Data Attribution for Segmentation Models

Anonymous Author(s)
Affiliation
Address
email

Abstract

The quality of segmentation models is driven by their training datasets labeled with detailed segmentation masks. How does the composition of such a training dataset contribute to the performance of the resulting segmentation model? In this work, we take a step towards attaining such an understanding by applying the lens of data attribution to it. To this end, We first identify specific behaviors of these models to attribute, and then provide a method for computing such attributions efficiently. We validate the resulting attributions, and leverage them to both identify harmful labeling errors and curate a 50% subset of the MS COCO training dataset that leads to a $2.79\% \pm 0.49\%$ increase in mIOU over the full dataset.

1 Introduction

Semantic segmentation is a fundamental task in computer vision. Indeed, segmentation models classify individual pixels in an image, enabling a detailed understanding of complex scenes with diverse applications such as autonomous driving [1, 2], medical image analysis [3, 4], and agricultural field monitoring [5]. However, the quality of segmentation models is impacted by the difficulty of collecting high-quality training data [6].

In particular, the process of manually creating the whole segmentation masks for an individual image is far more expensive than assigning a single label to it (as is done in the classification setting). Because of this complexity of creating pixel-wise annotations, the resulting masks often contain labeling errors [7, 8]. To what extent are these label errors detrimental to the performance of segmentation models though? And can we identify and remove problematic training examples in order to curate better training datasets?

More broadly, we might wonder about the relationship between training data and the resulting behavior of segmentation models. For example, we might want to understand the possible pitfalls of training on synthetic data [9, 10], an increasingly popular alternative to manually annotated data.

Our Contributions To answer such questions, in this work we study *data attribution* for segmentation models—that is, the task of tracing such models' behavior to individual training examples.

Performing data attribution in the segmentation setting requires addressing a number of complications. First, a segmentation model might aim to segment multiple distinct objects that can be of different classes, making it less clear how to choose a specific target of attribution (i.e., *what* to attribute). To this end, we begin by identifying specific behaviors (i.e., parts of the output) of segmentation models that we wish to attribute to the training dataset. Additionally, since the output of segmentation models is high-dimensional, computing gradients of this model output with respect to training inputs (a key step in most data attribution methods) might be difficult. However, by leveraging recent work in data attribution within classification settings [11], we provide an efficient method for computing data attributions for segmentation models, and validate the faithfulness of the resulting attributions.

Finally, we demonstrate that the attributions identified by our method surface harmful labeling errors in segmentation datasets, and leverage these attributions to curate a 50% subset of the MS COCO

Figure 1: ***What* do we want to attribute?** In the segmentation setting, there are many possible targets for attribution. Here, we visualize three possible targets of interest: a model's predictions for a particular class (left), a specific object (middle), or separation of background from foreground (right).

training dataset that achieves a $2.79\% \pm 0.49\%$ increase in mIOU over the full training dataset. Related works are in Appendix A and a conclusion is in Appendix B.

2 Preliminaries

Data Attribution Data attribution has been closely studied for classification models, and within this setting, previous works have employed data attribution for a number of tasks that can provide great value in the segmentation setting. For example, mislabeled or otherwise poisoned training examples can by identified by examining negatively influential training examples [12–17]. More broadly, evaluating the overall impact of individual training examples can be useful for curating training datasets [13, 14, 18]. Data attribution can also be useful for debugging model behavior [19] and comparing learning algorithms [20]. Below, we formalize the task of data attribution.

Consider a learning algorithm $\mathcal{A}$ (e.g., the training process for a neural network) paired with an n-element training dataset $S \in \mathcal{Z}^n$ from input space $\mathcal{Z}$. Broadly, data attribution aims at identifying how the behavior of models trained with algorithm $\mathcal{A}$ is impacted by each training point $z_i \in S$. In particular, given some held-out example $z \in \mathcal{Z}$, we can quantify the behavior of the model on this example via a *model output function* $f(z, \theta(S)) : \mathcal{Z} \times \mathbb{R}^d \to \mathbb{R}$, where $\theta(S) \in \mathbb{R}^d$ denotes the model parameters resulting from running the algorithm $\mathcal{A}$ on the dataset $\mathcal{S}$.

