Continual Learning for Deep Dense Prediction

Sanket A. Lokegaonkar

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Science and Applications

Naren Ramakrishnan, Chair
Jia-Bin Huang, Co-Chair
Bert Huang

May 08, 2016
Blacksburg, Virginia

Keywords: Computer Vision, Continual Learning, Image Segmentation , Dense Prediction

Continual Learning for Deep Dense Prediction

Sanket A. Lokegaonkar

(ABSTRACT)

Transferring a deep learning model from old tasks to a new one is known to suffer from the catastrophic forgetting effects. Such forgetting mechanism is problematic as it does not allow us to accumulate knowledge sequentially and requires retaining and retraining on all the training data. Existing techniques for mitigating the abrupt performance degradation on previously trained tasks are mainly studied in the context of image classification. In this work, we present a simple method to alleviate catastrophic forgetting for pixel-wise dense labeling problems. We build upon the regularization technique using knowledge distillation to minimize the discrepancy between the posterior distribution of pixel class labels for old tasks predicted from 1) the original and 2) the updated networks. This technique, however, might fail in circumstances where the source and target distribution differ significantly. To handle the above scenario, we further propose an improvement to the distillation based approach by adding adaptive l2-regularization depending upon the per-parameter importance to the older tasks. We train our model on FCN8s, but our training can be generalized to stronger models like DeepLab, PSPNet, etc. Through extensive evaluation and comparisons, we show that our technique can incrementally train dense prediction models for novel object classes, different visual domains, and different visual tasks.

Continual Learning for Deep Dense Prediction

Sanket A. Lokegaonkar

(GENERAL AUDIENCE ABSTRACT)

Modern deep networks have been successful on many important problems in computer vision viz. object classification, object detection, pixel-wise dense labeling, etc. However, learning various tasks incrementally still remains a fundamental problem in computer vision. When trained incrementally on multiple tasks, deep networks have been known to suffer from catastrophic forgetting on the older tasks, which leads to significant decrease in accuracy. Such forgetting is problematic and fundamentally constraints the knowledge that can be accumulated within these networks.

In this work, we present a simple algorithm to alleviate catastrophic forgetting for pixel-wise dense labeling problems. To prevent the network from forgetting connections important to the older tasks, we record the predicted labels/outputs of the older tasks for the images of the new task. While training on the images of the new task, we enforce a constraint to "remember" the recorded predictions of the older tasks while learning a new task. Additionally, we identify which connections in the deep network are important to the older tasks and prevent these connections from changing significantly. We show that our proposed algorithm can incrementally learn dense prediction models for novel object classes, different visual domains, and different visual tasks.

To my parents, for all their love and support and providing me with the best education possible. I appreciate their sacrifices and I wouldn't have been able to get to this stage without them.

Acknowledgments

First and foremost, I would like to extend my sincere gratitude to both my graduate advisors Prof. Jia-Bin Huang and Prof. Naren Ramakrishnan , for providing me with the opportunity to work on exciting research problems and the guidance to push through. I would like to especially thank my thesis advisor, Prof. Jia-Bin for his constant motivation to push harder and hours of insightful and valuable suggestions. I would also like to thank all my lab-mates from Vision & Learning Lab, especially Jinwoo, Chen, Yuliang , George Yeh for their suggestions and helpful comments. Last but not least, I would like to thank my parents , my grandparents and my family to whom I am grateful for all their unconditioned love and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Learning visual capabilities incrementally is a fundamental task in a computer vision. CNN based architectures [7, 14] work very well on the new task when trained on a new task, but show "catastrophic forgetting" [6] when trained on the older tasks. This problem only becomes harder with difficulty/diversity of the subsequent tasks. In this work, we focus our attention on learning dense prediction tasks like segmentation [2, 4, 36, 38] and depth prediction incrementally while minimizing the "forgetting" behavior on the previously learned dense prediction tasks. Here, we present a simple method which can work well to alleviate catastrophic forgetting for dense-prediction problems. Our objective in this work is given a set of dense prediction tasks to learn a model which can learn tasks incrementally and achieve performance close to optimal.

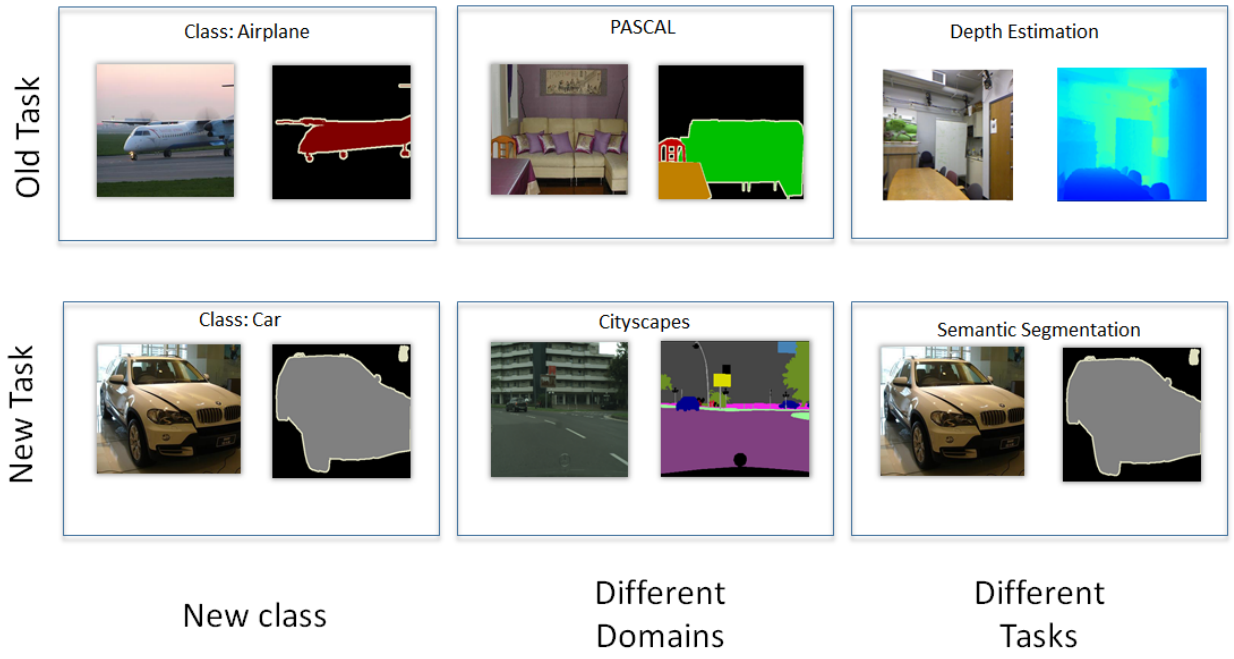


Figure 1.1: **Illustration of continual learning for learning classes, with different domains, with different modalities**

In the immediate future, there is a real need for systems to be able to learn new tasks incrementally. Autonomous vehicles would be expected to generalize well to different conditions quickly without learning a new task, Hazard robots working in a long-lived environment to learn new tasks(depth estimation in a low-light environment) or classes(hazardous materials) incrementally. Modern datasets are continuously evolving with new classes and more data. Fine-tuning allows to adapt the network to the new data and tasks but drops the performance on the older tasks dramatically. Multi-task/Joint learning [3] of all tasks is not always feasible solution in this case either. Especially for dense prediction tasks, Joint Learning becomes significantly harder due to time constraints, memory limitations and memory requirements to store all training data throughout the lifetime of the network.

