Dynamic Model Specialization for Efficient Inference, Training and Supervision

Ravi Teja Mullapudi

CMU-CS-21-128

August 2021

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was sponsored by National Science Foundation awards IIS1422767 and IIS1539069; Defense Advanced Research Projects Agency award HR001117C0051; and by University of Maryland awards 54548Z9108201, 72858Z9321201, and 92483Z9618201. This research was also sponsored by gifts to faculty from Google LLC, Nvidia Corporation, Intel Corporation, Lutron Electronics Company, and Activision Publishing, Inc. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: Model Specialization, Computer Vision, Machine Learning, Deep Learning

Abstract

Recent supervised learning approaches focus on designing and building models that generalize to a wide range of scenarios. The key ingredients for building these general models are large scale datasets that capture a diverse set of scenarios and computational resources to train large models. This large scale supervised learning approach has well known scalability challenges namely: 1) accurate general models are computationally expensive for training and inference 2) collecting and labeling large datasets requires extensive human effort and 3) datasets need to be repeatedly curated due to shifts in the target distribution. In this thesis, we argue that in many cases creating a set of highly specialized models that span the domain of interest can reduce model inference, training, and supervision costs, compared to creating a single monolithic model that generalizes across the entire domain. Specifically, we exploit *temporal* specialization for building efficient video segmentation models. We show that continuously specializing a compact model to the content in a video stream enables accurate and efficient inference. We leverage specialization to *visually similar* categories for building efficient image classification architectures. We show that by specializing model features to discriminate between visually similar categories, one can improve inference efficiency by only computing the subset of features necessary for classifying a specific image. We exploit specialization to *individual categories* for reducing human labeling effort in building models for rare categories. We show that models specialized for binary classification of individual rare categories reduce human effort in mining large unlabeled data collections for relevant examples. More broadly, we demonstrate that by dynamically specializing to a moment in time, to an input scene, or to a specific object category, it is possible to train accurate models quickly, reduce inference costs, and significantly reduce the amount of supervision required for training.

Acknowledgments

The work done in thesis would not have been possible without my amazing advisers Kayvon and Deva who have been extremely supportive while I transitioned through a range of research interests during my Ph.D. from compilers and high performance computing to machine learning and computer vision. Kayvon has constantly pushed me to critically think about the problems I choose to work on more than the solutions themselves. The amount of time and dedication he put in teaching me how to present and communicate my research has had a significant impact on my career. I still remember how he helped me prepare for my first major conference presentation. He sat through several iterations of the same talk with small deltas paying attention to every little detail and provided some of the most useful feedback that has been nothing short of transformative. Deva has been a great mentor and enabled my transition to machine learning and computer vision research. I always learn a thing or two every time I meet with him to discuss my research. My conversations with him have been some of the most productive discussions during my time at CMU. He has taught me the importance of having a positive outlook and the ability to chart a path forward even when things do not turn out the way you anticipated. I am fortunate to be advised by Kayvon and Deva and thank them for their unconditional support and mentoring.

Over the course of my graduate studies I have been very lucky to have great mentors and collaborators. I am grateful to Bill for his guidance and confidence in my abilities. He has enabled many fruitful collaborations with amazing people at Google. The long walks he took me on to discuss research were some of best brainstorming sessions I have ever done. Fait Poms has been a great friend and collaborator who has been down in the trenches with me working on a range of topics and patiently listening to my long winded rants. I would not have started a career in research if not for the collaboration with my masters adviser Uday Reddy. The path that lead to my Ph.D. started with him responding a cold email from me asking he was interested in working with me on high performance computing research. The freedom and encouragement Uday provided during my time at Indian Institute of Science has expanded my horizons and reaffirmed my choice to leave industry and pursue academic research. His encouragement and support ultimately led me to do pursue graduate research. The support and advice of both Prof. Govindarajan and Prof. Albert Cohen was instrumental in shaping my career. Working with Andrew Adams, Jonathan Ragan-Kelley and the Halide team was a great learning experience.

I would like to thank my thesis committee - Dave Andersen, Ross Girshick for their valuable suggestions and feedback. Much of the work I have done would not have been feasible without my phenomenal collaborators - Andrew Adams, Steven Chen, Saurabh Gupta, Fait Poms, William Qi, Dillion Sharlet, Noam Shazeer, Vinay Vasista, Keyi Zhang. A large part of the graduate school experience is defined by the friends and connections you build over the years.

My office mates Evan Shimizu, Yong He, Jenny Lin, Nico Feltman, Ada Zhang and folks from Deva's lab Achal Dave, Peiyun Hu, Martin Li, Ishan Nigam, Rohit Girdhar have made my graduate school experience a lot of fun. I really miss the lunches with Chris Yu, Nick Sharp, Rohan Sawhney, Jim McCann,Ioannis Gkioulekas and other folks in Smith Hall over the last year and a half. Deb and Catherine have been immensely helpful in everything administrative from immigration paperwork to final thesis submission.

Vidya is the person who has been my anchor for what seems like a lifetime. Thanks for being there and constantly supporting the unconventional decisions I have made so far. I thank my family for being patient and supportive of a career path they did not even know exists. My parents are surprisingly some of the most forward thinking people despite barely being educated by conventional standards. I would not even have pursued a bachelors at IIITH if not for my mother and grandfather who scrapped together all the funds they could find to pay for my education. I wish my grandfather was here to see his bet pay off to some degree.

Contents

List of Figures

List of Tables

Chapter 1

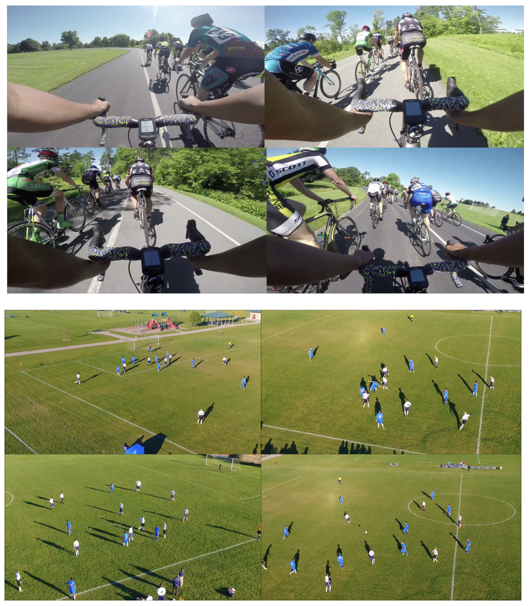
Introduction

Recent approaches to supervised deep learning are focused on designing and building a model that generalizes to a wide range of scenarios. The key ingredients in building general models are large scale datasets that capture a diverse set of scenarios and computational resources to train large models. Although general models that work in any scenario are desirable, training and using these models has the following scalability challenges that are well recognized: 1) accurate general models are computationally expensive to train and use for inference, 2) collecting and labeling a large representative dataset for a wide range of scenarios is expensive and time consuming, and 3) data distributions evolve over time requiring repeated dataset curation and model training.

Given that general models are difficult to build and use, we ask the question do we always need the most general model? If we look at many scenarios where models are used the data observed has very limited visual diversity and it is sufficient to be accurate for data observed in that specific scenario. For example, consider the scenario where a drone is capturing footage of a football match as shown in the bottom left of Figure 1.1 and following certain players by detecting and tracking them. Detecting players in the images captured by the drone is far easier compared to detecting people in a general distribution of images like the ones from COCO [85] dataset shown in the top of Figure 1.1. The reason the task is easier is due to limited variation is appearance of the players in the match, scale and background compared to the general distribution. Similarly, consider the task of detecting bicycles in the footage captured by an ego-centric head mounted camera as show in Figure 1.1 bottom left or detecting various animal species from wildlife cameras at a particular location. In both these cases the diversity of images observed is very limited compared to general distribution we are training models for. The limited diversity opens up the possibility of building models specialized to each context. Since the specialized model tackles a simpler task it can be compact and computationally cheaper compared to a more general model. Moreover, we only need to collect and label a dataset that is specific to the context reducing supervision and training costs. For instance, when building a model for detecting animals using wildlife cameras at a specific location, we can focus human labeling effort on data specific to that location as opposed to creating a general wildlife dataset.

However, models specialized to a subset of the distribution are only viable when one

(a) Sample of images from the COCO [85] dataset which captures a general distribution of objects.


(b) Images captured from a head-mounted and a drone camera during a short time window.

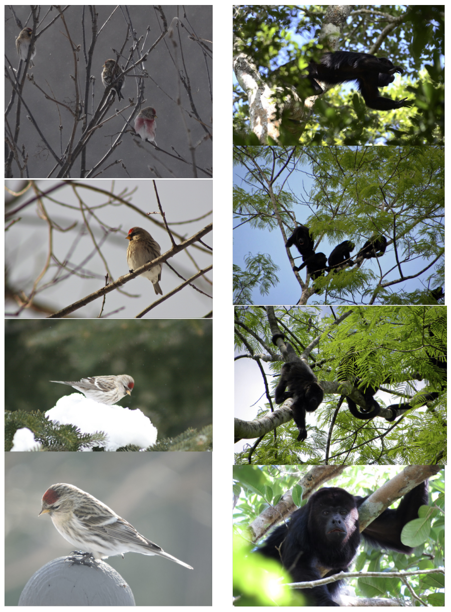

(c) Images capturing different animal species at specific locations.

Figure 1.1: The distribution of images in specific contexts (bottom) is very limited compared to the general distribution of images (top). Due to limited diversity in appearance, scale and background detecting players in the images captured by the drone (bottom left) or identifying animal species in images captured at a specific location (bottom right) is easier compared to performing the same task on a general distribution of images (top).

can rapidly adapt the model to cope with shifts in the target distribution. For instance, when the player detection model on the drone capturing the football match is specialized to daytime footage in clear weather conditions it is not going to be effective when there is change in weather or lighting conditions. In this thesis, rather than trying to build a general model that is accurate in a wide range of contexts, we embrace the context specific view and present techniques for rapidly building and using models specialized to a context. We show that by using these techniques for specialization to a moment in time, to an input image, or to a specific object category, it is possible to train accurate models quickly, reduce inference costs, and significantly reduce the amount of supervision required for training. More broadly, in a world where both the data distributions and tasks are constantly evolving we argue that:

In many cases creating a set of highly specialized models that span the

domain of interest can reduce model inference, training, and supervision costs, compared to creating a single monolithic model that generalizes across the entire domain.

1.1 Factors for Effective Model Specialization

There are several challenges that arise in building specialized models and using them in practice to reduce inference, training and supervision costs. We organize the challenges one needs to address for effective model specialization in terms of the following questions:

What is the model being specialized for? This is the first question we need to answer when specializing a model. Effective specialization for a task hinges on the choosing the appropriate strategy for partitioning the general distribution. For example, in the case detecting a rare animal species the model is specialized to detecting specific animal instances which are a small fraction of the general distribution of images containing other animals and flora. The model can further be specialized to the specific distribution of animals at a location since the appearance can vary across geographies. Similarly, the model can also specialized *temporally*, for instance, it can be specialized for different times of the day or across different seasons. More broadly, models for a task can be specialized to a slice or mode of the general data distribution.

How to curate a dataset specialized to each context? Once we decide on the dimension for partitioning the general distribution, we need to find the data corresponding to each partition and get supervision for training specialized models. For instance, when trying to detect vehicles from stationary cameras at traffic intersections we can choose to specialize based on location. When training a model specialized to a location we can focus labeling effort on data from the cameras at that location. Similarly, when trying to segment vehicles in a video stream we can choose to specialize temporally. In this setting, data from a few seconds earlier provides more relevant information compared to data from a few minutes or hours before, this suggests a more on-the-fly approach to dataset curation. In addition to finding data relevant to a context, an appropriate choice of supervision is crucial for effective specialization. For example, it might not feasible to leverage human supervision when specializing a model for monitoring traffic in real-time. However, when specializing models to mine and label rare categories in a large image collection, sparse human supervision is viable and can be cost effective.

What are model architecture and training algorithm changes required to induce and exploit specialization? Effective specialization relies on the ability to dynamically specialize models to different contexts. For instance, when trying to specialize for a rare category or a camera at a specific location one cannot wait to collect a representative dataset spanning all the visual variations of the category or the scene at that location to train a specialized model. Therefore, we need to be able to build models with limited amount of data pertaining to a context. Doing so requires both models and training techniques that are effective with small amount of data and supervision. Such training techniques and models need to strike a balance between utilizing representations learned on general distributions with task specific adaptation.

How do we identify when a specialized model is applicable? A major challenge in realizing the benefits of specialization is knowing when a particular specialized model is applicable. Consider two models one specialized for classifying animals and another specialized for classifying vehicles. Both specialized models can be more compact compared to a general model which can classify both vehicles and animals. Similarly, consider models for detecting an animal specialized for different weather conditions or different times of the day. Based on the context in question it is paramount that the appropriate specialized model is used for an accurate prediction.

1.2 Thesis Contributions

In addressing the challenges associated with effectively specializing models for reducing inference, training and supervision costs, this thesis makes the following contributions.

Temporal Specialization for Efficient Inference. In Chapter 2, we exploit *temporal* specialization for building efficient video segmentation models. Compact models can closely approximate the behavior of slower accurate models in short temporal windows. However, the compact models need to be frequently specialized to cope with distribution shifts in video streams that evolve over time. We show such continuous temporal specialization is feasible by designing a compact model architecture called JITNet and an online distillation algorithm for rapidly specializing the JITNet model. Moreover, our approach for specialization does not require upfront curation of training data or model training for individual video streams. We collected a Long Video Stream (LVS) dataset to facilitate evaluation of inference efficiency on long running video streams from a single camera.

Specialization to Similar Classes for Efficient Inference. In Chapter 3, we exploit model specialization to visually similar classes for building efficient image classification architectures. We show that by specializing different parts of a model to produce features to discriminate between visually similar categories, one can improve inference efficiency by only computing the subset of features necessary for classifying a specific image. However, for maintaining high accuracy across a range of categories, such a specialized model needs to dynamically choose and combine the appropriate specialized features based on the input. We enable accurate and efficient classification by designing a template architecture called HydraNet that can be used to transform an off-the-shelf deep architecture to an architecture the supports specialization. We present a training method which both specializes components of the HydraNet architecture and trains a sub-network to dynamically choose features to compute based on the input.

Specialization to Individual Categories for Efficient Supervision. In Chapter 4, we exploit binary classification models specialized for individual rare categories to effectively mine and label large unlabeled data collections. We show that by specializing models for binary classification of individual rare categories, as opposed to joint multi-way classification, one can accurately machine label a large number of negatives and identify a small subset of the unlabeled data relevant to each category. However, these specialized models need to be continuously updated as humans label more data to make better use of human effort. We design an active approach that utilizes the specialized binary labeling models to focus human effort on a small amount of hard examples and machine label easy negatives separately for each category. A key component in our approach is leveraging a large number of machine labeled negatives for each category to update models. However, the large number of negatives cause extreme imbalance in the labeled data making it difficult to train deep models. In Chapter 5, we describe a technique called background splitting which addresses the challenges in training deep models for rare categories with extremely imbalanced training data. We created iNat-BG and Places-BG using iNaturalist and Places datasets to enable evaluation of models for rare categories in settings where most of the data is background.

Chapter 2

Temporal Specialization for Efficient Inference

There are several axes one can specialize the model along and a natural set of axes for specialization is time and location. In this chapter, we focus on temporal model specialization to the data captured from a single camera in a short window of time. Most real-world video cameras capture scenes that feature a small fraction of the distribution of images observed across all cameras, and this distribution continuously evolves over time. For example, stationary cameras observe scenes that evolve with time of day, changing weather conditions, and as different subjects move through the scene. TV cameras pan and zoom, most smartphone videos are hand-held, and egocentric cameras on vehicles or robots move through dynamic scenes. We embrace this reality and move away from attempting to pre-train a model on camera-specific datasets curated in advance, and instead train models *online on a live video stream as new video frames arrive*. Specifically, we apply this methodology to the task of realizing high-accuracy and low-cost semantic segmentation models that continuously adapt to the contents of a video stream.

We employ the technique of model distillation [11, 55], training a lightweight "student" model to output the predictions of a larger, reliable high-capacity "teacher", but do so in an online fashion, intermittently running the teacher on a live stream to provide a target for student learning. We find that simple models can be accurate, provided they are continuously adapted to the specific contents of a video stream as new frames arrive (i.e. models can learn to cheat—segmenting people sitting on a park lawn might be as easy as looking for shades of green!). To achieve high efficiency, we require a new model architecture that simultaneously supports low-cost inference and fast training, as well as judicious choice of when to periodically run the teacher to obtain supervision.

We show that online model distillation yields semantic segmentation models that closely approximate their Mask R-CNN [51] teacher with 7 to 17× lower inference runtime cost (11-26× when comparing FLOPs), even when the target video's distribution is non-stationary over time. Our method requires no offline pretraining on data from the target video stream, has a small number of hyper parameters, and delivers higher accuracy segmentation output, than low-cost video semantic segmentation solutions based on flow. The output of our low-cost student models can be preferable (in terms of temporal stability) to that

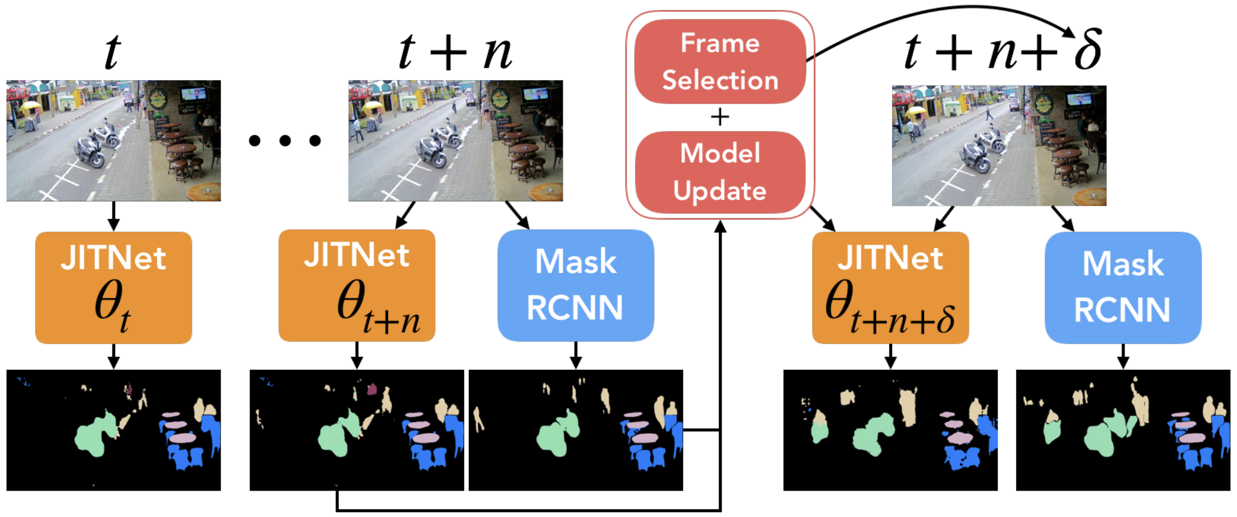


Figure 2.1: Online model distillation overview: A low-cost student model is tasked to generate a high-resolution, per-frame semantic segmentation. To retain high accuracy, as new frames arrive, an expensive teacher model's (MRCNN) output is periodically used as a learning target to adapt the student and selecting the next frame to request supervision. We call the student model "JITNet" since is designed to be specialized "just-in-time" for future frames.

of the expensive teacher. We also provide a new video dataset designed for evaluating the efficiency of inference over long running video streams.

2.1 Related Work

Distillation for specialization: Training a small, efficient model to mimic the output of a more expensive teacher has been proposed as a form of model compression (also called knowledge distillation) [11, 55]. While early explorations of distillation focused on approximating the output of a large model over the entire original data distribution, our work, like other recent work from the systems community [73], leverages distillation to create highly compact, domain-specialized models that need only mimic the teacher for a desired subset of the data. Prior specialization approaches rely on tedious configuration of models [87, 37] or careful selection of model training samples so as not to miss rare events [94]. Rather than treating model distillation as an offline training preprocess for a stationary target distribution (and incurring the high up-front training cost and the challenges of curating a representative training set for each unique video stream) like NoScope [73], we perform distillation online to adapt the student model dynamically to the changing contents of a video stream.

Online training: Training a model online as new video frames arrive violates the independent and identically distributed (i.i.d) assumptions of traditional stochastic gradient descent optimization. Although online learning from non-i.i.d data streams has been explored [18, 123], in general there has been relatively little work on online optimization of "deep" non-convex predictors on correlated streaming data. The major exception is the body of work on deep reinforcement learning [102], where the focus is on learning policies from experience. Online distillation can be formulated as a reinforcement or a meta-learning [36] problem. However, training methods [120, 101] employed in typical reinforcement settings are computationally expensive, require a large amount of samples, and are largely for offline use. Recent systems work [66] uses online model performance and accuracy profiling to choose the right model configuration from a collection of pre-trained models. Our goal is to train a single compact model which mimics the teacher in a small temporal window. In this context, we demonstrate that standard gradient decent is effective for online training our compact architecture.

Tracking: Traditional object tracking methods [71, 48, 54] and more recent methods built upon deep feature hierarchies [95, 151, 56, 105] can be viewed as a form of rapid online learning of appearance models from video. Tracking parameterizes objects with bounding boxes rather than segmentation masks and its cost scales in complexity with the number of objects being tracked. Our approach for online distillation focuses on pixel-level semantic segmentation and poses a different set of performance challenges. It can be viewed as learning an appearance model for the entire scene as opposed to individual objects.

Fast-retraining of compact models: A fundamental theme in our work is that low-cost models that do not generalize widely are useful, provided they can be quickly retrained to new distributions. Thus, our ideas bear similarity to recent work accelerating image classification in video via online adaptation to category skew [127] and on-the-fly model training for image super-resolution [131].

Video object segmentation: Solutions to video object segmentation (VOS) leverage online adaptation of high-capacity deep models to a provided reference segmentation in order to propagate instance masks to future frames [112, 161, 148, 14]. The goal of these algorithms is to learn a high-quality, video-specific segmentation model for use on subsequent frames of a short video clip, not to synthesize a low-cost approximation to a pre-trained general segmentation model like Mask R-CNN [51] (MRCNN). VOS solutions require seconds to minutes of training per short video clip (longer than directly evaluating a general segmentation model itself), precluding their use in a real-time setting. We believe our compact segmentation architecture and online distillation method could be used to significantly accelerate top-performing VOS solutions (see Section 2.4).

Temporal coherence in video: Leveraging frame-to-frame coherence in video streams, such as background subtraction or difference detection, is a common way to reduce computation when processing video streams. More advanced methods seek to activate different network layers at different temporal frequencies according to expected rates of change [77, 126] or use frame-to-frame flow to warp inference results (or intermediate features) from prior frames to subsequent frames in a video [38, 166]. We show that for the task of semantic segmentation, exploiting frame-to-frame coherence in the form of model specialization (using a compact model trained on recent frames to perform inference on near future frames) is both more accurate and more efficient than flow-based methods on a wide range of videos.

2.2 Just-In-Time Model Distillation

Figure 5.2 provides a high-level overview of online model distillation for high quality, low-cost video semantic segmentation. On each video frame, a compact model is run, producing a pixel-level segmentation. This compact student model is periodically adapted using predictions from a high-quality teacher model (such as MRCNN [51]). Since the student model is trained online (adapted just-in-time for future use), we refer to it as "JITNet". To make online distillation efficient in practice, our approach must: 1) use a student network that is fast for inference and fast for adaptation, 2) train this student *online* using imperfect teacher output, and 3) determine when and how to ask the teacher for labels as new frames arrive. We next address each of these challenges in turn.

2.2.1 JITNet Architecture

Efficient online adaptation requires a student architecture that (1) is efficient to evaluate even when producing high resolution outputs and (2) is amenable to fast learning. The ability to make high-resolution predictions is necessary for handling real-world video streams with objects at varying scales. Fast and stable adaptation is necessary for learning from the teacher in a small number of iterations.

Our JITNet architecture is a compact encoder-decoder [7] composed of three modified ResNet [52] blocks. To reduce computation, we replace the second 3×3 filter in each block with a separable filter (1×3 followed by a 3×1) and also limit the number of channels for high resolution feature maps. To ensure fast training, we add skip connections from each encoder block to the corresponding decoder block. This allows the gradient signal to propagate efficiently to lower layers. We include diagnostic experiments to evaluate the impact of these skip connections in section .

Table 2.1 gives the parameter count, number of floating-point operations, and runtime of both JITNet and MRCNN on a frame of 720p video on an NVIDIA V100 GPU. (We provide both inference and training costs for JITNet.) Compact segmentation models, such as those based on MobileNet V2 [119, 141], are 3-4× slower than JITNet at high resolution and are not designed for fast, stable online training. We evaluate the MobileNet V2 architecture as the student model and demonstrate that online distillation is viable using off-the-shelf architectures. However, we find that JITNet is more suitable for achieving both higher accuracy and efficiency. We also evaluate JITNet variants on standard semantic segmentation to ground it relative to other efficiency-oriented architectures.

Input Size	Operation	s	r	c
1280 x 720	conv 3x3	2		8
640 x 360	conv 3x3	2		8
320 x 180	enc_block 1	2		64
160 x 90	enc_block 2	2		64
80 x 45	enc_block 3	2		128
40 x 23	dec_block 3	1	2	64
80 x 45	dec_block 2	1	2	32
160 x 90	dec_block 1	1	4	32
640 x 360	conv 3x3	1		32
640 x 360	conv 3x3	1	2	32
1280 x 720	conv 1x1	1		32

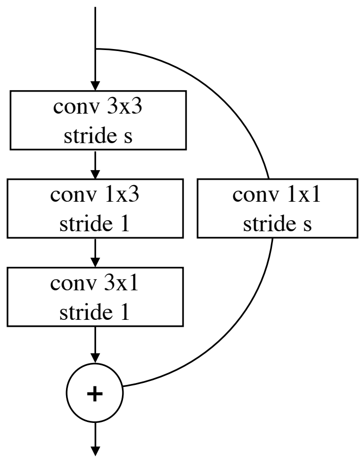


Figure 2.2: Left: JITNet architecture. Right: encoder/decoder block details. s = stride, r = resize, c = output channels.

Model	FLOPS (B)		Params (M)	Time (ms)	
	Infer	Train		Infer	Train
JITNet	15.2	42.0	3	7	30
MRCNN	1390.0	-	141	300	-

Table 2.1: FLOPS (inference, training), parameter count, and runtime for both JITNet and MRCNN. JITNet has 47× fewer parameters and requires 91× (inference) and 34× (training) fewer FLOPS than MRCNN inference.

2.2.2 Online Training with Gradient Descent

Online training presents many challenges: training samples (frames) from the video stream are highly correlated, there is continuous distribution shift in content (the past may not be representative of the future), and teacher predictions used as a proxy for "ground truth" at training can exhibit temporal instability or errors. The method for updating JITNet parameters must account for these challenges.

To generate target labels for training, we use the instance masks provided by MRCNN above a confidence threshold, and convert them to pixel-level semantic segmentation labels. All pixels where no instances are reported are labeled as background. On most video streams, this results in a significantly higher fraction of background compared to other classes. This imbalance reduces the ability of the student model to learn quickly, especially for small objects, due to most of the loss being weighted on background. We mitigate this issue by weighting the pixel loss in each predicted instance bounding box (dilated by 15%) five times higher than pixels outside boxes. This weighting focuses training on the challenging regions near object boundaries and on small objects. With these weighted labels, we compute the gradients for updating the model parameters using weighted cross-entropy loss and gradient descent. Since training JITNet on a video from a random initialization would require significant training to adapt to the stream, we pretrain JITNet on the COCO dataset, then adapt the pretrained model to each stream.