A valuable primitive that captures many of the underlying goals of data attribution is *datamodeling* [19]: the task of predicting the model output function $f(z, \theta(S'))$ that results from running algorithm $\mathcal{A}$ on arbitrary subsets $S' \subset S$. In fact, despite the complex dynamics of modern non-convex neural networks, prior works [19, 15, 11] have demonstrated that this prediction task can be well approximated by a *linear* mapping $\{0,1\}^{|S|} \to \mathcal{R}$. So, following Park et al. [11], we can formalize a *data attribution method* as a function $\tau \colon \mathcal{Z} \times \mathcal{Z}^n \to \mathbb{R}^n$ that assigns a score $\tau(z, S)_i \in \mathbb{R}$ to each training example $z_i \in S$, indicating the change in $f(z, \theta(S))$ induced by removing z_i from S. In particular, we can interpret these scores as weights of a *linear datamodel*: $f(z, \theta(S')) \approx \tau(z, S)^T \mathbf{1}_{S'}$ where $\mathbf{1}_{S'}$, is the indicator vector of S' in S.

Segmentation Models We focus on the task of *semantic segmentation*, in which each pixel in an image is assigned a class. Each training example $z \in \mathcal{Z}$ consists of an image x of size $H \times W \times N_{ch}$, where N_{ch} is the number of channels, and a label y of size $H \times W$. Each pixel x_{ij} has a label $y_{ij} \in \{0, 1, \ldots, N_{cls}\}$, where N_{cls} is the number of classes (and 0 represents the background class).

3 Attributing Segmentation Models

In order to perform data attribution for segmentation models, we first need to identify a specific target of attribution (and pair it with a model output function that formalizes this attribution target). So, we begin this section by identifying two such targets, towards which we will focus our attributions. Then, we introduce a method for computing such attributions efficiently by leveraging TRAK [11], a recent work in data attribution for supervised models.

3.1 *What* do we want to attribute?

Classification models aim to predict a single discrete value (the class label). So, in this setting, the target of attribution is generally the correctness of the model in predicting this label. To represent

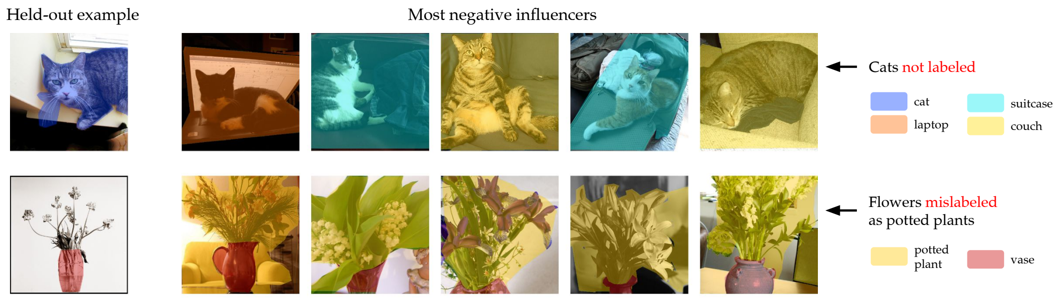

Figure 2: **Investigating negative influencers uncovers labeling errors.** For two MS COCO validation examples, we display the five most negatively influential training images according to our attribution method. Note that these negative influencers reflect consistent mislabeling errors: in the top row, each training example includes a cat that is not labeled, and in the bottom row, the flowers in each image are mislabeled as potted plants. See Appendix D for further examples.

correctness, a common choice for the model output function is the *correct-class margin* [19, 11], that is, the difference between the logit for the correct class and the highest incorrect logit.