Continual Learning would allow us to incrementally-learn and adapt to changing datasets, work in a limited data-storage environment and learn different tasks without a significant

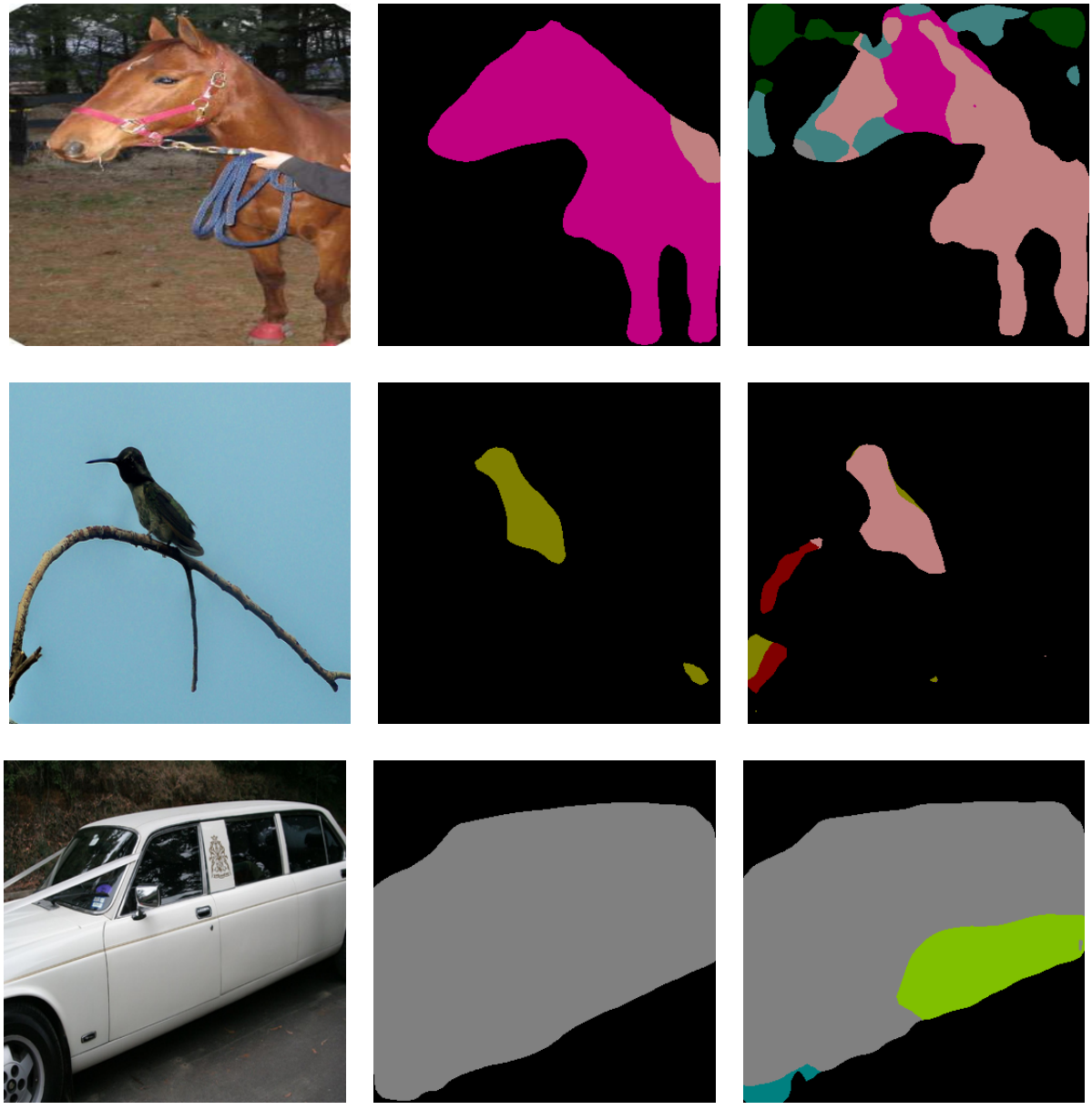
drop in older task performance. Ultimately, this research provides further development/impetus in the direction of the fundamental problem of learning sequentially without forgetting for general artificial intelligence.

Despite the importance of the problem, it remains underexplored for the modern deep learning paradigm. Most of the existing techniques have studied this problem from classification and reinforcement learning perspective [17, 23, 24, 33] for these models. Recently, Shmelkov et al. [29] proposed a method to continually learn object detector in a Fast-RCNN framework. *However, we are the first approach to extend the continual learning problem setting to pixel-wise tasks.*

In this work, we propose a generic method for continual learning in dense prediction tasks, which utilizes knowledge distillation method proposed by Hinton et al. [8] for regularizing older tasks, similar to work by Li et al. [17] in classification setting. The knowledge distillation objective imposes a constraint to minimize the changes to the posterior distribution of labels for older tasks when learning a new task. This simple technique works quite well for dense prediction tasks. We further try to improve the proposed distillation-based approach by leveraging a key observation.

The distillation-based regularization might fail when the distribution of images significantly differs from the core distribution. As noted by Rebuffi et al. [22] for classification setting, the distillation based approach suffers from a significant drop in older task performance if the following tasks have data from different distributions.

We propose an improvement upon the distillation-based method by adding an adaptive l2-regularization loss depending upon the importance of per-parameter towards the older task. The per-parameter importance is constructed for both the shared layers and the task-specific layers, using limited data samples from the previous tasks. The suggested improvement has the advantage of constraining parameters/weights important to the older task, while distill-



(a) Image(Pascal VOC)

(c) Before Training on New Task (Cityscapes)

(c) After Training on New Task (Cityscapes)

Figure 1.2: **Failure of finetuning** Without any regularization, the old task performance drops after training on the new task. This can be seen in the low quality of segmentation mask.

ing from the new task data distribution. We demonstrate experimental results on a variety of settings, seen in the real world (incremental classes, different domains, different tasks). Our results show the flexibility of our approach to a variety of tasks.

In this work, we make the following three contributions.

- We provide a model-agnostic solution to continual learning for dense prediction tasks using knowledge distillation.
- We propose an improvement upon the solution for handling different distributions of data in the tasks by introducing per-parameter importance for both the shared and task-specific feature layers.
- We experimentally show the flexibility of our approach to a variety of real-world settings (incremental classes, different domains, different tasks).

Chapter 2

Review of Literature

Continual learning has been a long sought-after goal in the recent machine learning and artificial intelligence [26, 32]. Continual learning/ Lifelong learning focuses on learning through a sequence of tasks while retaining knowledge from the older tasks and adapting to new tasks quickly. This problem is crucial from modern deep learning perspective as this behavior is seen strongly in modern deep network architectures.

2.1 Why forgetting happens

The problem of catastrophic forgetting has been diagnosed to be caused due to two major factors [6, 15]. First, the representations learned by the internal layers of the deep models are heavily correlated, and change in single neuron among these networks significantly affects the representations by higher layers [6]. Second, all the parameters in the network are used for forward propagation through the network, and in each backpropagation update for a training point, all the parameters in the network are potentially affected [15]. This problem was first seen in classical connectionist networks decades ago but becomes crucially important with

the larger deep network architectures of today [7, 31].

2.2 Relationship to Fine-tuning/ Multitask Learning/ Domain Adaptation

Our work also relates to the problem of fine-tuning (transfer learning), multi-task learning and domain adaptation.

