training data by presenting a human in the loop with information about subsets of the training data that are inaccurate. Flipper uses information that Socratic learning identifies and packages it using textual and visual explanations, which are easily understood by users. It draws upon existing work that generates explanations for machine learning model predictions [11, 19] and utilizes them to instead explain discrepancies in the training set. We describe some such methods in this paper and report a user study where the discriminative model performance improves by 2.4 points over Socratic learning with Flipper.

Our main contribution in this paper is Flipper, a framework for users to receive the information about how the generative model is lacking in a comprehensible form and help them systematically eliminate deficiencies in the training set creation process. Users can therefore debug their training set labels efficiently while avoiding the guessing-and-checking involved in inspecting individual data points to adjust their heuristics. In the next sections, we briefly describe data programming, which uses a generative model to label data noisily, and Socratic learning, which automatically improves generative models (Section 2). We then describe the Flipper workflow and describe how Flipper can work with textual and image data and with methods like crowdsourcing (Section 3). Finally, we

HILDA'17, Chicago, IL, USA

DOI: http://dx.doi.org/10.1145/3077257.3077263

demonstrate specifics of the Flipper prototype with a user study in Section 4.

2 BACKGROUND: DATA PROGRAMMING AND SOCRATIC LEARNING

We explain data programming, a specific technique that uses a generative model to create a noisy training set that a discriminative model can then train on. The data programming paradigm combines multiple sources of weak supervision by allowing users to programmatically encode these heuristics as labeling functions, as shown in Figure 1(a) and (d). These labeling functions are then used to assign multiple labels to the same set of data, and data programming uses the disagreements among these labels to learn an accuracy parameter for each of the labeling functions. Using a weighted average, each data point in the training set receives a probabilistic label. This noisy training set can then be used to train any noise-aware discriminative model – a model that accepts and optimizes over a set of probabilistic instead of binary labels. This process of writing and refining labeling functions iteratively and training a machine learning model is encapsulated in Snorkel [3], an interactive environment for users to utilize data programming.

We observed that some labeling functions users wrote were implicitly designed for a subset of the dataset, but these subsets were difficult for users to recognize merely via inspection. We proposed Socratic learning to address the above issue, a paradigm that automatically identifies subsets in the data that some heuristics are specialized for and incorporates this information in the generative process. In order to recognize these subsets, Socratic learning relies on the disagreement between the training set labels and the predicted labels from the trained discriminative model, and finds specific features of the data most correlated with this disagreement. With access to these features, the updated generative model outputs a more accurate training set, which in turn leads to a better discriminative model.

However, in our experience working with domain experts using data programming and Socratic learning for real-world experiments, we noticed that users could manually refine their labeling functions based on the features Socratic learning recognized. We propose Flipper, a framework that allows users to better understand why their generative process to create a training set has errors and provides users information about how to debug their training sets systematically.

3 FLIPPER FRAMEWORK

The vision for Flipper is inspired by the study of Explainable AI models [1], where performance and interpretability are not competing forces. Flipper allows users to understand the features that Socratic learning uses to automatically debug the training set generation process. Since the feedback process is transparent for the users, they have control over how additional corrections can be made to the training set, leading to improvements above what Socratic learning can provide. Users are given the opportunity to *identify* why their training set is being composed poorly as well as *refine* their heuristics to address the issues.

3.1 Machine Learning Pipeline with Flipper

We outline the steps a user goes through while developing a machine learning task and show where Flipper fits into the pipeline.

(a) **Initial Labeling and Training:** Users obtain or create noisily labeled data, whether via labeling functions in the Snorkel framework, by collecting crowdsourced labels, or by aggregating labels from a variety of sources. This imperfectly labeled data is then used to train and evaluate a machine learning model, as shown in Figure 1(a).

(b) **Automated Debugging:** Next, Socratic learning is used to automatically improve the generative process. This improvement is based on specific features, as shown in Figure 1(b), extracted from the trained discriminative model.

(c) **Flipper:** The automated improvement is usually not enough to correct all the errors in the training set. Flipper provides users with various tools that allow them to recognize a common theme that explains the errors in their training set labels. These tools and their possible uses are detailed in later in this section.

(d) **Refining Labeling Process and Retraining:** With information from the previous steps, users have the option to refine their labeling process, as shown in Figure 1(d). This step can be manual or partially automated. After the generative model is refined, a new training set is generated that the discriminative model can retrain on.