For segmentation models, however, the question of *what* to attribute is less clear. In particular, segmentation models output a high-dimensional predicted mask with a separate prediction for each pixel of the input image. This output mask for a single image often predicts the presence of *multiple* objects, which may be of different classes. Thus, given a specific image of interest, there are a number of possible questions we might want to ask about the behavior of segmentation models on this image.

For instance, consider the image (and corresponding ground-truth masks) in Figure 1. Beyond identifying the training examples that impact the model's overall ability to segment the objects in this image, there might be a number of more fine-grained questions we might want to ask in order to *isolate* attributions to specific "subtasks" of the segmentation task. For example, we might be interested in identifying the training examples that impact the model's ability to segment specifically the cats in this image, or the model's ability to generally separate the background from foreground objects. Clearly, there are many possible such subtasks of interest. In this work, we limit our focus to the following two possible targets of attribution:

1. **Full-image Attribution:** As a natural initial choice, we can directly attribute with respect to the full segmentation of a given image of interest. Specifically, we attribute the average of the per-pixel class predictions (each of which is its own classification task).
2. **Class-Specific Attribution:** As one avenue of identifying more fine-grained attributions, we additionally consider attributing the model's ability to segment a specific class. Specifically, given a class c we attribute the per-pixel binary classification task of identifying whether each pixel belongs to class c.

3.2 Adapting TRAK to Segmentation Models

One common strategy for assigning attribution scores $\tau(z)_i$ for a given input z is to compute the *leave-one-out* influence of removing that particular example on the model output function:

$$\tau(z)_i = f(z, \theta(S)) - f(z, \theta(S \setminus \{z_i\})).$$

In the linear regression setting, we can compute this influence directly, as there exists a closed form solution for the parameters $\theta(S')$ given a training subset $S' \subset S$. However, estimating the resulting model parameters when leaving out an example is difficult in the non-convex deep learning setting.

To overcome this difficulty, TRAK [11] first *linearize* the model and then apply classical methods for data attribution in the linear setting [21]. In particular, recent work in studying the neural tangent kernel (NTK) has shown that linearizing neural networks with their first-order Taylor expansion can closely approximate training dynamics [22–24]. This expansion allows us to view the model output function $f(z, \theta(S))$ as a linear model acting on inputs $\nabla_\theta f(z, \theta(S))$ (see Section 3.2 of Park et al. [11] for a more detailed explanation). Now, applying TRAK to a given attribution task requires two steps: (1) computing gradients of the training examples with respect to the training loss, and (2)

Figure 3: **Examples of positive influencers.** For two MS COCO validation examples, we show the most positive influencers identified by our method, along with the most similar training examples according to CLIP similarity.

Figure 4: **Per-class positive influencers.** An example MS COCO validation example that includes segmentations for the *person*, *car*, and *traffic light* classes, as well as the top two most positively influential training images according to class-specific TRAK scores for each of these classes.

computing gradients of each held-out example of interest with respect to the model output functions (see Section 3.4 of Park et al. [11] for more details).

At first glance, it may appear that since the output of a segmentation model is high-dimensional (with a multi-class prediction for each pixel in the image), computing the gradient with respect to each training example (the first step) might require tracking a large number of independent gradients. However, note that each training example impacts the final model parameters *only* through its contribution to the gradient of the training loss. Thus, we only need to compute the gradient of each training example with respect to this loss. Finally, to compute the second step, below we define a model output function for each of the two attribution targets introduced in Section 3.1.

Full-image Attribution In the classification setting, a common choice of model output function (as we introduced in Section 3.1), is the *correct-class margin* between the correct-class logit and the largest incorrect logit [19]. However, prior theoretical work [25] motivates the use of the model output function

$$f(z;\theta) = \log\left(\frac{p(z;\theta)}{1-p(z;\theta)}\right),$$

where $p(z;\theta)$ is the correct-class probability; this model output can be viewed as a "soft" version of the correct-class margin.

