Learning to Encode Secondary Data with Architecture Search

Jack Newsom
kcaj@berkeley.edu

Abstract

It is well known that most deep learning models can improve their performance by training on larger datasets, but data collection is expensive. In the process of building up datasets, one might collect an amount of secondary data, or data not intended to be used as part of the dataset. In the context of segmentation, secondary data might include pictures of the desired scene from a different angle. Although this secondary data may be superficially distinct from primary data, discarding it entirely seems wasteful, since it is likely correlated with the primary data. Additionally, this secondary data might help to improve performance on instances that are particularly difficult with primary data alone. Another context secondary data appears in is scientific experiments, often as a parallel set of detector measurements for a particular event. Scientists may collect multiple types of data for a particular event, but only use one for data analysis tasks, leaving the others for calibration, or discarding them entirely.

To this end, we propose Smoking Gun Model Search (SGMS), a technique for determining how much additional signal can be extracted from this secondary data in segmentation tasks. In particular, SGMS aims to improve segmentation quality by learning to encode both primary and secondary data into a common latent space, then decoding the combined latent data into a segmentation on the primary data. We first train a U-Net-like autoencoder to perform the segmentation task on the primary data alone [3]. Then, we learn a feature extractor to embed the secondary data in the same latent space as the encoder of the primary autoencoder. Although we expect primary and secondary data to be correlated, we do not necessarily expect that the same types of models can be used to encode them into the same latent space. Instead of attempting to handcraft a feature extractor for the secondary data, we frame the model design as a neural architecture search problem, then employ the REINFORCE algorithm to quickly search through the space of these secondary data models [13]. We test the effectiveness of our technique on a simulated physics dataset, and our model derived from SGMS improves on the validation loss of a primary data-only model by over fifteen percent.

1 Introduction

In the process of building a computer vision dataset, one may collect a significant amount of secondary data, or instances not intended for inclusion in the dataset itself, but containing meaningful correlations with dataset instances. This process of data collection and labelling can be highly expensive; for example, the instance spotting stage alone of the design state of the popular COCO dataset took 20,000 hours [2]. Despite this cost, secondary data are often ignored or discarded. Consider the case of a team building a semantic segmentation dataset for a self-driving car. While the primary data might consist of labeled images from cameras in the front of the car, the team may collect secondary data of the same scenes from different angles to calibrate the car's cameras. Although these secondary

Preprint. Under review.

data would not be included in the dataset, they still contain a large amount of information about the scenes in question.

Secondary data also appears in a number of scientific tasks. For example, as part of the large DUNE collaboration, scientists aim to collect information about neutrinos by studying the structure of the tracks they leave in liquid argon detectors [12]. Current techniques take only the geometry of the tracks as input and attempt to produce segmentation maps, where distinct events correspond to distinct class labels [10]. However, it is known that these tracks will often have significant overlap in the real data, which will make this geometry-only segmentation approach very difficult.

It has been proposed that DUNE also collect light emission data from the events, effectively time-varying projections of the track geometries onto the detector walls. Although it is expected to be able to resolve this piled-up geometry problem, it is currently unknown what an effective approach to this strategy would be.