3.2 Flipper Tools

We outline possible methods that users can run using Flipper that allow them to further understand the errors in the training set. The first three methods describe how Flipper could help find an interpretable explanation for users to understand where their training data has errors. The last two provide a vision for how the training set generation process can be partially automated or made more efficient by utilizing the information from the previous methods.

Low-Dimensional Data Representation. As described in Section 2, Socratic learning uses features of the data to automatically correct the generative model. While these features usually represent some aspect of the data, we have observed that users can manually refine the generative model significantly if they understand what these features represent. In the case of text data, users can use techniques like latent semantic analysis (LSA) [9] to recognize a set of words or topics closely related to the one the feature from Socratic learning represents. Given the word or phrase that Socratic learning identifies, LSA provides users a list of similar words or phrases, which can be manually analyzed in order to recognize what common theme the feature is representative of.

A specific example comes from an experiment we conducted with Socratic learning, where a group of biomedical experts wrote labeling functions in the Snorkel environment to identify mentions of diseases in PubMed abstracts [16]. After Socratic learning identified a feature related to the phrase "induction of", users ran LSA on the phrase and saw that the most relevant phrase was "induction [of] anesthesia" – a common phrase that appears in PubMed abstracts. Domain experts pointed out that the phrase "induction of" usually appears with non-disease names, such as "anesthesia" and "labor"

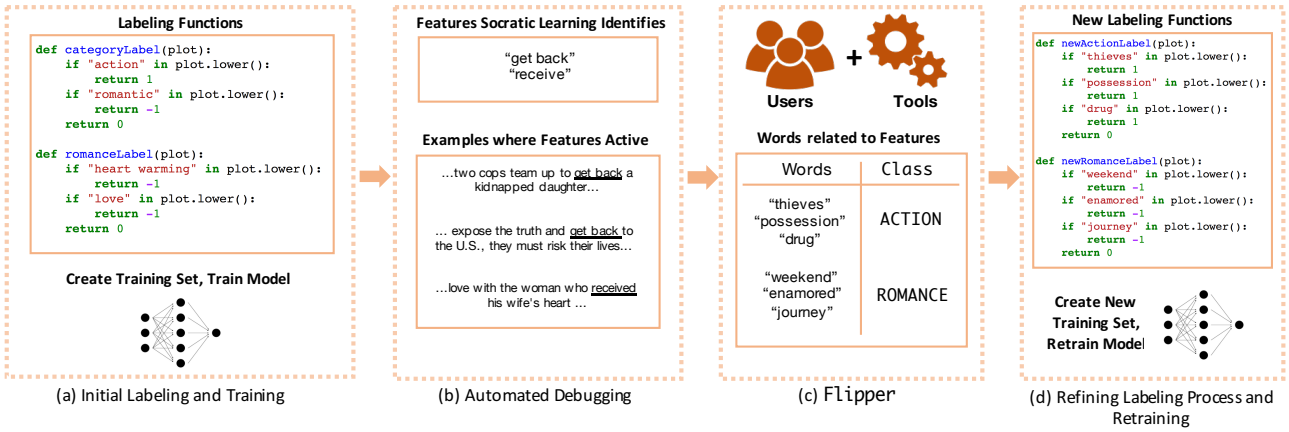

Figure 1: The machine learning pipeline with Flipper, further described in Section 3.1. The task, as described in Section 4, is to categorize movie plots as belonging to the Action or Romance genre. (a),(b) relate to the initial labeling, training and automated debugging processes. (c) shows how Flipper can help users identify key words they missed while writing the labeling functions in (a). (d) shows the new labeling functions that users wrote using the information from Flipper.

and is therefore a good predictor of whether the word that follows the phrase is indeed a disease or not. Adding another heuristic that marked the word that appeared after said phrase as a non-disease led to a 1.15 F1 point improvement over what automated Socratic learning had provided. Note that this improvement is measured on the final discriminative model, the downstream model whose accuracy increases as a direct result of a better generative model and training set.

Visual Explanations for Image Data. While features for textual data are interpretable and can be analyzed through techniques like LSA, image features are more difficult to interpret and find common themes for. In this case, Flipper could draw on two existing techniques for explaining model prediction and instead use it to explain errors in the training set.