Since the training loss takes the average of individual pixel-wise cross-entropy losses over all pixels in an image, a natural choice of model output function for the semantic segmentation setting is to adapt the above model output in the same manner. Specifically, in the full-image attribution setting, we calculate the above model output function for each pixel-wise classification problem, then average over pixels as follows:

$$f\left(z;\theta(S)\right) = \frac{1}{HW}\sum_{i=1}^{H}\sum_{j=1}^{W}\log\left(\frac{p(z_{ij};\theta)}{1-p(z_{ij};\theta)}\right), \tag{1}$$

where $p(z_{ij};\theta)$ is the correct-class probability predicted for pixel x_{ij}.

Class-Specific Attribution To attribute a model's segmentation predictions for a specific class c, we treat the segmentation task as a binary classification task with respect to c. That is, we adapt the

Figure 5: **Counterfactual experiments.** We quantify the counterfatual impact of removing the highest scoring training examples and retraining according TRAK scores targeted at either full-image or per-class attribution, or according to CLIP similarity. (Left) We apply this intervention to 15 validation examples and plot the average increase in cross-entropy loss of the segmentation task. (Right) We apply this intervention to 15 pairs of (validation example, class) and plot the average increase in cross-entropy loss over the pixel-wise binary classification task of predicting the presence of that class, with per-class attributions targeted at the this class. Error bars represent standard error.

above model output such that all classes that other than c are treated as the same "not-c" class. Our model output for an example $z = (x, y)$ then becomes:

$$f(z; \theta(S)) = \frac{1}{HW} \sum_{i=1}^{H} \sum_{j=1}^{W} \left(\mathbf{1}\{y_{ij} = c\} \log \left(\frac{p_c(x_{ij}; \theta)}{1 - p_c(x_{ij}; \theta)} \right) + \mathbf{1}\{y_{ij} \neq c\} \log \left(\frac{1 - p_c(x_{ij}; \theta)}{p_c(x_{ij}; \theta)} \right) \right), \tag{2}$$

where $p_c(x_{ij}; \theta)$ refers to the predicted probability that pixel x_{ij} belongs to class c.

4 Experiments

We now evaluate our attribution method on segmentation models trained on the MS COCO dataset [26]. After visually inspecting the computed attributions, we validate their counterfactual significance, and then demonstrate their value for curating training datasets.

Experimental Setup We use DeepLabV3 [27] segmentation models trained on segmentations from the MS COCO dataset [26]. When validating our attribution scores, we use image similarity according to a pretrained CLIP [28] ViT-L/14 [29] model as a baseline.

4.1 Visually Inspecting Our Attributions

In Figures 2 and 3, we visualize the most negative and most positive influencers, respectively, for images from the MS COCO validation set. We find that positive influencers identify semantically similar (or even nearly-duplicated) training examples, while negative influencers often surface consistent labeling errors. In Figure 4, we additionally visualize an example of top positive class-specific influencers. See Appendix D for further examples of attributions identified by our method.

4.2 Validating Our Attributions

To validate our full-image attributions, we measure the counterfactual impact of the following intervention: removing the most positive influencers for a given held-out example, retraining, and then measuring the change in loss between the original and new models. If the attributions are meaningful, we would expect this intervention to cause a significant increase in loss. We repeat this process over 25 randomly selected validation examples and compare to the baseline of removing the most visually similar training examples according to CLIP (see Figure 5, left). Our results suggest that that the full-image TRAK scores indeed identify influencers that are counterfactually significant.