Finetuning Fine-tuning is the classical technique proposed for adapting to a new task. The method for transferring knowledge in dense prediction tasks is similar to that of classification. The trained Fully Convolutional Network (FCN) for the older task is used as the initialization. The randomly initialized output / task-specific layers are added to the FCN and trained with decreased learning rate. As there is no additional regularization on the older task, the performance on the old task suffers dramatically.

Multitask Learning Multitask learning [3] is defined as jointly learning the tasks simultaneously, where all the training data is available at once. This is equivalent to learning the combined joint model for all the tasks jointly, where each task data is stored at all times. This problem focuses on the learning the best representation which can work with the all the tasks. This is an active area of research, and many new approaches have been proposed recently for the dense prediction tasks [11, 13]. However, this method represents the upper bound / ideal representation for continual learning approaches.

Domain Adaptation Domain adaptation [9, 20, 34] transfers or adapts the learned representation from source domain task to a representation that can work well with the target

domain. Domain adaptation techniques usually share the same label space but have different data distributions. Domain adaptation differs from continual learning in two key aspects. Firstly, it does not attempt to preserve the performance in the older task. Secondly, the new task is usually unlabeled or sparsely labeled.

2.3 Categorizing Continual Learning Literature

Most of the previous work in this domain focuses only on the simple task of classification [17, 23, 24, 25] . Recent works in this domain can loosely be categorized into the (1) learning continually with architectural changes (2) learning continually with architectural changes by constraining the outputs (3) learning continually with architectural changes by constraining the weights.

Learning continually with architectural changes Approaches in this category focus on altering the architecture to reduce interference between tasks and reduce catastrophic forgetting. Early approaches use simple strategies like freezing specific weights in a network [27], using decreased learning rate while fine-tuning [35]. Rusu et al. [25] proposed the new architecture where the network for the previous task is kept frozen and additional lateral layers are added while solving a new task. It completely avoids forgetting on older tasks, but the model complexity grows with the increase in the number of tasks. Rosenfeld et al. [24] and Rebuffi et al. [23] proposed architectures which utilized a strong base network and suppressing activations with task-specific filters. The architecture prevents catastrophic forgetting, with a smaller memory footprint, but is bounded by the performance of the base network and does not in its true spirit learn continually. Lee et al. [16] proposed an approach of dynamically varying the capacity of the neural network for individual tasks using sparse

connectivity. Their approach achieves performance close to the multi-task performance and better than progressive networks. The fundamental weakness of this category of approaches is that they require architecture-specific modifications and are difficult to scale for a large number of tasks.

Learning continually by constraining the outputs Approaches in this category focus on using regularizing terms which penalizes changes in the functional outputs of the older tasks. Li et al. [17] proposed regularizing networks to maintain the activations of the previous tasks network to be close to the activations of the current network while applying to a new task, using knowledge distillation. This works well when the data distribution in the new task is similar to the old tasks but does not regularize effectively when not. Followup work by Triki et al. [33] uses intermediate-layer regularization with under-complete auto-encoders to regularize when the two tasks are not very similar to each other. We follow a similar approach as Li et al. [17] using knowledge distillation for preserving the activations of the older task in the fully convolutional model setting. Instead of intermediate-layer regularization which requires expensive training of an under-complete auto-encoder, We take advantage of the per-parameter importance methods (explored in structural regularization methods) and propose an improvement which chooses to adaptively regularize in the parameter space, based on the importance of the parameter. Other generative models based approaches [28] learn a generator for the old task and use samples generated from the generator to "remember" the old task. These class of approaches is limited in capacity by the generative model, and it requires expensive training of generative model for each previous task. Recent work by Lopez-Paz et al. [19] proposes using constraints on the gradients to prevent significant loss in old task connections. This method suffers from the limitation that the performance of this method is dependent on the samples and bounded by the memory.

Learning continually by constraining the weights/parameters Approaches in this category focus on regularizing by the parameters of the networks in such a way that the parameters important to previous tasks stay close to their original values. Kirkpatrick et al. [12] introduced quadratic regularization on the parameters of the network, scaled by their relative "importance" of the parameters, computed by Fisher Information Matrix. We use this paradigm proposed by Kirkpatrick et al. and extend to both the internal shared task parameters and task-specific parameters while distilling the older task. Followup work in this domain by Zenke et al. [37] focuses on computing importance of the parameters as the network is being trained. These methods are heavily inspired by biology, primarily working on the intuition that synapses in the brain have more complex dynamics than scalar weights in deep learning networks.

2.4 Deep Dense Prediction using Fully Convolutional Networks

We follow Convolutional Neural Network based architecture for dense prediction problems. Precisely, we follow the fully-convolutional approach developed by Long et al. [18] to predict and regress inputs (images) of arbitrary size/resolution to the corresponding size output (depth/segmentation maps/flow, etc.) The critical difference between standard deep learning networks and the fully convolutional networks is the convolutions are used for mapping to the output space instead of fully connected operations seen in classification networks. Fully-convolutional methods have since become the state-of-the-art method for semantic segmentation, single-image depth estimation, and many pixel-wise prediction problems. Following the FCN8s, there have many recent advances in the dense prediction space viz replacing the base network (VGG with ResNet), multi-scale training, coarse-to-fine structure with de-

convolution networks, dilated convolutions. Our proposed solution for continual learning is model-agnostic and should be able to work with any modern architectures without significant changes.

Chapter 3

Approach

3.1 Overview

Our overall approach for continual learning in dense prediction tasks is described in figure 3.1. Given an FCN model with shared feature layers defined by f_s and task-specific layers defined by f_i, the objective of continual learning is to learn a new task-specific layer t, such that the performance on the set of old tasks $i \in \{1..(t-1)\}$ is reasonably maintained and the new task-specific layers are learnt.

To prevent catastrophic forgetting behavior in the shared layers f_s, we need to constrain the internal layers with regularization. Similar to the Li et al. [17], we first propose to constrain the network by knowledge distillation loss function, extending to pixel-wise space. Our algorithm works in two stages:

Stage 1 For each new training image X_t of new task t, we first record the responses of the older tasks $R_i(X_t)$. In case of dense prediction, the responses of the older task mean differently depending on the nature of the tasks. E.g. In case of segmentation, the response

corresponds to the $C \times H \times W$ pixel-wise per-class probability map. In case of flow/depth prediction tasks, the responses correspond to the predicted depth or flow with the dimensions $1 \times H \times W$.

Stage 2 The training image X_t is fed through all the full-convolutional classifiers/regressors for all tasks (old tasks and new task t). For learning the new task t, label supervision loss is applied ($l1/l2$ regression loss or cross-entropy loss defined as L_s) is applied to the predictions of the new task. For older tasks, we want the predicted posterior distribution/ predicted values to remain close to its original distribution/ recorded values. To do so, we can apply a similarity inducing loss i.e. (knowledge distillation for probability maps or $l1/l2$ regression loss).

This technique works well when the distribution of training images in the new task is similar to the old task training distribution, but is likely to fail when the distributions differ significantly. In order to handle this issue, in addition to regularizing just by distillation, we regularize parameters which are important for the corresponding old task in its subnetwork (Shared feature layers f_s and task-specific layers f_i). For computing the per-parameter importance, we follow the paradigm proposed by Kirkpatrick et al. [12] of using the first-order diagonal approximation of Fisher Information Matrix. This approach requires storing certain samples for each of the older tasks. The diagonal approximation is computed using first-order gradient information from the samples. We use the per-parameter importance to scale the $l2$-regularization. We set the total number of samples stored per task as a hyperparameter.