The first tool Flipper can work with is Lime [11], which generates locally faithful explanations for model predictions. Extending this approach, we can use Lime in order to explain why the discriminative and generative models disagree – that is, why the training set labels and the predictions do not match. Since Lime allows users to visualize which specific patches in an image correspond to the predictions, it can help users understand what parts of the image are being missrepresented in their training set.

Image classification tasks are usually conducted using CNNs, where the features are automatically generated by the neural network. Recent work [19] allows users to visualize what the features in each layer of the neural network represent. These can range from specific colors, textures, and pose variations within the images. Flipper can determine which features (or layer of features) best explains the discrepancies between the training set labels and the labels assigned by the trained CNN. Drawing upon these techniques, Flipper can provide users a visual map of what characteristics in the images cause mislabeled images in the training set.

Textual Explanations for Image Data. Captioning images automatically [6, 15, 18] is a popular challenge in machine learning, which Flipper can draw upon to generate short explanations for why the training set is inaccurate. If Socratic learning identifies a feature that relates to errors in the generative process, these features can be used to cluster image data. Moreover, automated captioning techniques can be used to generate noisy captions for different clusters of images. Flipper can use simple techniques like listing the most common words in the captions for each cluster, which can allow users to determine what common theme or object explains errors in their training set.

We used a similar technique in a binary image classification task conducted with Socratic learning, where we wrote labeling functions to identify images as containing "Humans" vs. "Objects". Roughly analyzing the captions for the mislabeled images, we found that the training set was mislabeling images with groups of people. This resulted from labeling functions that only marked captions with words like "man", "woman" and "person" as Human, without accounting for the plural versions of these words. Adding in a labeling function that accounted for this led to a 1.28 point increase in accuracy for AlexNet [7], the discriminative model being trained.

Automated Labeling Function Generation. Currently, users receive feedback in the form of phrases or keywords related to a topic that explains why the training set is inaccurate. However, the burden still remains on the user to further interpret and decide how to include this information in the form of refined or additional labeling functions. This process could also be automated to a certain extent such that Flipper not only provides keywords to the users, but also simple labeling function suggestions that mimic if-then rules for re-labeling existing training data. Once the users view this labeling function, they would have the option to refine it further or add it to the existing set of labeling functions. Ideally, such feedback could be provided in near real-time, making the debugging and labeling process more efficient.

Enhancing Crowdsourced Labeling Process. Flipper can also be used to generate questions for crowdsourcing workers rather than

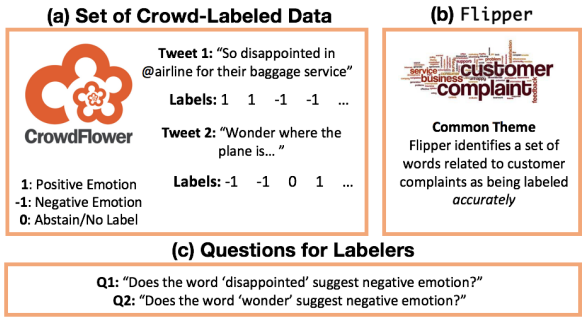

Figure 2: An example of what `Flipper` could do in the crowdsourcing setting. (a) shows the initial, small subset of data labeled via crowdsourcing. (b) shows the common theme `Flipper` recognizes related to where crowd labels were more accurate than average. (c) shows a generated set of questions related to the theme from (b) that workers can answer to automatically label certain tweets.

having them hand-label multiple data points. The motivation is that answering questions takes less time than hand labeling examples and since data programming can account for noisy labels, this saves a significant amount of time for the labelers without sacrificing quality of training set data. An example of such a setting is shown in Figure 2. We used a crowdsourced sentiment analysis task where the task was to classify tweets related to airplanes (Figure 2(a)). Socratic learning identifies phrases related to complaints (Figure 2(b)) and we gather that tweets related to complaints are easier for labelers to accurately classify compared to other tweets. Given such information, we propose that `Flipper` can reduce the amount of data that has to be hand-labeled and replace it with questions about specific phrases, as shown in Figure 2(c). Answering these questions allows a large amount of data to be labeled efficiently, and in most cases, fairly accurately.