When fine-tuning models offline, it is common to only update a few layers or use small learning rates to avoid catastrophic forgetting. In contrast, for online adaptation, the goal is to minimize the cost of adapting the JITNet model so that it maintains high accuracy for current and near future video content. Rapidly specializing the compact JITNet to the temporal context retains high accuracy at low-cost. *Therefore, we update all layers with high learning rates.* Empirically, we find that gradient descent with high momentum (0.9) and learning rate (0.01) works remarkably well for updating JITNet parameters. We believe high momentum stabilizes training due to resilience to teacher prediction noise. *We use the same parameters for all online training experiments.*

Algorithm 1: Online distillation

```
Input: $S_{0...n}$, $u_{max}$, $\delta_{min}$, $\delta_{max}$, $a_{thresh}$, $\theta_0$
Output: $P_{0...n}$
1  $\delta \leftarrow \delta_{min}$
2  for $t \leftarrow 0$ to $n$ do
3      if $t \equiv 0 \pmod{\delta}$ then
4          $L_t \leftarrow$ MaskRCNN($S_t$)
5          $u \leftarrow 0$, update $\leftarrow$ true
6          while update do
7              $P_t \leftarrow$ JITNet($\theta_t$, $S_t$)
8              $a_{curr} \leftarrow$ MeanIoU($L_t$, $P_t$)
9              if $u < u_{max}$ and $a_{curr} < a_{thresh}$ then
10                 $\theta_t \leftarrow$ UpdateJITNet($\theta_t$, $P_t$, $L_t$)
11             else
12                 update $\leftarrow$ false
13             $u \leftarrow u + 1$
14         if $a_{curr} > a_{thresh}$ then
15             $\delta \leftarrow \min(\delta_{max}, 2\delta)$
16         else
17             $\delta \leftarrow \max(\delta_{min}, \delta/2)$
18     else
19         $P_t \leftarrow$ JITNet($\theta_t$, $S_t$)
20     $\theta_{t+1} \leftarrow \theta_t$
```

2.2.3 Adaptive Online Distillation

Finally, we need to determine *when* the student needs supervision from the teacher. One option is to run the teacher at a fixed rate (e.g., once every n frames). However, greater efficiency is possible using a dynamic approach that adapts JITNet with teacher supervision only when its accuracy drops. Therefore, we require an algorithm that dynamically determines when it is necessary to adapt JITNet without incurring the cost of running the teacher each frame to assess JITNet's accuracy.

Our strategy is to leverage the teacher labels on prior frames not only for training, but also for *validation*: our approach ramps up (or down) the rate of teacher supervision based on recent student accuracy. Specifically, we make use of exponential back-off [42], as outlined in Algorithm 1. Inputs to our online distillation algorithm are the video stream (S_t), maximum number of learning steps performed on a single frame (u_{max}), the minimum/maximum frame strides between teacher invocations (δ_{min}, δ_{max}), a desired accuracy threshold (a_{thresh}), and the initial JITNet model parameters (θ_0).

The algorithm operates in a streaming fashion and processes the frames in the video in temporal order. The teacher is only executed on frames which are multiples of the current stride (δ). When the teacher is run, the algorithm computes the accuracy of the current JITNet predictions (P_t) with respect to the teacher predictions (L_t). If JITNet accuracy is less than the desired accuracy threshold (mean IoU), the model is updated using the teacher predictions as detailed in the previous section. The JITNet model is trained until it either reaches the set accuracy threshold (a_{thresh}) or the upper limit on update iterations (u_{max}) per frame. Once the training phase ends, if JITNet meets the accuracy threshold, the stride for running the teacher is doubled; otherwise, it is halved (bounded by minimum and maximum stride). The accuracy threshold is the only user-exposed knob in the algorithm. As demonstrated in our evaluation, modifying the threshold's value allows for a range of accuracy vs. efficiency trade-offs.

Even when consecutive video frames contain significant motion, their overall appearance may not change significantly. Therefore, it is better to perform more learning iterations on the current frame than to incur the high cost of running the teacher on a new, but visually similar, frame. The maximum stride was chosen so that the system can respond to changes within seconds (64 frames is about 2.6 seconds on 25 fps video). The maximum updates per frame is roughly the ratio of JITNet training time to teacher inference cost. We set δ_{min} and δ_{max} to 8 and 64 respectively, and u_{max} to 8 for all experiments. *We do an ablation study of these parameters, choices in network design, and training method in section* **??**.

2.3 Long Video Streams (LVS) Dataset

Evaluating fast video inference requires a dataset of long-running video streams that is representative of real-world camera deployments, such as automatic retail checkout, player analysis in sports, traffic violation monitoring, and wearable device video analysis for augmented reality. Existing large-scale video datasets have been designed to support training high-quality models for various tasks, such as action detection [78, 137], object detection, tracking, and segmentation [113, 158], and consist of carefully curated, diverse sets of short video clips (seconds to a couple minutes).

We create a new dataset designed for evaluating techniques for efficient inference in real-world, long-running scenarios. Our dataset, named the Long Video Streams dataset (LVS), contains 30 HD videos, each 30 minutes in duration and at least 720p resolution. (900

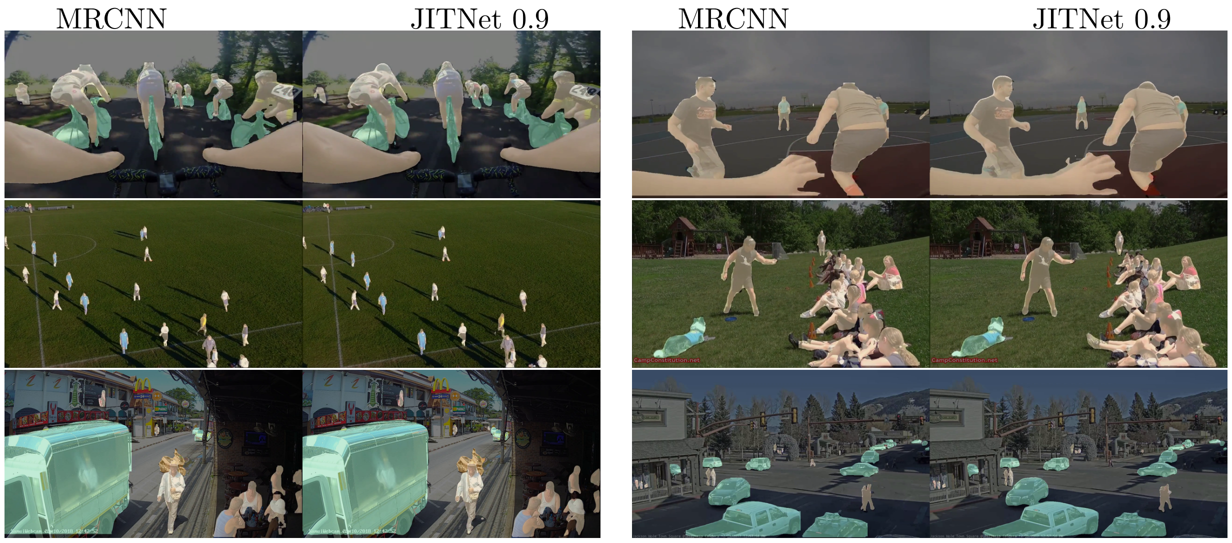


Figure 2.3: Frame segmentations generated by MRCNN (left) and JITNet 0.9 (right) from a subset of videos in the LVS dataset.

minutes total; for comparison, YouTube-VOS [158] is 345 minutes.) Unlike other datasets for efficient inference, which consist of streams from fixed-viewpoint cameras such as traffic cameras [67], we capture a diverse array of challenges: from fixed-viewpoint cameras, to constantly moving and zooming television cameras, and hand-held and egocentric video. Given the nature of these video streams, the most commonly occurring objects include people, cars, and animals.

It is impractical to obtain ground truth, human-labeled segmentations for all 900 minutes (1.6 million frames) of the dataset. Therefore, we curate a set of representative videos and use MRCNN [51] to generate predictions on all the frames. (We evaluated other segmentation models such as DeepLab V3 [20] and Inplace ABN [12], and found MRCNN to be produce the highest quality labels.) We use the highest-quality MRCNN [34] without test-time data augmentation, and provide its output for all dataset frames to aid evaluation of classification, detection, and segmentation (semantic and instance level) methods. Figure 2.3 shows a sampling of videos from the dataset with their corresponding MRCNN segmentations (left image in each group).

2.4 Evaluation

To evaluate online distillation as a strategy for efficient video segmentation, we compare its accuracy and cost with an alternative motion-based interpolation method [166] and an online approach for video object segmentation [14]. While our focus is evaluating accuracy and efficiency on long video streams (LVS), we also evaluate on the DAVIS video benchmark [113] to show possible applications to video object segmentation.

	Offline	Flow [166]		Online Distillation		
Video	Oracle (20%)	Slow (2.2×) (12.5%)	Fast (3.2×) (6.2%)	JITNet 0.7	JITNet 0.8	JITNet 0.9
Overall	80.3	76.6	65.2	75.5 (17.4×, 3.2%)	78.6 (13.5×, 4.7%)	82.5 (×7.5, 8.4%)
Category Averages						
Sports (Fixed)	87.5	81.2	71.0	80.8 (24.4×, 1.6%)	82.8 (21.8×, 1.8%)	87.6 (10.4×, 5.1%)
Sports (Moving)	82.2	72.6	59.8	76.0 (20.6×, 2.1%)	79.3 (14.5×, 3.6%)	84.1 (6.0×, 9.1%)
Sports (Ego)	72.3	69.4	55.1	65.0 (13.6×, 3.7%)	70.2 (9.1×, 6.0%)	75.0 (4.9×, 10.4%)
Animals	89.0	83.2	73.4	82.9 (21.7×, 1.9%)	84.3 (19.6×, 2.2%)	87.6 (14.3×, 4.4%)
Traffic	82.3	82.6	74.0	79.1 (11.8×, 4.6%)	82.1 (8.5×, 7.1%)	84.3 (5.4×, 10.1%)
Driving/Walking	50.6	69.3	55.9	59.6 (5.8×, 8.6%)	63.9 (4.9×, 10.5%)	66.6 (4.3×, 11.9%)
Subset of Individual Video Streams						
Table Tennis (P)	89.4	84.8	75.4	81.5 (24.7×, 1.6%)	83.5 (24.1×, 1.6%)	88.3 (12.9×, 3.4%)
Kabaddi (P)	88.2	78.9	66.7	83.8 (24.8×, 1.6%)	84.5 (23.5×, 1.7%)	87.9 (7.8×, 6.3%)
Figure Skating (P)	84.3	54.8	37.9	72.3 (15.9×, 2.8%)	76.0 (11.4×, 4.1%)	83.5 (5.4×, 9.4%)
Drone (P)	74.5	70.5	58.5	70.8 (15.4×, 2.8%)	76.6 (6.9×, 7.2%)	79.9 (4.1×, 12.5%)
Birds (Bi)	92.0	80.0	68.0	85.3 (24.5×, 1.6%)	85.7 (24.2×, 1.6%)	87.9 (21.7×, 1.8%)
Dog (P,D,A)	86.1	80.4	71.1	78.4 (19.0×, 2.2%)	81.2 (13.8×, 3.2%)	86.5 (6.0×, 8.4%)
Ego Dodgeball (P)	82.1	75.5	60.4	74.3 (17.4×, 2.5%)	79.5 (13.2×, 3.4%)	84.2 (6.1×, 8.2%)
Biking (P,Bk)	70.7	71.6	61.3	68.2 (12.7×, 3.5%)	72.3 (6.7×, 7.3%)	75.3 (4.1×, 12.4%)
Samui Street (P,A,Bk)	80.6	83.8	76.5	78.8 (8.8×, 5.5%)	82.6 (5.3×, 9.5%)	83.7 (4.2×, 12.2%)
Driving (P,A,Bk)	51.1	72.2	59.7	63.8 (5.7×, 8.8%)	68.2 (4.5×, 11.5%)	66.7 (4.1×, 12.4%)

Table 2.2: Comparison of accuracy (mean IoU over all the classes excluding background), runtime speedup relative to MRCNN (where applicable), and the fraction of frames where MRCNN is executed. Classes present in each video are denoted by letters (A - Auto, Bi - Bird, Bk - Bike, D - Dog, E - Elephant, G - Giraffe, H - Horse, P - Person). Overall, online distillation using JITNet provides a better accuracy/efficiency tradeoff than baseline flow based methods [166] and has accuracy comparable to oracle offline models.

2.4.1 Experimental Setup

Our evaluation focuses on both the efficiency and accuracy of semantic segmentation methods relative to MRCNN. Although MRCNN trained on the COCO dataset can segment 80 classes, LVS video streams captured from a single camera over a span of 30 minutes typically encounter a small subset of these classes. For example, none of the indoor object classes such as appliances and cutlery appear in outdoor traffic intersection or sports streams. Therefore, we measure accuracy only on classes which are present in the stream and have reliable MRCNN predictions. Our evaluation focuses on object classes which can independently move, since stationary objects can be handled efficiently using simpler methods. We observed that MRCNN often confused if an instance is a car, truck, or a bus, so to improve temporal stability we combine these classes into a single class "auto" for both training and evaluation. Therefore, we only evaluate accuracy on the following classes: bird, bike, auto, dog, elephant, giraffe, horse, and person. Table 2.2 shows the classes that are evaluated in each individual stream as an abbreviated list following the stream name.

All evaluated methods generate pixel-level predictions for each class in the video. We use mean intersection over union (mean IoU) over the classes in each video as the accuracy metric. All results are reported on the first 30,000 frames of each video (≈16-20 minutes due to varying fps) unless otherwise specified. Timing measurements for JITNet, MR-CNN (see Table 2.1), and other baseline methods are performed using TensorFlow 1.10.1 (CUDA 9.2/cuDNN 7.3) and PyTorch 0.4.1 for MRCNN on an NVIDIA V100 GPU. All speedup numbers are reported relative to wall-clock time of MRCNN. Note that MRCNN performs instance segmentation whereas JITNet performs semantic segmentation on a subset of classes.

2.4.2 Accuracy vs. Efficiency of Online Distillation

Table 2.2 gives the accuracy and performance of online distillation using JITNet at three different accuracy thresholds: JITNet 0.7, 0.8, and 0.9. Performance is the average speedup relative to MRCNN runtime, *including the cost of teacher evaluation and online JITNet training*. To provide intuition on the speedups possible on different types of videos, we organize LVS into categories of similar videos and show averages for each category (e.g., Sports (Moving) displays average results for seven sports videos filmed with a moving camera), as well as provide per-video results for a selection of 10 videos. We also show the fraction of frames for which MRCNN predictions are used. For instance, on the Kabaddi video stream, JITNet 0.8 is 23.5 times faster than MRCNN, with a mean IoU of 84.5, and uses 510 frames out of 30,000 (1.7%) for supervision.

On average, across all sequences, JITNet 0.9 maintains 82.5 mean IoU with 7.5× runtime speedup (11.3× in FLOPs). In the lower accuracy regime, JITNet 0.7 is 17.4× faster on average (26.2× in FLOPs) while maintaining a mean IoU of 75.5. Mean IoUs in the table exclude the background class, where all the methods have high accuracy. As expected, when the accuracy threshold is increased, JITNet improves in accuracy but uses a larger fraction of teacher frames for supervision. Average speedup on sports streams from fixed cameras is higher than that for moving cameras. Even on challenging egocentric sports videos with significant motion blur, JITNet 0.9 provides 4.9× speedup while maintaining 75.0 mean IoU.

Although JITNet accuracy on the Sports (Fixed), Sports (Moving), Animals, and Traffic categories suggests potential for improvement, we observe that for streams with large objects, it is often difficult to qualitatively discern if JITNet or MRCNN produces higher quality predictions. Figure 2.3 displays sample frames with both MRCNN (left) and JITNet (right) predictions (zoom in to view details). The boundaries produced by JITNet on large objects (1st row) are smoother than MRCNN, since MRCNN generates low-resolution masks (28×28) that are upsampled to full resolution. However, for videos containing small objects, such as traffic camera (Figure 2.3, 3rd row, right) or aerial views (2nd row, left), MRCNN produces sharper segmentations. JITNet's architecture and operating resolution would need to be improved to match MRCNN segmentations on small objects.

Streams from the Sports (Ego) category exhibit significant motion blur due to fast motion. Teacher predictions on blurred frames can be unreliable and lead to disruptive model updates. The Driving/Walking streams traverse a busy downtown and a crowded

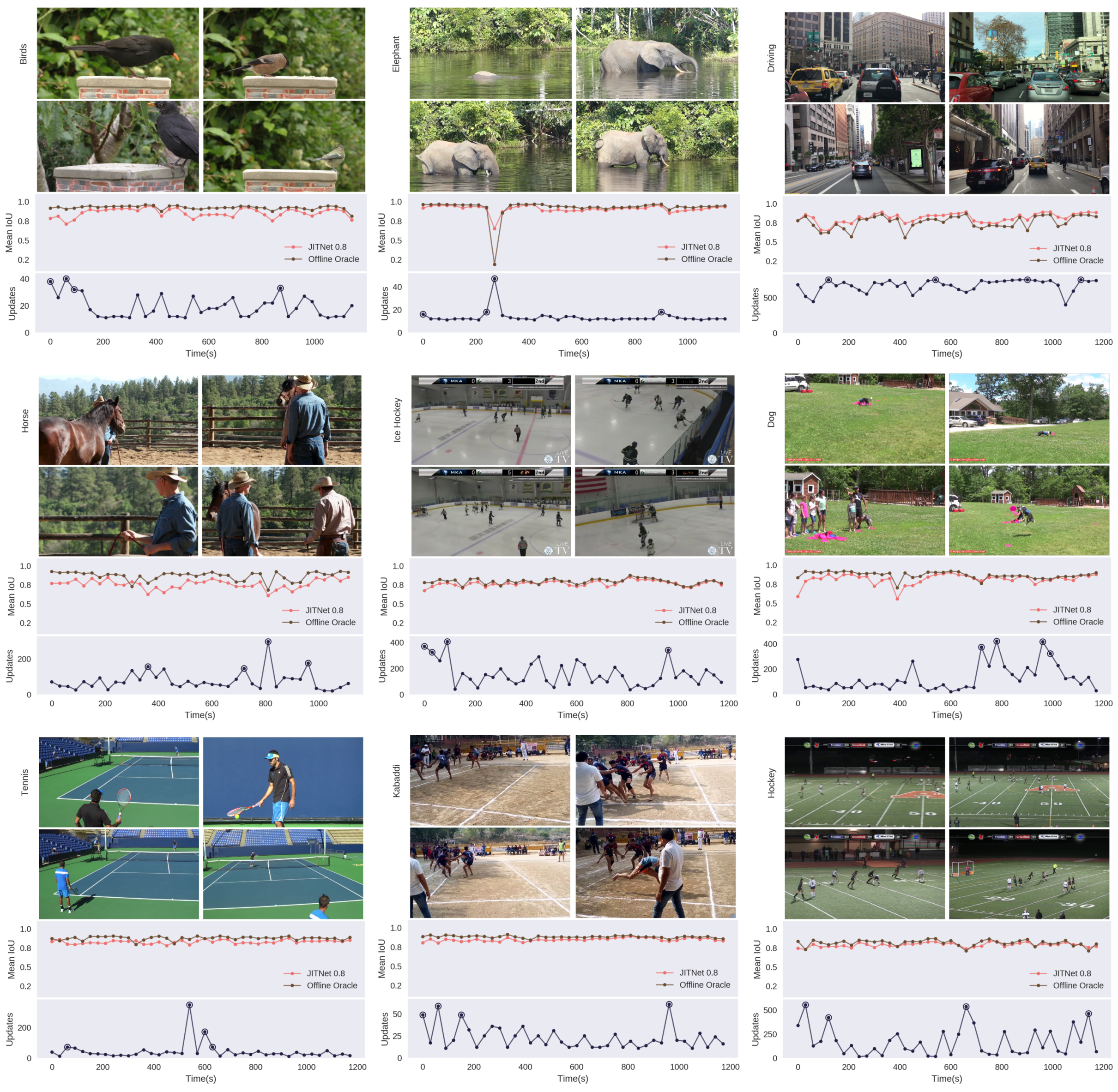


Figure 2.4: Top graph: the accuracy of JITNet 0.8 and Offline Oracle relative to MRCNN. Bottom graph: the number of updates to JITNet during online distillation. Plotted points are averages over a 30 second interval of the video. Images correspond to circled points in bottom plot, and show times where JITNet required frequent training to maintain accuracy.

beach, and are expected to be challenging for online distillation since object instances persist on screen for only short intervals in these videos. Handling these scenarios more accurately would require faster methods for online model adaptation.

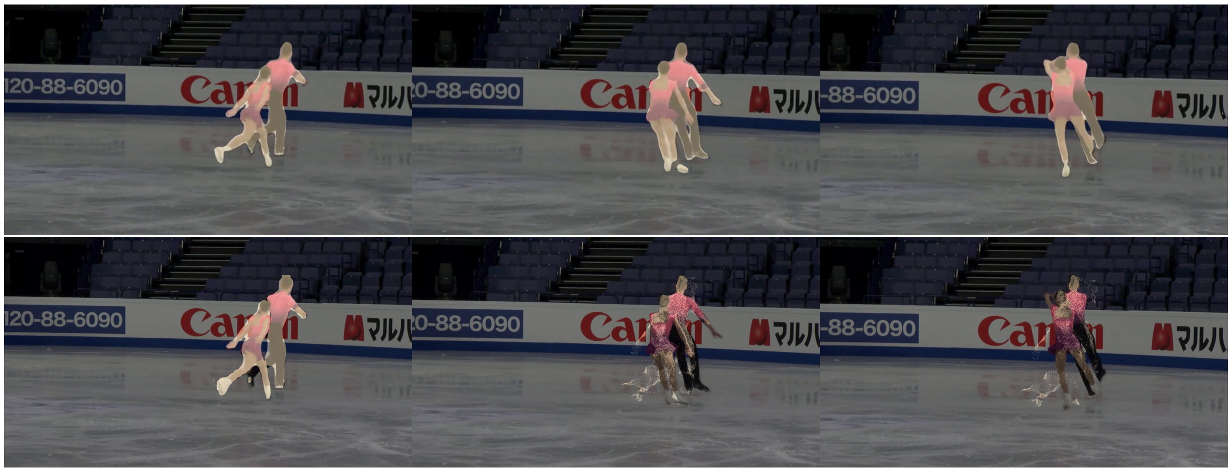


Figure 2.5: Top: JITNet 0.9 predictions on a sequence of three frames which are roughly 0.13 seconds apart (4 frames apart) in the Figure Skating video. Bottom: Large deformations, object and camera motion prove challenging to flow based interpolation.

2.4.3 Comparison with Offline Oracle Specialization

The prior section shows that a JITNet model pre-trained only on COCO can be continuously adapted to a new video stream with only modest online training cost. We also compare the accuracy of just-in-time adaptation to the results of specializing JITNet to the contents of the each stream *entirely offline*, and performing no online training. To simulate the effects of near best-case offline pre-training, we train JITNet models on every 5th frame of the entire 20 minute test video sequence (6,000 training frames). We refer to these models as “offline oracle” models since they are constructed by pre-training on the test set, and serve as a strong baseline for the accuracy achievable via offline specialization. All offline oracle models were pre-trained on COCO, and undergo *one hour* of pre-training on 4 GPUs using traditional random-batch SGD. Note that in contrast, *online adaptation incurs no pre-training cost* and trains in a streaming fashion.

As shown in Table 2.2, JITNet 0.9 is on average more accurate than the offline oracle. Note that JITNet 0.9 uses only 8.4% of frames on average for supervision, while the oracle is trained using 20%. This trend also holds for the subcategory averages. This suggests that the compact JITNet model does not have sufficient capacity to fully capture the diversity present in the 20 minute stream.

Figure 2.4 shows mean IoU of JITNet 0.8 and the offline oracle across time for three videos. The top plot displays mean IoU of both methods (data points are averages over 30 second time intervals). The bottom plot displays the number of JITNet model updates in each interval. Images above the plots are representative frames from time intervals requiring the most JITNet updates. In the Birds video (left), these intervals correspond to events when new birds appear. In comparison, the Elephant video (center) contains a single elephant from different viewpoints and camera angles. The offline oracle model incurs a significant accuracy drop when the elephant dips into water. (This rare event makes up only a small fraction of the offline training set.) JITNet 0.8 displays a smaller drop since it specializes immediately to the novel scene characteristics. The Driving video (right) is challenging for both the offline oracle and online JITNet since it features significant visual diversity and continuous change. However, while the mean IOU of both methods is lower, online adaptation consistently outperforms the offline oracle in this case as well.

2.4.4 Comparison with Motion-Based Interpolation

An alternative approach to improving segmentation efficiency on video is to compute teacher predictions on a sparse set of frames and interpolate the results using flow. Table 2.2 shows two baselines that propagate pixel segmentations using Dense Feature Flow [166], although we upgrade the flow estimation network from FlowNet2 [63] to modern methods. (We propagate labels, not features, since this was shown to be as effective [166].) The expensive variant (Flow (Slow)) runs MRCNN every 8th frame and uses PWC-Net [139] to estimate optical flow between frames. MRCNN labels are propagated to the next seven frames using the estimated flow. The fast variant (Flow (Fast)) uses the same propagation mechanism but runs MRCNN every 16th frame and uses a faster PWC-Net. Overall JITNet 0.7 is 2.8× faster and more accurate than the fast flow variant, and JITNet 0.9 has significantly higher accuracy than the slow flow variant except in the Driving/Walking category.

Figure 2.5 illustrates the challenge of using flow to interpolate sparse predictions. Notice how the ice skaters in the video undergo significant deformation, making them hard to track via flow. In contrast, online distillation trains JITNet to learn the appearance of scene objects (it leverages temporal coherence by reusing the model over local time windows), allowing it to produce high-quality segmentations despite complex motion. The slower flow baseline performs well compared to online adaptation on rare classes in the Driving (Bike) and Walking (Auto) streams, since flow is agnostic to semantic classes. Given the orthogonal nature of flow and online adaptation, it is possible a combination of these approaches could be used to handle streams with rapid appearance shifts.

2.4.5 Comparison with Video Object Segmentation

Although not motivated by efficiency, video object segmentation (VOS) solutions employ a form of online adaptation: they train a model to segment future video frames based on supervision provided in the first frame. We evaluate the accuracy of the OSVOS [14] approach against JITNet on two-minute segments of each LVS video. (OSVOS was too expensive to run on longer segments.) For each 30-frame interval of the segment, we use MRCNN to generate a starting foreground mask, train the OSVOS model on the starting mask, and use the resulting model for segmenting the next 29 frames. We train OSVOS for 30 seconds on each starting frame, which requires approximately *one hour* to run OSVOS on each two-minute video segment. Since segmenting all classes in the LVS videos would require running OSVOS once per class, we run OSVOS on only one class per video (person or animal class in each stream) and compare JITNet accuracy with OSVOS on the designated class. (Recall JITNet segments all classes.) Furthermore, we run two

Category	OSVOS (3.3%)		JITNet 0.8
	A	B	
Overall	59.9	60.0	77.4 (14.5×, 4.6%)
Sports (Fixed)	75.7	75.7	82.3 (24.0×, 1.6%)
Sports (Moving)	69.1	69.3	78.7 (16.3×, 2.9%)
Sports (Ego)	67.6	68.1	74.8 (9.5×, 5.9%)
Animals	79.3	79.8	86.0 (19.7×, 2.1%)
Traffic	22.3	21.9	70.8 (8.4×, 7.7%)
Driving/Walking	36.7	36.3	66.8 (4.3×, 11.8%)

Table 2.3: JITNet 0.8 generates higher accuracy segmentations than OSVOS on LVS and is two orders of magnitude lower cost. Percentages give the fraction of frames used for MRCNN supervision.

configurations of OSVOS: in mode (A) we use the OSVOS model from the previous 30-frame interval as the starting point for training in the next interval (a form of continuous adaptation). In mode (B) we reset to the pre-trained OSVOS model for each 30-frame interval.

Table 2.3 compares the accuracy of both OSVOS variants to online distillation with JITNet. The table also provides model accuracy, runtime speedup relative to MRCNN, and the fraction of frames used by JITNet 0.8 for supervision in the two-minute interval. Overall JITNet 0.8 is more accurate than OSVOS and *two orders of magnitude* faster. On Traffic streams, which have small objects, and Driving/Walking streams with rapid appearance changes, OSVOS has significantly lower accuracy than JITNet 0.8. We also observe that the mode A variant of OSVOS (continuously adapted) performs worse than the variant which is re-initialized. We believe the JITNet architecture could be employed as a means to significantly accelerate online VOS methods like OnAVOS [148] or more recent OSVOS-S [96] (uses MRCNN predictions every frame).

2.4.6 Online Distillation Ablation Study

Our online distillation approach has several parameters (maximum updates (u_{max}), minimum stride (δ_{min}), learning rate and network size) that enable different trade-offs between accuacy and efficiency. Here, we study the impact of these parameters on the accuracy vs. efficiency trade-off on a subset of six video streams (which are representative of different scenarios) in the LVS dataset. We also evaluate the impact of skip connections and resolution on accuracy and efficiency. Table 2.4 compares the accuracy, speedup (relative to running the teacher on every frame), fraction of frames used for supervision, and number of FLOPS (floating point operations) for both training and inference on each of the variants. The baseline is JITNet 0.8, the online distillation algorithm run with an accuracy threshold of 0.8. For JITNet 0.8, the maximum updates, minimum stride, and learning rate were set to 8, 8 and 0.01 respectively. We vary one parameter at a time, and each column in the

	JITNet												MobileNet	
	Baseline	Max Updates		Learning Rate		Min Stride		Width		Skip	Scale		Output Stride	
		4	16	0.001	0.1	4	16	0.5	2.0	No	0.5	0.75	8	16
Accuracy (mIoU)	78.7	77.3	78.0	75.6	16.7	79.8	76.1	62.0	80.3	76.0	75.9	78.1	75.3	74.6
Speed Up	19.2×	17.2×	22.8×	14.3×	7.3×	17.1×	22.9×	14.9×	12.4×	12.4×	20.0×	22.0×	9.5×	16.8×
Teacher Samples	5.0%	6.1%	3.7%	6.7%	10.6%	7.7%	3.3%	5.1%	4.2%	6.3%	6.2%	5.3%	5.0%	5.8%
Inference (FLOPS)	15.2							11.8	47.9	14.3	4.6	10.3	60.3	18.3
Training (FLOPS)	42.0							31.8	140.4	39.4	8.6	22.4	176.1	53.0

Table 2.4: Comparison of different input parameter settings to the online distillation algorithm. The algorithm is robust to all parameter changes except very high learning rates, where online training becomes unstable.

table corresponds to a variation of the JITNet 0.8 baseline.