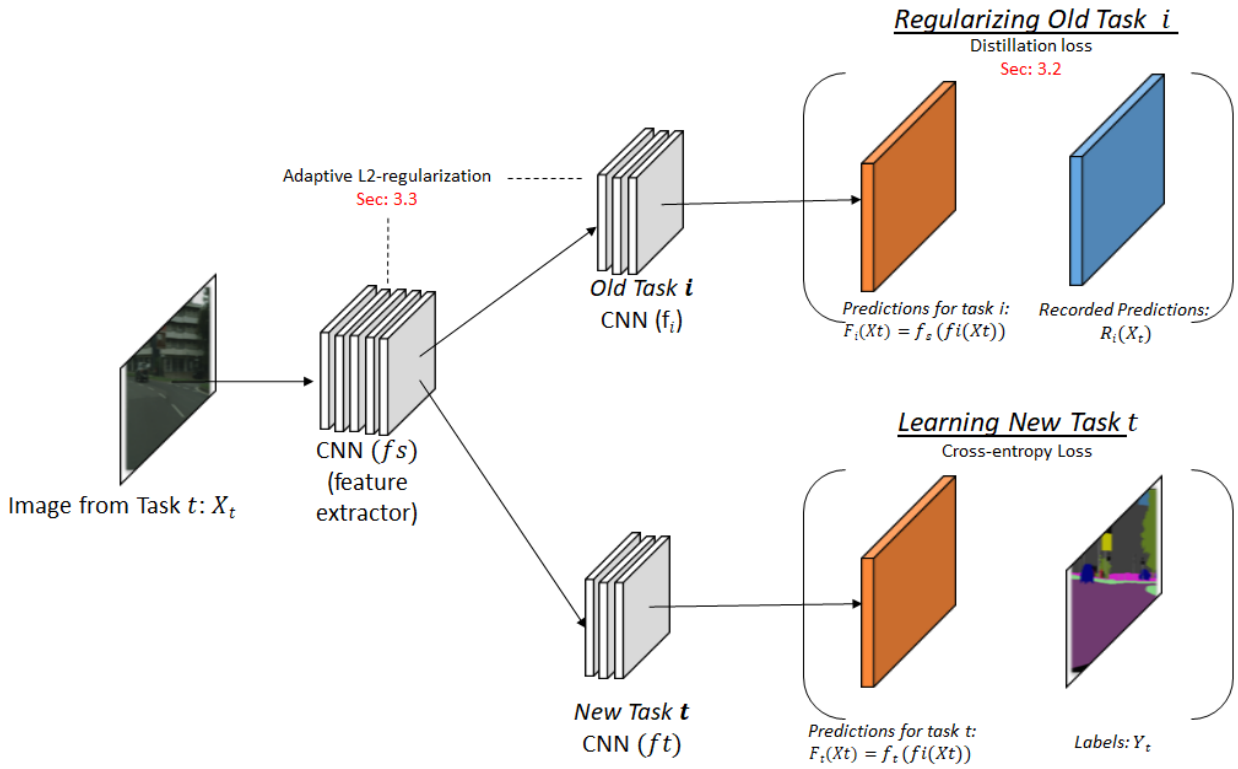


Figure 3.1: **Overview of the method.** Given an input image from the new task, we first record the responses for the input image in the first stage. Second, while learning on the new task, we regularize the network based on two losses: 1. Constraining output predictions 2. Constraining parameters based on their importance.

3.2 Distillation-based Objective

For preserving information of the older task-specific layers, we build on top of the approach proposed by LwF for classification [17], extending it to the dense prediction setting. For each new task t with a training image X_t, we record the responses from the old task R_t. The responses for the old task that we record are the final prediction map.

For segmentation tasks, the final prediction map will be $C \times H \times W$ dimensional per-pixel probability map, where $H \times W$ are dimensions of the training image and C is the number of classes.

The distillation objective for segmentation tasks is defined as follows:

$$L_d(y_o, \hat{y_o}) = \sum_{h=1}^{H} \sum_{w=1}^{W} \sum_{c=1}^{C} -y_o^{(hwc)} \log \hat{y_o}^{(hwc)} \tag{3.1}$$

where y_o and $\hat{y_o}$ are the current and the recorded pixel-wise probability maps of the old task and the new task respectively. For regression tasks, the final prediction map will be $H \times W$ dimensional map, we propose to use simple regression loss for preserving/distilling information. The distilling objective for regression tasks is defined as follows:

$$L_d(y_o', \hat{y_o}') = \beta \sum_{h=1}^{H} \sum_{w=1}^{W} \left\| y_o^{(hw)} - \hat{y_o}^{(hw)} \right\| \tag{3.2}$$

3.2.1 Shortcomings of Distillation

When the distributions between the training samples are similar, the distillation based method works very well. However, the distillation-based regularization might fail when the distribution of images significantly differs from the core distribution. As noted by Rebuffi et al. [22], for classification setting, the distillation based approach suffers from a significant drop in older task performance, if the following tasks have data from different distributions. We propose an improvement on the distillation-based objective by further regularizing weights which are essential to the previous tasks. The per-parameter importance can be computed using few samples stored for each task. The per-parameter importance approach tries to provide additional structural regularization if the new task domain varies significantly from the older task.

3.3 Adaptive l2-regularization objective

Our adaptive regularization follows up from the work of Kirkpatrick et al. [12] with Elastic Weight Consolidation (EWC). The EWC structural regularization model derives from specific neurological behavior seen in brains, where synapses(neurons) reduce their plasticity to remember crucial previously learned information.

One formal way of interpreting the importance of the parameter in artificial neural network space is to define stiffness of the weights/parameter. viz. how likely is the parameter/weight to change in a backpropogation step.

Kirkpatrick et al. [12] proposes to use diagonal of approximate Fisher information matrix among parameters to identify the stiffness per parameter. The approximation is calculated using squared gradient per parameter averaged over stored samples. i.e Let N be the number of samples, θ_k be one of the parameters in the network and $\nabla\theta_k$ be the gradient associated with the parameter. Per-parameter importance is computed as follows:

$$F_k = \frac{\sum_{k=1}^{N} \nabla\theta_k^2}{N} \tag{3.3}$$

Given per-parameter importance for each parameter i, the adaptive l2-regularization objective now becomes:

$$L_a = \sum_i \lambda F_i(\theta_i - \hat{\theta}_i)^2 \tag{3.4}$$

where λ is the memory strength hyper-parameter associated with the adaptive L2-regularization and θ_i is the current parameter value and $\hat{\theta}_i$ is stored parameter value for the network. We differ from the original EWC in one key aspect, the adaptive regularization loss is applied to both the shared feature layers f_s and task-specific layers f_i. This provides better regularization effect as compared to regularizing just the shared feature layers f_s naively.

Our proposed method acts as a better regularizer and provides a better update signal/

gradients to the old and new task.

3.4 Overall Loss

The final objective loss for the continual learning problem now composes of 3 terms: supervised loss for the new task L_s, similarity-inducing distillation loss for the older task L_d and adaptive l2-regularization loss L_a to compensate when the training distribution for the new task is significantly different from training distribution of the older tasks. The overall objective can be defined with the following equation:

$$L = L_s + L_d + L_a \tag{3.5}$$

where L is the overall loss.