4 USER STUDY WITH FLIPPER

To see how `Flipper` would help in a real experimental setting, we describe a simple labeling exercise a group of graduate students completed using `Flipper` and Snorkel. The task at hand is to write rules that determine whether a specific movie belongs to the "Action" or "Romance" genres using the content of the movie's plot. The data is extracted from the Internet Movie Databse (IMDb) and pre-processed to remove any examples that did not belong to either category, or belonged to both. We split this dataset further into train, test and development sets of with 1136, 500, and 284 samples, respectively.

4.1 Initial Labeling and Training

The initial set of labeling functions were not difficult for users to write and took less than an hour. The users were only required to ensure that their labeling functions were better than chance, that is, they had accuracies of more than 50%. Even though we had the ground truth for all data in this particular case, we allowed users to test their labeling function accuracies on the development set.

Example 4.1. A group of three graduate students wrote a set of five labeling functions in the Snorkel framework that checked for certain words in the plot to decide whether the movie belonged to the "Action" or "Romance" genre. Examples of these labeling functions are shown in Figure 1(a). If words like "chase", "criminal" and"attack" are present in a plot summary, they are marked as an "Action" movie and if words like "heart warming" and "love" are in the plot summary, they are marked as "Romance". These labeling functions had accuracies ranging from 66.67% to 93.33% on the data points they did assign labels to. Only 17.6% of the training data had at least one label from one of these labeling functions.

The data programming algorithm was used to generate a noisy training set where each data point was assigned a probabilistic label, which was used to train a noise-aware logistic regression model. The features for the logistic regression model were generated using a bag-of-words representation for the plot summaries, where each binary feature signified the presence or absence of a certain word or phrase in the plot. Note that both the labeling functions and features map to the words and phrases in the plot summaries. However, this is not always the case and the labeling function and features can work over different domains. The trained model had an accuracy of 68.8% when evaluated on a held-out test set that the users or the models never had access to.

4.2 Automated Debugging

Socratic learning relies on features to transfer knowledge from the trained model to the training set creation process. In some cases, the features provided to the machine learning model have a direct correspondence to some aspect of the data being analyzed which makes this feedback interpretable.

Example 4.2. The features that Socratic learning identifies in this case represent the phrases "get back" and "receives" (Figure 1(b)). Socratic learning improves the accuracy of the training set, which in turn improves the accuracy of the retrained model to 69.6%. Since such generic phrases provide users little intuition about why their labeling functions were inaccurate, users are unable to manually edit their labeling functions at this stage.

This user study encapsulates why only having features that map to some aspect of the data might not be enough to make the Socratic learning process truly interpretable. Users are unable to determine what kinds of movies their labeling functions were failing on given these phrases. Therefore, `Flipper` is key to understanding why the training set is inaccurate and how these inaccuracies can be corrected systematically.

4.3 `Flipper`

A highlight of the `Flipper` framework is that it provides users the opportunity to address the shortcomings in their training set by providing tools to analyze the features that Socratic learning identified. For the user study, we used LSA to find meaningful relations among the identified phrases from step (b) and other words present in the database of movie plots being classified.

Example 4.3. With LSA, users discover the words "drugs", "thieves", and "possession" are among the top words associated with the phrase "get back", while "weekend", "enamored", and "journey" are associated with "receive". Since the two categories are "Action" and "Romance", it is easy for users to conclude that the first set of words correspond to action movies and the second to romantic movies.

It is important to highlight that the users had to parse through the list of words LSA provided in order to recognize these patterns. Users agreed that adding these words to their labeling functions would make their generative process more expressive and cover more data points in the training set. Even though it would be easy to add words users *thought* were associated with the two genres, going through this exercise provides more direction in terms of what specific words to include.

4.4 Refining Labeling Process and Retraining

With knowledge of the shortcomings in the labeling functions, the users decide to either refine existing or write new labeling functions. Since each labeling function encodes some useful information, users usually do not remove existing labeling functions.

Example 4.4. Users decide to add two labeling functions shown in Figure 1(d). One marks plot summaries with the words "drugs", "thieves", or "possession" as "Action". The other marks plot summaries with the words 'weekend", "enamored", or "journey" as "Romantic". The first labeling function has an accuracy of 73.5% *while the second has an accuracy of* 90% *on the data points they each label. With these additional labeling functions, the coverage increases to* 24.5% *of the training set.*

Users regenerate the training set with after adding these new labeling functions within Snorkel and retrain the same model. The training set accuracy increases by 6.8% and the test set accuracy increases by 2.4%, compared to the training set and model in step (b). The final accuracy of the discriminative model was 72% when evaluated on the test set.