Learning rate: High learning rates allow for faster adaptation. Therefore, we chose the highest learning rate at which online training is stable for all our experiments. As one can see in Table 2.4, a lower learning rate of 0.001 reduces both accuracy and speedup. Increasing the learning rate to 0.1 destabilizes training and yields low accuracy.

Max updates and stride: The number of updates needed on a single frame depends on how much the model can learn from one frame, and how useful that information is in the immediate future. Increasing the number of updates leads to overfitting, reducing accuracy while increasing speedup, and reducing the number of teacher samples used. This suggests some room for improvement in choosing how many updates to perform on a given frame over our simple accuracy-based heuristic. As one would expect, increasing and decreasing minimum stride increase and decrease accuracy respectively.

JITNet capacity: Intuitively, as the capacity of the student architecture is increased, the student model should require less help from the teacher. We verify this by varying the width of JITNet (the number of channels in each layer), and observe that a smaller capacity network (width 0.5) requires more supervision from the teacher, and also results in a significant drop in accuracy. Doubling JITNet width improves overall accuracy and reduces the number of teacher samples used. However, overall speedup is lower than the baseline due to the increased inference and training cost of the wider JITNet model.

JITNet resolution: High resolution is necessary for maintaining high accuracy on video streams that have small objects. When the input resolution to JITNet is halved (scale 0.5), there is an overall increase in the number of frames on which teacher supervision is used, and also a drop in accuracy relative to the baseline. However, reducing the resolution to 75% (scale 0.75) retains high accuracy while being slightly faster than the baseline. This suggests the possibility of varying resolution based on the contents in a video stream, which could be explored in the future.

Skip connections: We added encoder-to-decoder skip connections to facilitate better gradient propagation and make JITNet suitable for fast online adaptation. We evaluate the impact of the skip connections by removing them (Table 2.4, No Skip). JITNet without

Method	JM	JR	JD	FM	FR	FD
JITNet A	0.642	0.731	0.238	0.680	0.761	0.235
JITNet B	0.796	0.927	0.018	0.798	0.904	0.060
JITNet C	0.811	0.924	-0.004	0.831	0.913	0.004
OSVOS-S [96]	0.856	0.968	0.055	0.875	0.959	0.082
OSVOS [14]	0.798	0.936	0.149	0.806	0.926	0.150

Table 2.5: Accuracy comparison of different methods using the JITNet architecture and recent methods for semi-supervised video object segmentation on the DAVIS 2016 benchmark.

skip connections requires more teacher samples and adaptation, reducing both accuracy and speedup relative to the baseline.

Overall, the online distillation algorithm is reasonably robust to different parameter settings and provides a range of options for accuracy vs. efficiency.

2.4.7 MobileNet Student

We compare JITNet with a popular efficiency-oriented MobileNetV2 [119, 141] architecture in the context of online distillation. Table 2.4 shows online accuracy and speedup of online distillation when the MobileNetV2 architecture is used as the student. The two MobileNetV2 variants produce outputs at $1/8^{th}$ and $1/16^{th}$ of the input resolution. As one can see, the higher resolution variant of the MobileNetV2 architecture is significantly slower and has lower accuracy than the JITNet baseline. Even the lower resolution (scale 0.75) version of JITNet has higher accuracy and speedup compared to the MobileNetV2 student. These results demonstrate that off-the-shelf models can be used in our online distillation framework. However, JITNet provides a better accuracy vs. efficiency spectrum compared to MobileNetV2 for online distillation, since it is designed for fast adaptation. Note that we measure FLOPS, which is a platform-agnostic metric, to ensure fair comparison, since wall-clock time (MobileNetV2 takes 38ms for inference compared to 7ms for JITNet on a Nvidia V100 GPU) depends on various factors, including target platform, underlying libraries, and specific implementation.

2.4.8 DAVIS Evaluation

Online distillation as a technique can be used to mimic an accurate teacher model with a compact model, improving runtime efficiency. The main focus of this work is to demonstrate the viability of the online distillation technique for semantic segmentation on streams captured from typical use case scenarios. In this section, we show preliminary results on the viability of online distillation combined with the JITNet architecture for accelerating semi-supervised video object segmentation methods. Specifically, we evaluate how the JITNet architecture can be combined with state-of-the-art methods such as OSVOS-S [96].

We evaluate three different configurations of JITNet at varying levels of supervision. In configuration A, we train JITNet on only the first ground truth frame of each sequence, and evaluate JITNet over the rest of the frames in the sequence without any additional supervision (the standard video object segmentation task). On many sequences in DAVIS, object appearance changes significantly and requires prior knowledge of the object shape. Note that JITNet is a very low capacity model designed for online training, and cannot encode such priors. Configuration A is not an online distillation scenario, but even with its low capacity, the JITNet architecture trained on just the first frame yields reasonable results.

Recent methods such as OSVOS-S [96] leverage instance segmentation models such as Mask R-CNN for providing priors on object shape every frame. We take a similar approach in configuration B, where the goal is to mimic the expensive OSVOS-S model. We train JITNet on the first ground truth frame, then adapt using segmentation predictions from OSVOS-S [96] every 16 frames. Note that in configuration B, our combined approach does not use additional ground truth, since OSVOS-S predictions are made using only the first ground truth frame. Finally, in configuration C, we train on the first ground truth frame, and adapt on the ground truth mask every 16 frames. This gives an idea of how the quality of the teacher effects online distillation.

We use the validation set of the DAVIS 2016 [113] dataset for our evaluation. The dataset contains 50 video sequences of 3455 frames total, each labeled with pixel-accurate segmentation masks for a single foreground object. We evaluate using the main DAVIS metrics: region similarity J and contour accuracy F, with precision, recall, and decay over time for both. We present metrics over the entire DAVIS 2016 validation set for all three JITNet configurations, alongside a subset of state-of-the-art video object segmentation approaches. In all configurations, we start with JITNet pre-trained on YouTube-VOS [158], with max updates per frame set to 500, accuracy threshold set to 0.95, and use standard data augmentation (flipping, random noise, blurring, rotation). JITNet A performs similarly to OFL [143], a flow-based approach for video object segmentation, while JITNet B, using OSVOS-S predictions, performs comparably to OSVOS, with significantly lower runtime cost. Finally, JITNet C, which uses ground truth masks for adaptation, performs comparably to only using OSVOS-S predictions. This suggests that even slightly noisy supervision suffices for online distillation. Overall, these results are encouraging with regards to further work into exploring architectures well suited for online training.

2.4.9 Standalone Semantic Segmentation

The JITNet architecture is specifically designed with low capacity so that it can support both fast training and inference. To understand the accuracy vs. efficiency trade-off relative to other architectures such as MobileNetV2 [119, 141], we trained a JITNet model with twice the number of channels and encoder/decoder blocks than the one used in other experiments. This modified architecture is 1.5× faster than the semantic segmentation architecture based on MobileNetV2. The larger JITNet gives a mean IoU of 67.34 on the cityscapes [28] validation set and compares favorably with the 70.71 mean IoU of the MobileNetV2 based model [141]. We started with the larger JITNet architecture in the online distillation experiments, but lowered the capacity even further, with half the number of channels and encoder/decoder blocks, since it provided a better cost vs. accuracy trade-off for online distillation.

2.5 Discussion

We demonstrate that for common, real-world video streaming scenarios, it is possible to perform online distillation of compact (low cost) models to obtain semantic segmentation accuracy that is comparable with an expensive high capacity teacher. We summarize our approach as answers to the questions we need to address for effective specialization.

- *What is the model being specialized for?* The JITNet model is specialized for short temporal windows.
- *How to curate a dataset specialized to the context?* The data required for temporal specialization comes naturally from the recent past in a video stream and the large accurate model is used to provide sporadic supervision on recent frames.
- *What are model architecture and training algorithm changes required to induce and exploit specialization?* The JITNet model is designed to support not only real-time inference but also real-time training. Unlike the conventional training online distillation continuously trains the JITNet model on a stream of data and noisy supervision.
- *How do we identify when a specialized model is applicable?* Deviation from the predictions of the accurate teacher model is used to indicate when the JITNet model is no longer applicable and needs to be updated.

The solution space becomes more apparent when thinking in terms of these questions. For instance the question of how to curate a dataset specialized to the context corresponds to how we use the prior data observed in the stream for improving the model. We show that continuously specializing to the most recent data is more accurate and also sidesteps the need to curate datasets and train models offline. The question of what the model is being specialized for encompasses the choice of specializing to short windows of time as opposed to individual cameras or a different temporal partitioning (day vs. night). The question of identifying when a specialized model is applicable highlights the reactive nature of our approach since it can only identify the JITNet model failing after it performs poorly. Our approach is more suitable for throughput oriented applications since it recovers fairly quickly (few seconds) and reduces the overall cost. However, it is not suitable for mission critical settings where a failure cannot be tolerated. In such settings, a predictive method for JITNet performance is more applicable. For instance, one can rely on low-level cues like large changes in depth or significant motion of the camera to preemptively identify situations where the JITNet model might need to adapt.

Going forward, we hope that our results encourage exploration of online distillation for domain adaptation and self-supervised learning. More generally, with continuous capture of high-resolution video streams becoming increasingly commonplace, we believe it is relevant for the broader community to think about the design and training of models that are not trained offline on carefully curated datasets, but instead continuously evolve each day with the data that they observe from specific video streams. We hope that the Long Video Streams dataset serves this line of research.

Chapter 3

Specialization to Similar Classes for Efficient Inference

One of the scalability challenges with accurate general models is the inference cost which can consume a significant fraction of data center capacity [70] and is not practical on resource-constrained mobile devices. As a result, there is growing interest in improving the design of deep architectures to be both accurate and computationally efficient. In many cases, the solution has been to create new architectures that achieve a better accuracy/cost balance. Like most architectures, these new designs are *static architectures* that activate the entire network for each input, regardless of the input's characteristics. In contrast, *dynamic architectures* attempt to specialize the work performed during inference to properties of a particular input. For example, a cascade is a dynamic architecture that employs multiple models with different computational costs and uses low cost models to "early out" on easy-to-process inputs.

In this chapter, we explore a dynamic architecture, which we call a *HydraNet*, which achieves efficiency gains by dynamically determining which subset of the architecture should be run to best perform inference on a given input. In other words, a HydraNet retains accuracy by maintaining large capacity that can be specialized to aspects of the input domain. However, a HydraNet retains performance by only using a small fraction of this capacity when conducting inference on any one input. The HydraNet architecture template, shown in Figure 3.1, is a recipe for transforming state-of-the-art static network designs into conditional architectures that offer a better accuracy-per-unit-cost trade-off. We evaluate the HydraNet architecture in the context of image classification, where multiple branches of the network are specialized for different visual appearance. Although the idea of dynamically routing inputs specialized subnetworks is conceptually simple, successfully exploiting dynamic execution during inference requires subtle network design choices that balance the need to aggressively, but accurately, determine what subset of the architecture to run for a given task with the cost of making these dynamic decisions.

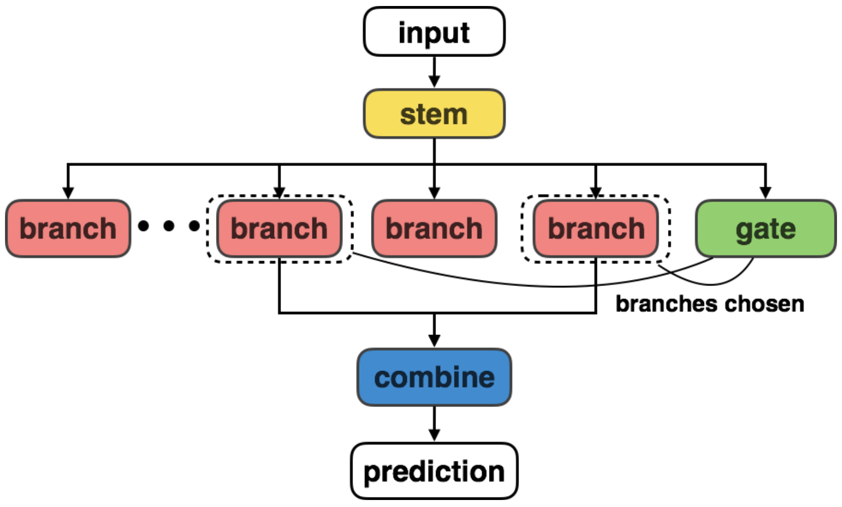


Figure 3.1: **HydraNet architecture template** supports conditional execution by specializing the branches to visually similar classes. During inference, only a subset of the branches relevant to each image chosen by the gate are executed reducing computational cost.

3.1 Related Work

Sparsity. The common approach to improving DNN efficiency is to enforce sparsity in network connectivity [86, 157, 24, 57]. This can be achieved via manual design of new DNN modules (Inception [140], SqueezeNet [62], MobileNet [57]) or via automated techniques that identify and remove the least important connections from a dense network [47, 46]. In either case, determination of the network topology is a static preprocessing step, and all connections are evaluated at the time of inference.

A complimentary optimization is to employ conditional execution at the time of inference to exploit sparsity in activations (skipping computation and memory accesses for model weights when activations are known to be zero). While attractive for reducing the energy consumption of DNN accelerators [45], fine-grained, per-element sparsity is difficult to exploit on CPUs and GPUs, which rely heavily on wide vector processing for performance. The subtask specialization we exploit in HydraNets can be viewed as a mechanism for designing and training a network architecture that, through coarse-grained conditional execution, is able to more effectively exploit dynamic sparsity.

Cascades. Cascades [147] are a common form of conditional model execution that reduces inference cost (on average) by quickly terminating model evaluation on inputs that are easy to process ("early out"). Region proposal [116] based models for object detection are canonical example of cascades in DNNs. Recent work has shown that integrating cascades into deep network architectures [35, 60] can improve the accuracy vs. cost of state-of-the-art architectures, where the later stages in a cascade specialize for difficult problem instances. Recent systems work [128, 58] also addresses the challenges in scaling up data analysis with deep models by using a using cascade like approach. Focus [58] uses a cheap model to first index the data during ingest and selectively uses a more expensive model during query time. The HydraNet approach of specializing network components for different subtasks is orthogonal and complementary to the benefits of cascades.

Mixture of experts. The idea of specializing components of a model for different subtasks is related to mixture-of-experts models where the experts are specialized for different inputs or tasks. Recent work on training very large DNNs for language modeling [125] has used conditional execution of experts for evaluating only a small fraction of experts for each training instance. One of the key aspects addressed in [125] is the design of the mechanism for choosing which experts to evaluate and trade-offs in network architecture to maintain computational efficiency. These design choices are tailored for recurrent models and cannot be directly applied to state-of-the-art image classification models which are feed-forward convolutional networks.

Hierarchical classification. Categories in ImageNet [30, 117] are organized into a hierarchy using an external knowledge base. The hierarchy can be used to first predict the super class and only perform fine grained classification within the super class [29, 31]. HD-CNN [160] is a hierarchical image classification architecture which is similar in spirit to our approach. HDCNN and Ioannou [75, 64] improve accuracy with significant increase in cost relative to the baseline architecture. In contrast, HydraNets on ImageNet improve top-1 accuracy by 1.18-2.5% with the same inference cost as corresponding baseline architectures.

Both HDCNN and Ioannou model the routing weights as continuous variables which are used to linearly combine outputs from multiple experts. Jointly learning the routing weights with the experts is similar to LEARN in Table 8, and performs poorly due to optimization difficulties (collapse and poor utilization of experts). HDCNN uses complex multi-stage training to mitigate optimization issues and provides robustness to routing error by overlapping the classes handled by each expert. HydraNets use binary weights for experts during both training and inference by dropping out all but the top-k experts. This enables joint training of all the HydraNet components while allowing flexible usage of experts.

Architectural structures similar to HydraNet [2] have been used for learning the partition of categories into disjoint subsets. Our main contribution is a gating mechanism which reduces inference cost by dynamically choosing components of the network to evaluate at runtime. Recent work [88, 99] has explored directly incorporating inference cost in the optimization and explore training methods for jointly learning the routing and the network features. In contrast to the complex training regime required for joint learning, our approach enables a simple and effective training strategy which we comprehensively evaluate cost on both ImageNet and CIFAR-100 datasets.

3.2 HydraNet Architecture Template

The HydraNet template, shown in Figure 3.1, has four major components.

- *Branches* which are specialized for computing features on visually similar classes. We view computing features relevant to a subset of the network inputs as a *subtask* of the larger classification task.
- A *stem* that computes features used by all branches and in deciding which subtasks to perform for an input.
- The *gating* mechanism which decides what branches to execute at inference by using features from the stem.
- A *combiner* which aggregates features from multiple branches to make final predictions.

Realizing the HydraNet template requires partitioning the classes into visually similar groups that the branches specialize for, an accurate and cost-effective gating mechanism for choosing branches to execute given an input, and a method for training all the components. The following sections describe how we address these key questions.

3.2.1 Subtask Partitioning

To create a HydraNet with n_b branches, we partition the classes into n_b groups of equal size. Similar to hierarchical classification we group visually similar classes so that branches can specialize in discriminating among these classes. While it might be possible to manually create groups when the number of classes is small, for large classification problems we need a mechanism for automatically creating visually similar groups. We compute a feature representation for each class by averaging the features from the final fully connected layer of an image classification network for several training images of the same class. Clustering these average class features using k-means with n_b cluster centers results in a partitioning of the feature space. Directly assigning each class to its nearest cluster center leads to an imbalanced class partitioning. Instead, each cluster center is assigned the class that is nearest to it and this process is repeated for $\frac{C}{n_b}$ steps, where C is the total number of classes, resulting in a balanced partitioning. Each of the n_b class partitions is assigned to one of the branches, which we refer to as a subtask.

3.2.2 Cost Effective Gating

Given a subtask partitioning, traditional hierarchical classification uses the gating function to classify among the subtasks and the branches for classifying among the classes within a subtask. For dynamic execution to be accurate, both the gating function and the chosen branch need to perform their respective classification tasks accurately. In practice, the accuracy of the gating function depends on the capacity (computational cost and parameters) of the gating component. Making the gating function highly accurate incurs significant computational cost negating the benefits of dynamic execution.

Our key insight in making the gating both computationally efficient and accurate is to change the function of both the subtasks and the gating function. In a HydraNet architecture, the branches do not perform final classification, instead they *only compute features relevant to the subtask assigned to the branch.* For example, a branch corresponding to a cluster of bird classes might compute features that distinguish between the different species of birds. Since the branches compute features instead of final predictions, executing more branches translates to computing more features that can be combined to make final predictions. Therefore, the job of the gating function is to choose which features (which branches) to compute. Since the gating function only needs to narrow down the final classification problem to determining which k subtasks to compute (rather than a single precise subtask), a lower capacity gating function is sufficient. Note that in the HydraNet template the *stem* computation is shared across different *branches* making the execution of k branches computationally efficient by design (Section 3.3.2).

We denote gating function score for an input I by $g(I) \in [0,1]^{n_b}$, the output of branch b by $branch(b)$, and indices of the top-k branches by $topk(g(I))$. The combined output of the branches $comb$ is given by:

$$comb(I) = \sum_{b \in topk(g(I))} branch(b) \tag{3.1}$$

We find that combining the features from the branches by concatenation works equally well in practice, but requires more memory and parameters than linear combination.

3.2.3 Training

Given a subtask partitioning, the stem, the branches, and the gating function of the HydraNet architecture are trained jointly. While direct supervision for branch outputs is not available, the labels for the gating function are given by the class to branch mapping determined by the subtask partitioning. The gating function and the downstream combiner indirectly provide supervision for the features that each branch needs to compute. Since the mapping of the class clusters to branch is fixed and determined by subtask partitioning, consistently using the features from the same branch for a subtask drives the features to specialize for the subtask. This approach is similar to the modular network approach for visual question answering [3, 69], where the questions define the set of modules to execute and a consistent mapping from questions to the modules encourages approximately learning the function designated to each module. The branches are indirectly supervised by the classification predictions after combining the features computed by the top-k branches.

Both the gating function and the branches are supervised using ground truth labels for image classification. During training, the top-k branches chosen for different inputs in the mini-batch can be different. We evaluate all branches on all inputs and mask out features from branches not picked by the gating function. In effect, the branch outputs from the branches other than the top-k for each input are ignored and not seen by the combiner. Masking ensures that weights of the branches not chosen by the gating remain unchanged during back propagation. This can also be view as an adaptive form of drop out [6] where entire paths are dropped instead of units within a layer. We use the cross entropy loss for both the gating function and the final predictions.

Model	Configuration		Params	MADDs	Accuracy
	d	w	($\times 10^6$)	($\times 10^6$)	(Top-1)
ResSep-A	2	0.50	1.96	181	61.88
ResSep-B	3	0.50	2.68	290	65.27
ResSep-C	2	0.75	3.98	380	67.16
ResSep-D	3	0.75	5.58	620	69.90
ResSep-E	3	1.00	9.53	1060	72.02
ResNet-18	-		11.69	1800	69.30
MobileNet	-		4.2	569	70.60

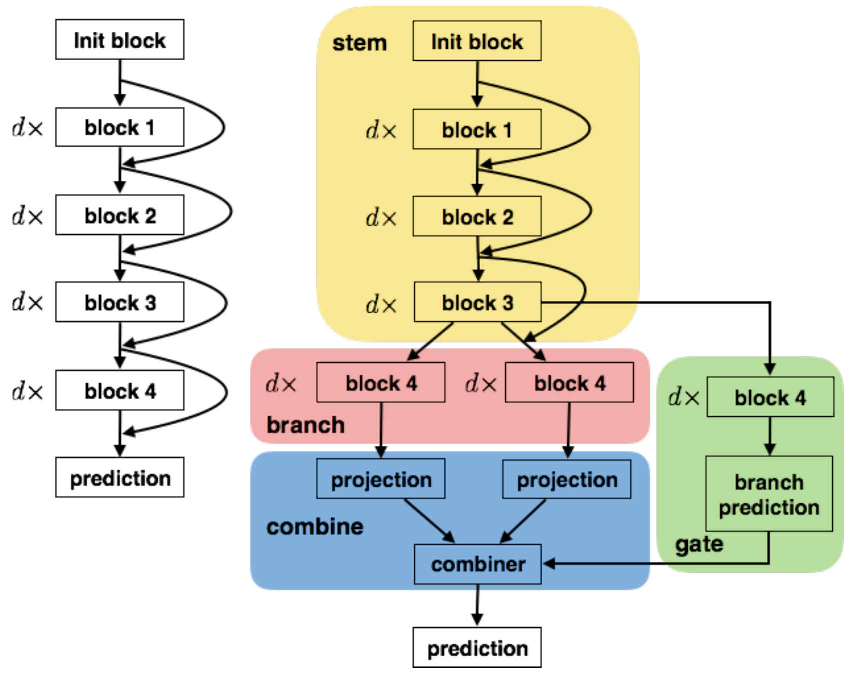
Table 3.1: Number of multiply-add operations and Top-1 ImageNet classification accuracy of variants of ResSep, ResNet-18, and MobileNet architectures for image classification. ResSep-D model is more accurate than Resnet-18, and 2.5× less expensive.

3.3 Architecture for ImageNet Classification

To analyze the computational cost and accuracy trade-offs enabled by HydraNet designs, we need a baseline architecture that is both computationally efficient at inference like MobileNet [57] and allows fast training like ResNet [52] in order to rapidly train multiple models in the design space. In response we created a static architecture, called ResSep which incorporates desirable architectural choices from both the MobileNet and the ResNet architectures, namely depth wise separable convolutions and residual connections.

3.3.1 Baseline Architecture

The high-level structure of the ResSep architecture (Figure 3.2) is similar to the ResNet architecture for ImageNet classification. There are four blocks (excluding the initial block) that can be stacked multiple times to create deep networks. Residual connections are added between the input and output of each block to accelerate training and provide regularization. Unlike ResNet, instead of using convolution layer for each of the blocks we use depth-wise separable convolutions, which have been shown to be effective at reducing computation costs with only minor reduction in accuracy [57]. The configuration of layers comprising each block in ResSep is shown in Figure 3.2. The amount of computation and number of parameters can be varied by scaling the number of filters in each block by the parameter w and the depth of the network by the stacking factor d. Table 3.1 compares the computation cost, number of parameters and Top-1 ImageNet classification accuracy of several variants of the baseline ResSep architecture to popular image classification architectures. The ResSep-D model is more accurate than Resnet-18, and is 2.5× less expensive; it also compares favorably to the MobileNet architecture which is highly tuned for low cost inference on mobile devices.



(a) Transforming the baseline ResSep architecture for ImageNet classification into a HydraNet architecture. Left: ResSep with separable convolutions in each block. The hyper parameter d is number of times each block is stacked. Residual connections are added from each block's input to its output. Right: the corresponding HydraNet architecture with branches, gate, and combiner components.

Name	Layers	Stride	Channels
Init block	conv 3×3	2	64
	max pool 3×3	2	64
Block 1	sep conv 3×3	[1, 2]	$128 \times w$
	sep conv 3×3	1	$128 \times w$
Block 2	sep conv 3×3	[1, 2]	$256 \times w$
	sep conv 3×3	1	$256 \times w$
Block 3	sep conv 3×3	[1, 2]	$512 \times w$
	sep conv 3×3	1	$512 \times w$
Block 4	sep conv 3×3	[1, 2]	$1024 \times w$
	sep conv 3×3	1	$1024 \times w$
Projection	conv 1×1	1	$1024 \times w$
Combiner	add		$1024 \times w$
	sep conv 3×3	1	$1024 \times w$
Prediction	avg pool 7×7	1	$1024 \times w$
	fully connected		1000

(b) Layer configuration for each of the block in the ResSep architecture. The parameter w controls the number of filters in each block. The first convolution layer in the first instance of each blocks uses stride 2 and subsequent instances use stride 1.

Figure 3.2: ResSep and the corresponding HydraNet Architecture for ImageNet.

Parameter	Description
d	Number of times each block is stacked. Controls overall network depth.
n_b	Total number of branches
k	Number of branches selected by the gating mechanism for an input image
w_s, w_b, w_g	Multipliers for controlling number of filters in layers of stem, branches, and gating components respectively

Table 3.2: Hyper parameters for exploring various trade-offs in the design of HydraNet architectures.

3.3.2 Design Space for HydraNet Transformation

Transforming a baseline static architecture into a dynamic HydraNet architecture involves several design choices for each of the components. We describe these choices in the context of ResSep, but the design space transforming ResNet [52] and DenseNet [61] architectures for CIFAR-100 classification is similar (Section 3.4.1).

The first of these choices is determining the architecture of the *stem* and the *branches*. Having a deeper *stem* allows sharing higher level features across the different branches or subtasks and allows features from deeper layers to be used to make more accurate gating decisions. On the other hand, making the gating decision after a deep *stem* diminishes the potential savings of dynamic execution. Therefore, deciding where to branch is crucial for retaining both computational efficiency and accuracy. We empirically observe that partitioning the first three ResSep blocks into the *stem* and replicating the fourth block in each of the *branches* gives better accuracy per unit computation cost.

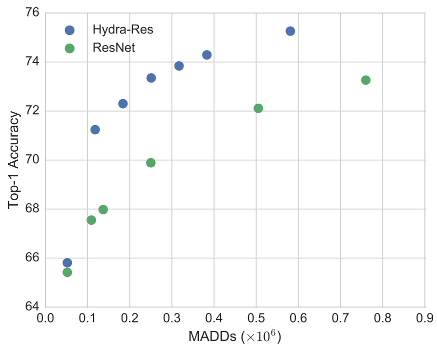
In designing the branches, directly replicating ResSep Block 4 into branches results in an inordinate increase in the number of model parameters and the number of floating point operations (by 3-4$\times$ for $n_b = 10$) during training, even for a small number of branches. Since training on ImageNet is computationally expensive, we retain the structure of Block 4 but scale the number of filters in each convolution layer by w_b to reduce training time. Balancing the computation in the *branches*, *stem*, and the number of branches is essential for improving computational efficiency of inference. The hyper parameters that govern this balance are listed in Table 3.2: n_b controls the number of branches, w_s and w_b control the number of filters in the layers in the *stem* and *branches* respectively. When transforming architectures for smaller datasets (CIFAR-100) where training is relatively cheap we do not scale down the number of filters.