Chapter 4

Experimental Results

We conduct extensive experiments to validate if our proposed methods can work on various real-world continual learning scenarios viz learning from more data, learning new annotations, learning new tasks from similar and different domains. We follow and extend the experimental design proposed by Li et al. [17] to evaluate each method on how much performance is preserved on old/previous tasks and the final performance on the new task. For dense prediction tasks, we focus on segmentation and depth prediction. For segmentation tasks, we report the mean class IOU as the evaluation metric (details in subsection 3), and For depth prediction tasks, we report the relative absolute error for the predicted depth.

4.1 Baselines and Method Variants

We compare our proposed variants against four baseline approaches, namely fine-tuning, feature extraction, joint/multi-task learning, uniform weight constraint.

Feature Extraction In Feature Extraction, the pre-trained strong model backbone is shared by all the tasks, but the trainable parameters for each task are their respective classification layer. The stronger model essentially acts as a feature extractor over which the classifiers are trained. This method severely restricts the learning capacity for the task.

Fine-tuning Fine-tuning is a standard method to train a CNN on new tasks or data points. The classification layers for the new task are initialized randomly. The shared backbone and the classification layers are learned jointly to minimize the loss on the new task. Since there is no "regularization" loss, the older task connections/network might change significantly, which causes the "catastrophic forgetting" behavior that we usually see. Often, the gradients are strong during the initial iterations, when the network is randomly initialized. In order to prevent early degradation caused by strong gradients, we follow the approach of warm-start (pre-training "classifiers" first) for the fine-tuning and all subsequent tasks.

Multi-task learning Multi-task learning focuses on learning all of the tasks simultaneously. Conventionally, it is the strongest method and establishes the upper bound of our method. The fundamental limitation for multi-task learning is that it requires data from all the tasks to be stored on the same machine. Additionally, this technique works best when we have the labels for the single image on both the tasks. This might not be the case in some of the setting, for example, when the two tasks come from different domains (Pascal / Cityscapes). In cases where we do not have training labels for images across tasks, we define multi-task learning as alternating between two datasets/tasks per-epoch.

Uniform Weight Constraint This baseline corresponds to uniformly applying regularization to each parameter in the shared network to keep the parameter same as before training. For this baseline, we apply L2-regularization loss on the shared backbone of the

network, constraining the network to stay in its original state prior to training.

We propose three variants of our method:

Constraining the weights This variant consists of only using adaptive L2 regularization loss from Section 3.3. The importance of each parameter is computed before training of the new task classifier. During the training of new task classifier, the network receives two losses: 1. The supervised loss for the new task classifier 2. adaptive regularization loss for shared backbone network. The loss helps in regularizing the shared backbone to the original state, but the classifier weights do not adapt to these changes in the shared backbone.

Constraining the old task outputs This variant consists of staged distillation-based model for dense prediction from Section 3.2

Joint Constraint This variant consists of the joint method with the overall loss as distillation + adaptive l2 regularization, as discussed in Section 3.4.

4.2 Datasets and Settings

In this work, we validate the effectiveness of our proposed method on various tasks based on two segmentation datasets, Pascal VOC 2012 and Cityscapes and depth estimation dataset NYUv2.

Pascal VOC 2012 Pascal VOC 2012 is a semantic segmentation dataset consisting of 8496 training images and around 2000 validation set images. The dataset originally defines 20 object categories (cars, people, animals, etc.) and background class. We use this dataset

configuration in various settings viz. incrementally adding data, learning on different domains, learning on different tasks.

We use the additional annotations on the Pascal VOC 2010 provided by Mottaghi et al. [21] with Pascal Context dataset to validate our method in incremental classes setting. We report performance on the validation set for the Pascal VOC 2012 dataset.

Cityscapes Cityscapes [5] is a semantic segmentation dataset focusing on the urban scene understanding. The dataset consists of 5000 high-resolution pixels annotated images covering a total of 50 cities. We work with the 2975 train set images and 500 images in the validation set. The dataset defines 19 categories containing a variety of objects seen in urban scenes (buildings, fences, signals, car, vans, etc.). We use this dataset configuration in the scenario to evaluate the effect of different domains in a continual learning.

NYUv2 NYUv2 dataset [30] consists of RGBD examples from 27 indoor scene categories taken from 464 different scenes. For training and testing splits, we use the same configuration as specified by the authors (795 training examples and 654 test examples).

4.3 Evaluation metrics

For evaluation of segmentation tasks, we report the common evaluation metric *mean of class-wise intersection over union(mIoU)* which is defined as :

$$mIoU = \sum_i \frac{n_{ii}}{(t_i + \sum_j n_{ji} - n_{ii})n_{cl}} \tag{4.1}$$

where n_{ij} is the number of pixels of class i predicted to class j and n_{cl} are number of classes and $t_i = \sum_j n_{ij}$ is the total number of pixels of class i. Higher mIoU indicates better

performance.

For evaluation of depth estimation tasks, we report the common evaluation metric *absolute relative difference(RD)* which is defined as :

$$RD = \frac{\sum_{y \in T} |y - y^*|}{|T|} \tag{4.2}$$

where y is the predicted depth at each pixel and T is the total number of pixels. Lower relative error indicates better performance.

4.4 Implementation details

For all our experiments, we use FCN8s as the base network. We initialize our FCN8s model with the conv1-5 VGG16 convolution weights. The fc6-fc7 convolution weights are initialized with Imagenet classification weights. The scoring layers for FCN8s were zero-initialized as suggested by Long et al. [18], as they reported neither better performance or faster convergence with random initialization. Dropout is included between the fc6 and fc7 layers where it used to be for the original classifier network. We initialize the transposed convolutions with up-sampling weights. More details about the FCN8s architecture can be found in Figure 4.1.

For optimization, we use the standard stochastic gradient descent with 10^{-10} as our initial learning rate, a momentum of 0.9 and weight decay of 0.0002 and unnormalized losses. For data augmentation, we random crop the images, randomly flip them across the horizontal axis, randomly rotate the images up to 10 degrees, scale to the resolution of 350×350 and normalize the images with ImageNet mean and variance. We choose the maximum number of epochs dependent on the setting and size of the dataset. Our models are trained on single

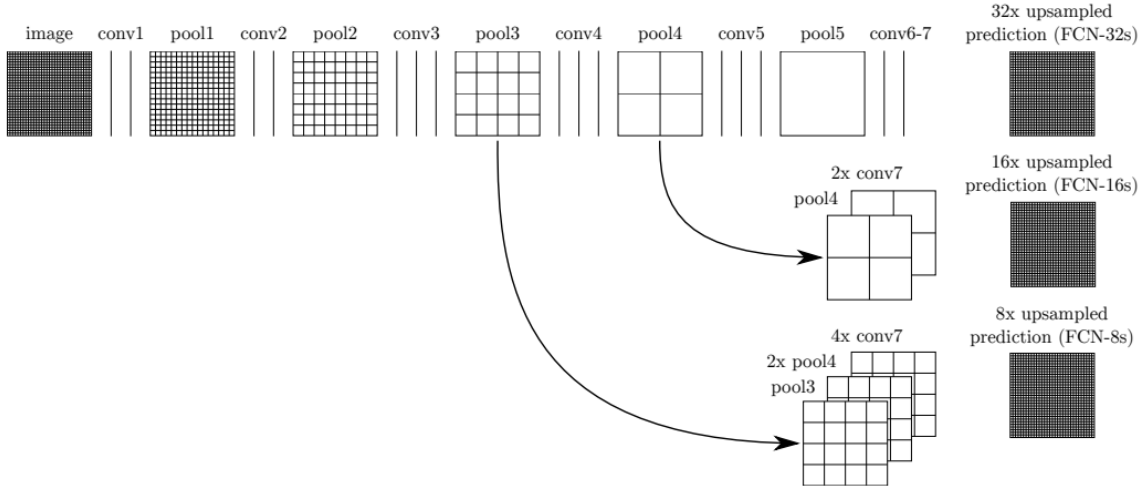


Figure 4.1: **Architecture of Fully convolutional networks from Long et al. [18].** We follow the FCN8s architecture as specified in the paper. The conv6 - conv5 layers are intialized with the Imagenet classification layers and dropout is included as is in the original VGG16 network.

instances of P100 or V100 Nvidia GPUs. Choice of the number of epochs is dependent on the setting.

