

Few-Shot Meta Learners for Domain Adaptation

DISSERTATION SUBMITTED IN PARTIAL
FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE OF

Master of Technology
in
Computer Science

by

Surjayan Ghosh

[Roll No: CS-1926]

under the guidance of

Dr. Swagatam Das

Associate Professor

Electronics and Communication Sciences Unit



Indian Statistical Institute

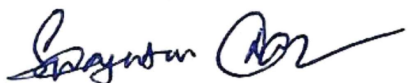
Kolkata-700108, India

July, 2021

To my Parents

CERTIFICATE

This is to certify that the dissertation entitled “**Few Shot Meta Learning for Domain Adaptation**” submitted by **Surjayan Ghosh** to Indian Statistical Institute, Kolkata, in partial fulfillment for the award of the degree of **Master of Technology in Computer Science** is a bonafide record of work carried out by him under my supervision and guidance. The dissertation has fulfilled all the requirements as per the regulations of the institute and, in my opinion, has reached the standard needed for submission.




Swagatam Das
Associate Professor,
Electronics and Communication Sciences Unit,
Indian Statistical Institute,
Kolkata-700108, INDIA

Acknowledgements

I would like to express my highest gratitude to my advisor, *Dr. Swagatam Das*, Associate Professor, Electronics and Communication Sciences Unit, Indian Statistical Institute, Kolkata, for his guidance and continuous encouragement. Without his support this work would not have been possible.

My utmost thanks goes to all the teachers of Indian Statistical Institute for their suggestions and discussions whenever I have needed them, which has undoubtedly helped me to improve my research work.

I would also like to thank all of my friends for their help and support. Last but not least, I am very much thankful to my parents and family for their everlasting support.



Surjayan Ghosh
Indian Statistical Institute
Kolkata - 700108 , India

Contents

1	Introduction	8
2	Related Works	9
3	Model Agnostic Meta Learning	11
3.0.1	What is Meta Learning?	11
3.0.2	Model Agnostic Meta Learning	12
4	Dataset for our Experiments	14
5	Model Architecture	15
6	MAML Experiment	16
6.0.1	Training :	16
6.0.2	Evaluation :	17
7	Results from Base MAML	18
7.1	Congiguration for Base MAML :	18
7.2	Accuracy plots in the four domains :	18
7.2.1	Procut Domain :	18
7.2.2	Art Domain :	19
7.2.3	Clipart Domain :	19
7.2.4	Real World Domain :	21
8	Modifications for Model Performance Enhancements	22
8.0.1	Dropout Layer:	22
8.0.2	Random Augmentation :	23
8.0.3	Label Smoothing :	23
8.0.4	Pre-trained Weights :	24
8.0.5	Learning Rate Scheduling :	24
9	MAML performance post Enhancements :	26
9.1	Configuration terminology and their descriptions :	26
9.2	Accuracy Plots under various configurations:	27
9.2.1	Product Domain :	27
10	Discussion : MAML Enhancement Configuration	30
10.1	Product Domain :	30
10.2	Art Domain :	30
10.3	Clipart Domain :	31

10.4 Real World Domain :	31
11 Model Agnostic Meta Learning with Gradient Surgery	32
11.0.1 What are contradicting tasks ?	32
11.0.2 Gradient Surgery :	32
12 Result & Discussion : MAML with Gradient Surgery	34
13 Domain Adaptation using Meta Learning	35
13.0.1 Meta Learning General Approach :	35
13.0.2 Supervised Domain Adaptation	36
14 Domain Adaptation Results	38
14.0.1 Baseline	38
14.0.2 Supervised Domain Adaptation	38
15 Discussion & Conclusion	41

List of Algorithms

1	Model-Agnostic Meta-Learning	13
2	MAML for Few-Shot Supervised Learning	17
3	Gradient Surgery MAML for Few-Shot Supervised Learning	33
4	MAML for Few-Shot Supervised Domain Adaptation	36

Abstract

Modern developments in Deep Learning and Machine Learning have shown great capacity to learn from labelled training data and perform remarkably well. However, there lies an inherent assumption that training data and the data which the learning algorithm might see during testing or deployment have the same distribution. However this might not be true in most cases. We call this problem domain shift. In addition to the fact that it might not be possible to collect data from every possible domain gathering labelled data is expensive and resource consuming. So there is a need to build a learning algorithm that can adapt to new domains effectively and from small training samples. Hence, we propose a Meta Learning based approach using a Few Shot Model Agnostic Meta Learning (MAML) Algorithm to tackle problem of domain adaptation.

Extensive experimentation is performed on the office-home dataset using various configurations of MAML and using recent advances in multi task learning like Gradient Surgery which is incorporated for efficient learning from tasks. We combine these approaches and build a Few Shot Meta Learner that has the best domain adaptation capability and is able to generalize to new tasks in new domains from small training samples.

Keywords: Meta Learning, Domain Adaptation, Model Agnostic Meta Learner, Gradient Surgery, Office Home

Chapter 1

Introduction

Modern Developments in the field of Learning Algorithms and Deep Learning have shown great capacity to learn from labelled training data and perform remarkably. However, there lies an inherent assumption that the data used during training and the data which the learning algorithm might see during testing i.e. performing in the field are having the same distribution. However this might not be true in most cases.