The architecture of the gating function is the same as that of a branch, but followed by a prediction block. The gating function predicts the top-k branches relevant for the given input. Features from these branches are combined by first projecting them from a w_b-dimensional space into a w_s-dimensional space and adding them together. The *combiner* block uses the w_s features to make final class predictions. The hyper parameters w_g and k (Table 3.2) control the cost of the gating function and the number of branches dynamically evaluated for each input. We further elaborate on how these hyper parameter choices impact gating accuracy in Section 3.4.2.

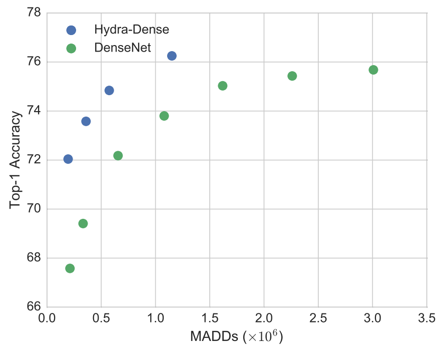
3.4 Evaluation

In this section, we demonstrate the effectiveness of HydraNet architectures by evaluating the computational cost vs. accuracy for image classification using both ImageNet and CIFAR-100 datasets. We use the total number of multiply-add (MADD) operation performed by the network during inference as measure of cost. Although the number of MADDs does not directly translate to inference runtime, it is indicative of the runtime and allows for comparing network cost independent of hardware and associated software implementation details.

We first analyze the computational cost and accuracy of HydraNet architectures based on two state-of-the-art architectures on CIFAR-100. We then use the ResSep architecture, which is designed to be computationally efficient for both training and inference, on the ImageNet dataset for evaluating the design choices in each component of the HydraNet template.



(a) ResNet architectures.



(b) DenseNet architectures.

Figure 3.3: Inference cost vs accuracy of HydraNet architectures for CIFAR-100 classification. HydraNet architectures have the same accuracy as the baseline ResNet and DenseNet architecture at lower dynamic cost.

3.4.1 CIFAR

We use the HydraNet template to transform ResNet (plain residual units) and DenseNet architectures for CIFAR-100 classification into Hydra-Res and Hydra-Dense architectures. The overall structure of the Hydra-Res and Hydra-Dense models is similar to that of the Hydra-ResSep architecture for ImageNet, however both ResNet and DenseNet for CIFAR-100 classification architectures have three blocks instead of four. We replicate the third block in each of the $n_b = 20$ branches and dynamically run $k = 4$ branches for each input. Unlike the Hydra-ResSep model for ImageNet, the number of filters in the stem, branches and gating function are not scaled down (w_b, w_s, w_g=1) from the design of the static baseline architecture. We create multiple variants of both Hydra-ResNet and Hydra-DenseNet by varying the depth (which is encoded in the variant name in Table 3.3). The hyper parameters k, n_b and the point to branch were chosen empirically by searching over

Model	Params ($\times 10^6$)	Inference MADDs ($\times 10^6$)	Accuracy (Top-1)
Hydra-Res-d1	1.28	52	65.81
Hydra-Res-d2	2.86	118	71.24
Hydra-Res-d3	4.43	184	72.30
Hydra-Res-d4	6.01	251	73.35
Hydra-Res-d5	7.59	317	73.84
Hydra-Res-d6	9.17	383	74.29
Hydra-Res-d7	10.74	449	74.71
Hydra-Res-d9	13.90	581	75.26
ResNet-14	0.17	52	65.42
ResNet-20	0.27	81	66.18
ResNet-26	0.37	109	67.55
ResNet-32	0.47	137	67.98
ResNet-56	0.85	250	69.89
ResNet-110	1.73	505	72.11
ResNet-164	2.60	760	73.26
Hydra-Dense-d2	2.77	195	72.04
Hydra-Dense-d3	5.43	360	73.58
Hydra-Dense-d4	8.97	574	74.84
Hydra-Dense-d6	18.72	1151	76.25
DenseNet-d3	0.36	212	67.58
DenseNet-d4	0.58	333	69.41
DenseNet-d6	1.19	655	72.18
DenseNet-d8	2.00	1080	73.80
DenseNet-d10	3.03	1619	75.03
DenseNet-d12	4.27	2260	75.43
DenseNet-d14	5.72	3008	75.68

Table 3.3: Computation cost and CIFAR Top-1 accuracy of several variants of HydraNet architectures and baseline architectures. The HydraNet architectures have the same accuracy as the baseline networks with much lower dynamic cost. All the HydraNet architectures have $n_b = 20$ branches and use $k = 4$ branches during inference.

$n_b = [5, 10, 20, 25]$, $k = [3, 4, 5]$ and branching at block 2 or block 3. The hyper parameter sweep was done using the smallest HydraNet architecture, Hydra-Res-d1 and Hydra-Dense-d2 in Table 3.3 and reused for the more expensive variants. We found that increasing the total number of branches (n_b) and dynamic branches (k) beyond 20 and 4 respectively had diminishing returns, while branching at block 2 delivered the better cost-accuracy trade-off.

The cost (in MADDs), number of model parameters and accuracy of variants of both Hydra-Dense and Hydra-Res are shown in Table 3.3. Both the Hydra-Res and Hydra-Dense architectures give the same level of accuracy as the static baseline architectures with significantly lower computational cost. For example, Hydra-Res-d4 yields the same accuracy as ResNet-164 while incurring one third the cost. As seen in Figure 3.3, similar trends hold for several variants of the HydraNet architectures. The improved accuracy of the HydraNet models is due to the overall increased capacity of the model, i.e., the total number of parameters and MADDs. However, at inference time the model dynamically picks a small subset of branches to execute; reducing the inference cost dramatically.

Model	Branch Point	d	w_s	Configuration w_b	w_g	k	n_b	Params ($\times10^6$)	MADDs ($\times10^6$)	Accuracy (Top-1)
Hydra-ResSep-A	Block 3	2	0.50	0.125	0.125	5	50	10.89	193	64.40
Hydra-ResSep-B	Block 3	3	0.50	0.125	0.250	5	50	13.30	299	68.13
Hydra-ResSep-C	Block 3	2	0.75	0.125	0.250	5	50	15.70	374	69.09
Hydra-ResSep-D	Block 3	3	0.75	0.125	0.500	5	50	19.51	630	71.52
Hydra-ResSep-E	Block 3	3	1.00	0.125	0.500	5	50	26.83	1075	73.20

Table 3.4: Computation cost, hyper parameters, and Top-1 accuracy several variants of the Hydra-ResSep architecture for ImageNet classification. All the architectures are transformed from corresponding baseline ResSep architecture.

3.4.2 ImageNet

As with CIFAR, we trained multiple HydraNets to explore the accuracy vs. cost landscape. However, due to the high cost of training large models on ImageNet classification we focus on a single baseline family of architectures (ResSep from Section 3.3.2). The ResSep-D model takes $\sim$ 12 hours to train with distributed asynchronous training using 32 GPUs. Relative to ResSep-D, the Hydra-ResSep-D architecture shown in Table 3.4 takes $\sim$ 2.5× longer to train. Training time and GPU memory make it impractical to explore large values for n_b and w_b. For example, setting $w_b = 0.5$ instead of $w_b = 0.125$ in the Hydra-ResSep-D architecture makes the training time $\sim$ 4× longer. The particular configurations in Table 3.4 were chosen to have similar inference cost as the baseline ResSep architectures (the hyper parameters d, w_s directly correspond to the baseline architectures and n_b, k, w_g are balanced such that the dynamic inference cost closely matches the corresponding baseline). We further elaborate on the choice of hyper parameters in the rest of the section with associated experiments.

HydraNet architecture improves accuracy while having similar computation costs as the baseline ResSep architecture. Figure 3.4 shows the cost vs. accuracy of both Hydra-ResSep and the baseline ResSep architectures. As shown in Table 3.4, the HydraNet architectures are more accurate, and have more parameters compared to the baseline ResSep architectures with similar inference cost, especially in the low-cost regime. However, at inference time only $k = 5$ of the $n_b = 50$ branches are evaluated, making the inference cost comparable to the corresponding baseline architecture. The accuracy improvements of HydraNets in the low cost regime are more pronounced. We believe this is because of the increase in total capacity (parameters and total static MADDs) of the HydraNet architectures relative to the baseline is larger in this regime (HydraNet-ResSep-A has 5.6× more parameters and 3.3× more MADDs). In the high accuracy regime, the accuracy improves by 1.18% relative to the baseline (HydraNet-ResSep-D has 2.8× more parameters and 1.5× more multiply-adds).

Increasing the total number of branches results in better accuracy and increases training costs. In order to analyze the impact of the number of branches (n_b) we vary n_b while keeping the other hyper parameters the same as HydraNet-ResSep-A (Figure 3.4).

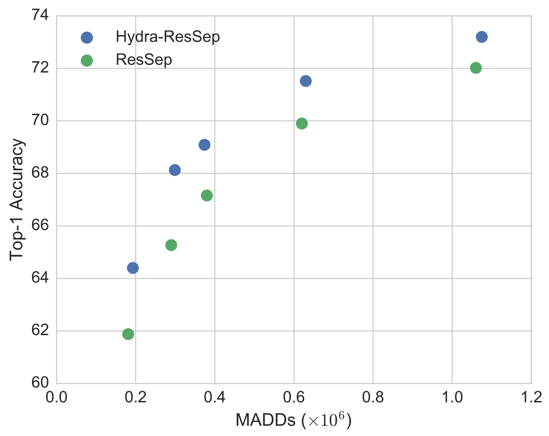


Figure 3.4: Cost vs. accuracy trend for Hydra-ResSep and the baseline ResSep architectures.

Total branches	$n_b = 10$	$n_b = 25$	$n_b = 50$
Accuracy (Top-1)	61.76	63.21	64.40
Params ($\times 10^6$)	3.30	6.14	10.89
Total MADDs ($\times 10^6$)	239	376	605

Table 3.5: Increasing number of branches (n_b) gives better accuracy but increases training cost and the number of model parameters. Total MADDs is the cost of all branches which is proportional to training time. Other hyper parameters are the same as Hydra-ResSep-A in Table 3.4.

Increasing the number of branches increases the total capacity of the network and allows for more specialization, but makes learning the subtask gating function harder. However, since we rely on a soft top-k gating mechanism, increasing the total number of branches results in a net improvement in overall accuracy as shown in Table 3.5. Note that increasing the total number of branches increases the training cost, but not inference cost. Training cost and available GPU memory make it impractical to scale n_b and w_b to large values. We set $n_b = 50$ and $w_b = 0.125$ to keep the training time under ~ 2 days (asynchronous training with 32 GPUs) for all the variants of HydraNet-ResSep.

Dynamic branches	$k = 1$	$k = 3$	$k = 5$	$k = 10$
Accuracy	60.08	63.38	64.40	64.89
Inference MADDs ($\times 10^6$)	157	175	193	239

Table 3.6: Increasing the number of dynamically executed branches (k) of HydraNet-ResSep-A results in better accuracy with diminishing returns.

Split point	Params	MADDs ($\times 10^6$) Total	Cond	Accuracy
Block 2	9.66	695	174	61.35
Block 3	9.66	605	193	64.40

Table 3.7: Branching at Block 2 of ResSep-A gives lower accuracy compared to branching at Block 3. All other hyper parameters are the same as Hydra-ResSep-A in Table 3.4.

Method	Accuracy
RANDOM	63.07
LEARN	61.50
LEARN-BALANCE	63.17
CLUSTER	64.40
NO GATING	66.21

Table 3.8: Accuracy of HydraNet-ResSep-A with different subtask partitioning and gating strategies. Other hyper parameters are the same as in Table 3.4 except for *no gating* which uses outputs of all the branches.

Executing a small number of branches dynamically improves accuracy significantly relative to executing a single branch. Table 3.6 shows the importance of the soft-gating mechanism by varying k while keeping other hyper parameters the same. Dynamically executing only a single branch results in lower accuracy than the ResSep-A model. This is due to inaccuracy in gating as well as the low capacity of a single branch. However, executing a small number of branches recovers accuracy and outperforms the ResSep-A model by 2.5%. There are diminishing improvements in accuracy with increasing k. We set $k = 5$ for all the HydraNet-ResSep architectures.

Branching earlier in the network hurts accuracy with only a slight reduction in dynamic compute cost. An important aspect of transforming a baseline architecture using the HydraNet template is deciding where the model is partitioned into branches. Table 3.7 shows the accuracy when the baseline architecture ResSep-A is transformed into a HydraNet architecture by branching at Block 2 (each branch and the gating function now has both Block 3 and 4 with the number of filters scaled by w_b and w_g respectively). Branching earlier in the network increases the overall training cost and results in significantly lower accuracy. We believe this is due to inaccurate decisions by the gating function when relying on low-level features. In principle, one could empirically search for the best split point at the layer granularity rather than block granularity. We restricted our experiments to block granularity to limit the time spent on this search.

Clustering based subtask partitioning works well in practice. We evaluate the effectiveness of clustering based subtask partitioning by comparing with several baselines. Table 3.8 shows different methods for partitioning and their corresponding accuracy. CLUSTER is the k-means based subtask partitioning described in Section 3.2.1. RANDOM partitions the classes into equally sized subsets randomly. NO GATING does not require subtask partitioning since all the branches are used both at training and inference, and serves as an upper bound on the accuracy that can be achieved since any gating function will utilize a subset of the branches per input. CLUSTER is 1.33% more accurate than RANDOM demonstrating that partitioning the classes based on visual similarity allows the branches to specialize more effectively. The accuracy gap between CLUSTER and NO GATE is 1.8%, which is narrow given that CLUSTER only uses five branches out of the total 50 per input.

We also explored alternative methods to CLUSTER which learn the subtask partitioning jointly with the training of the branches. Instead of supervising the gating function with a branch classification loss, LEARN uses the scores generated by the gating function as weights for linearly combining output features of the top-k branches. The gating function learns to weigh the relevant branches higher for a given input based on the overall classification loss. Table 3.8 shows that LEARN gives lower accuracy than the baseline. We observe the gating function tends to favor branches which initially get more training signal. This results in a self-reinforcing loop: branches which are not picked do not learn to compute useful features and the gating function learns to ignore these branches. Prior work in language modeling has overcome the under utilization problem by adding a loss term encouraging equal utilization of branches for a batch of inputs [125]. Given a batch size of N and the gating function $g(\text{batch\_id}, \text{branch\_id})$, the utilization of the branch b across the batch is given by $\sum_{i=0}^{N} g(i,b)$. We encourage equal utilization of branches by adding an additional loss term $\lambda_{bal} \sum_{b=0}^{n_b}(\sum_{i=0}^{N} g(i,b))^2$, this corresponds to the LEARN-BALANCE row in the table. LEARN-BALANCE does significantly better than LEARN but still falls short of CLUSTER.

3.5 Discussion

The HydraNet architecture reduces computational cost by specializing components of a network for subtasks and exploiting this specialization at inference time. We summarize our approach as answers to the questions we need to address for effective specialization.

- *What is the model being specialized for?* The branches in the HydraNet model are specialized for computing features to discriminate between groups of visually similar looking classes.
- *How to curate a dataset specialized to the context?* The data and supervision required for specializing the branches is generated by clustering the training data using a general feature representation.
- *What are model architecture and training algorithm changes required to induce and exploit specialization?* The HydraNet template is designed to support conditional execution and the training algorithm makes the branches specialize for different groups of classes.
- *How do we identify when a specialized model is applicable?* The gating function determines the set of branches that should be picked based on the input.

Thinking in terms of these questions organizes the solution space for each key component enabling specialization. For instance, clustering using general feature representation trained using supervised data is one way of generating the class grouping and labels for supervising the gating function. With more recent advances in self-supervised learning one can revisit both how to generate the visual grouping as well as the supervision for gating and specializing branches. Similarly, more recent work has show that it is feasible learn the grouping classes each branch jointly with the gating function which is a training method for inducing specialization.

HydraNet architectures enable more efficient inference than static architectures, this efficiency comes at the cost of longer training times due to the network's many branches. One approach to reducing the training time is to leverage dynamic execution in the training process. Unlike inference, training is sensitive to batch size and efficient dynamic execution in small batch settings is challenging since each item in the batch may need to execute a different set of branches. Current deep learning frameworks are not designed for efficient execution in these scenarios. Extending deep learning frameworks with support for conditional training would be necessary to scale up training of conditional models like HydraNet. Although, we focus on classification we believe that sparse dynamic execution will be increasingly important for building multi-task models which perform a wide range of tasks using a shared representation. Especially with the current trend of training larger and more accurate models on increasing amounts of data.

Chapter 4

Specialization to Individual Categories for Efficient Supervision

Training image classification models for a single (or small number of) rare categories is common in real-world settings and is a natural scenario where models need to target a small subset of the general distribution. For instance, autonomous vehicle development requires recognizing rare entities, like construction vehicles, in the video logs collected by the fleet. Ecological monitoring requires recognition of rare animal species. A major challenge in building models for rare categories is acquiring training examples - precisely because they are rare!. Fortunately, in many real-world settings one has access to a large amount of *unlabeled* data. In this chapter, we focus on leveraging models specialized to small amounts of human labeled data to continuously guide human and machine effort in improving classification models for rare categories.

Because naive labeling is likely to waste human effort on easy background examples, a natural strategy is *to interactively mine* the data by combining [98, 104, 44, 121, 150, 134] active [122] and semi-supervised [19, 165, 136, 156] learning. However, most prior active and semi-supervised learning approaches assume upfront access to a modest amount of labeled data and assume that every image belongs to a common class of interest (Figure 4.1). In contrast, we are interested in building models for rare categories starting with only a few labeled positives (as little as 5) and where most of the unlabeled data (as much as 99.9%) is background. Simply put, our goal is to maximize model accuracy for rare categories given a small fixed amount of human labels (500–1000).

Typical active and semi-supervised approaches use an initial labeled set to train a model for identifying and labeling relevant data. However, we have so few labeled positives for each rare category that it is difficult to train a reliable deep model using only the initial labeled set. Prior work in the rare category setting resorts to training simple linear models on pre-defined features [111, 26]. We show that it is feasible to improve deep features with a limited amount of labeled data by integrating active and semi-supervised learning to address the following challenges:

- Because humans can only label a small fraction of the full dataset, we make use of semi-supervised learning to pseudo-labels additional examples. Pseudo-labeling positives works poorly because they are rare. **We let humans label "hard" examples**

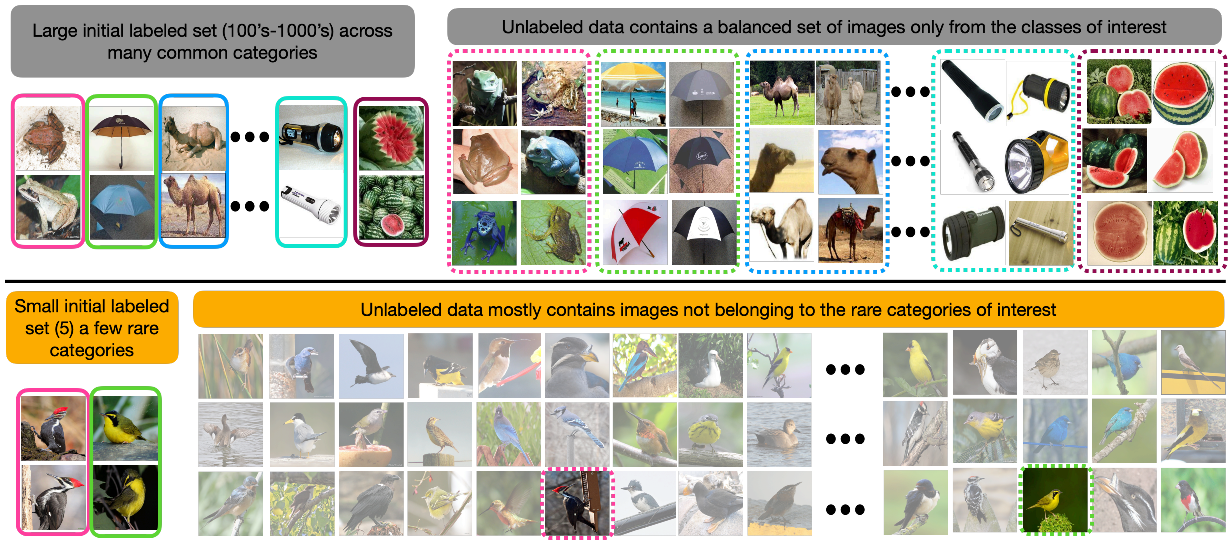

Large initial labeled set (100's-1000's) across many common categories
Unlabeled data contains a balanced set of images only from the classes of interest
Small initial labeled set (5) a few rare categories
Unlabeled data mostly contains images not belonging to the rare categories of interest

Figure 4.1: **Problem setup:** Top half shows the setting widely studied in active and semi-supervised literature; where the goal is to build a model for many common categories starting from 100's-1000's of labeled images using unlabeled data that is balanced and only contains images belonging to the categories of interest. Bottom half shows our setting that is commonplace in many practical scenarios; where the goal is to build a model for a few rare categories starting from a small number of exemplars using unlabeled data that mostly (as much as 99.9%) does not belong to the categories of interest.

that may contain positives and hard negatives and let machines pseudo-label "easy" negatives.

- Training on large amounts of pseudo-labeled negatives leads to difficult, highly-imbalanced learning. **We adapt background splitting [103], a recently proposed technique for learning from highly imbalanced human labeled datasets, for iterative semi-supervised learning with both human and machine labels.**
- Iteratively learning (deep) features significantly improves accuracy but is computationally expensive and adds significant delays between model updates and human labeling. **To reduce overall computational costs and latency between model updates and human labeling, we update the features at a low frequency but train linear models on cached features at a high frequency to pick the samples to query humans on.**

4.1 Related Work

Our work overlaps with research in *active learning*, *semi-supervised learning*, and the intersection of both of these methods with *deep learning*. Broadly unlike prior work, we explore semi-supervised active learning in the context of learning deep models for a small number of rare categories starting with a few labeled positives and a large amount of unalabeled data mostly consisting of images not relevant to the categories of interest.

Active learning The problem of choosing data to label to learn a model has be extensively studied in active learning [122] literature. Much of the early work has focused on training a classifier on top of pre-defined features where the unlabeled dataset has a *balanced* category distribution, and the goal is to build a model to classify all categories. State-of-the-art methods in this setting label data that the current classifier model is confused about (as measured by entropy or margin distance) [25] or data with the most information gain, often measured by the expected model gradient [122, 149]. When unlabeled data is balanced simple strategies like uniform random sampling are quite competitive [100]. Prior work has shown that uniform sampling as well as standard active learning techniques struggle in our imbalanced setting [5, 4]. Some work has explored active learning in the imbalanced/rare category setting [155, 13, 1], where unlabeled data only contains categories of interest and the imbalance studied is minor (2-99×) compared to the setting we study (384-10,000×).

Our work is most closely related to the Tropel system [111], which trains a rare-category classifier via active learning starting with just a few samples of the rare class. Tropel uses a fixed feature representation from a pre-trained deep model for classifier training. Similarly more recent work trains a logistic regression classifier on top of frozen deep features by limiting candidates for human labeling to the nearest-neighbors of the samples that have already been labeled [26]. In contrast, we update the deep feature representation used in the active process. Even though we have only a small amount of human labeled data, we make this update viable by also utilizing the unlabeled data.

Deep active learning & semi-supervised learning In traditional active learning methods, a classifier is learned on top of fixed features, whereas when training a complete deep model from actively-acquired samples the goal is to learn both a feature representation and a classifier. This difference requires fundamental changes to active learning techniques. Recent work [100] demonstrates that active learning with deep models in the common category setting benefits enormously from data augmentation or semi-supervised techniques using the unlabeled data. Therefore, straightforward adaptations of traditional active learning methods to training deep models [121, 49] are less efficient in their use of human samples than methods which augment deep active learning with *semi-supervised learning* [150, 134, 108, 39]. These combined techniques build on earlier work from the pre-deep-model era [98, 104, 44, 81].

Semi-supervised methods rely on an existing model to infer labels on the unlabeled data either using *proxy-labeling* or *graph-based methods* [108]. The existing model used to infer labels must be reasonably accurate for the methods to successfully improve the model. Most prior semi-supervised methods are evaluated in the many common category setting and start with 100's-1000's of labeled examples to train a good initial model [108, 65, 115]. However, with rare categories it is unrealistic to start with such a large number of labeled positive instances. In the realistic scenario that trains an initial model with just a few positive instances, we find that the model's *positive* predictions on unlabeled data are unreliable (the false positive rate is high). Therefore, we limit semi-supervised labeling to just a subset of the samples strongly predicted to be *negative*, confirming previous results in the classical setting [23]. The semi-supervised method that we use is a modified variant of the relatively simple *proxy-labeling* method [108].

Knowledge transfer and representation learning Techniques like knowledge distillation [21, 159, 115] enable transferring knowledge from a model in a related domain to learn better representations. We rely on a recent distillation based technique developed specifically for the rare category settings with extreme imbalance [103]. More recently self-supervised representation learning techniques have been successful for balanced datasets like ImageNet [16, 43, 21, 22]. We find that representations learned using such methods are good starting points but they need to be updated with supervision relevant to the rare categories of interest.

Few-shot learning Much work on active learning assumes a small but sizable portion of data is labeled (e.g., 5%), enough to *warm-start* with a reasonable model [129, 59]. In our setting, we assume only a small number of initial examples per class (5). This setting is close to that of a *cold start* in active learning [40, 68]. Our initial setup is similar to a few-shot learning problem, which often focuses on learning general representations [135] or aggressive data augmentation [154]. Rather than focusing on complex few-shot learning strategies our approach focuses on learning a good feature representations using unlabeled data echoing recent work [142].

Heterogeneous models for active learning Actively training deep models produces more accurate models but incurs a high computational cost. Prior work has shown that simpler models such as linear classifiers can be trained quickly compared to complex models such as deep CNNs, and are often good enough for sample selection in active learning [80, 27]. We use a similar approach and interleave linear classifier training with deep model training to retain advantages of both.

4.2 Approach

Our goal is to build an accurate model for a rare category given the following inputs: a small set of labeled positives (I_p); a large amount of unlabeled data (>99.9% of data is negative instances) from the target domain U; a pre-trained model M_{pre} from a related domain; and the ability to query a human for labels on a set of samples. Figure 5.2 provides an overview of our approach and Algorithm 2 gives concrete details. At a high level our approach iteratively builds a model by utilizing the best feature representation (F) and labeled positives (L_p), negatives (L_n), and pseudo negatives (W_n) available on hand to rapidly train linear models. Then we use the linear models to rank the unlabeled images and to query humans for labels on the top images in the ranking R. In addition to human labels, the linear model is used to accurately pseudo label a large number of *easy negatives* (W_n) by picking images from the bottom 50% of the ranking. Using fixed features not trained for the task limits the accuracy of the linear models. Therefore, the deep features are periodically updated at a slower rate by training the deep model (M^d) with both human

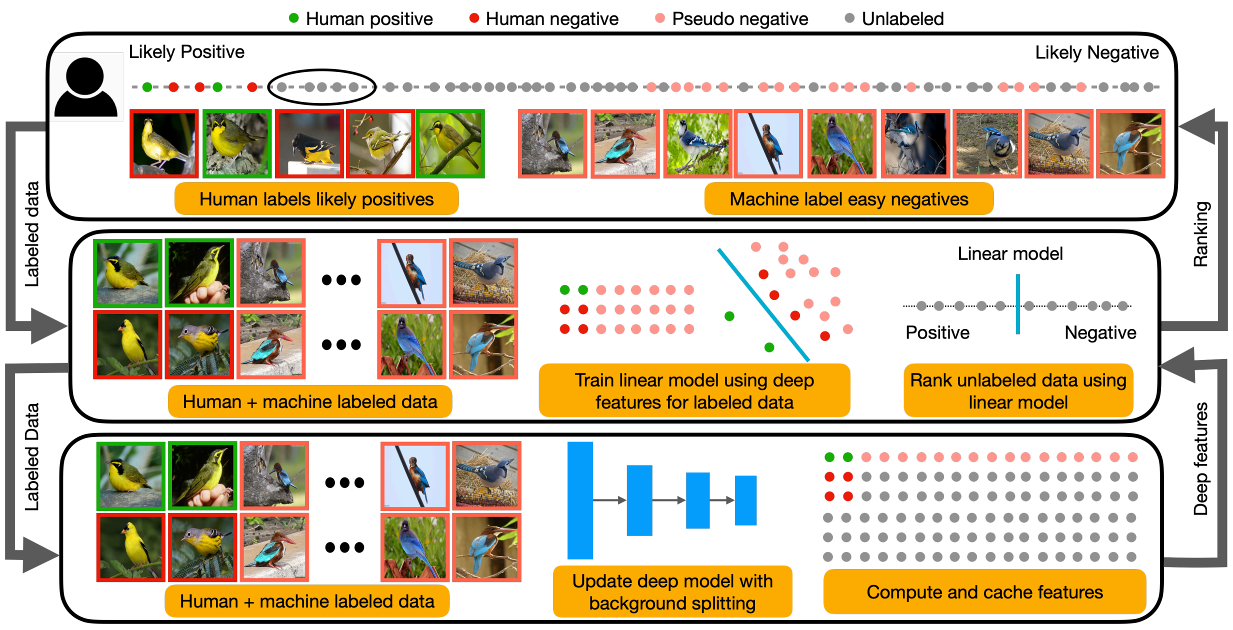

Human positive
Human negative
Pseudo negative
Unlabeled
Likely Positive
Likely Negative
Human labels likely positives
Machine label easy negatives
Labeled data
Ranking
Human + machine labeled data
Train linear model using deep features for labeled data
Linear model
Positive
Negative
Rank unlabeled data using linear model
Labeled Data
Deep features
Human + machine labeled data
Update deep model with background splitting
Compute and cache features

Figure 4.2: **Overview:** Our approach builds rare category models by alternating (top cycle) between selecting unlabeled samples for a human to label and training a linear model on top of frozen deep features from labeled images. Our pipeline focuses human effort on images likely to contain positives by ranking unlabeled images with the linear model and querying humans on high scoring unlabeled images. In addition to human labels, the linear model is used to automatically label a large number of *easy negatives* by picking images from the bottom 50% of the ranking. Using fixed features not trained for the task limits the accuracy of the linear models. Therefore, the deep feature representation is periodically updated at a slower rate (bottom cycle) by training the deep model with both human and automatically labeled data.