4.5 Sequentially adding object classes

In this section, we focus on the task of learning classes incrementally on Pascal VOC. For evaluating our proposed method on how well classes are learned incrementally, we divide the Pascal VOC Segmentation task into two tasks splitting the 45 object classes. The first task contains 30 object classes, and the second task contains 15 object classes. We obtain this split by using Pascal in Detail challenge repository [1]. From an implementation perspective, we train all the baselines and our method variants for 30 epochs on both the tasks. For each task, the training data consists of images from the training set of Pascal VOC which contain at least one of the 30/15 classes. Similarly, for each task, the testing data consists of

images from the validation set of Pascal VOC which contain at least one of the 30/15 classes.

Table 4.1: **Evaluation of the baselines and method variants for the continual learning setting of incremental classes (30 class Pascal to 15 class Pascal).** Before column indicates performance after training on the first task. After column indicates performance after training on the second task.

Sequential Classes Pascal				
		Before	After	
		Pascal-30-class	Pascal-30-class	Pascal-15-class
Feature Extraction		-	36.79	36.10
Multitask		-	47.09	41.34
Finetuning		48.95	31.92	41.91
Uniform Weight Constraint		48.95	36.22	38.58
Constrain Weights (Adaptive)	Constrain Outputs			
✓	✗	48.95	47.50	39.54
✗	✓	48.95	47.86	42.19
✓	✓	48.95	48.86	40.05

Table 4.2: **Detailed evaluation of selected classes from the Pascal-30-classes after training on Pascal 15-classes task.**

Pascal 30 classes → Pascal 15 classes																			
Method	background	aeroplane	track	motorbike	fence	bicycle	diningtable	person	boat	train	bottle	tvmonitor	bus	pottedplant	car	chair	keyboard	dog	mIoU
Feature Extraction	71.4	24.5	3.05	24.35	1.03	19.57	6.22	25.3	10.57	21.73	10.57	13.24	39.64	4.31	21.09	5.52	1.67	24.97	36.79
Finetuning	73.64	31.09	23.15	19.84	6.94	21.65	9.16	29.92	18.46	21.02	6.92	19.37	28.06	9.12	19.45	7.77	12.06	31.17	31.92
Multitask	**81.11**	**46.49**	**22.80**	**30.39**	**9.60**	**29.60**	**13.75**	**45.14**	**27.52**	**44.61**	**17.32**	**33.30**	**52.19**	**16.44**	**31.83**	**10.55**	**17.79**	**40.82**	**47.09**
Adaptive Weight Constraint	80.93	49.39	31.36	30.80	7.80	32.45	11.51	44.21	27.89	38.07	18.58	32.66	51.85	15.48	36.89	9.48	17.95	42.27	47.50
Output Constraint	81.37	44.93	30.07	30.31	7.77	30.46	12.64	44.37	25.68	40.58	17.66	32.37	44.34	15.75	33.46	10.54	15.31	42.05	47.86
Joint Constraint	**81.16**	**48.56**	**27.10**	**35.90**	**7.41**	**32.03**	**10.18**	**45.56**	**26.81**	**39.86**	**17.73**	**34.14**	**53.70**	**12.91**	**37.08**	**9.07**	**13.53**	**42.58**	**48.86**

Table 4.1 compares the performance of the proposed method to the baselines. We can see that all three variants of our method surpass the fine-tuning and feature extraction baseline and achieve performance close to that of multi-task baseline. Among the three variants, the joint constraint performs the best in both tasks and even outperforms the multi-task baseline in the previous task. We believe this behavior is due to a synergistic interplay between regularizing on certain features and the constraining/regularizing the outputs. This improvement in the performance can be mainly seen in the following classes: airplane, car, dog, motorbike, person, bus. The relative comparison among the previous task classes can

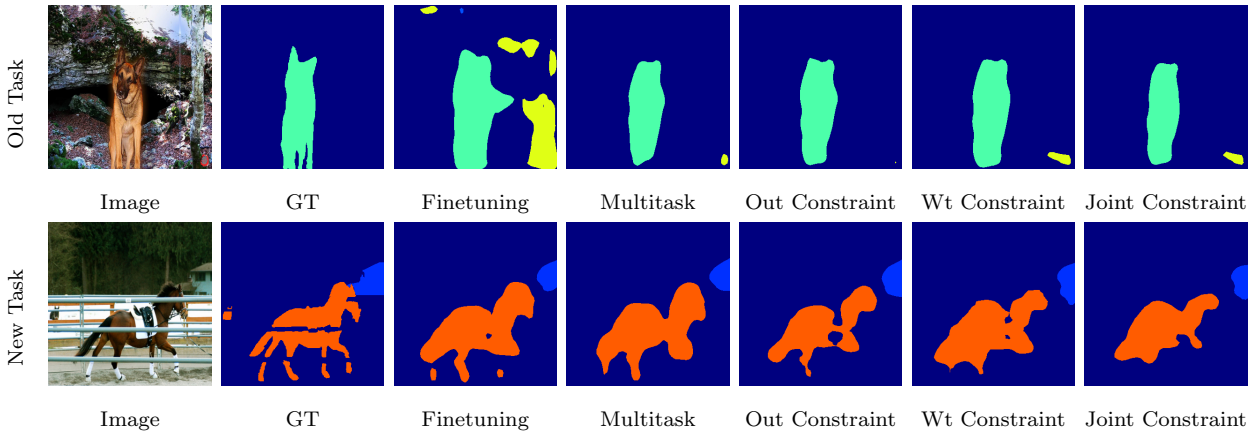


Figure 4.2: **Visualization of the segmentation map across the baselines and method variants after training on Pascal-30 and Pascal-15 classes.** We can see in the above visualization that the quality of the segmentation map for the proposed methods is quite better than fine-tuning for the new task.

be seen in table 4.2. We can see additional qualitative examples comparing the baselines and the variants in Figure 4.2.

4.6 Sequentially adding more training images

In this subsection, we evaluate the capabilities of our proposed approach to work well with new training images coming in batches. The goal of this experiment is to achieve performance as close as to the optimal joint training (multitask). One key difference between the sequential classes setting and sequential data setting is: we transfer the classifier weights to the new task and apply a regularization to the older task classifier. Our intuition is that information about the previously seen half of the data should be encoded in the classifier of the previous task (1st half of the dataset). We train all the baselines and our method variants for 30 epochs on both the tasks. We evaluate our experiments on the validation set of Pascal VOC.

Table 4.3: **Evaluation of the baselines and method variants for the continual learning task of incremental data (Pascal VOC 1/2 dataset to Pascal VOC 1/2 dataset).** Before column indicates performance after training on the first task. After column indicates performance after training on the second task. The numbers indicate the mIoU on the complete PASCAL validation set.