To validate our class-specific attributions, we repeat the above intervention with top influencers identified by either the class-specific attributions fora given class within an image or full-image

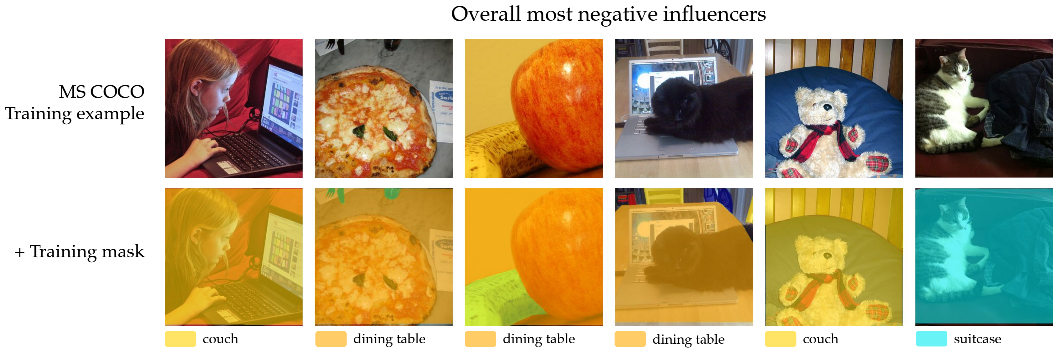

Figure 6: **Overall Most Negative Influencers.** Six of the twenty overall most negatively influential training examples across the full MS COCO validation set, according to our attribution method. Note that *person*, *cat*, *pizza*, *laptop*, *apple*, and *teddy bear* are all MS COCO classes, but labels for these classes are missing in the above training masks.

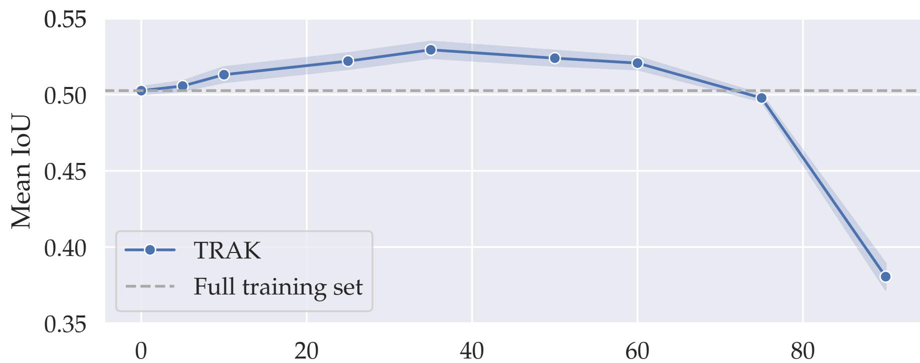

Figure 7: **Dataset curation.** Mean IoU of models trained on MS COCO after removing the most negatively influential examples of the training set according to our attribution scores. The shaded region represents standard error.

attributions, and measure the impact of this intervention on the loss of the binary classification task with respect the corresponding class. We find that when targeting attributions towards a specific class, the impact of the intervention is greater than that of full-image attributions (see Figure 5, right).

4.3 Curating Datasets for Segmentation

In Section 4.1, we saw that the most negative influencers often surface training examples with labeling errors. Intuitively, such training examples should be detrimental to the segmentation model. By removing such problematic examples from the training dataset, can training on the resulting *curated* dataset improve the quality of the resulting model?

As proxy for each individual training image's effect on overall model's performance, we average its attribution scores across the entire validation set. In Figure 6, we visualize samples among the most negative "overall" influencers according to this proxy and find that the surfaced training examples include notable cases of mislabeling, often missing multiple objects within a single image. Now, for various values of N, we remove the bottom $N\%$ of training examples (according to averaged attribution scores) and re-train new models on the resulting smaller, curated dataset. To prevent leakage, we randomly split the validation set into two halves, using one half to calculate average influence scores and the other half to evaluate our re-trained models. We evaluate models via mean intersection-over-union (mIoU) of predicted masks versus ground-truth masks. We find that our curated datasets are able to outperform the full MS COCO training set (see Figure 7), and in particular, we achieve a $2.79\% \pm 0.49\%$ increase in mean IoU with a 50% subset of the training set.

References

[1] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3061–3070, 2015.