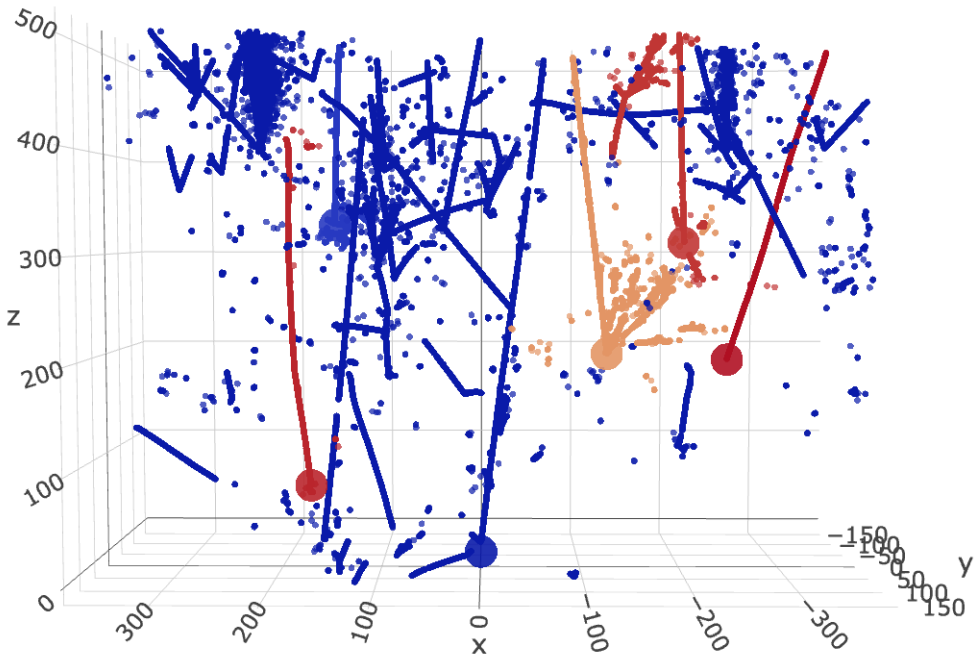


Figure 1: Six semi-overlapping neutrino tracks

It is fairly obvious that given a body of secondary data with meaningful correlations to a dataset's primary data, using the primary data alone is wasteful. However, given that this secondary data can vary significantly in form and type from the primary, it is generally not obvious how to find an effective way to make use of both types simultaneously.

To this end, we propose Smoking Gun Model Search to enable the discovery of optimal models for using primary and secondary data for segmentation tasks. SGMS finds performant models through an efficient neural architecture search technique related to AutoHAS. We employ the REINFORCE algorithm to search over the space of candidate models for the feature extractor and make use of several techniques to speed up this search process, including procedural initialization of the decoder weights and weight sharing.

2 Related Work

2.1 Neural Architecture Search

Much of the success of recent deep learning models can be attributed to very clever architecture design by the authors. Tricks like residual collections [5] and inception blocks [6] produced state-of-the-art performance on landmark tasks like ImageNet. Given the sensitivity of deep neural networks to these types of improvements, the ability to design new types of architectures is important. Frustratingly, discovering new performant architectures by hand can be quite difficult. Neural architecture search (NAS) aims to address this issue by automating away the model design process. In 2016, Zoph and Le introduced a technique (confusingly, also named Neural Architecture Search) that framed NAS as a reinforcement learning problem and obtained state-of-the-art accuracy on the CIFAR-10 and Penn Treebank datasets by training a recurrent network to predict model architectures [11].

Several significant improvements to the approach presented by Zoph and Le in 2016 have been produced since the original paper was published, primarily aimed at reducing the high cost of iterative model search. A key flaw of the original proposed search strategy was that once a candidate model was evaluated and used to update the controller, its trained weights were discarded entirely, effectively wasting all of the work done to train the child in the previous steps. Pham et al. addressed this problem in 2018 by reusing the weights of particular layers that had already been trained, reducing the total amount of compute needed to get the same results on CIFAR-10 by a factor of 1000 [4].

A second significant flaw with Zoph and Le's original approach was that it used the same set of hyperparameters to train and evaluate each child model. In 2020, Dong et al. extended the original NAS algorithm to also produce a distinct set of hyperparameters for each child model, allowing children to reach even higher performance [1].

2.2 Segmentation Models

Segmentation is a common task in computer vision that requires a model to produce a class or instance label for each pixel in an input image. One of the first deep models to reach reasonable performance on a segmentation task was the Fully Convolutional Network, proposed by Shelhamer et al. in 2014 [8]. The authors proposed using a deep network composed only of convolution and pooling layers, then upsampling the final feature maps to the original resolution to produce a segmentation map. This technique suffers from poor output resolution, since it attempts to upsample directly from the encoder's output feature maps, which are only a fraction of the size of the original input.