Sequential Pascal VOC				
		Before	After	
		Pascal-1	Pascal-1	Pascal-2
Feature Extraction		-	42.73	42.69
Multitask		-	59.81	59.81
Finetuning		55.52	56.43	59.81
Uniform Weight Constraint		55.52	55.62	58.01
Constrain Weights (Adaptive)	Constrain Outputs			
✓	✗	55.52	55.09	57.60
✗	✓	55.52	56.41	59.52
✓	✓	55.52	56.29	57.34

Experiments in Table 4.3 suggest that fine-tuning works best in this setting, attaining performance similar to multi-task learning surprisingly. We hypothesize this may be due to the distribution of training images between the two tasks being quite similar and most of the "information" can be contained within the first half of the dataset. This is consistent with the findings of Huh et al. [10], which shows that it is possible to achieve nearly the same performance on Imagenet even when the number of images per-class has been reduced by half. We believe that in real-world dataset setting, where the distribution between the two tasks is dissimilar, then our method should work optimally.

4.7 Adapting to different semantic class labels

In this subsection, we evaluate the capabilities of our proposed approach to work well with different training distributions and different semantic class labels. We choose the Pascal VOC and Cityscapes dataset for the two tasks as training distributions of the images among

them is mostly dissimilar, with some overlap among their labels. For the continual learning experiment, we first train on the generic Pascal VOC and then learn on the Cityscapes dataset. For the first task, we train over the training split of Pascal VOC and test over the validation split of Pascal VOC. For the second task, we train over the Cityscapes training split and test over the validation split of Cityscapes dataset.

Table 4.4: **Evaluation of the baselines and method variants for the continual learning setting of Pascal VOC to Cityscapes dataset.** Before column indicates performance after training on the first task . After column indicates performance after training on second task. The numbers indicate the mIoU on the complete Pascal validation set and Cityscapes validation set respectively.

Pascal VOC to Cityscapes				
		Before	After	
		Pascal VOC	Pascal VOC	Cityscapes
Feature Extraction		-	42.57	19.84
Multitask		-	59.39	44.96
Finetuning		57.52	22.02	46.14
Uniform Weight Constraint		57.52	25.22	43.74
Constrain Weights (Adaptive)	Constrain Outputs			
✓	✗	57.52	48.47	39.25
✗	✓	57.52	46.04	45.39
✓	✓	57.52	55.25	35.23

The performance of the proposed variants and baselines can be seen in table 4.4. Here, we see that our proposed variants perform significantly better than the fine-tuning. Among the proposed variants, weight constraint provides the best of both worlds (reasonable performance on both the datasets). Joint constraint regularizes very strongly on the older task and restricts growth on the new task.

4.8 Adapting to different tasks

In this subsection, we evaluate our proposed approaches to work well with tasks of different modalities (semantics and low-level vision, e.g., depth). Specifically, we evaluate on Pascal

VOC semantic segmentation dataset and NYUv2 depth estimation dataset. For the continual learning experiment, we first train on the generic Pascal VOC segmentation dataset and then train on the NYUv2 depth estimation dataset. For the first task, we train over the training split of Pascal VOC and test over the validation split of Pascal VOC. For the second task, we train over the NYUv2 training split and test over the test split of NYUv2 depth estimation dataset.

Table 4.5: **Evaluation of the baselines and method variants for the continual learning setting of Pascal VOC segmentation to NYUv2 depth.** Before column indicates performance after training on the first task . After column indicates performance after training on second task. The numbers correspond to the mIoU on the complete Pascal validation set and absolute relative error on NYUv2 test set. For absolute relative error, lower is better. For mIoU , higher is better.

Pascal Segmentation to NYUv2 Depth				
		Before	After	
		Pascal Segmentation↑	Pascal Segmentation↑	NYUv2 Depth ↓
Feature Extraction		-	43.71	1.19
Multitask		-	46.13	0.464
Finetuning		57.90	40.56	0.226
Uniform Weight Constraint		57.90	57.90	0.38
Constrain Weights (Adaptive)	Constrain Outputs			
✓	✗	57.90	50.67	0.225
✗	✓	57.90	57.90	0.39
✓	✓	57.90	56.19	0.29

The performance of the proposed variants and baselines can be seen as seen in the table 4.5. Here, we see that the output-constraint based regularization performs the best, getting better performance on the NYUv2 (lower relative absolute error) while retaining most of the performance on the Pascal VOC task. Weight-constraint and Joint-constraint based regularization preserve the performance on the older task almost entirely, while achieving a relative absolute error of 0.39 and 0.29 respectively. Varying the scaling for the EWC loss might lead to better performance trade-off with joint constraint.

4.9 Adapting to 3 tasks

In this subsection, we evaluate the proposed approaches to work well with more than two tasks. Specifically, we evaluate on 3 task setting where classes are added incrementally in batches of 15.

We divide the Pascal VOC Segmentation task into three tasks. We split the 45 object classes into three tasks of 15 classes namely Pascal-1, Pascal-2, and Pascal-3. We obtain this split by using Pascal in Detail challenge repository [1]. We train all the baselines and our method variants for 20 epochs on all the tasks. For each task, the training data consists of images from the training set of Pascal VOC which contain at least one of the 15-classes. Similarly, for each task, the testing data consists of images from the validation set of Pascal VOC which contain at least one of the 15-classes.

Table 4.6: **Evaluation of the baselines and method variants for the 3-task incrementally learning setting with Pascal VOC.** Phase-1 column indicates performance after training on the first task (Pascal-1). Phase-2 column indicates performance after training on the second task (Pascal-2). Phase-3 column indicates performance after training on the third task(Pascal-3).

Sequential Classes Pascal

		Phase-1	Phase-2		Phase-3		
		Pascal-1	Pascal-1	Pascal-2	Pascal-1	Pascal-2	Pascal-3
Feature Extraction		-	-	-	26.13	26.00	26.19
Multitask		-	-	-	35.39	35.36	35.53
Finetuning		33.83	31.20	34.69	30.77	34.39	35.86
Uniform Weight Constraint		34.11	32.18	35.33	32.19	35.44	35.91
Constrain Weights (Adaptive)	Constrain Outputs						
✓	✗	34.11	35.13	36.19	35.21	36.12	36.20
✗	✓	34.22	34.91	36.53	34.39	36.51	36.34
✓	✓	34.52	34.62	36.23	34.13	35.75	35.84

In Table 4.6, we see that the regularizing on output achieves the best overall performance, followed by adaptive weight-based regularization. Joint constraint-based regularization performs significantly better than fine-tuning and gets close to the multi-task performance but

does not out-perform the individual regularization methods. We show with this experiment that our methods can scale to increasing number of sequential tasks.

4.10 Controlling relative importance

In this subsection, we show how the relative importance between the previous tasks and new task can be controlled.

We propose to use weighing hyperparameter to control the relative importance. The weighing hyperparameter a is selected to be in the range 0.0 to 1.0, where $a = 0.0$ corresponds to no regularization on previous tasks and $a = 1.0$ corresponds to no learning of the new task. The overall equation can be described as follows:

$$L_t = L_{reg} \times a + L_s \times (1 - a) \tag{4.3}$$

Where L_t corresponds to total loss, L_{reg} corresponds to the total regularization loss, L_s corresponds to the supervised loss for the new task and a corresponds to the weighing hyperparameter.