5 CONCLUSION

In this paper, we described `Flipper`, a system that provides users interpretable, high-level feedback about debugging their training set. With the growing need for large, labeled training data sets, it has become critical to find alternative ways of hand-labeling data. In such a setting, we surmise that creating training sets programmatically and debugging them in a systematic manner are key components in the overall data analysis pipeline. We continue enhancing `Flipper` via avenues mentioned in this paper while testing its performance on various real-world tasks.

ACKNOWLEDGEMENTS

Thanks to Henry Ehrenberg, Alex Ratner, Stephen Bach, Sen Wu, Jason Fries and Theodoros Rekatsinas for their helpful conversations and feedback. Thanks to Shoumik Palkar, Yuval Gannot and Deepak Narayanan for participating in the user study for `Flipper`.

We gratefully acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) SIMPLEX program under No. N66001-15-C-4043, the National Science Foundation (NSF) CAREER Award under No. IIS- 1353606, the National Science Foundation (NSF) Graduate Research Fellowship under award No. DGE-114747, the Office of Naval Research (ONR) under awards No. N000141210041 and No. N000141310129, the Sloan Research Fellowship, the Moore Foundation, the Okawa Research Grant, the Joseph W. and Hon Mai Goodman Stanford Graduate Fellowship, Toshiba, and Intel. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA, NSF, ONR, or the U.S. government.

REFERENCES

[1] Mark G. Core, H. Chad Lane, Michael van Lent, Dave Gomboc, Steve Solomon, and Milton Rosenberg. 2006. Building Explainable Artificial Intelligence Systems. In *Proceedings of the 18th Conference on Innovative Applications of Artificial Intelligence - Volume 2 (IAAI'06)*. AAAI Press, 1766–1773. http://dl.acm.org/citation.cfm?id=1597122.1597135

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 248–255.

[3] Henry R Ehrenberg, Jaeho Shin, Alexander J Ratner, Jason A Fries, and Christopher Ré. 2016. Data programming with DDLite: putting humans in a different part of the loop.. In *HILDA@ SIGMOD*. 13.

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*. 2672–2680.

[5] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[6] Andrej Karpathy and Li Fei-Fei. 2015. Deep Visual-Semantic Alignments for Generating Image Descriptions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[7] Alex Krizhevsky. 2014. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997* (2014).

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

[9] Thomas K Landauer, Peter W Foltz, and Darrell Laham. 1998. An introduction to latent semantic analysis. *Discourse processes* 25, 2-3 (1998), 259–284.

[10] Yann LeCun, Corinna Cortes, and Christopher JC Burges. 1998. The MNIST database of handwritten digits. (1998).

[11] Carlos Guestrin Marco Tlio Ribeiro, Sameer Singh. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *KDD*. 1135–1144.

[12] Matt Post, Gaurav Kumar, Adam Lopez, Damianos Karakos, Chris Callison-Burch, and Sanjeev Khudanpur. 2013. Improved Speech-to-Text Translation with the Fisher and Callhome Spanish–English Speech Translation Corpus. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT)*. Heidelberg, Germany.

[13] Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data Programming: Creating Large Training Sets, Quickly. In *Advances in Neural Information Processing Systems*. 3567–3575.

[14] Paroma Varma, Bryan He, Dan Iter, Peng Xu, Rose Yu, Christopher De Sa, and Christopher Ré. 2017. Socratic Learning: Correcting Misspecified Generative Models using Discriminative Models. *arXiv preprint arXiv:1610.08123* (2017).

[15] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3156–3164.

[16] Chih-Hsuan Wei, Yifan Peng, Robert Leaman, Allan Peter Davis, Carolyn J Mattingly, Jiao Li, Thomas C Wiegers, and Zhiyong Lu. 2015. Overview of the BioCreative V chemical disease relation (CDR) task. In *Proceedings of the fifth BioCreative challenge evaluation workshop*. 154–166.

[17] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. 2015. Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2691–2699.

[18] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention.. In *ICML*, Vol. 14. 77–81.

[19] Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*. Springer, 818–833.