Algorithm 2: Our rare category active approach.

```
Input: $U$, $I_p$, $M^d_{pre}$, $B$, $N$, $Q$, $f_a$
Output: $M^d_{1...N}$
1  $F \leftarrow$ cacheFeatures($M^d_{pre}$, $U$)
2  $A_l \leftarrow$ auxiliaryLabels($M^d_{pre}$, $U$)
3  $c \leftarrow$ computeFeatureCentroid($M^d_{pre}$, $I_p$)
4  $R_{pos} \leftarrow$ rankByCosineSimilarity($c$, $F$)
5  $L_p \leftarrow I_p$, $L_n \leftarrow \{\}$, $W_n \leftarrow \{\}$
6  $M^d_0 \leftarrow M^d_{pre}$
7  for $i \leftarrow 1$ to $N$ do
8      for $j \leftarrow 1$ to $Q$ do
9          if $|L_p| < |L_n|$ then
10             $H_p, H_n \leftarrow$ queryHumanTopK($B$, $R_{pos}$)
11         else
12             $H_p, H_n \leftarrow$ queryHumanTopK($B$, $R_{ent}$)
13         $L_p \leftarrow H_p \cup L_p$, $L_n \leftarrow H_n \cup L_n$
14         $W_n \leftarrow$ pseudoNegativeLabels($R_{pos}$, $f_a$)
15         $M^l \leftarrow$ trainLinearModel($F$,$L_p$,$L_n$,$W_n$)
16         $U \leftarrow U - (H_p \cup H_n)$
17         $R_{pos} \leftarrow$ sortPositiveScore($M^l$, $U$)
18         $R_{ent} \leftarrow$ sortMarginDistance($M^l$, $U$)
19     $M^d_i \leftarrow$ trainBGSplit($M^d_{i-1}$,$L_p$, $L_n$, $W_n$, $A_l$)
20     $F \leftarrow$ cacheFeatures($M^d_i$, $U$)
```

and automatically labeled data. We use the recently proposed background splitting [103] technique to learn good representations even with the extreme imbalance encountered in the training data. The rest of the section elaborates on the motivation and the resulting design choices in the key components of our approach.

Bootstrapping the initial model and ranking. The first challenge we need to tackle is building a model for ranking the unlabeled data using the starting set of positives we call the prototype positives (I_p). Given only the prototype positives it is difficult to train a classifier since there are no negatives on hand. One could sample negatives randomly from the unlabeled data, but we find that randomly sampled negatives are not effective for building an accurate classifier. Instead it is more effective to rely on feature similarity to the positive instances for finding hard negative examples. Therefore, we generate a feature-similarity based ranking using the prototype positives. `cacheFeatures` (Line 1) uses the pre-trained model M^d_{pre} to pre-compute the features F for all the samples in the unlabeled data U. Then `computeFeatureCentroid` (Line 3) normalizes the feature representations of all the prototype positive examples I_p and computes an average feature representation c (category prototype) for the rare category. Finally, `rankByCosineSimilarity` (Line 4) generates the initial ranking R_{pos} of unlabeled images in U by sorting them by cosine similarity to the category prototype c.

Querying for human labels on high-ranking samples. Given the initial bootstrapped model c or the linear models (M^l) in later iterations we need to choose which unlabeled images to present to the human for labeling. The initial ranking (Line 4) is based on similarity and surfaces data that is visually similar to the category of interest. However, the ranking can be unreliable (especially for fine-grained categories) since the similarity is computed using only a few positive instances (I_p) based on a representation learned for a different task (ImageNet classification or a self-supervised auxiliary task). Even in subsequent iterations we only request a small number of $(B = 10)$ additional labels every iteration to update the ranking model (M^l). For instance, Figure 5.2 top shows the highest ranking unlabeled images during the active loop. Most of the top images surfaced by the model are visually similar, but only a small fraction of those images belongs to the category of interest making it difficult to automatically label positives. Therefore, we focus human labeling effort on the top scoring positive images in the ranking.

Each iteration of the inner loop j, `queryHumanTopK` (Lines 10,12) queries the human for labels on the B top ranked images, where B is the labeling budget per iteration. The human labeled positives and negatives H_p, H_n are added to the appropriate labeled sets L_p, L_n and removed from the unlabeled data pool. As we request more labels in the iterative process and update the model, the model becomes more accurate and scores easy positives highly. Once the model is reasonably accurate, labeling more easy positives doesn't improve the model much, but the model is then able to identify samples near the true margin (hard positives and hard negatives). Based on this intuition, we use a simple heuristic to estimate model quality and adaptively choose a strategy for picking the samples for human labeling. The key insight is that a high quality model finds more positives than negatives. Our approach keeps track of the number of positives and negatives labeled so far $(|L_p|, |L_n|)$. If the number of positives labeled so far is less than the number of negatives, we ask the human to label the most-likely-positive images (R_{pos}), otherwise we ask the human to label the images closest to the linear model's margin (R_{ent}).

Generating negative pseudo labels on low ranking samples. Given only the small number of human labeled images $(|L_p|, |L_n|)$ it is difficult to improve the deep feature representation. Our approach addresses this issue by machine labeling a large subset of the unlabeled images U. Firstly, our approach avoids labeling errors by restricting pseudo labels to negatives and more precisely the low ranked samples in the current ranking R_{pos} which are unlikely to be positives. Secondly, our algorithm takes a progressive approach to machine labeling based on the number of human-labeled positives and negatives (L_p and L_n). The routine `pseudoNegativeLabels` (Line 14) chooses $f_a(|L_p| + 0.1|L_n|)$ samples uniformly from the bottom half of the ranking as pseudo negatives. f_a is a scalar hyper parameter that controls the amount of pseudo labels relative to the human-labeled data the algorithm uses. The goal of this heuristic is to keep the label errors relative to the amount of human-labeled data low while maintaining diversity in the negatives. Although easy negatives can be automatically labeled reliably with simple heuristics, similar heuristics cannot be used to generate automatic labels for positives of the rare category. We show that even conservatively pseudo labeling positives results in lower accuracy in the active process compared to our approach.

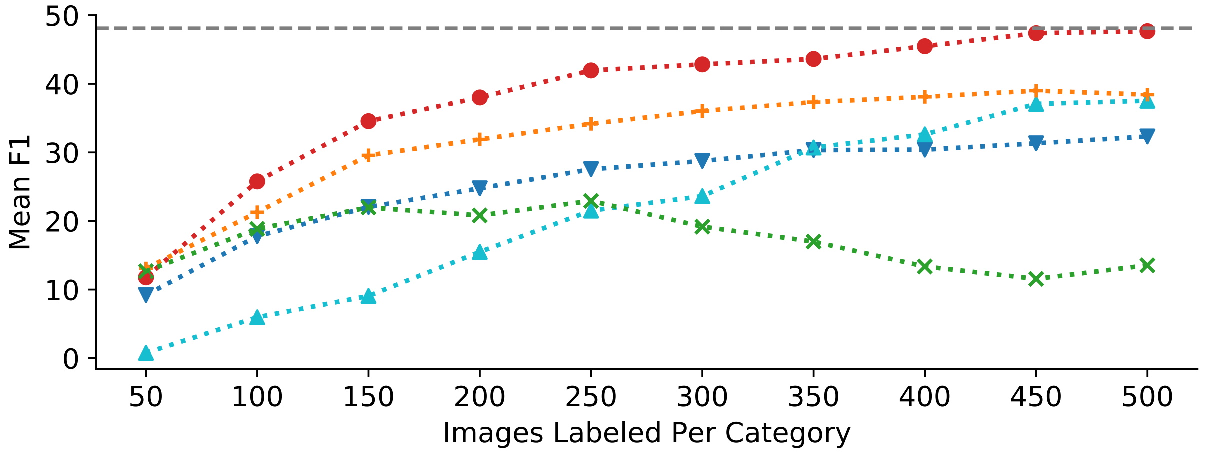
Updating feature representation with extreme imbalance. Training the deep models M_i^d with a small number of human labeled images and a large number of pseudo labeled negatives remains challenging due to the extreme imbalance in the training data. Our approach mitigates this issue while distilling information from a related domain by using a recent deep-model training technique called background splitting [103] designed for coping with extreme imbalance. **trainBGSplit** (Line 19) trains the deep representation using all the human-labeled and pseudo-labeled data, plus auxiliary labels A_l computed using the pre-trained model M_{pre}^d.

Interleaved feature representation and ranking updates. In principle, we can update the deep model with every human label to make best use of subsequent human effort. However, updating the deep representation frequently is computationally expensive and gives diminishing returns as we show in our experiments. Conversely, training linear models on fixed features is computationally cheap but results in lower quality models. Our approach strikes a balance between the two extremes by interleaving low frequency updates of the feature representation (every QB human labels) with high frequency training of a linear model (every B human labels) used to update the ranking. The outer loop i corresponds to feature updates and the inner loop j corresponds to ranking updates with a fixed feature representation. The parameters N and Q specify the number of feature and ranking update iterations respectively. The routine `trainBGSplit` starts with the model from the previous iteration and only trains it for a few epochs on the current labeled data (L_p, L_n, W_n). Continuous training of the deep model across the active iterations keeps the training cost close to that of training a fully supervised deep model on all the data. After every update the features for the unlabeled data are cached (Line 20) for training linear models used to choose images for human labeling. In between the feature updates, the routine `trainLinearModel` uses the current features to rapidly train a linear model (M^l) using both human and pseudo negative labels. The resulting linear model is used to update the ranking used to pick images for human labeling as well as pseudo negatives.

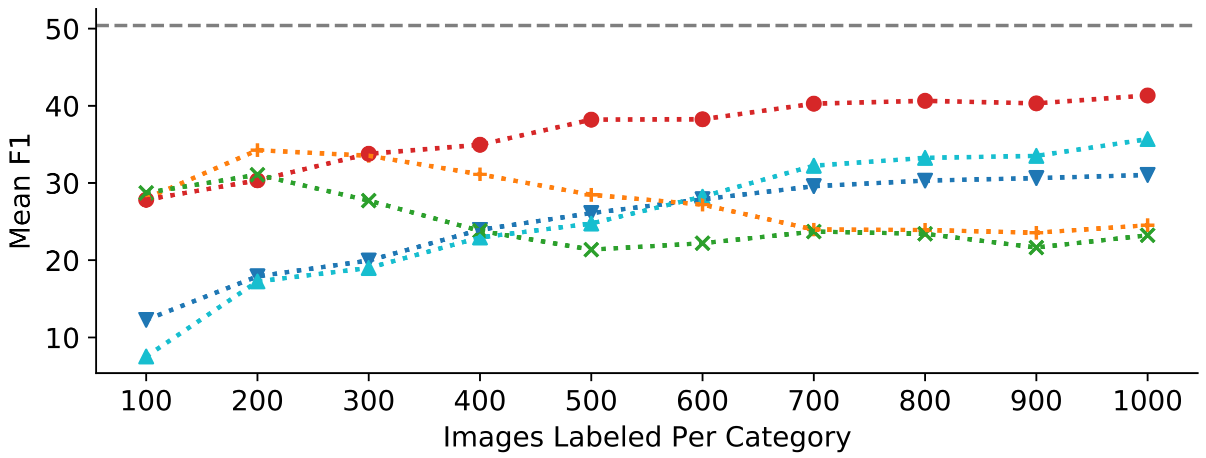
4.3 Evaluation

Our approach has three components: active sampling, pseudo labeling and deep model training on imbalanced data. Prior work has largely focused on individual components of our approach. We compare our approach to prior work by replacing individual components with prior methods and evaluating the overall performance. We also present ablation studies for key ideas.

Approach		Pseudo Positives	Pseudo Negatives	BG Splitting	Train Features	Active Learning
Ours	🟥 ∘	✗	✓	✓	✓	✓(adaptive)
KD-Semi	🟧 +	✓	✓	✓	✓	✓(adaptive)
DeepProp	🟩 ×	✓	✓	✓	✓	✓(adaptive)
Tropel-Deep	🟦 △	✗	✗	✗	✓	✓(most-likely-positive)
Tropel	🟦 ▽	✗	✗	✗	✗	✓(most-likely-positive)
Fully supervised	⬛	✗	✗	✓	✓	✗



(a) iNaturalist 50 categories (each is ∼0.01-0.26% of data)



(b) Places 20 categories (each is ∼0.19-0.27% of data)

Figure 4.3: **Comparison to baselines.** Plots show average F1 accuracy of our approach and baselines on iNaturalist (5 × 10) and Places (2 × 10) categories as a function of human labeling effort. **Our** approach gives higher accuracy per human effort compared to baselines, especially when a small number of images are labeled. The **Tropel** baseline trains a linear model on fixed features and is competitive with **Tropel-Deep** baseline in the low data regime. **DeepProp** and **KD-Semi** baselines pseudo label positives in addition to negatives. The false positives in the pseudo positive labels leads to lower accuracy compared to our approach.

4.3.1 Experimental setup

We evaluate our approach for building rare category models using two datasets: the iNaturalist [144] fine-grained species recognition dataset and the Places [162] scene classification dataset. We randomly choose 50 categories from the iNaturalist dataset and 20 categories from the Places dataset for evaluation as rare categories. For the iNaturalist dataset we restrict the random choice to categories with more than 50 instances in the training set, so that the active learning process has something to find. Given this setup, positive instances for individual categories are extremely rare in both datasets. For iNaturalist, the number of positives for each selected category is in the range 50-1500 (∼0.01-0.26% of the training set). For Places, the categories are more uniformly distributed, with ∼3500-5000 instances per category (∼0.19-0.27% of the training set). For each category we pick five randomly chosen positives as the initial labeled set and treat the rest of the training data as the unlabeled pool of data. We evaluate the trained models on the full validation set of each dataset. All images not in a category of interest are considered negatives in our evaluation i.e., our validation set reflects the imbalance in the training set. We use the F1 metric for evaluation instead of classification accuracy since classification accuracy is easily gamed for rare categories by just predicting everything as negative.

Rare-category active learning baselines: **Tropel** [111] is a prior active learning approach for rare categories. **Tropel** iteratively trains a linear model (on top of frozen pre-trained deep features) by labeling the top-k scoring predictions from the current linear model. We also evaluate a new variant, **Tropel-Deep**, which trains a deep model instead of a linear model at each iteration. **Tropel-Deep** uses the same incremental deep-model training that we use in our approach.

Semi-supervised baselines As the results show, our method outperforms **Tropel** (because we train a deep model) and **Tropel-Deep** (because our method augments the human labels with pseudo-labels). This second result raises the question as to whether traditional semi-supervised techniques could perform just as well as our enhanced methods. To study this question, we implemented additional baselines based on label propagation [65, 134] and knowledge distillation [21, 159, 115]. We replace our pseudo negative labeling approach with these approaches for generating automatic labels, naming them **DeepProp** and **KD-Semi**. For the DeepProp baseline, instead of only using the pseudo negatives W_n to train the deep model, we also use pseudo positive labels generated with label propagation [65]. For the KD-Semi baseline, instead of only using the pseudo negatives W_n produced by the linear model, we use the high confidence predictions of the deep model from the previous iteration on the unlabeled data as pseudo positive and negative labels.

Implementation and configuration details We use a ResNet-50 model [53] pre-trained on ImageNet or using self-supervised methods (M^d_{pre}) to initialize the deep model and compute feature representations for all the unlabeled data. The same ResNet-50 model is used to generate labels for the background splitting auxiliary loss. We run our active approach and its variants with $N = 10$, $Q = 5$, and $B = 10$ for iNaturalist and $N = 10$,

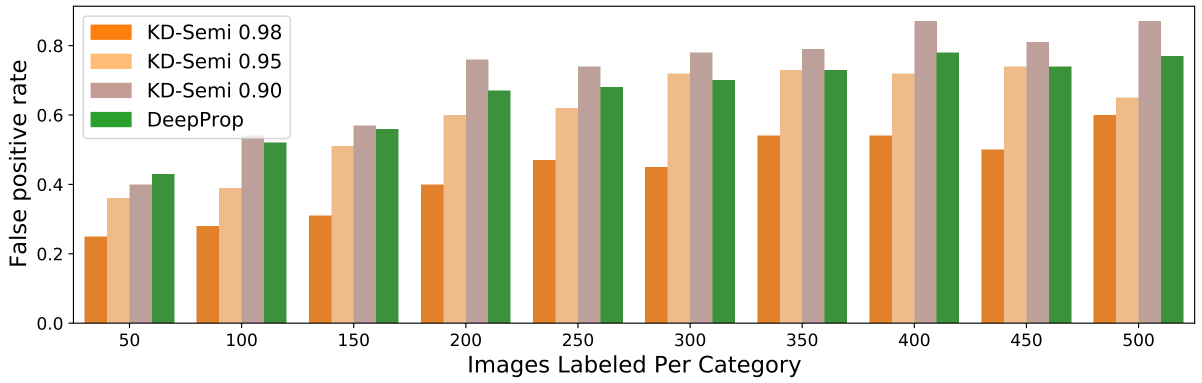


Figure 4.4: **Analysis of automatic positives.** Plot shows the average false positive rate in the automatic positives labeled by **KD-Semi** and **DeepProp** baselines for the 50 iNaturalist categories. Lowering the confidence threshold for automatic positives with **KD-Semi** increases the false positive rate. The high false positive rates showcase the difficulty in automatically labeling positives.

$Q = 5$, and $B = 20$ for Places unless specified otherwise. We use the same set of initial positives (I_p) for each category across all the methods and set f_a to 100 unless specified otherwise. We show that our approach is robust to a range of values for f_a.

Ideally, we would evaluate our approach and all the baselines individually for each category. However, due to computational limitations it is not feasible to train deep models with pseudo negatives individually for each category. Therefore, we run our approach and baselines for groups of ten categories at a time to reduce the compute costs. We maintain the same grouping of categories across all the experiments. When training the deep model we use binary cross entropy loss for each category as opposed to cross entropy loss across all the ten categories and supervise the model with only binary labels for each category. The linear models used to rank unlabeled data are trained **independently for each category** and humans are queried for binary labels **for each category separately**. Therefore, images might only be labeled for one of the ten categories. When training the deep models we deal with the missing labels by masking the corresponding binary loss for categories that are not labeled.

When updating the feature representation we train the deep model for 15 epochs on the currently labeled data in the active loop. In the $N = 10$, $Q = 5$, and $B = 10$ configuration we thus run 150 total epochs of training. However, on average an epoch in our approach only uses a quarter of the unlabeled data. Therefore, the overall cost of training is approximately equal to 40 epochs of training on the full dataset. When training deep models in the active active loop we train for 15 epochs on the currently labeled data. We start with a learning rate of 0.025 and follow a step schedule for decaying the learning rate every 5 epochs by a factor of 10. When training the fully supervised models we train for 50 epochs and follow a step schedule for decaying the learning rate every 20 epochs by a factor of 10. We use SGD with momentum and a batch size 256 for training all the models.

4.3.2 Comparison with baselines

Figure 4.3 shows that mean F1 accuracy of our approach and several baselines on the 50 and 20 categories of interest in the iNaturalist and Places datasets as a function of human labeling effort. We measure labeling effort as the number of binary labels per category. Our approach gives higher accuracy for the same human labeling effort compared to baselines, especially when a small number of images are labeled. The **Tropel** baseline trains a linear model on fixed features computed using a deep model pre-trained on ImageNet. The **Tropel-Deep** baseline trains a deep model only using the human labeled data. When only a small amount of human labeled data is available training a deep model can overfit leading to low accuracy. As one can see, **Tropel-Deep** performs worse than **Tropel** when a small amount of images are labeled. In contrast, our approach uses pseudo negatives in addition to human labeled data allowing us to update the feature representation without overfitting. On the iNaturalist categories, our approach matches fully supervised accuracy when 500 images per category are labeled since the dataset contains less than 500 positives for each of the categories. On the Places categories, there is a significant gap with fully supervised accuracy but this expected since there are 5000 positives per category and we only label a total of 1000 images per category.

DeepProp and **KD-Semi** use pseudo labels on the unlabeled data to update the deep feature representation. Unlike our approach these baselines do not restrict pseudo labels to easy negatives. **DeepProp** propagates labels to all the unlabeled data based on the feature similarity graph and assigns weights to the pseudo labels, whereas **KD-Semi** uses the current model to predict pseudo labels on the unlabeled data. Both **DeepProp** and **KD-Semi** perform worse that our approach due to errors in the labels propagated on positives. Figure 4.4 shows the false positive rate of **KD-Semi** and **DeepProp** on the iNaturalist categories. A high false positive rate is expected since the models are trained from very few labeled examples. The false positive rate keeps increasing as errors in labeling propagate and distort the definition of the category. **KD-Semi** can be tuned to reduce errors in the positive pseudo labels by increasing the confidence threshold at which the labels are trusted. Figure 4.4 shows the false positive rates for **KD-Semi** at different confidence thresholds. Even with confidence threshold as high as 0.98 the false positive rate is significant and is even worse with lower confidence thresholds. Setting the confidence threshold higher than 0.98 effectively results in almost no pseudo positive labels and tends to our strategy in the limit. **DeepProp** performs worse than **Tropel** due to the high amount of errors in label propagation which is difficult to control unlike **KD-Semi** as shown in Figure 4.4. We note that **KD-Semi** and **DeepProp** baselines only vary the pseudo labeling in our approach for a fair comparison.

4.3.3 Ablation study

Figure 4.5 shows F1 accuracy for variants of our approach on selected categories from the iNaturalist (5×10) dataset as a function of human labeling effort. The variants evaluate the impact of different aspects of our approach. **-Pseudo Neg** and **-BG Splitting** are variants without pseudo negatives and background splitting respectively. As one can see

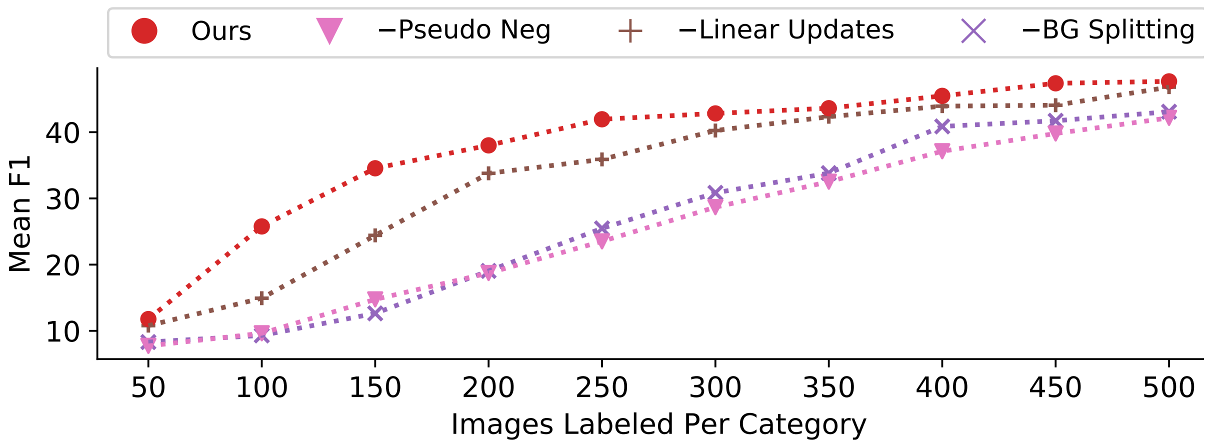


Figure 4.5: **Ablation study.** Plot shows average F1 accuracy for variants of our approach on iNaturalist (5×10) categories as a function of human labeling effort. Not using pseudo negatives **-Pseudo Neg** or background splitting to handle imbalance **-BG Splitting** results in lower model accuracy, especially when a small amount of images are labeled. Not updating linear models between representation updates **-Linear Updates** also results in lower accuracy.

not using the pseudo negatives when updating the deep representation results in lower accuracy for the same human labeling effort. As one would expect, the effect of pseudo negative and background splitting is more pronounced when a small amount of images are labeled. Since our approach only uses a fraction of the easy negatives along with the human labeled positives and hard negatives for training, it also reduces the imbalance between positive and negative instances when compared to fully supervised training using all the data. **-Linear Updates** only updates the linear model once every representation update, this corresponds to the parameter setting $N = 10, B = 50, Q = 1$. Skipping the linear model updates results in a lower accuracy for the same labeling effort. The linear model updates enable quickly adapting the ranking used to choose images for human labeling and hence lead to better overall model.

4.3.4 Using self-supervised representations

Until now we have evaluated our approach using a model pre-trained on ImageNet using supervised labels shown as **Ours-Sup-ImageNet** in Figure 4.6. Instead **Ours-Self-iNat** and **Ours-Self-ImageNet** use self-supervised models trained on iNaturalist and ImageNet with SwAV [16] as the starting point. We generate auxiliary labels for background splitting by clustering the iNaturalist data with the self-supervised feature representations. Similarly, **Tropel-Self-iNat** and **Tropel-Self-ImageNet** actively train linear models using fixed self-supervised representations. Our approach produces more accurate models showing the importance of updating feature representation with actively labeled data even when starting with a self-supervised representation trained on the same data. These results show that our approach when combined with self-supervised pre-training produces

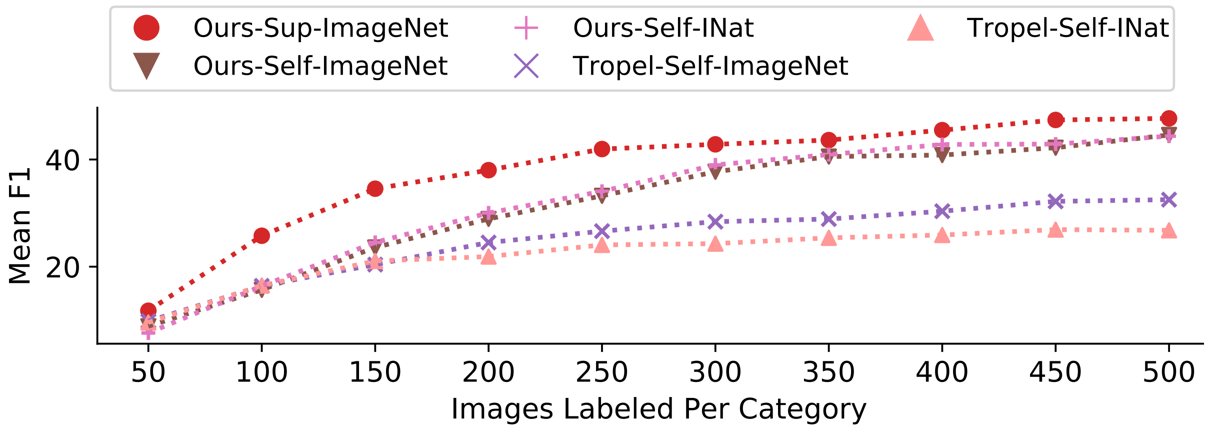


Figure 4.6: **Self-supervised representations.** Plot shows average F1 accuracy of our approach and Tropel on iNaturalist (5×10) categories as a function of human labeling effort. **Ours-Self-iNat** and **Ours-Self-ImageNet** use self-supervised models trained on iNaturalist and ImageNet as the starting point. **Tropel-Self-iNat** and **Tropel-Self-ImageNet** train linear models using fixed self-supervised representations. Our approach is more label efficient showing the importance of actively updating features. **Ours-Sup-ImageNet** uses ImageNet supervised representation as the starting point and outperforms self-supervised representations.

accurate models with just 500 binary labels for each category. However, our approach produces better models with supervised pre-trained representations on ImageNet. This suggests room for improvement in self-supervised representation learning approaches.