In 2015, Ronneberger et al. introduced a new model for medical image segmentation called U-Net [3]. The model was an improvement on the original FCN architecture that improved output resolution by allowing the upsampling layers to take the output of the downsampling layers as input, allowing more spatial information to flow past the deepest parts of the network. Since then, U-Net and derivative models have shown promising performance on a number of different segmentation-related problems, including particle-identification tasks [10].

3 Smoking Gun Model Search

In this section, we describe our framework for discovering optimal secondary data models and some of the techniques we applied to handle noisy reward signals and decrease training time.

3.1 Objective

The goal of SGMS is to discover three models: a feature extractor for primary data, a feature extractor for secondary data, and a decoder to produce segmentation maps. Specifically, our objective is

$$\begin{aligned} \min_{\alpha_p, \alpha_s, \alpha_d, h} \quad & \mathcal{L}_{\mathbb{D}_{\text{val}}}(\alpha_p(w_p), \alpha_s(w_s), \alpha_d(w_d), h) \\ \text{subject to} \quad & w_p = f(\alpha_p, \mathbb{D}_{\text{train}}), \\ & w_s, w_d = g_h(\alpha_s, \alpha_d, \mathbb{D}_{\text{train}}) \end{aligned}$$

where $\mathcal{L}$ is some loss function, α_p is the primary encoder, α_s is the secondary encoder, α_d is the decoder, h is the sampled hyperparameters, f is the optimization algorithm for the weights of the primary encoder, and g is the optimization algorithm for the weights of the decoder and secondary encoder.

3.2 Overview

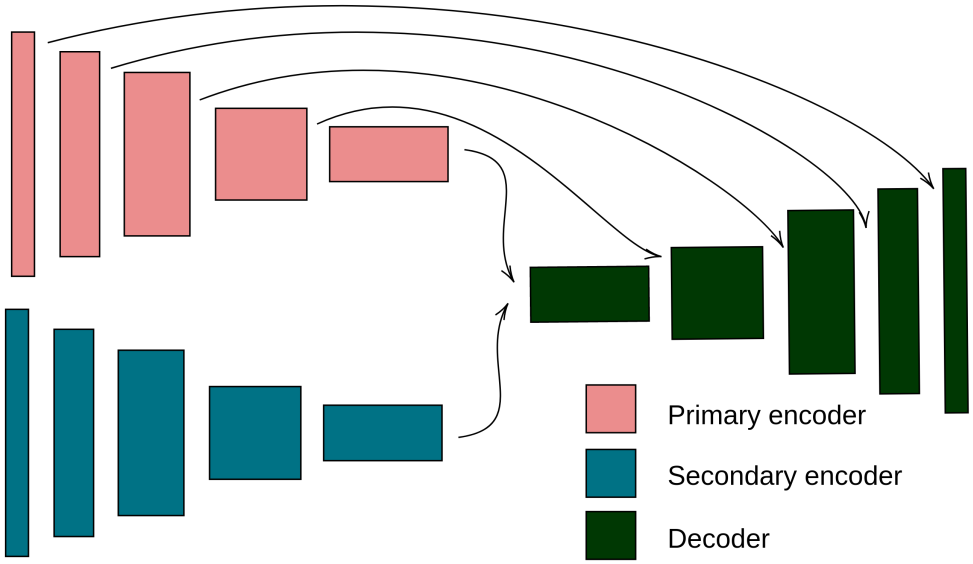


Figure 2: Smoking Gun Model Search Architecture Structure

There are effectively two stages of SGMS. The first is the training to convergence of a U-Net-like autoencoder on only the primary data. We then detach the encoder and decoder of this U-Net and save their weights.

The second step is the fast architecture search step for the secondary feature extractor. Just as in AutoHAS, we define a search space

$$\mathcal{S} = \mathcal{A} \times \mathcal{H}$$

where $\mathcal{A}$ is the space of feature extractor models and $\mathcal{H}$ is the space of hyperparameters. Then, we instantiate a probabilistic controller parameterized by an independent multinomial variable for each layer in the architecture space and hyperparameter choice in the hyperparameter space. To sample a child architecture and hyperparameter, we sample each multinomial variable in the controller.