To show the effect of the hyperparameter a, we focus on incremental learning from Pascal VOC (previous task) to Cityscapes (new task). In Figure 4.3, we see that as we decrease the importance of the previous tasks (Pascal VOC), the performance of the new task (Cityscapes) increases. This can be explained due to decreased regularization loss over the network.

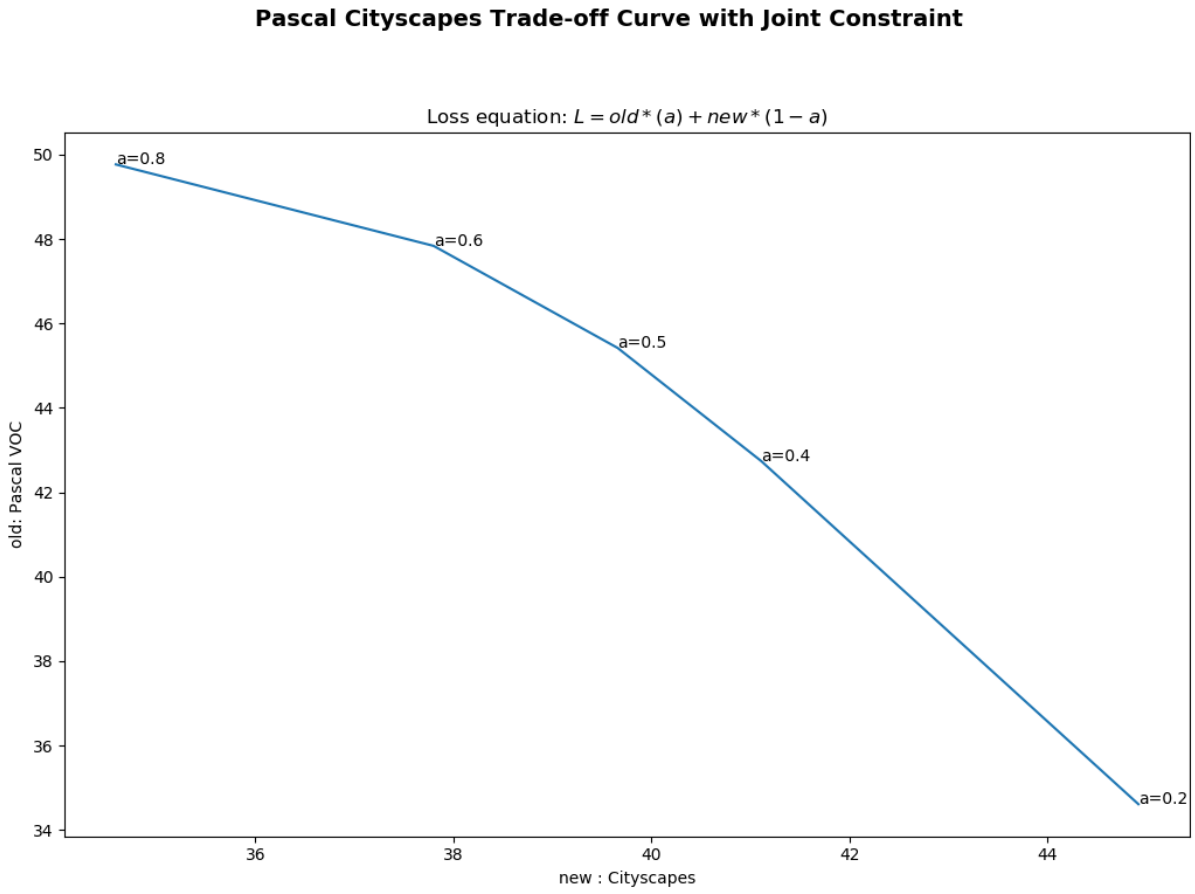


Figure 4.3: **Visualizing trade-off between the previous tasks and new task with hyperparameter a.**

Chapter 5

Discussion & Conclusion

In this work, we proposed a generic method for continual learning on the dense prediction based on the combination of constraining weights and posterior distribution. We showed with various experiments, that our method is capable of working in a variety of real-world settings viz learning from incremental dense labels, learning on different modalities, learning on different data domains. Our work is the first work to explore continual learning from a structured prediction perspective.

Several exciting avenues can be examined as a follow-up with this work. It would be interesting to see how the joint constraint-based approach works with the ResNet-based architectures. Additionally, we have only explored uniform output regularization with this work. Adaptively regularizing in the output space is one possible extension that could help in regularizing the deep network more effectively. The current incremental learning framework is limited to tasks which have the same training image modality. It should be possible to extend the incremental learning framework to images with different modalities (RGB, hyperspectral images, infrared) by having individual lower-level features and sharing intermediate-level features.

Bibliography

[1] Pascal in detail challenge. URL https://sites.google.com/view/pasd.

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015.

[3] Rich Caruana. Multitask learning. In *Learning to learn*, pages 95–133. Springer, 1998.

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016.

[5] 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.

[6] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[7] 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.

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

[9] Judy Hoffman, Dequan Wang, Fisher Yu, and Trevor Darrell. Fcns in the wild: Pixel-level adversarial and constraint-based adaptation. *arXiv preprint arXiv:1612.02649*, 2016.

[10] Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. What makes imagenet good for transfer learning? *arXiv preprint arXiv:1608.08614*, 2016.

[11] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. *arXiv preprint arXiv:1705.07115*, 2017.

[12] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, page 201611835, 2017.

[13] Iasonas Kokkinos. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory.

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

[15] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[16] Jeongtae Lee, Jaehong Yoon, Eunho Yang, and Sung ju Hwang. Lifelong learning with dynamically expandable networks, 2017.

[17] Zhizhong Li and Derek Hoiem. Learning without forgetting. In *European Conference on Computer Vision*, pages 614–629. Springer, 2016.

[18] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[19] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continuum learning. *arXiv preprint arXiv:1706.08840*, 2017.

[20] Zelun Luo, Yuliang Zou, Judy Hoffman, and Li Fei-Fei. Label efficient learning of transferable representations across domains and tasks. In *Conference on Neural Information Processing Systems (NIPS)*, 2017.

[21] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[22] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, and Christoph H Lampert. icarl: Incremental classifier and representation learning. *arXiv preprint arXiv:1611.07725*, 2016.

[23] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. *arXiv preprint arXiv:1705.08045*, 2017.

[24] Amir Rosenfeld and John K Tsotsos. Incremental learning through deep adaptation. *arXiv preprint arXiv:1705.04228*, 2017.

[25] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[26] Jeffrey C Schlimmer and Douglas Fisher. A case study of incremental concept induction.

[27] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014.

[28] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *arXiv preprint arXiv:1705.08690*, 2017.

[29] K. Shmelkov, C. Schmid, and K. Alahari. Incremental learning of object detectors without catastrophic forgetting. In *Proceedings of International Conference on Computer Vision*, 2017.

[30] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, pages 746–760. Springer, 2012.

[31] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[32] Sebastian Thrun. Is learning the n-th thing any easier than learning the first? In *Advances in neural information processing systems*, pages 640–646, 1996.

[33] Amal Rannen Triki, Rahaf Aljundi, Mathew B Blaschko, and Tinne Tuytelaars. Encoder based lifelong learning. *arXiv preprint arXiv:1704.01920*, 2017.

[34] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. *arXiv preprint arXiv:1702.05464*, 2017.

[35] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[36] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[37] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995, 2017.

[38] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. *arXiv preprint arXiv:1612.01105*, 2016.