4.3.5 Feature update rate

Figure 4.7 shows variants our approach where we update features with different frequencies. The plot shows average F1 accuracy on iNaturalist (5×10) categories. The three variants in Figure 4.7 update the deep feature representation after every 25, 50 and 100 images are labeled by a human and correspond to the following configurations ($N = 5, B = 20, Q = 5$), ($N = 10, B = 10, Q = 5$) and ($N = 20, B = 5, Q = 5$) respectively. As once can see updating the features at low frequency (every 100 human labeled images) results in lower accuracy compared to more frequent updates, especially in the early iterations where only a small amount of data is labeled. However, as one would expect increasing the frequency of updates gives diminishing returns in accuracy since the representation change between iterations would be minimal with a very few additional labeled images.

4.3.6 Fraction of auto negatives f_a

Figure 4.8 shows variants of our approach where we change the ratio (f_a) of pseudo negatives labeled per human labeled image. The plot shows average F1 accuracy on iNaturalist (5×10) categories and each series corresponds to a different value of f_a. Using a small value for f_a (25) results in slightly lower accuracy. However, our approach is robust to

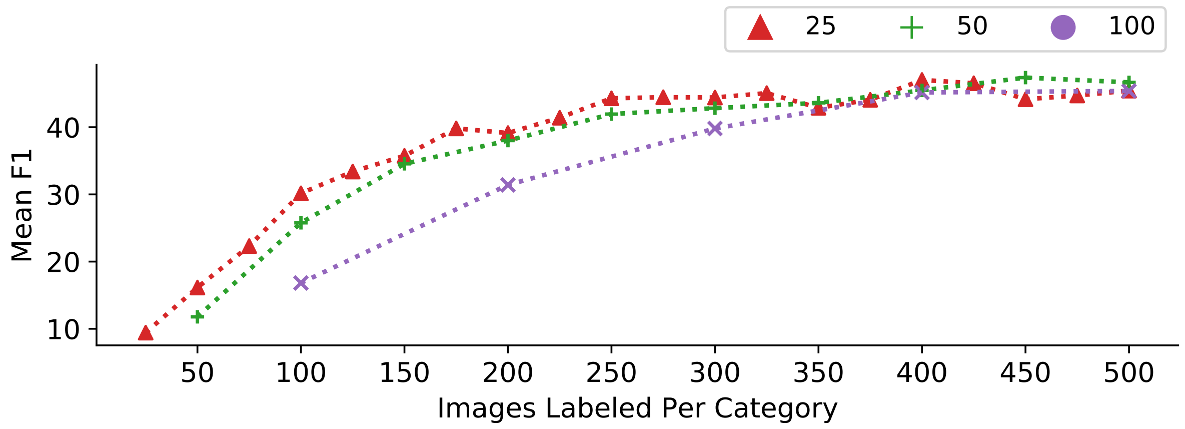


Figure 4.7: **Increasing feature update rate.** Plot shows average F1 accuracy for variants of our approach on iNaturalist (5×10) categories as a function of human labeling effort. The variants use different rates (number of labeled images) of updating the deep features. Updating at a low frequency (every 100 images) leads to lower accuracy models, especially in the early iterations. Very high frequency updates (every 25 images) gives diminishing improvements in model accuracy.

a range of values of f_a and provides significant improvement compared not using pseudo negatives $f_a = 0$.

4.3.7 Impact of adaptive sampling

Figure 4.9 shows variants of our approach where we change the sampling strategy for picking images to query humans. The plot shows average F1 accuracy on Places (2×10) categories. When there is a sufficient number of positives in the unlabeled data set always sampling the most likely positives is not effective since we would query humans on a lot of easy negatives. This scenario occurs with the Places dataset where there are 5000 positives in the unlabeled data for most of the categories. In such scenarios, as Figure 4.9 shows our adaptive strategy that switches between samples on the margin and most likely positives performs better than a fixed strategy which queries humans on images which are likely to be positives (strategy used by **Tropel**).

4.4 Discussion

In many real world scenarios it is often necessary to build a model for a rare category starting from a small positive set with access to a large unlabeled data collections which contain instances of the rare category. We demonstrate that specializing models to mine unlabeled data for instances of rare categories significantly lowers human effort in building accurate models for rare categories. We summarize our approach as answers to the questions we need to address for effective specialization.

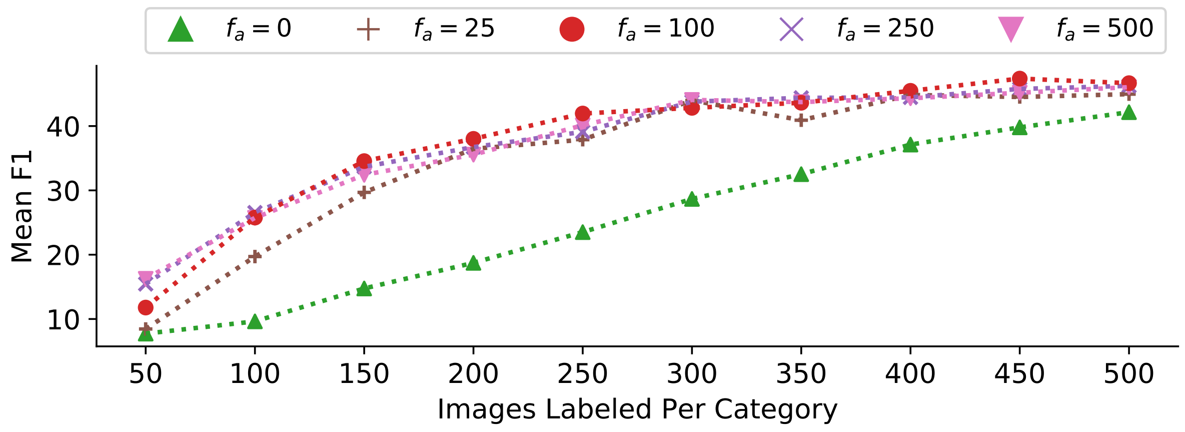


Figure 4.8: **Varying number of pseudo negatives f_a.** Plot shows average F1 accuracy for variants of our approach on iNaturalist (5 × 10) categories as a function of human labeling effort. The variants use different ratio (f_a) of pseudo negatives relative to the human labeled data. Low values of f_a leads to slightly lower accuracy but our overall approach is robust for a wide range of f_a.

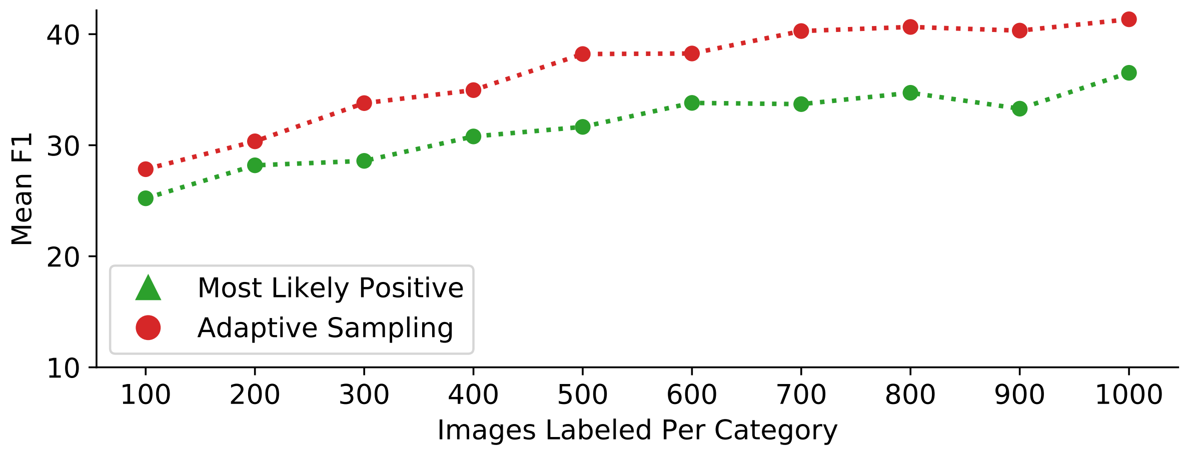


Figure 4.9: **Impact of adaptive sampling.** Plot shows average F1 accuracy for variants of our approach on Places (2 × 10) categories as a function of human labeling effort. Our adaptive sampling strategy for choosing the images to query humans is more effective compared to sampling most likely positives (strategy used by **Tropel**).

- *What is the model being specialized for?* The ranking models are specialized to individual categories and small amount of human labeled positives.
- *How to curate a dataset specialized to the context?* The data for training the per category specialized models is curated using separate ranking models for each category. Instead of using k-way classification labels the ranking for each individual category is used to query humans for binary labels on a small number of images. In addition to human labels a large number of per category negative labels are automatically inferred using the ranking model.
- *What are model architecture and training algorithm changes required to induce and exploit specialization?* Background splitting is used to update feature representation with a small amount of positives and large number of negatives for each category.
- *How do we identify when a specialized model is applicable?* The specialized ranking models are only used to automatically label negatives in bottom half of the ranking.

Our approach can also be viewed as combining semi-supervised learning on the large amount of easy negative data with human supervision on a small subset of rare category positives and hard negatives. We find that techniques that work well in the common multi category setting like propagating positive labels or training a deep model using standard losses do not work off-the-shelf and need to be adapted to rare category settings with extreme imbalance in the training data. We hope that our work showcases the challenges in building models for rare categories by mining large unlabeled collections and encourages further work into the different under studied aspects of the problem.

Chapter 5

Background Splitting

One appeal of specializing models to a subset of the distribution is that we only need to label data for that subset. However, only labeling a subset of data can lead to impoverished supervision and imbalance in the training data both of which pose challenges for deep model training. Training classification models for rare categories is a natural scenario where the need for specializing to a narrow domain arises. In many real-world settings, categories of interest are sufficiently rare that it is far more common for images to *not exhibit any categories of interest*. For example, consider a newly collected dataset (e.g., obtained by an autonomous vehicle fleet) where a new category – an e-scooter – is annotated. This is a highly imbalanced binary image classification problem, as the vast majority of collected images will *not* contain an e-scooter (sparse positives). In this chapter we focus on the problem of training accurate deep models for image classification of a small number of rare categories assuming we have labeled a large amount of negative data and small amount of positive data for these categories. We first study this problem because it is relatively easy to machine label a large number of negatives for rare categories with a high degree of accuracy. In these scenarios, not only is there a small number of positives per category, but the overall number of positive examples for all categories is dominated by the number of background images (e.g., our experiments include cases where 99.98% of the dataset is background). We find that even state-of-the-art methods for deep imbalanced classification based on data sampling [72] or loss reweighting [15] fail to produce accurate models in this extremely imbalanced setting.

We propose a surprisingly simple method to address learning challenges posed by severe background dominance: we *split the visually diverse, but monolithic background category* into many smaller, visually similar categories during training. However, rather than modify the main (rare category) classification loss of the model to identify more categories, our approach adds an auxiliary loss that forces the model to mimic the predictions of an existing, pre-trained image classification model *on all training examples*, the vast majority of which constitute "easy" background instances in the main rare category classification task. Jointly learning using the main classification loss and the auxiliary classification loss regularizes the rare category classification model by forcing it to discriminate between pseudo-categories for all training examples and helps reduce overfitting to the small number of rare category positives. Most importantly, this solution transfers knowledge of an

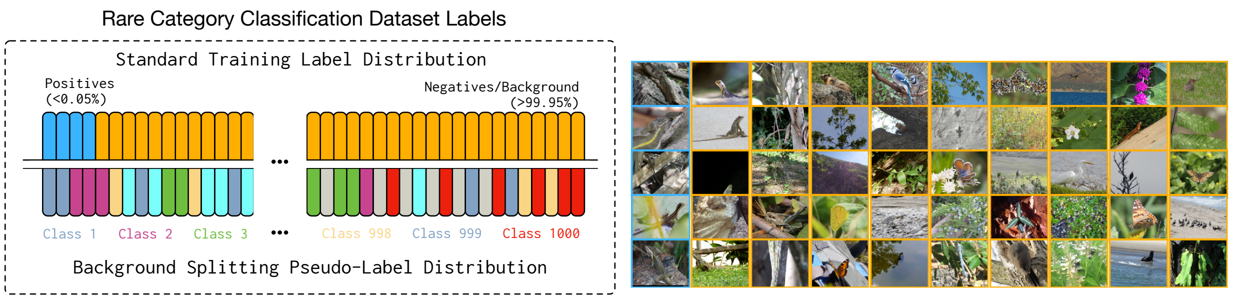

Rare Category Classification Dataset Labels
Standard Training Label Distribution
Positives (<0.05%)
Negatives/Background (>99.95%)
Class 1
Class 2
Class 3
Class 998
Class 999
Class 1000
Background Splitting Pseudo-Label Distribution

Figure 5.1: Top: Training classification models for rare categories is difficult because of the limited information content and extreme label imbalance provided by a large "background" category. During training we address these issues by *splitting the background category* into a large set of smaller categories according to pseudo-labels produced by an existing, pre-trained model. Bottom: positive (blue outline) and negative (orange outline) images for a fine-grained lizard category in iNaturalist. Even though the negatives all have the same label (background) they encompass a diverse set of hard negatives (hard to distinguish from the positives) and a vast number of easy negatives. Background splitting extracts value from all negatives by forcing the model to predict pseudo-labels for all images.

auxiliary classification task into the target model via distillation—it only requires access to the pre-trained model, *not access to any additional labels* beyond those used for the main classification task.

Benchmarking: Although background dominant scenarios are common in real-world applications of computer vision, no academic datasets directly target this important task. To evaluate our methods and facilitate future rare category learning research, we contribute modified label distributions for two existing large-vocabulary datasets: the highly-imbalanced iNaturalist dataset [145], and the Places365 dataset [162]. In both cases we select only a small number of categories as targets of interest, and merge all other categories into a single, large background category. These new label sets model real-world training scenarios in that they contain a small number of rare categories, and a vast majority of images exhibit only background. (We will release these modified label distributions to the public.)

Analysis: We find that state-of-the-art techniques for imbalanced classification [72, 15] perform *worse* than standard fine-tuning in scenarios where the background category makes up more than 98% of the dataset. In contrast, background splitting outperforms prior art by 8.9 mAP points. In extreme cases of a single, rare foreground category (e.g., 99.98% background), background splitting yields mAP improvements from 10.6 to 52.9. We also evaluate the benefits of background splitting under different amounts of background dominance, and varying semantic overlap between the auxiliary task and main task categories.

5.1 Related Work

Category imbalance. Training deep classifiers on heavily imbalanced data is challenging because standard losses focus on majority categories (failing to learn minority categories) or overfit to the few positive examples of minority categories. Most methods for learning under long-tail, imbalanced datasets [50] rely on techniques such as re-balancing, re-weighting/category-conditioned adjustments to final loss values, and category clustering. **Re-balancing methods** alter the training distribution to simulate traditional balanced training sets [15, 146]. Recent work showed re-balancing should be limited to a final stage of model training to encourage more general feature representations [72, 15, 153]. **Re-weighting/category-conditioned methods** adjust the loss attributed to a sample based on its category [163, 15, 72] or how hard the sample is. **Category clustering methods** use clustering of embeddings from training samples to improve transfer learning from head to tail categories [92] and to train sub-models specialized to individual categories (avoiding imbalance) [164, 109, 82].

Instead of altering the training distribution through rebalancing or reweighting, background splitting keeps the training distribution for the main loss fixed and instead adds an additional auxiliary loss that forces the model to make fine-grained distinctions among images in the background category (the model must perform a challenging task even for "easy" background images). This approach encourages the feature representation to be more discriminative while still learning from the unaltered primary training distribution.

Large, diverse background. Background dominant scenarios are an important real-world case of imbalanced classification where instances of a large, diverse background category are more common than any foreground category. Prior work has modeled the background by adding an additional "N+1" category to represent the background [97, 93, 90] (which we use in our method, Sec. 5.3.2), modeling the background as a Gaussian distribution [107], or by training the model to predict low scores for background instances and then applying a threshold to filter them [32, 97].

Handling a large, diverse background in classification is related to the foreground-background class imbalance problem in object detection [106], where ideas such as re-balancing (hard-negative mining [132]) and re-weighting (focal loss [84]) are employed to improve object detection performance. We focus on background imbalance problem in the context of image classification, but our findings may also benefit object detection tasks as well.

Open-world, open-set, and out-of-distribution recognition. Unlike the background category setting (in which models are given access to training instances which are from the same distribution as the test instances), Open World, Open Set, or Out-of-distribution recognition datasets test on data from categories (Open World/Open Set) or entire datasets (Out-of-distribution) not seen during training [92, 8, 9, 32]. In this setting, the model is tasked with identifying "unknown unknowns" in the test distribution that are not present in the training distribution. Typically, these "unknown unknowns" are not a significant majority of the test distribution. In contrast, we are interested in "known unknowns" which are present in the training distribution, but make up a majority of the data.

Knowledge transfer and sharing. Our approach leverages knowledge contained in a pre-trained model to improve performance on the rare category classification task. Our approach modifies standard **transfer learning** methods [110, 114, 124, 76] by introducing an auxiliary distillation loss to improve model performance in background dominant scenarios. Thus our solution is similar in spirit to **multi-task learning** methods [17, 79], in particular those addressing the challenges of incremental learning [83, 133, 74, 130]. However, our approach uses supervision from an existing classifier to regularize training on the rare-category classification task. It does not aim to train a model that accurately classifies both prior and novel categories. Instead of transferring knowledge exclusively via fine tuning, we transfer knowledge using a combination of fine tuning and **knowledge distillation** [55, 152], where the distillation is performed using data from the new domain rather than data from the original domain. This approach to knowledge transfer has been shown to outperform fine tuning for some tasks [138, 89], and is also useful when the source and destination models are different [41].

5.2 Large Background Datasets

To support image classification research in real-world, large-background settings, we created modified variants of the iNaturalist 2017 dataset and the Places365 dataset, which we call iNaturalist-BG and Places-BG. We selected iNaturalist 2017 because of its large (and imbalanced) category vocabulary (5089 categories, including many visually similar species), and because it is based on a real-world use case of identifying the world's flora and fauna. (iNaturalist was collected by thousands of real individuals interested in species identification, not by choosing a list of categories then collecting images by querying internet search engines [92, 15]). Places365 exhibits a more modestly sized vocabulary (365 categories), but contains content that differs significantly from iNaturalist, making the pair of datasets a good test of generalizability of rare category methods.

To construct our modified datasets, we consider a small number (N) of the original dataset's categories to be "labeled" and combine the remainder of the categories from the original dataset into a single background category. To study how the size of the background category affects model performance, we provide training and test set pairs for varying N. For example, for iNaturalist we create pairs for $N = 5089$ (equivalent to the standard iNaturalist 2017 category distribution), 1100, 100, 10, and 1). Figure 5.2-right shows the distribution of images to categories for the iNaturalist-BG ($N = 100$) training set. In total, the 100 chosen categories constitute only 1.7% of the dataset (98.3% background).

Note that in all cases, models are trained on all images in the original dataset's training set (579,184 for iNaturalist, 1.8 million for Places), and tested on the dataset's full test set. (Images not belonging to the N categories of interest are labeled "background".)

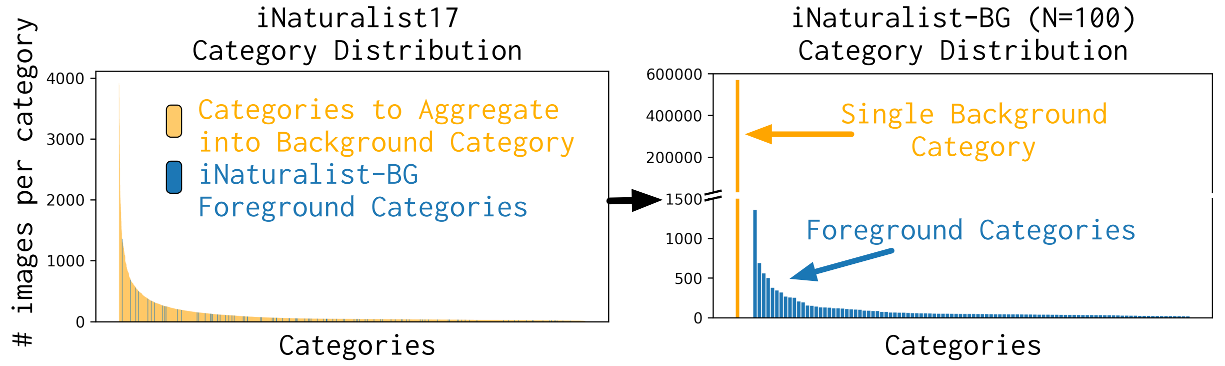


Figure 5.2: Left: distribution of images per category in the iNaturalist-2017 training set, sorted by category frequency. Although there is no "background" category in iNaturalist, we color categories based on whether they are placed in the background category in our modified dataset, iNaturalist-BG. Right: distribution of examples in iNaturalist-BG ($N = 100$). The background category (yellow) contains 98.3% of the images in the dataset.

5.3 Method

When training a model for classifying a small set of rare categories (*main task*), we want to avoid overfitting to the small set of foreground positives and avoid solutions which predict all instances are part of the background category. We address these challenges by transforming the model optimization problem into a easier one with significantly lower distribution skew. We also describe how we combine our background splitting technique with the common softmax thresholding method introduced by Matan et al. [97].

5.3.1 Regularization via Auxiliary Loss

In addition to supervising the model with the main task's cross-entropy loss for (N+1)-way classification during training (the additional +1 category representing the background), we also add an *auxiliary task* by attaching a second classification head to the network supervised by pseudo-labels generated with using pre-trained model. (Figure 5.3-right, "Auxiliary Task"). Let $y \in \{0, \dots N\}$ be the main $(N+1)$-way classification task where 0 is treated as the background class. Given a training set with a large fraction of background examples, we assume access to an *auxiliary model* with K classes that provides pseudo-labels on all training pairs $\{(x_i, y_i)\}$:

$$\{(x_i, y_i, t_i)\} \qquad y_i \in \{0, \dots N\}, \quad t_i \in \{1, \dots K\}.$$

where t_i is the auxiliary model's pseudo-label. We then learn a classifier with a multi-task loss:

$$\min_{\theta, \mathbf{w}, \mathbf{v}} \sum_i \text{loss}\Big(y_i, G_{\mathbf{w}}(F_\theta(x_i))\Big) + \lambda_H \text{loss}\Big(t_i, H_{\mathbf{v}}(F_\theta(x_i))\Big)$$

where G and H refer to the main and auxiliary task classification heads, respectively, of a base network trunk F with shared features θ and task-specific features $\mathbf{w}$ and $\mathbf{v}$. λ_H is

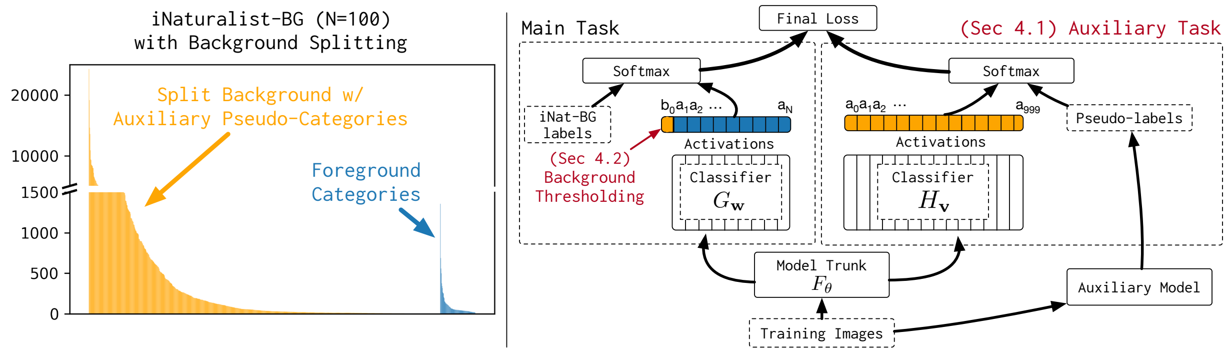


Figure 5.3: Left: distribution of images for the iNaturalist-BG ($N=100$) dataset after partitioning into pseudo-categories determined by the output of a pre-trained auxiliary model (in this example, 1000 ImageNet categories). The largest pseudo-category is only 4.5% of the dataset, far smaller than the iNaturalist-BG "background" category (98.3%). Right: the training configuration for our method. We use two tasks, one supervised by labels from the target dataset (e.g., iNaturalist-BG) and one supervised by pseudo-labels from the auxiliary model. The main loss uses a fixed background logit to improve background classification.

the weight for the auxiliary loss. The loss function is the standard softmax cross-entropy loss. Note that the pseudo-labels t_i are generated by evaluating the auxiliary model on all training data. *No additional human labeling effort* is required beyond the labels y_i provided for the main rare category classification task. Figure 5.3-right visualizes the full training graph for our multi-task network, which defines task-specific linear weights $(\mathbf{w}, \mathbf{v})$ on the shared feature trunk θ. Note that at test time, performing the main rare category classification task requires only evaluating $G_{\mathbf{w}}(F_\theta(x))$ (for test sample x).

During training, the role of the auxiliary task is to effectively "split up" the large background category into a large number of visually coherent sub-categories. This reduces distribution skew and forces learning of robust features θ that discriminate between the many psuedo-categories, even for the large majority of training data that would otherwise serve as a "easy" background examples for the main classification task. Figure 5.3-left shows the results of splitting the iNaturalist-BG ($N = 100$) background category into K=1000 categories defined by an auxiliary model that classifies images into ImageNet categories. Although the background category is 98.3% of the training set in this case, the most frequently occurring auxiliary task category contains only 4.5% of the dataset.

Importantly, a direct mapping between auxiliary model categories and the content of the dataset being evaluated on is not necessary for visually coherent groupings to occur. Figure 5.4 illustrates that visual coherence of sub-categories that emerge when evaluating an ImageNet category classification model on the iNaturalist dataset. Rows 1, 3, 4, and 5 contain visually similar images, even though they do not contain the animal the auxiliary model category was trained for. (Row 1's category is "flamingo", but the birds at right are a different species of pink bird; Row 4's category is "European fire salamander" but the images to the right are of owls). In Section 5.4.3, we evaluate how different choices for the

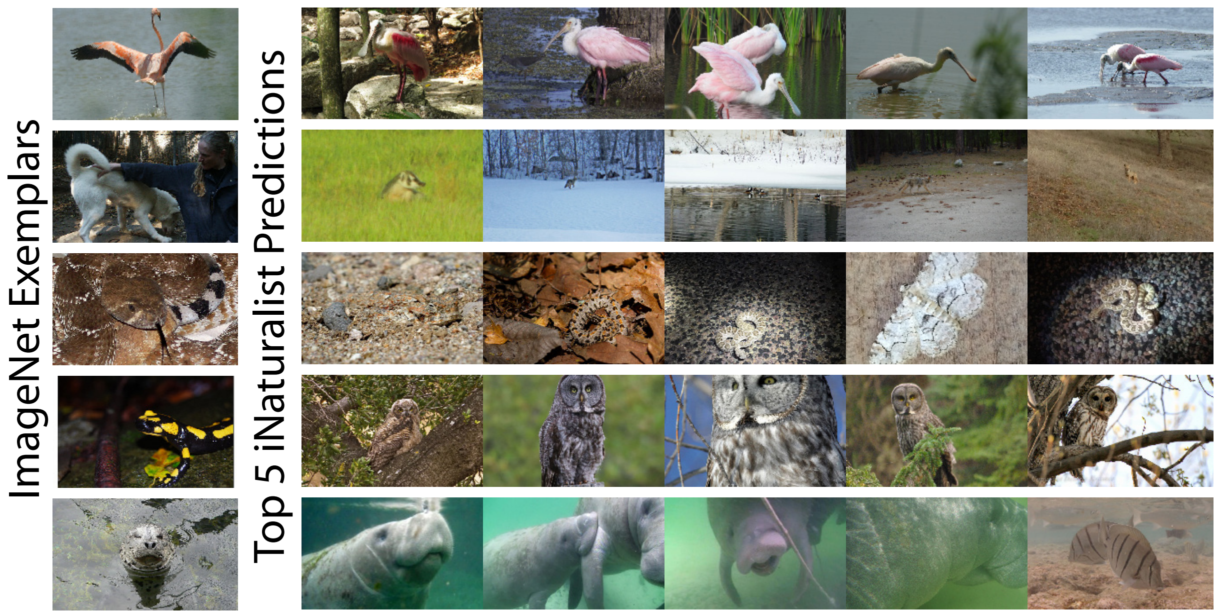


Figure 5.4: An auxiliary model pre-trained to perform ImageNet category classification groups semantically similar iNaturalist images together, resulting in pseudo-labels that split the large background category in iNaturalist-BG. Each row represents a category chosen at random from the auxiliary model categories (ImageNet categories). The image on the left is an example of the category taken from the auxiliary model training set (ImageNet). Images on the right are the auxiliary model's five most confident predictions for that category on the iNaturalist dataset. Notice that rows 1, 3, 4, and 5 contain visually similar images even though the auxiliary model category is different from the animals in the iNaturalist images.

auxiliary task influence the extent auxiliary loss helps main task classification performance.