A significant issue with AutoHAS is that it only samples a single architecture before performing a REINFORCE update to its controller's policy parameters, which can tend to make the reward very noisy if there is a lot of potential variance in the performance of the models in the architecture space. To remedy this, we sample some number $m > 1$ child models, training each to convergence and calculating its quality before performing a controller update step. We also use a simple average reward baseline to maintain low variance.

We then perform the REINFORCE update on the controller using the zero-mean rewards from the child models. To determine if the controller has converged, we obtain the highest probability child model from the controller and calculate its reward signal. In our experiments, we found that this approach significantly reduced the number of iterations required to reach convergence, since it allows us to ignore the significant variance in model performance (that also caused issues with the policy gradient updates to the controller).

Algorithm 1: Smoking Gun Model Search

Input: $\mathbb{D}_{\text{train}}, \mathbb{D}_{\text{val}}, \mathcal{A}, \mathcal{H}$
Result: $\alpha_p, \alpha_s, \alpha_d$
$w_p = f(\alpha_p, \mathbb{D}_{\text{train}})$
initialize controller $\mathcal{C}$
while $\mathcal{C}$ *is not converged* **do**
  for $t = 1$ **to** m **do**
    sample α_s, h from $\mathcal{C}$
    $w_s, w_d = g_h(\alpha_s, \alpha_d, \mathbb{D}_{\text{train}})$
    determine validation signal associated with sampled child α_s, q_t
  end
  update controller $\mathcal{C}$ parameters using $q_1, ..., q_m$
  obtain most likely child model and hyperparameters α_s^*, h^*
  $w_s^*, w_d^* = g_h(\alpha_s, \alpha_d, \mathbb{D}_{\text{train}})$
  determine validation signal associated with most likely child α_s^*, q^*
  if $q^* > threshold$ **then**
    $\mathcal{C}$ is converged
  end
end

4 Discussion

The dataset we use to evaluate our technique is made of simulated neutrino interaction events. Each instance in the dataset contains a list of points defining the event's geometry, a sparse representation of the light emission data for each event, and a target heatmap created by calculating a normal distribution at the start and end of each track with a variance of one pixel. For this task, we use the list of points as the primary data and the light emission data as secondary data, where our goal is to produce a heatmap close to the target. Each input is scaled to lie within a cube with side length 128 voxels. The dataset contains 800 training samples, and 200 validation samples.

The data is naturally extremely sparse, with less than 3% of voxels being active in each geometry input. For this reason, we use submanifold sparse convolutions to reduce computational overhead and maintain sparsity in our output heatmaps [9].

Table 1: Liquid Argon Simulated Dataset Model Design

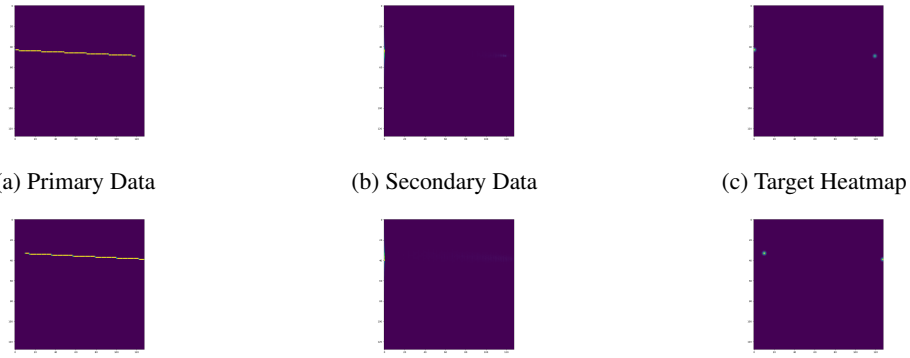
Design Choice	Options
Latent Space Spatial Size	$4 \times 4 \times 4$
Convolution Kernel Size	3, 5
Pooling Operation	$MaxPool, AveragePool$

We force our primary and secondary encoders to downsample the input images to a 4x4x4 voxel volume. We allow the model search space for the secondary encoder to be the set of models as deep as the primary encoder, with each layer either being a convolution with kernel size 3 or 5 or a pooling operation, and number of filters varying $\pm 20\%$ from 2^d, where d is the layer's depth in the network. Additionally, we set $m = 5$, thus sampling five child models before each REINFORCE step.