[2] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[3] Nathan Moon, Elizabeth Bullitt, Koen Van Leemput, and Guido Gerig. Automatic brain and tumor segmentation. In *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2002: 5th International Conference Tokyo, Japan, September 25–28, 2002 Proceedings, Part I 5*, pages 372–379. Springer, 2002.

[4] Fabian Isensee, Paul F Jaeger, Simon AA Kohl, Jens Petersen, and Klaus H Maier-Hein. nnu-net: a self-configuring method for deep learning-based biomedical image segmentation. *Nature methods*, 18(2): 203–211, 2021.

[5] Furkat Safarov, Kuchkorov Temurbek, Djumanov Jamoljon, Ochilov Temur, Jean Chamberlain Chedjou, Akmalbek Bobomirzaevich Abdusalomov, and Young-Im Cho. Improved agricultural field segmentation in satellite imagery using tl-resunet architecture. *Sensors*, 22(24):9784, 2022.

[6] Şerban Vădineanu, Daniël Maria Pelt, Oleh Dzyubachyk, and Kees Joost Batenburg. An analysis of the impact of annotation errors on the accuracy of deep learning for cell segmentation. In *International Conference on Medical Imaging with Deep Learning*, pages 1251–1267. PMLR, 2022.

[7] Matthias Rottmann and Marco Reese. Automated detection of label errors in semantic segmentation datasets via deep learning and uncertainty quantification. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3214–3223, 2023.

[8] Vedang Lad and Jonas Mueller. Estimating label quality and errors in semantic segmentation data via any model. *arXiv preprint arXiv:2307.05080*, 2023.

[9] Yuhua Chen, Wen Li, Xiaoran Chen, and Luc Van Gool. Learning semantic segmentation from synthetic data: A geometrically guided input-output adaptation approach. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1841–1850, 2019.

[10] Virginia Fernandez, Walter Hugo Lopez Pinaya, Pedro Borges, Petru-Daniel Tudosiu, Mark S Graham, Tom Vercauteren, and M Jorge Cardoso. Can segmentation models be trained with fully synthetically generated data? In *International Workshop on Simulation and Synthesis in Medical Imaging*, pages 79–90. Springer, 2022.

[11] Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. Trak: Attributing model behavior at scale. In *Arxiv preprint arXiv:2303.14186*, 2023.

[12] Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning (ICML)*, 2019.

[13] Rajiv Khanna, Been Kim, Joydeep Ghosh, and Sanmi Koyejo. Interpreting black box predictions using fisher kernels. In *The 22nd International Conference on Artificial Intelligence and Statistics*, 2019.

[14] Ruoxi Jia, Fan Wu, Xuehui Sun, Jiacen Xu, David Dao, Bhavya Kailkhura, Ce Zhang, Bo Li, and Dawn Song. Scalability vs. utility: Do we have to sacrifice one for the other in data importance quantification? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[15] Jinkun Lin, Anqi Zhang, Mathias Lecuyer, Jinyang Li, Aurojit Panda, and Siddhartha Sen. Measuring the effect of training data on deep learning predictions via randomized experiments. *arXiv preprint arXiv:2206.10013*, 2022.

[16] Zayd Hammoudeh and Daniel Lowd. Identifying a training-set attack's target using renormalized influence estimation. In *arXiv preprint arXiv:2201.10055*, 2022.

[17] Shuming Kong, Yanyan Shen, and Linpeng Huang. Resolving training biases via influence-based data relabeling. In *International Conference on Learning Representations (ICLR)*, 2022.

[18] Zhuoming Liu, Hao Ding, Huaping Zhong, Weijia Li, Jifeng Dai, and Conghui He. Influence selection for active learning. In *ICCV*, 2021.

[19] Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Predicting predictions from training data. In *International Conference on Machine Learning (ICML)*, 2022.

[20] Harshay Shah, Sung Min Park, Andrew Ilyas, and Aleksander Madry. Modeldiff: A framework for comparing learning algorithms. In *arXiv preprint arXiv:2211.12491*, 2022.