5.3.2 Background Thresholding

The straightforward approach to performing classification with a background category modifies the N-way classification problem into an $(N+1)$-way classification problem. However, linear (softmax) classification encourages examples that fall into the same category to have similar features that can be linearly separated from other classes. This may be problematic for the background class, which we expect to be very large and diverse in our setting (Figure 5.2-left). We address this issue by *choosing to not* learn a classifier for the background category. Instead, we adopt the method of Matan et al. [97], which assigns the background category (category 0) a fixed activation, as represented by b_0 in Figure 5.3-right. We write the softmax classifier for $G_{\mathbf{w}}(F_\theta(x))$ as follows:

$$p(y = n|x) \propto e^{w_n \cdot F_\theta(x) + b_n}, n \in \{0, 1, \ldots N\} \tag{5.1}$$

where $F_\theta(x)$ is the nonlinear (deep) embedding shared across our heads and $\mathbf{w} = \{w_0, b_0, \ldots, w_{N+1}, b_{N+1}\}$. Because the weight associated with the background class $(n = 0)$ can be difficult to estimate given the large expected variability in appearance, we *clamp* it to be the 0 vector:

$$w_0 = \mathbf{0}, b_0 = \text{constant} \tag{5.2}$$

and do not update either during learning. This modification is equivalent to setting the background class logit to a fixed constant during learning:

$$p(y = n|x) = \frac{e^{w_n \cdot F_\theta(x) + b_n}}{e^{b_0} + \sum_{i=1}^{N+1} e^{w_i \cdot F_\theta(x) + b_i}}, n \in \{1, \dots N\} \tag{5.3}$$

where the sum over N class probabilities can now be less than 1 (where the remaining probability mass is assigned to the background class). The above model no longer learns a hyperplane w_0 to separate background examples from the foreground category examples, but instead classifies an example as background only if the model produces logit values lower than b_0 for all foreground categories.

5.4 Evaluation

We perform an in-depth analysis of background splitting on the iNaturalist-BG dataset by comparing its performance to strong baselines in multiple different regimes of background dominance. We also provide diagnostic experiments that explore the sensitivity of background splitting to different choices of auxiliary task and show the necessity of jointly optimizing the main task loss with the auxiliary loss. Finally, we also demonstrate the benefits of background splitting on Places-BG, suggesting applicability to a wide range of classification tasks.

5.4.1 Experimental Setup

Evaluation metrics. Traditional metrics for image classification include top-1 or top-5 error, but these metrics can be manipulated by constructing models that favor the background. (Always predicting background yields a top-1 accuracy of 98.3% on iNaturalist-BG N=100.) Instead, we propose two evaluation protocols for datasets with N category labels plus a large background category. The first protocol is motivated by top-1 error: algorithms must report a single $(N + 1)$-way label for each test sample. It computes the F1 accuracy (harmonic mean of precision and recall) for each of the N classes and reports their mean. The second protocol, motivated by those used for large-background tasks such as object detection [33], recasts the problem as N retrieval tasks corresponding to each foreground category. For each category, algorithms must return a *confidence* for each test sample. These are ranked to produce N precision-recall curves, which in turn are summarized by their average area underneath (mAP).

For iNaturalist-BG, we evaluate using label sets that range from no background (N= 5089) to increasing levels of background dominance (N=1100, 100, 10, 1). Even though the foreground category distribution varies with N, we construct training/set pairs so that evaluation metrics remain comparable across N. Specifically, we evaluate performance using a fixed set of 100 categories. When $N > 100$, we only compute mean F1 or mAP from the selected 100 categories. The $N = 10$ and $N = 1$ setups do not contain all 100 categories, so we provide 10 test set pairs for $N = 10$ (spanning all 100 test categories) and average performance across these subsets. Computational constrains required us to limit evaluation to 10 pairs for $N = 1$. (Evaluation averages over only 10 of the 100 test categories, so $N = 1$ results are not directly comparable to results for other N.)

Model and training details. In our background splitting configurations (referred to as BG-SPLIT) we use the ResNet-50 architecture [53] (initialized via ImageNet pre-training) in all experiments. We set the background threshold value to $b_0 = 0.1$, the weight on the auxiliary loss to $\lambda_G = 0.1$, and batch size to 1024. We find that large batch sizes are crucial for training models when the background category is dominant, and provide an additional evaluation of the effect of batch size on different methods.

Unless otherwise stated, we use a standard ResNet50 model trained on the 1000-category ImageNet [118] dataset, as an auxiliary model. We evaluate alternative choices for auxiliary models in Sec. 5.4.3.

Baselines. We compare our method to standard fine-tuning with cross entropy loss (denoted as FT), as well as two state-of-the-art baselines for training on the iNaturalist dataset: LWS [72] which performs sample re-balancing; and LDAM [15], which performs loss re-weighting. LDAM and LWS have been shown to perform better than a broad set of other imbalanced learning methods, including recent methods from the object detection literature such as focal loss [84]. LWS trains in two stages: first training a standard cross-entropy model with uniform sampling and no re-weighting for 90 epochs; then freezing all layers but the final classification layer and retraining for 15 epochs using class-balanced sampling. LDAM also uses a two-stage training approach: first training a classification model using uniform sampling and weighting samples with a Label-Distribution Aware Margin (LDAM) loss for 60 epochs; and then continue training the same model by re-weighting the loss for individual examples based upon their frequency for 30 epochs. We train these baseline models using the official code provided by each method [1] [2].

We tuned the learning rate and batch size for FT via hyperparameter search. For LWS when $N = 5089$ we train the base representation model for the first stage using published hyperparameters [72]. When $N < 5089$ we use the same hyperparameters as FT. We report results for the best-performing second-stage method (*Tau Norm* for $N = 100$ and 1100, *LWS* for $N = 5089$).

5.4.2 Comparison with Baselines

iNaturalist-BG comparison. Table 5.1 compares the mAP and F1 scores of models trained using BG-SPLIT against all baselines on iNaturalist-BG. LDAM and LWS perform

[1]LWS: https://github.com/facebookresearch/classifier-balancing
[2]LDAM: https://github.com/kaidic/LDAM-DRW

Model Performance: iNaturalist-BG

	$N=1$ (99.98%)		$N=10$ (99.83%)		$N=100$ (98.30%)		$N=1100$ (77.95%)		$N=5089$ (0%)	
	mAP	F1	mAP	F1	mAP	F1	mAP	F1	mAP	F1
FT	10.6	10.8	9.3	8.9	38.3	38.4	50.9	**44.8**	57.7	49.9
LDAM	-	-	-	-	24.7	21.1	41.6	37.1	57.4	55.1
LWS	-	-	-	-	35.5	36.6	42.5	37.3	**60.0**	**56.9**
FOCAL	-	-	-	-	5.97	3.94	25.2	21.7	58.3	50.7
BG-SPLIT	**52.9**	**51.7**	**44.6**	**39.4**	**47.2**	**40.7**	**51.4**	44.7	59.9	52.5

Table 5.1: **bg-split outperforms all baselines when the background frequency exceeds 98% on iNaturalist-BG.** Comparison of mAP and F1 scores of BG-SPLIT to baselines on the iNaturalist-BG dataset (percentages indicate background frequency). BG-SPLIT is 8.3 mAP points more accurate than state-of-the-art baselines in the $N = 100$ setting, and *42.3 points higher than fine tuning* (FT) in the extremely imbalanced $N=1$ setting that is typical of many real-world image classification scenarios. Prior state-of-the-art baselines for imbalanced training (LDAM [15], LWS [72]) perform worse than FT in background dominated settings. ($N=1$ results are not directly comparable to other choices of N, see Section 5.4.1).

Model Performance: Places-BG

	$N=1$ (99.72%)		$N=10$ (97.22%)	
	mAP	F1	mAP	F1
FT	44.0	41.3	56.8	51.4
BG-SPLIT	**53.8**	**48.7**	**61.0**	**56.6**

Table 5.2: **bg-split outperforms ft on Places-BG.** The benefit of BG-SPLIT over FT increases with increasing background size (percentages indicate background frequency). Overall, the magnitude of benefit of BG-SPLIT is less on Places-BG than on iNaturalist-BG (Table 5.1) because categories in Places-BG are on average over 13× more frequent than iNaturalist-BGcategories (0.27% vs. 0.02%). Places-BG categories are also less fine-grained than those in iNaturalist-BG, making the background easier to discriminate.

slightly better than BG-SPLIT in the N=5089 case they were designed for (original iNaturalist setting, no background), however as background size increases (N=1100, 78.0% background) and (N=100, 98.3% background), both baselines degrade rapidly, *performing worse* than the FT baseline, even when using the best hyperparameter configurations found for each N. In the N=100 configuration BG-SPLIT outperforms the best results of LDAM and LWS by 11.7 (mAP) and 4.1 (F1) points. We observe that class-balanced sampling (LWS) and loss re-weighting based on class frequency (LDAM) cause accuracy on the dominant background category to decrease, yielding a significant increase in false positives for foreground categories. This results in a minor increase in foreground category recall, but a

Aux Task Ablations: iNaturalist-BG (N=100)

Auxiliary Task		BG-SPLIT	
Source Dataset	Label Method	mAP	F1
— *No auxiliary loss* —		41.1	37.7
None	Random-1K	37.2	35.0
Places365	Classifier	39.3	34.1
ImageNet	Cluster-1K	45.9	39.0
ImageNet	Cluster-5K	45.9	**41.0**
ImageNet	Classifier	**47.2**	40.7

Table 5.3: **Choice of auxiliary task significantly impacts bg-split model performance.** Due to similarity between ImageNet classification and iNaturalist-BG classification tasks, pseudo-labels from ImageNet-based auxiliary tasks increase the performance of models trained using BG-SPLIT on iNaturalist-BG, even when categories are machine-defined (IMAGENET/CLUSTER-1K, IMAGENET/CLUSTER-5K). Auxiliary tasks based on random pseudo-labels (RANDOM-1K) or significantly different classification tasks (PLACES/CLASSIFIER) are destructive to model training for iNaturalist-BG. Note that the *no auxiliary loss* configuration is similar to that of FT (N=100) in Table 5.1, but with two differences for consistency within this table: *no auxiliary loss* uses batch size 1024 instead of 512, and uses the background thresholding loss modification.

large reduction in precision and overall worse performance for LDAM and LWS in the sparse positive setting. Since the performance of LDAM and LWS further degrades with decreasing N, we do not evaluate these methods for $N = 10$ and $N = 1$ to save experimental costs.

The value of BG-SPLIT increases as background dominance is further increased. In the extreme case of N=1 (98.98% background), BG-SPLIT *beats* FT *by 42.3 (mAP) and 40.9 (F1) points.* We note that the extreme background dominance in the N=1 configuration is typical of many real-world applications. The overall performance of BG-SPLIT decays gracefully with decreasing N without modifying key hyperparameters (e.g., learning rate, batch size, b_0, λ_G). Hyperparameter robustness across different levels of background imbalance is an attractive property of our method which makes it easy to use in practice. FOCAL implements the Focal Loss method [84], which increases the loss value for difficult to classify examples and decreases the loss value for easy to classify examples. We find that the FOCAL baseline performs slightly better than FT for $N = 5089$, but overall performs worse than any other baseline.

Places-BG comparison. Table 5.2 compares BG-SPLIT to FT on two configurations of Places-BG: N=10 (97.2% background) and N=1 (99.7% background). Despite Places-BG containing notably different categories and image content than iNaturalist-BG, the overall trends on Places-BG are similar. The benefit of BG-SPLIT is significant and increases with increasing background dominance. However, for a given N, the benefits of BG-SPLIT are lower on Places-BG than for iNaturalist-BG because Places-BG contains a smaller number of categories with higher per-category frequency, making the foreground category classification problem easier (helping the FT baseline). Note that N=10 background frequency on Places-BG is similar to that of N=100 on iNaturalist-BG.

5.4.3 Sensitivity to Choice of Auxiliary Task

A goal of background splitting's auxiliary task is to leverage the vast number of background examples to learn useful representations that aid the main classification task. To understand the sensitivity of background splitting's benefits to auxiliary tasks that exhibit varying degrees of similarity to the main classification task, we trained BG-SPLIT models on iNaturalist-BG with the following auxiliary tasks: pseudo-labeling via a pre-trained ImageNet classification model (IMAGENET/CLASSIFIER), pseudo-labeling via pre-trained image classification model for the Places365 dataset[162] (PLACES/CLASSIFIER), performing approximate k-means clustering [3] on features generated by the pre-trained ImageNet model (IMAGENET/CLUSTER-1K, IMAGENET/CLUSTER-5K), and assigning 1000 pseudo-categories to training images at random (RANDOM-1K).

Table 5.3 summarizes results for iNaturalist-BG ($N = 100$), where all methods are trained using a batch size of 1024 and include the background thresholding loss modification. RANDOM-1K, which splits the background into equal-sized categories, but yields categories that lack visual or semantic coherence, *degrades performance* compared to a training configuration that does not employ any auxiliary task (it is destructive to feature learning). PLACES/CLASSIFIER also degrades performance because scene-specific classification on Places365 is a significantly different task than species classification. (Table 5.2 showed ImageNet classification is a beneficial auxiliary task for Places-BG but Places365 classification is a destructive auxiliary task for iNaturalist-BG classification.) However, since the ImageNet dataset has a large number of categories (1000), including some with semantic category overlap with iNaturalist (e.g., many animal categories), there is substantial improvement from auxiliary tasks based on ImageNet models. For example, even though IMAGENET/CLUSTER-1K and IMAGENET/CLUSTER-5K generate pseudo-labels for machine-generated categories, these auxiliary tasks significantly improve final model performance. Although IMAGENET/CLASSIFIER yields the highest mAP gain, the fine-scale clusters of IMAGENET/CLUSTER-5K are the pseudo-labels that produce the highest F1 gain.

5.4.4 Value of Joint Training

Can the auxiliary loss alone produce an effective feature representation without target foreground category labels? If so, one could use the auxiliary loss for dataset pre-training, and leverage the resulting representations to rapidly learn a model for novel foreground categories. To isolate the value of background splitting's joint training approach vs. supervision from main task and pseudo-labels alone, we conducted an ablation experiment where we separated the aux loss training and main loss training into separate phases, and froze the features after the first phase.

Table 5.4 compares the performance of the resulting models on iNaturalist-BG (N=100). Even though pre-training using only the auxiliary loss (row 2) keeps 98.3% of the labels the same as the full BG-SPLIT solution (row 3), the BG-SPLIT method is over *over 18 mAP*

[3] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html

Feature Learning Ablations: iNaturalist-BG (N=100)

Method	Step 1 loss	Freeze/FT	Step 2 loss(es)	mAP
Linear model on ImageNet features	–	Freeze	Main	25.5
Decoupled aux/main	Aux	Freeze	Main	28.3
BG-SPLIT	–	Fine tune	Main & Aux	47.2

Table 5.4: **Combining sparse foreground category labels with pseudo-labels is critical to feature learning.** Feature learning using only auxiliary task loss (row 2) provides improvement over initial ImageNet features (row 1). However a model trained using BG-SPLIT (row 3) is over 18 mAP points higher.

BG-SPLIT Component Analysis: iNaturalist-BG (N=100)

None (FT)		Aux loss only		BG thresh only		Both (BG-SPLIT)	
mAP	F1	mAP	F1	mAP	F1	mAP	F1
36.0	35.6	46.0	40.4	41.1	37.7	47.2	40.7

Table 5.5: **Most of the benefits of the full bg-split solution are due to the use of an auxiliary loss.** However, use of auxiliary loss and background logic clamping are complementary techniques (additive benefits), suggesting auxiliary losses could be combined with other techniques for learning on imbalanced data.

points higher than this decoupled method. These results suggest that feature learning via joint training from sparse positive examples and abundant pseudo-labels from the auxiliary task is critical.

5.4.5 Component Analysis

The full BG-SPLIT solution has two new components: an auxiliary loss and the background thresholding modification to the main task loss. Table 5.5 isolates the relative gains due to each of these components on iNaturalist-BG (N=100). The majority of the benefit of BG-SPLIT comes from the use of the auxiliary loss (10 mAP points "aux loss only"). Background logit thresholding in isolation yields 1.1 mAP points ("bg thresh only"). These techniques are complementary techniques (additive effects in the final solution), suggesting that the benefits of using an auxiliary loss could be combined with other state-of-the-art techniques for imbalanced learning such as LDAM.

5.4.6 iNaturalist-BG and Places-BG Details

The original iNaturalist 2017 dataset has 579,184 training images and 95,986 validation images. The dataset is long-tailed and each individual category is rare.

- $N = 5098$: Train a single model with all the 5098 categories in the dataset labeled. This corresponds to the standard setting on the iNaturalist dataset, with 0% of the data in the background class.
- $N = 1100$: Single model with 1100 categories with the 100 test categories included in N. 77.95% of data is in the background category.
- $N = 100$: Single model for all the 100 random test categories. 98.3% of data is in the background category.
- $N = 10$: 10 models each trained with a subset of 10 categories from the 100 randomly chosen categories. 99.83% of data is in the background category on average.
- $N = 1$: 10 binary models for a subset of 10 categories from the randomly chosen categories. We limit evaluate our evaluation to 10 categories in the case of binary models since it is too expensive to train separate deep binary models for all the 100 categories evaluated in other settings. As such, the performance for this setting is not directly comparable to the other settings. 99.98% of data is in the background category on average.

The original Places365 dataset has 1,803,460 training images and 21,700 validation images across 365 categories. The number of images per category ranges from 3,068 to 5,000.

- $N = 10$: 5 models each trained with a subset of 10 categories from the 365 places365 categories. 97.22% of data is in the background category on average.
- $N = 1$: 10 binary models for a subset of 10 categories from the $N = 10$ chosen categories. We limit evaluate our evaluation to 10 categories in the case of binary models since it is too expensive to train separate deep binary models for all the 100 categories evaluated in other settings. As such, the performance for this setting is not directly comparable to the other settings. 99.72% of data is in the background category on average.

5.4.7 Generalization of BG Splitting learned features

Are the features learned with Background Splitting more useful for training a classifier for a new set of categories than the features learned from a standard fine-tuning baseline? That is, are the features learned by BG Splitting just better for classifying the target set of categories, or does BG Splitting produce features which can be effectively adapted to classify new categories? Table 5.6 shows the accuracy of a linear classifier trained to classify a set of 100 new classes S2 using features produced by two methods trained on a different set of 100 classes S1: a standard fine-tuning baseline (FT), and our method (BG-SPLIT). The bottom row is the result of training the BG-SPLIT method to directly classify the S2 classes and thus represents an upper bound. We find that the features produced

Method	Weight Initialization	mAP on 100 new categories (S2)
Fine-tune last layer	FT on 100 categories (S1)	15.4
Fine-tune last layer	BG-SPLIT on 100 categories (S1)	32.4
BG-SPLIT (full model training on S2 categories)		44.4

Table 5.6: Feature generalization study. Using BG-SPLIT to train a model for one set of categories (S1) results in features than generalize well for a different set of categories (S2) compared to FT.

Batch size	FT				BG-SPLIT			
	AP	F1	Recall	Precision	AP	F1	Recall	Precision
128	28.4	19.0	**53.8**	12.9	48.2	44.8	**53.0**	44.0
256	37.7	30.8	50.8	24.7	**48.8**	**46.6**	42.8	61.3
512	**38.3**	**38.4**	42.5	41.1	47.9	44.3	37.4	68.1
1024	36.0	35.6	35.2	**44.1**	47.2	40.7	33.4	**67.1**

Table 5.7: Performance of standard fine-tuning and our approach with different batch sizes. Larger batch sizes than the ones traditionally used in balanced training (64 or 128) significantly improves performance of FT. BG-SPLIT (our approach) is more robust to batch size but also benefits from a larger batch size.

by the BG-SPLIT method are very useful for classifying new sets of categories, as seen by the 17 point mAP increase from using the BG-SPLIT features versus the FT features. Note that there is still significant advantage to training the entire model for the target set of new categories S2, as seen by further 12 point boost from 32.4 to 44.4 mAP, echoing our observation in Section 5.4 that using the small set of positives for the foreground categories is critical.

5.4.8 Batch Size Study

Classification networks are typically trained with a batch size of 64 or 128. However, we empirically observed that, in settings with a large background category, larger batch sizes (512 for traditional fine-tuning and 256 for our approach) result in improved performance over small batch sizes. To understand this effect, we evaluated the performance of both traditional fine-tuning and our approach for different batch sizes (Table 5.7). For traditional fine-tuning, a batch size of 512 results in the best performance. Our approach is more robust to smaller batch sizes and marginally benefits from increasing the batch size from 128 to 256. A clear empirical trend we observed is that increasing the batch size results in higher model precision at the cost of lower recall. Our hypothesis for this trend is that as the batches become larger, the model sees more positives and hard negatives in the same batch, requiring the model to be more discriminative, thus increasing precision at the cost of reducing recall. Further exploration of this behavior could lead to better sampling and batch size selection techniques in scenarios with extreme imbalance and a majority

	Setting ($N = 100$)							
	1% (36.64%)		5% (74.30%)		25% (93.52%)		100% (98.30%)	
	AP	F1	AP	F1	AP	F1	AP	F1
FT	23.5	18.7	27.9	22.4	36.3	32.5	38.3	38.4
LDAM	23.0	5.7	14.8	3.3	13.2	2.7	24.7	21.1
LWS	15.0	3.9	12.7	3.8	38.8	39.1	35.5	36.6
BG-SPLIT	-	-	-	-	-	-	**47.2**	**40.7**

Table 5.8: AP and F1 scores for baseline methods using a downsampled background category. In all cases, using the downsampled background category dataset does not result in performance which exceeds our method. In nearly all cases, the downsampled dataset results in reduced performance because it drastically decreases precision, resulting in a large number of false positives.

background category. We note that all other experiments in this are performed with a batch size of 1024 in order to produce results in a reasonable time–although training with smaller batch sizes results in a small increase in performance, it requires training much longer and with lower learning rates to reach convergence.

5.4.9 Background Category Downsampling

As the training issues caused by the extremely large background category are due to imbalance between the number of foreground and background examples, one might ask if simply 'downsampling' the number of background category instances would solve the problem. We implemented this approach by selecting increasingly smaller fractions of the background category images in the training set, and creating new training sets using all the foreground images plus this smaller background set. Table 5.8 shows the AP and F1 score for the baselines when trained with the downsampled background datasets for 100%, 25%, 5%, and 1% of all background instances. In nearly all cases, using the downsampled versions of the dataset results in lower overall performance across both AP and F1. In particular, we find that using these downsampled datasets increases the recall of the baseline methods, but at drastic cost to precision, resulting in a lower overall F1 and AP score.

5.5 Discussion

A key component in specializing deep models to narrow distributions is the ability to train accurate deep models with a small amount of data pertaining to task and a large amount of irrelevant or background data which can often be obtained without human labeling effort. State-of-the-art classification methods for handling imbalanced data do not perform well in the presence of a large and diverse background. In response we contribute a new approach that handles the imbalance by jointly training the main classification task along with an auxiliary classification task that involves categories that are better balanced and less rare. This approach improves on baselines by over 42 mAP points in situations of significant background dominance, making it feasible to train useful binary classification models for rare categories without additional human-provided labels. We also contribute the iNaturalist-BG and Places-BG datasets, which we hope will encourage further research in this important training regime. Our approach raises the broader question of integrating self-supervised representation learning methods with small amounts of human labeled data to learn accurate and highly specialized models. An interesting aspect of our approach is that it agnostic of how the feature representation used to formulate the auxiliary task is trained. We can in fact use self-supervised representations to cluster and generate labels for the auxiliary task and given the simplicity of our approach it can be more readily integrated with training for a specialized task.

Chapter 6

Conclusion

We outlined the challenges one needs to address for effectively specializing models and demonstrated the benefits of dynamically specializing models in several scenarios. Chapters 2 and 3 focus on efficient inference using dynamically specialized models. Chapters 4 and 5 focus on efficient training and leveraging specialized models for lowering human labeling effort. The are many avenues for further research in exploiting and exploring different aspects of model specialization. In the following sections we outline some of the directions for future work and point out broader implications of model specialization techniques presented in the thesis.

6.1 Exploring Different axes for Specialization

We have demonstrated the benefits of specializing models to short windows in time, to visually similar categories and to individual rare categories. There are a range of different axes to exploit model specialization beyond the ones explored in this thesis; especially in the context of autonomous robots and digital assistants personalized to specific users. Many autonomous robots are deployed in a specific spatial context like a user's home in the case of vacuuming robot or a neighborhood in an city in the case of a self-driving car or a warehouse in the case of a robotic arm that moves boxes around. In all these cases a model specialized to the specific spatial context could potentially be more accurate and efficient to train and deploy. For instance, models specialized to different geographic locations or indoor spaces can capture nuances that are specific to a city or a home like the kind of vehicles or furniture encountered. Another interesting axis for specialization is personalizing models to a specific user. For instance, in the case of digital assistants like Alexa or Siri, speech recognition models can be specialized to individual users to enable a better experience. Augmented and virtual reality experiences can benefit from specializing models both to user's spatial context as well as personal characteristics. We believe that specialization techniques proposed in the thesis can push the boundaries of model specialization in these application areas. Interesting avenues for future work include hierarchically specializing models at different time scales or spatial contexts and preserving user privacy when personalizing models.

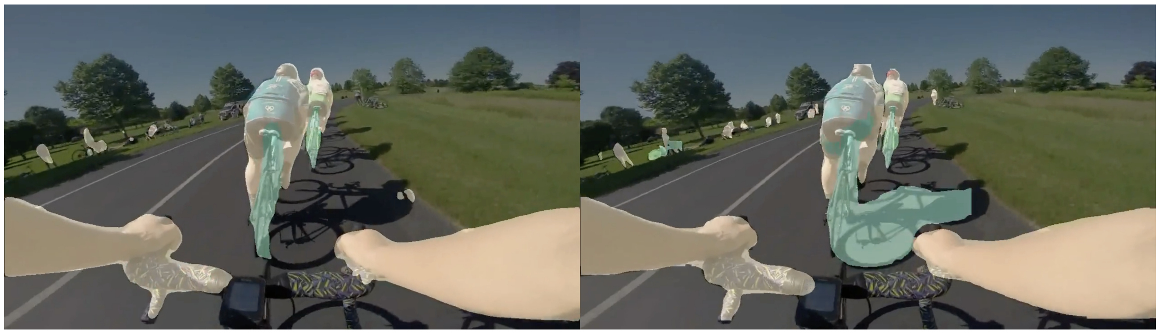
Another way of looking a model specialization is through the lens of multi-task learning. Solving a prediction problem on different subsets of the general distribution can be viewed as different tasks. For instance, detecting cars from an aerial satellite images is very different from detecting cars in google street view images. Taking this perspective establishes the connection between the model specialization perspective and heterogeneous multi-task learning. In multi-task learning models are typically specialized to perform different tasks on the same data. Instead of completely specializing models to each individual task like depth estimation, detection or motion estimation the recent trend especially in robotics settings has been to build models that share a representation which is then specialized for individual tasks. This is similar to the HydraNet architecture proposed in this thesis but for heterogeneous tasks as opposed to classification for different groups of visually similar categories. The outputs of these heterogeneous multi-task networks are used to predict and plan actions that other agents and the robot should execute. We believe that conditional execution architectures and training techniques similar to the ones we proposed in the thesis can be beneficial for both training and deployment of large heterogeneous multi-task models.