When running SGMS, we only trained each sampled child for ten epochs to iterate as quickly as possible. Ideally, we would train each model to convergence and then evaluate its performance because that is the metric we are interested in optimizing, but training to convergence would make the search step much slower and would only provide a marginally more accurate reward.

5 Experiments and Analysis

We trained the primary-only encoder for twenty epochs, then ran SGMS for a number of iterations and recorded the primary-secondary model performance at each step. Both models were training using mean squared error as loss on an NVidia RTX 2070.



(a) Primary Data (b) Secondary Data (c) Target Heatmap

Figure 3: Inputs and target for two events

Table 2: Model performance on Liquid Argon Dataset

Model	Training Epochs	Validation Loss
Primary-only Autoencoder	20	$1.70 \cdot 10^{-3}$
SGMS Iteration 1	10	$1.64 \cdot 10^{-3}$
SGMS Iteration 5	10	$1.47 \cdot 10^{-3}$
SGMS Iteration 30	10	$1.43 \cdot 10^{-3}$

After a single iteration of SGMS, our most likely primary+secondary model was already more performant than the primary-only autoencoder. Of course, we expect this to some degree, since if there was no useful information to be extracted from the secondary data, the secondary encoder could learn to zero out these features before they reached the latent space. After thirty SGMS iterations, the controller's most likely model has fifteen percent lower validation loss than the primary-only autoencoder.

6 Conclusion

In this paper, we presented Smoking Gun Model Search, a novel technique for improving the quality of segmentation models by learning to encode primary and secondary data sources into a common latent space. Our approach improves on previous architecture search techniques primarily by lowering variance in the reward signal, allowing it to produce optimal models much more quickly. We test SGMS on a simulated physics dataset and demonstrate that it is able to lower validation loss over a primary-only model by over fifteen percent in fewer than thirty iterations.

References

[1] Xuanyi Dong, Minxing Tan, Adams Wei Yu, Daiyi Peng, Bogdan Gabrys, and Quoc Le. AutoHAS: Efficient Hyperparameter and Architecture Search. https://arxiv.org/pdf/2006.03656.pdf

[2] Tsung-Yi Lin, Michael Maire, Serge Belongie, et al. Microsoft COCO: Common Objects in Context. https://arxiv.org/pdf/1405.0312.pdf

[3] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. https://arxiv.org/pdf/1505.04597.pdf

[4] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, Jeff Dean. Efficient Neural Architecture Search via Parameter Sharing. https://arxiv.org/pdf/1802.03268v2.pdf

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition. https://arxiv.org/pdf/1512.03385.pdf

[6] Christian Szegedy, Wei Liu, Yangqing Jia, et al. Going deeper with convolutions. https://arxiv.org/pdf/1409.4842v1.pdf

[7] Hanxiao Liu, Karen Simonyan, Yiming Yang. DARTS: Differentiable Architecture Search. https://arxiv.org/pdf/1806.09055.pdf

[8] Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. https://arxiv.org/pdf/1605.06211.pdf

[9] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3D Semantic Segmentation with Submanifold Sparse Convolutional Networks. https://arxiv.org/pdf/1711.10275.pdf

[10] Laura Domine and Kazuhiro Terao. Scalable Deep Convolutional Neural Networks for Sparse, Locally Dense Liquid Argon Time Projection Chamber Data. https://arxiv.org/pdf/1903.05663.pdf

[11] Barret Zoph and Quoc Le. Neural Architecture Search with Reinforcement Learning. https://arxiv.org/pdf/1611.01578.pdf

[12] R. Acciarri et al. Long-Baseline Neutrino Facility (LBNF) and Deep Underground Neutrino Expirement (DUNE) Conceptual Design Report Volume 1: The LBNF and DUNE Projects

[13] Ronald Williams. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.129.8871