[21] Daryl Pregibon. Logistic regression diagnostics. In *The Annals of Statistics*, 1981.

[22] Philip M Long. Properties of the after kernel. In *arXiv preprint arXiv:2105.10585*, 2021.

[23] Alexander Wei, Wei Hu, and Jacob Steinhardt. More than a toy: Random matrix models predict how real-world neural representations generalize. In *ICML*, 2022.

[24] Sadhika Malladi, Alexander Wettig, Dingli Yu, Danqi Chen, and Sanjeev Arora. A kernel-based view of language model fine-tuning. In *arXiv preprint arXiv:2210.05643*, 2022.

[25] Nikunj Saunshi, Arushi Gupta, Mark Braverman, and Sanjeev Arora. Understanding influence functions and datamodels via harmonic analysis. 2023.

[26] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision (ECCV)*, 2014.

[27] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 2017.

[28] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *arXiv preprint arXiv:2103.00020*, 2021.

[29] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.

[30] R Dennis Cook and Sanford Weisberg. *Residuals and influence in regression*. New York: Chapman and Hall, 1982.

[31] Frank R Hampel, Elvezio M Ronchetti, Peter J Rousseeuw, and Werner A Stahel. *Robust statistics: the approach based on influence functions*, volume 196. John Wiley & Sons, 2011.

[32] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, 2017.

[33] Pang Wei Koh, Kai-Siang Ang, Hubert HK Teo, and Percy Liang. On the accuracy of influence functions for measuring group effects. In *Neural Information Processing Systems (NeurIPS)*, 2019.

[34] Andrea Schioppa, Polina Zablotskaia, David Vilar, and Artem Sokolov. Scaling up influence functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8179–8186, 2022.

[35] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J. Spanos. Towards efficient data valuation based on the shapley value. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, 2019.

[36] Bojan Karlaš, David Dao, Matteo Interlandi, Bo Li, Sebastian Schelter, Wentao Wu, and Ce Zhang. Data debugging with shapley importance over end-to-end machine learning pipelines. *arXiv preprint arXiv:2204.11131*, 2022.

[37] Chih-Kuan Yeh, Joon Sik Kim, Ian E. H. Yen, and Pradeep Ravikumar. Representer point selection for explaining deep neural networks. In *Neural Information Processing Systems (NeurIPS)*, 2018.

[38] Garima Pruthi, Frederick Liu, Mukund Sundararajan, and Satyen Kale. Estimating training data influence by tracing gradient descent. In *Neural Information Processing Systems (NeurIPS)*, 2020.

A Related Work

Data Attribution The task of data attribution has been studied closely, with early work focusing on the linear regression setting [30, 21]. In the modern machine learning settings, there is a long line of work in *influence functions* [31–34], as well as works that perform data attribution through Shapley values [12, 35, 36] or other heuristic approaches [37, 38]. Recently, Park et al. [11] proposed a method that significantly improves upon tradeoffs between computational efficiency and accuracy by leveraging the empirical kernel structure of neural networks (see Section 3 for further discuss on this work).

Detecting Labeling Errors in Segmentation Models. Recent methods for detection labeling error [8, 7] compare each training example's mask label to predictions from a segmentation model train trained without when excluding training example. For example, Lad and Mueller [8] propose an efficient strategy for label error detection using a label quality score for segmentation models based on taking a soft minimum over pixel-wise correct-class probabilities estimated by a trained segmentation model. Rather than directly processing pixel-wise predictions, Rottmann and Reese [7] instead identify labeling errors by comparing the predicted "connected components" (i.e., pixel-wise connected regions in an image with the same class label) from a trained segmentation model to the connected components in the ground truth mask to identify mismatches.

B Conclusion

In this work, we study the task of data attribution for segmentation models. We first identify two specific behaviors of these models that we hope to attribute: their overall ability to segment an image (i.e., full-image attribution), and their ability to properly segment a specific class within an image (i.e., class-specific attribution). We then provide a method for computing such attributions and instantiate this method on the MS COCO dataset. We validate the counterfactual significant of our computed attributions, and leverage them to both uncover labeling errors and curate subsets of the MS COCO training dataset that lead to increased accuracy over training on the full dataset.