6.2 Utilizing Specialized Models for Supervision

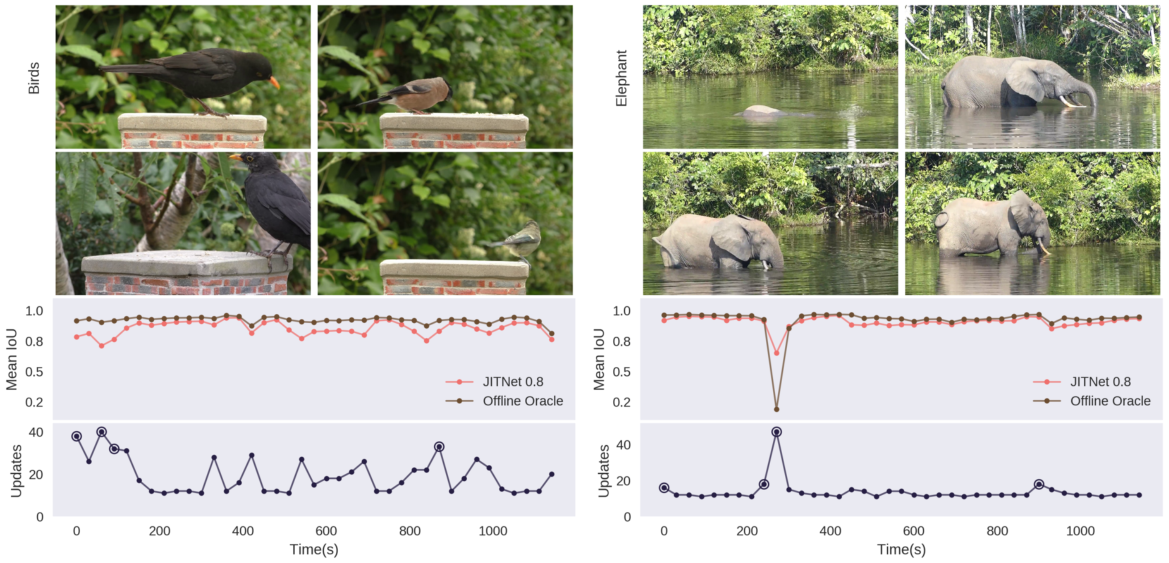
We exploited models specialized to individual rare categories to mine unlabeled data and focus human effort on the hard examples (chapter 4). We envision that temporally specialized models can similarly be used to reduce human effort in mining large video collections for hard instances. For instance, In Figure 6.1 the images on the top show an instance where the temporally specialized JITNet model does not predict the shadow as a bicycle but the more general Mask R-CNN model does. The graphs in the bottom of Figure 6.1 show the number of updates required for the temporally specialized JITNet model to adapt as a function of time. We find that the adaptation effort correlates well with interesting events that occur in a video stream. For instance, the peaks in the adaptation effort in the bottom left graph correspond to different birds and the peak in the bottom right graph corresponds to an elephant taking a dip in the water. Combining both these properties temporally specialized models can be used to identify interesting events in video and suggest labels to be verified by humans or used directly as a source of weak supervision. More broadly we believe that model specialization is an addition to the toolbox of self-supervised and semi-supervised techniques for reducing human labeling effort.

6.3 Rapid and Effective Model Specialization

A major challenge we address in the thesis is effectively training and adapting deep models with small amounts of labeled training data. We believe that techniques which enable specialization can be utilized for few-shot learning and domain adaptation. One way to look at both domain adaptation and few-shot learning problems is to think of them as specializing models to a new distribution and a new task respectively. Similarly, techniques



(a) On the left are the predictions of the temporally specialized JITNet model and on the right are predictions from the general Mask R-CNN model. The temporally specialized model does not predict the shadow as a bicycle while the general model does.



(b) Number of updates to JITNet during online distillation on two video streams. Plotted points are averages over a 30 second interval of the video. Images correspond to circled points in bottom plot, and show times where JITNet required frequent training to maintain accuracy.

Figure 6.1: **Top:** Temporally specialized JITNet models can often be more accurate than a general model because of coherence in the temporal context. **Bottom:** The effort required for temporal specialization is often indicative of interesting events in a video stream.

from style transfer, domain adaptation and low-shot learning can be leveraged for more effective specialization.

6.4 Datasets and Evaluation Metrics

A significant challenge in enabling future research on model specialization and more broadly continuous learning is datasets and evaluation metrics. Current datasets in machine learning and computer vision are largely geared towards supporting research on improving modeling and training techniques assuming the training and test distributions are identical. The standard academic metrics for evaluation focus on the average model performance on large scale datasets. In many real-world settings the target distribution continuous evolves over time violating the assumptions under which most large scale datasets are constructed. Moreover, in many settings high average case performance by itself is not sufficient or indicative of model reliability. Constructing more datasets like LVS which focus on evolving distributions and evaluation metrics which capture model performance on subsets of the distribution instead of average performance will be crucial going forward. Datasets and competitions that emulate settings like autonomous robot fleets or personalized digital assistants would bring attention to a host of challenges that are currently under explored in academic research. In addition to datasets, frameworks for model training and inference are largely geared towards static datasets and long development cycles. Exploiting the full benefits of rapid specialization would require rethinking the systems we use for training and deploying models today.

6.5 Continuous Learning Systems

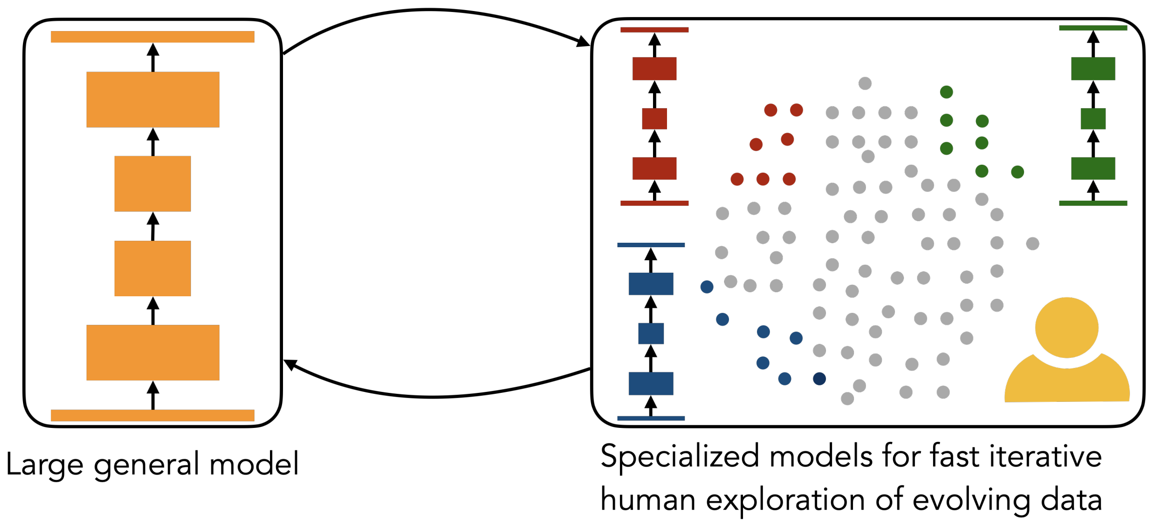

Large general model
Specialized models for fast iterative human exploration of evolving data

Figure 6.2: Effective model specialization can pave the way for rapid continuous learning on evolving streams of data. As general purpose models (*left*) become larger and larger (e.g., recent trends in language modeling [10, 91]), specializing compact models (*right*) to subsets of the overall distribution will be more crucial for inference efficiency. In turn, efficient models will enable humans to rapidly explore streams of unlabeled data to further improve general models.

The need for continuous learning systems is ubiquitous in many real-world scenarios where data driven models are widely used and there is a constant stream of new data being observed. For example, perception models on self-driving cars are continuously updated using data collected from the fleet of vehicles. An interesting aspect that emerges when we take a holistic view of the life cycle of a model is the relationship between model specialization, large scale modeling and continuous learning. Specializing models for efficient inference is intertwined with efficient supervision as illustrated in Figure 6.2. As larger and larger general models are trained with more data, computational resources and emerging self-supervised techniques (e.g., recent trends in language modeling [10, 91]), specializing compact models to subsets of the general distribution will be even more crucial for inference efficiency. These efficient specialized models can help humans label rapidly explore unlabeled data streams and provide supervision with low effort. The supervision in-turn improves the general model enabling a cycle of continuous human-in-the loop learning. We hope that the model specialization techniques presented in this thesis help pave the path towards machine learning systems the continuous learn and evolve with the data with low human effort.

Bibliography

[1] Umang Aggarwal, Adrian Popescu, and Céline Hudelot. Active learning for imbalanced datasets. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 1428–1437, 2020.

[2] Karim Ahmed, Mohammad Haris Baig, and Lorenzo Torresani. Network of experts for large-scale image categorization. In *European Conference on Computer Vision*, pages 516–532. Springer, 2016.

[3] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[4] Josh Attenberg and Seyda Ertekin. Class imbalance and active learning. *Imbalanced Learning: Foundations, Algorithms, and Applications*, pages 101–149, 2013.

[5] Josh Attenberg and Foster Provost. Why label when you can search? alternatives to active learning for applying human resources to build classification models under extreme class imbalance. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 423–432, 2010.

[6] Jimmy Ba and Brendan Frey. Adaptive dropout for training deep neural networks. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3084–3092. Curran Associates, Inc., 2013.

[7] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017.

[8] Abhijit Bendale and Terrance Boult. Towards open world recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1893–1902, 2015.

[9] Abhijit Bendale and Terrance E Boult. Towards open set deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1563–1572, 2016.

[10] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[11] Cristian Buciluă, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 535–541. ACM, 2006.

[12] Samuel Bulo, Lorenzo Porzi, and Peter Kontschieder. In-place activated batchnorm for memory-optimized training of DNNs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[13] Mausam C Lin. Active learning with unbalanced classes & example-generated queries. In *AAAI Conference on Human Computation*, 2018.

[14] Sergi Caelles, Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. One-shot video object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017.

[15] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. In *Advances in Neural Information Processing Systems*, pages 1565–1576, 2019.

[16] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *NeurIPS*, 2020.

[17] Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

[18] Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

[19] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. *Semi-Supervised Learning*. MIT Press, 2006.

[20] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[21] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*, 2020.

[22] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

[23] Yukun Chen and Subramani Mani. Active learning for unbalanced data in the challenge with multiple models and biasing. In *Active Learning and Experimental Design workshop In conjunction with AISTATS 2010*, pages 113–126, 2011.

[24] François Chollet. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357, 2016.

[25] Galen Chuang, Giulia DeSalvo, Lazaros Karydas, Jean-François Kagy, Afshin Rostamizadeh, and A Theeraphol. Active learning empirical study. NeurIPS 2019 Workshop on Learning with Rich Experience: Integration of Learning Paradigms, 2019. Available at workshop website http://sites.google.com/view/neurips2019lire.

[26] Cody Coleman, Edward Chou, Sean Culatana, Peter Bailis, Alexander C Berg, Roshan Sumbaly, Matei Zaharia, and I Zeki Yalniz. Similarity search for efficient active learning and search of rare concepts. *arXiv preprint arXiv:2007.00077*, 2020.

[27] Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via proxy: Efficient data selection for deep learning. *ICLR*, 2020.

[28] 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.

[29] Jia Deng, A. C. Berg, and Li Fei-Fei. Hierarchical semantic indexing for large scale image retrieval. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 785–792, Washington, DC, USA, 2011. IEEE Computer Society.

[30] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[31] Jia Deng, Sanjeev Satheesh, Alexander C. Berg, and Fei Li. Fast and balanced: Efficient label tree learning for large scale object recognition. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 567–575. Curran Associates, Inc., 2011.

[32] Akshay Raj Dhamija, Manuel Günther, and Terrance Boult. Reducing network agnostophobia. In *Advances in Neural Information Processing Systems*, pages 9157–9168, 2018.

[33] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015.

[34] FAIR. Detectron Mask R-CNN. https://github.com/facebookresearch/Detectron, 2018.

[35] Michael Figurnov, Maxwell D. Collins, Yukun Zhu, Li Zhang, Jonathan Huang, Dmitry P. Vetrov, and Ruslan Salakhutdinov. Spatially adaptive computation time for residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[36] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[37] Francois Fleuret, Jerome Berclaz, Richard Lengagne, and Pascal Fua. Multicamera people tracking with a probabilistic occupancy map. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):267–282, 2008.

[38] Raghudeep Gadde, Varun Jampani, and Peter V Gehler. Semantic video CNNs through representation warping. *CoRR, abs/1708.03088*, 2017.

[39] Mingfei Gao, Zizhao Zhang, Guo Yu, Sercan Ö Arık, Larry S Davis, and Tomas Pfister. Consistency-based semi-supervised active learning: Towards minimizing labeling cost. In *European Conference on Computer Vision*, pages 510–526. Springer, 2020.

[40] Roman Garnett, Yamuna Krishnamurthy, Xuehan Xiong, Jeff Schneider, and Richard Mann. Bayesian optimal active search and surveying. *arXiv preprint arXiv:1206.6406*, 2012.

[41] Rohit Girdhar, Du Tran, Lorenzo Torresani, and Deva Ramanan. Distinit: Learning video representations without a single labeled video, 2019.

[42] Jonathan Goodman, Albert G Greenberg, Neal Madras, and Peter March. Stability of binary exponential backoff. *Journal of the ACM (JACM)*, 35(3):579–602, 1988.

[43] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Moham-

mad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

[44] Mohamed Farouk Abdel Hady and Friedhelm Schwenker. Combining committee-based semi-supervised and active learning and its application to handwritten digits recognition. In *International Workshop on Multiple Classifier Systems*, pages 225–234. Springer, 2010.

[45] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A. Horowitz, and William J. Dally. Eie: Efficient inference engine on compressed deep neural network. *SIGARCH Comput. Archit. News*, 44(3):243–254, June 2016.

[46] Song Han, Jeff Pool, Sharan Narang, Huizi Mao, Shijian Tang, Erich Elsen, Bryan Catanzaro, John Tran, and William J. Dally. DSD: regularizing deep neural networks with dense-sparse-dense training flow. *CoRR*, abs/1607.04381, 2016.

[47] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1135–1143. Curran Associates, Inc., 2015.

[48] Sam Hare, Stuart Golodetz, Amir Saffari, Vibhav Vineet, Ming-Ming Cheng, Stephen L Hicks, and Philip HS Torr. Struck: Structured output tracking with kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(10):2096–2109, 2016.

[49] Elmar Haussmann, Michele Fenzi, Kashyap Chitta, Jan Ivanecky, Hanson Xu, Donna Roy, Akshita Mittel, Nicolas Koumchatzky, Clement Farabet, and Jose M Alvarez. Scalable active learning for object detection. *arXiv preprint arXiv:2004.04699*, 2020.

[50] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284, 2009.

[51] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988. IEEE, 2017.

[52] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[53] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[54] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015.

[55] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[56] Seunghoon Hong, Tackgeun You, Suha Kwak, and Bohyung Han. Online tracking by learning discriminative saliency maps with convolutional neural network. In *International Conference on Machine Learning*, pages 597–606, 2015.

[57] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[58] Kevin Hsieh, Ganesh Ananthanarayanan, Peter Bodik, Shivaram Venkataraman, Paramvir Bahl, Matthai Philipose, Phillip B Gibbons, and Onur Mutlu. Focus: Querying large video datasets with low latency and low cost. In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, pages 269–286, 2018.

[59] Peiyun Hu, Zachary C Lipton, Anima Anandkumar, and Deva Ramanan. Active learning with partial feedback. *arXiv preprint arXiv:1802.07427*, 2018.

[60] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q. Weinberger. Multi-scale dense convolutional networks for efficient prediction. *CoRR*, abs/1703.09844, 2017.

[61] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[62] Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. *CoRR*, abs/1602.07360, 2016.

[63] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2462–2470, 2017.

[64] Yani Ioannou, Duncan P. Robertson, Darko Zikic, Peter Kontschieder, Jamie Shotton, Matthew Brown, and Antonio Criminisi. Decision forests, convolutional networks and the models in-between. *CoRR*, abs/1603.01250, 2016.

[65] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. Label propagation for deep semi-supervised learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5070–5079, 2019.

[66] Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodik, Siddhartha Sen, and Ion Stoica. Chameleon: scalable adaptation of video analytics. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 253–266, 2018.

[67] Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodik, Siddhartha Sen, and Ion Stoica. Chameleon: Scalable adaptation of video analytics. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '18, pages 253–266, New York, NY, USA, 2018. ACM.

[68] Shali Jiang, Roman Garnett, and Benjamin Moseley. Cost effective active search. In *Advances in Neural Information Processing Systems*, pages 4880–4889, 2019.

[69] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Inferring and executing programs for visual reasoning. *arXiv preprint arXiv:1705.03633*, 2017.

[70] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Daniel Killebrew, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary,

Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ISCA '17, pages 1–12, New York, NY, USA, 2017. ACM.

[71] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-Learning-Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1409–1422, 2012.

[72] Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. In *International Conference on Learning Representations*, 2020.

[73] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. Noscope: Optimizing neural network queries over video at scale. *Proceedings of the VLDB Endowment*, 10(11):1586–1597, 2017.

[74] Marvin Klingner, Andreas Bär, Philipp Donn, and Tim Fingscheidt. Class-incremental learning for semantic segmentation re-using neither old data nor old labels. *CoRR*, abs/2005.06050, 2020.

[75] Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Bulo. Deep neural decision forests. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[76] Simon Kornblith, Jonathon Shlens, and Quoc V. Le. Do better imagenet models transfer better? *CoRR*, abs/1805.08974, 2018.

[77] Jan Koutnik, Klaus Greff, Faustino Gomez, and Juergen Schmidhuber. A clockwork RNN. In *Proceedings of the International Conference on Machine Learning*, pages 1863–1871, 2014.

[78] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. HMDB: A large video database for human motion recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2556–2563. IEEE, 2011.

[79] Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 2, 2013.

[80] David D Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Machine learning proceedings 1994*, pages 148–156. Elsevier, 1994.

[81] Hui Li, Xuejun Liao, and Lawrence Carin. Active learning for semi-supervised multi-task learning. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1637–1640. IEEE, 2009.

[82] Yu Li, Tao Wang, Bingyi Kang, Sheng Tang, Chunfeng Wang, Jintao Li, and Jiashi Feng. Overcoming classifier imbalance for long-tail object detection with balanced group softmax. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10991–11000, 2020.

[83] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[84] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[85] 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*, pages 740–755. Springer, 2014.

[86] Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Pensky. Sparse convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[87] Honghai Liu, Shengyong Chen, and Naoyuki Kubota. Intelligent video systems and analytics: A survey. *IEEE Trans. Industrial Informatics*, 9(3):1222–1233, 2013.

[88] Lanlan Liu and Jia Deng. Dynamic deep neural networks: Optimizing accuracy-efficiency trade-offs by selective execution. *CoRR*, abs/1701.00299, 2017.

[89] Qing Liu, Lingxi Xie, Huiyu Wang, and Alan Yuille. Semantic-aware knowledge preservation for zero-shot sketch-based image retrieval, 2019.

[90] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. *Lecture Notes in Computer Science*, page 21–37, 2016.

[91] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[92] Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X. Yu. Large-scale long-tailed recognition in an open world. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[93] Martin Loncaric. Handling "background" classes in machine learning. `https://thehive.ai/insights/handling-background-classes-in-machine-learning`, Jul 2018. Accessed 2020-05-31.

[94] Wei-Lwun Lu, Jo-Anne Ting, James J Little, and Kevin P Murphy. Learning to track and identify players from broadcast sports videos. *IEEE transactions on pattern analysis and machine intelligence*, 35(7):1704–1716, 2013.

[95] Chao Ma, Jia-Bin Huang, Xiaokang Yang, and Ming-Hsuan Yang. Hierarchical convolutional features for visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 3074–3082, 2015.

[96] K. Maninis, S. Caelles, Y. Chen, J. Pont-Tuset, L. Leal-Taixe, D. Cremers, and L. Van Gool. Video object segmentation without temporal information. 2018.

[97] Ofer Matan, RK Kiang, CE Stenard, B Boser, JS Denker, D Henderson, RE Howard, W Hubbard, LD Jackel, and Yann Lecun. Handwritten character recognition using neural network architectures. In *Proceedings of the 4th US Postal Service Advanced Technology Conference, Washington DC, November 1990*, 1990.

[98] Andrew Kachites McCallumzy and Kamal Nigamy. Employing em and pool-based active learning for text classification. In *Proc. International Conference on Machine Learning (ICML)*, pages 359–367. Citeseer, 1998.

[99] Mason McGill and Pietro Perona. Deciding how to decide: Dynamic routing in artificial neural networks. In *International Conference on Machine Learning*, pages 2363–2372, 2017.

[100] Sudhanshu Mittal, Maxim Tatarchenko, Özgün Çiçek, and Thomas Brox. Parting with illusions about deep active learning. *arXiv preprint arXiv:1912.05361*, 2019.

[101] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[102] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[103] Ravi Teja Mullapudi, Fait Poms, William R. Mark, Deva Ramanan, and Kayvon Fatahalian. Background splitting: Finding rare classes in a sea of background, 2020.

[104] Ion Muslea, Steven Minton, and Craig A Knoblock. Active+ semi-supervised learning= robust multi-view learning. In *ICML*, volume 2, pages 435–442, 2002.

[105] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4293–4302. IEEE, 2016.

[106] Kemal Oksuz, Baris Can Cam, Sinan Kalkan, and Emre Akbas. Imbalance problems in object detection: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[107] Margarita Osadchy, Daniel Keren, and Bella Fadida-Specktor. Hybrid classifiers for object classification with a rich background. In *European Conference on Computer Vision*, pages 284–297. Springer, 2012.

[108] Yassine Ouali, Céline Hudelot, and Myriam Tami. An overview of deep semi-supervised learning. *arXiv preprint arXiv:2006.05278*, 2020.

[109] Wanli Ouyang, Xiaogang Wang, Cong Zhang, and Xiaokang Yang. Factors in finetuning deep model for object detection with long-tail distribution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 864–873, 2016.

[110] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[111] Genevieve Patterson, Grant Van Horn, Serge Belongie, Pietro Perona, and James Hays. Tropel: Crowdsourcing detectors with minimal training. In *Third AAAI Conference on Human Computation and Crowdsourcing*, 2015.

[112] Federico Perazzi, Anna Khoreva, Rodrigo Benenson, Bernt Schiele, and Alexander Sorkine-Hornung. Learning video object segmentation from static images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[113] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[114] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. *CoRR*, abs/1403.6382, 2014.

[115] Sylvestre-Alvise Rebuffi, Sebastien Ehrhardt, Kai Han, Andrea Vedaldi, and Andrew Zisserman. Semi-supervised learning with scarce annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 762–763, 2020.

[116] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 91–99. Curran Associates, Inc., 2015.

[117] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[118] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[119] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[120] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[121] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.

[122] Burr Settles. Active learning literature survey (computer sciences technical report 1648). *University of Wisconsin-Madison*, 2010.

[123] Shai Shalev-Shwartz et al. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2012.

[124] L. Shao, F. Zhu, and X. Li. Transfer learning for visual categorization: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 26(5):1019–1034, 2015.

[125] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *CoRR*, abs/1701.06538, 2017.

[126] Evan Shelhamer, Kate Rakelly, Judy Hoffman, and Trevor Darrell. Clockwork convnets for video semantic segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 852–868. Springer, 2016.

[127] H. Shen, S. Han, M. Philipose, and A. Krishnamurthy. Fast video classification via adaptive cascading of deep models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[128] Haichen Shen, Seungyeop Han, Matthai Philipose, and Arvind Krishnamurthy. Fast video classification via adaptive cascading of deep models. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3646–3654, 2017.

[129] Yanyao Shen, Hyokun Yun, Zachary C Lipton, Yakov Kronrod, and Animashree Anandkumar. Deep active learning for named entity recognition. *arXiv preprint arXiv:1707.05928*, 2017.

[130] K. Shmelkov, C. Schmid, and K. Alahari. Incremental learning of object detectors without catastrophic forgetting. In *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.

[131] Assaf Shocher, Nadav Cohen, and Michal Irani. "Zero-shot" super-resolution using deep internal learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3118–3126, 2018.

[132] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 761–769, 2016.

[133] Daniel L Silver and Robert E Mercer. The task rehearsal method of life-long learning: Overcoming impoverished data. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 90–101. Springer, 2002.

[134] Oriane Siméoni, Mateusz Budnik, Yannis Avrithis, and Guillaume Gravier. Rethinking deep active learning: Using unlabeled data at model training. *arXiv preprint arXiv:1911.08177*, 2019.

[135] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pages 4077–4087, 2017.

[136] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *arXiv preprint arXiv:2001.07685*, 2020.

[137] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[138] Jong-Chyi Su and Subhransu Maji. Adapting models to signal degradation using distillation. In *Proc. British Machine Vision Conference*, 2017.

[139] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[140] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[141] TensorFlow. TensorFlow DeepLab Model Zoo. `https://github.com/tensorflow/models/blob/master/research/deeplab/g3doc/model_zoo.md`, 2018.

[142] Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need? *arXiv preprint arXiv:2003.11539*, 2020.

[143] Yi-Hsuan Tsai, Ming-Hsuan Yang, and Michael Black. Video segmentation via object flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[144] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8769–8778, 2018.

[145] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, 2018.

[146] Grant Van Horn and Pietro Perona. The devil is in the tails: Fine-grained classification in the wild. *arXiv preprint arXiv:1709.01450*, 2017.

[147] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages I–511–I–518 vol.1, 2001.

[148] Paul Voigtlaender and Bastian Leibe. Online adaptation of convolutional neural networks for video object segmentation. In *BMVC*, 2017.

[149] Carl Vondrick and Deva Ramanan. Video annotation and tracking with active learning. In *Advances in Neural Information Processing Systems*, pages 28–36, 2011.

[150] Keze Wang, Dongyu Zhang, Ya Li, Ruimao Zhang, and Liang Lin. Cost-effective active learning for deep image classification. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(12):2591–2600, 2016.

[151] Lijun Wang, Wanli Ouyang, Xiaogang Wang, and Huchuan Lu. Visual tracking with fully convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 3119–3127, 2015.

[152] Lin Wang and Kuk-Jin Yoon. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. *arXiv preprint arXiv:2004.05937*, 2020.

[153] Tao Wang, Yu Li, Bingyi Kang, Junnan Li, Jun Hao Liew, Sheng Tang, Steven Hoi, and Jiashi Feng. Classification calibration for long-tail instance segmentation. *arXiv preprint arXiv:1910.13081*, 2019.

[154] Yu-Xiong Wang, Ross Girshick, Martial Hebert, and Bharath Hariharan. Low-shot learning from imaginary data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7278–7286, 2018.

[155] Manfred K Warmuth, Jun Liao, Gunnar Rätsch, Michael Mathieson, Santosh Putta, and Christian Lemmen. Active learning with support vector machines in the drug discovery process. *Journal of chemical information and computer sciences*, 43(2):667–673, 2003.

[156] Colin Wei, Kendrick Shen, Yining Chen, and Tengyu Ma. Theoretical analysis of self-training with deep networks on unlabeled data. *arXiv preprint arXiv:2010.03622*, 2020.

[157] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2074–2082. Curran Associates, Inc., 2016.

[158] Ning Xu, Linjie Yang, Yuchen Fan, Dingcheng Yue, Yuchen Liang, Jianchao Yang, and Thomas Huang. YouTube-VOS: A large-scale video object segmentation benchmark. *arXiv preprint arXiv:1809.03327*, 2018.

[159] I Zeki Yalniz, Hervé Jégou, Kan Chen, Manohar Paluri, and Dhruv Mahajan. Billion-scale semi-supervised learning for image classification. *arXiv preprint arXiv:1905.00546*, 2019.

[160] Zhicheng Yan, Vignesh Jagadeesh, Dennis DeCoste, Wei Di, and Robinson Piramuthu. HD-CNN: hierarchical deep convolutional neural network for image classification. *CoRR*, abs/1410.0736, 2014.

[161] Linjie Yang, Yanran Wang, Xuehan Xiong, Jianchao Yang, and Aggelos K Katsaggelos. Efficient video object segmentation via network modulation. *Proceedings of the International Conference on Robotics and Automation*, 2018.

[162] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[163] Wei Zhu, Haofu Liao, Wenbin Li, Weijian Li, and Jiebo Luo. Alleviating the incompatibility between cross entropy loss and episode training for few-shot skin disease classification. *arXiv preprint arXiv:2004.09694*, 2020.

[164] Xiangxin Zhu, Dragomir Anguelov, and Deva Ramanan. Capturing long-tail distributions of object subcategories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 915–922, 2014.

[165] Xiaojin Zhu and Andrew B Goldberg. Introduction to semi-supervised learning (synthesis lectures on artificial intelligence and machine learning). *Morgan and Claypool Publishers*, 14, 2009.

[166] Xizhou Zhu, Yuwen Xiong, Jifeng Dai, Lu Yuan, and Yichen Wei. Deep feature flow for video recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, page 7, 2017.