C Experimental details

Dataset and models. We train and test all models on segmentations from the MS COCO dataset (2017 version) [26]. We center-crop images to a size of 224×224 and use random horizontal flip as an augmentation. For our models, we train DeepLabV3 [27] models for semantic segmentation for 120 epochs with the Adam optimizer with initial learning rate 0.0001, batch size 64, weight decay 0.0005, and a learning rate scheduler that reduces when loss plateaus. We use cross-entropy loss with no smoothing applied.

TRAK hyperparameters. For all uses of TRAK, we use a projection dimension of $k = 2048$. We find that attributions using a projection dimension of $k = 8192$ did not appear much stronger visually, and yielded very similar results when running preliminary data curation experiments. We also do not apply soft-thresholding to improve the quality of TRAK's attributions, though this is discussed in [11].

D Additional Results

We show additional top positive influencers for four randomly MS COCO validation examples in Figure 8. We also show the most negative influencers for two MS COCO validation examples in Figure 9, and highlight the labeling errors reflected in these influencers.

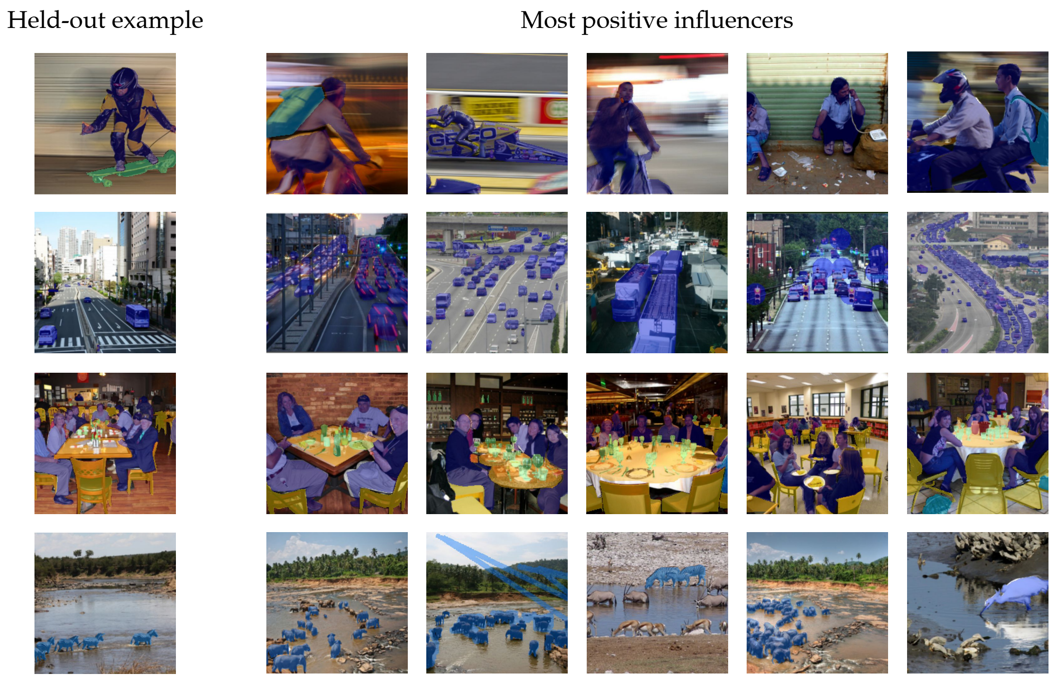

Figure 8: **More examples of top influencers.** For four MS COCO validation examples, we show the most positive influencers identified by TRAK.

Figure 9: **More examples of negative influencers.** For two MS COCO validation examples, we show the most negative (lowest-scoring) influencers identified to TRAK.