When our training data distribution is not same as our testing or working condition data, our learning algorithm behaves unpredictably and generally leads to poor or unreliable performance. This phenomenon is generally called Domain Shift.

There is also another aspect to domain shift. for example data from every domain might not be easy or at all possible to collect from. Hence, in this case it might become pretty useful if we could train our algorithm on domain data that is easy to gather from, and have our algorithm pick up generalized meta-information which transfers to other domains as well.

This is where Few Shot Learning comes in. It will be extremely useful in this case to have a Few Shot learner that can be trained to perform on data coming from the new hard to get data from domain with only one or a few samples.

Hence, for this purpose we will explore a Model Agnostic Meta Learning algorithm that is used for domain adaptation. There has been numerous studies on Meta Learning recently which explores the idea of learning to learn. We will use one of this Meta Learning techniques called Model Agnostic Meta Learner (MAML) which can be made to work on any learning algorithm trained with gradient descent.

We propose an algorithm to tackle the problem of supervised domain adaptation using MAML with Gradient Surgery. We propose a new way to combine tasks from selected domains keeping aside a unseen domain and train our learning algorithm and study its performance on the same tasks in the unseen domain and also on new tasks in the unseen domain.

We study our algorithm on the officehome dataset [1]. which contains images from 65 classes from 4 different domains.

Chapter 2

Related Works

Meta learning or as it is better known learning to learn [2] is a very well known topic which looks into training a meta learner that learns how to learn other learning-algorithms. In the current times methods based on gradient descent have been applied with huge success to Few Shot learning.

The authors of [3] propose a Meta-Learning technique that works for any learning algorithm that is trained with gradient descent. It can be applied to classification, regression, policy gradient reinforcement learning with few modifications. In the paper the authors introduces an algorithm that is capable of generalizing to new unseen tasks by only seeing a few examples. Hence this has a lot of application in Few Shot Learning techniques.

The model has an outer and an inner loop. The outer loop essentially takes a batch of tasks as an input and the inner loop runs on each each tasks separately. The parameters of the model are updated based on the learning's from all the tasks taken together, hence the model parameters are optimized to perform well on any new incoming tasks. The Model is not only agnostic to the learning techniques but also to the tasks. Even to methods in One Shot Learning belonging to state of the art category which are particularly designed for task of classification under supervision this technique holds up favourably.

A good number of work in the area of Adaptation to different Domains are based on keeping in check the the target error by a metric of discrepancy between target and source and source error [4]. In usual circumstances the issue is given a solution by lowering the shift between domain either in output space, feature space or input space [5] by using adversarial learning or a metric of discrepancy.

Another way tackle Domain Adaptation is : Domain generalization , which without the information of taget domain during training attempts to generalize the model to domain not seen before. There are various methods that have been proposed to learn different representations that can be generalized and transferred across domains. One way is to learn features which are specific to a task but invariant across

domains [6]. [6] learn auto-encoders of multi-task type to extract features which are in quality robust to domain variations and immune to variation.

The authors of [7] minimizes discrepancy of joint distribution after considering the conditional distribution of label space over input space.

Chapter 3

Model Agnostic Meta Learning

3.0.1 What is Meta Learning?

First, let us look at the word Meta. Meta essentially means one level of abstraction higher. For example Meta information means information about information.

Similarly usual learning algorithms like supervised classification learns how to classify 2 or more classes. A Meta Learning algorithm learns how to learn. This is the key idea of Meta Learning.

If we look at how we humans learn, we essentially learns new concepts and things fairly easily. For example a man who knows how to ride a bicycle finds it easy to learn to ride a motorbike or any other 2 wheeler easily. This essentially happens since the man is able to transfer a lot of the skills learned like balancing from riding a cycle to riding a motorbike.

The Meta Learning algorithm essentially attempts to mimic this fundamental aspect of learning, it tries to learn from other learning tasks, if there are n different learning tasks, then the meta learner is able to learn how to learn from the different tasks and is able to adapt to new tasks from only a few examples

Now, the tasks for the meta learning algorithm can be any well-defined task like supervised regression, classification, reinforcement learning etc.

Some examples:

- A supervised classification algorithm is able to learn how to differentiate between dogs and birds, upon only seeing a few samples of dog and bird images, even though it might have been initially trained to classify other non-related objects.
- A algorithm trained for playing one game using reinforcement learning learns to play a different game only from a few episodes.

3.0.2 Model Agnostic Meta Learning

MAML or Model Agnostic Meta Learning is a gradient descent based learning algorithm that is agnostic to the type of algorithm i.e it can be applied to any algorithm that is trained with gradient descent.

MAML trains over a plethora of tasks, it learns a representation that is simple to generalize or adapt to never before seen tasks via only a few steps of gradient descent. So, the MAML learns to find an optimal point (i.e. starting point) for the parameters of the model from where it can quickly adapt to a variety of task and also achieve that optimal parameter specific to a particular task quickly that is using only a limited count of gradient steps.

Below in Fig(1) is a visualization. As stated MAML tries to find a set of parameters that are optimal and is highly adaptable to different tasks. during meta learning represented by the bold line, MAML finds the set of optimal parameters that can be adapted just by a few learning/gradinet steps to optimal paremeters θ_i^* for task i

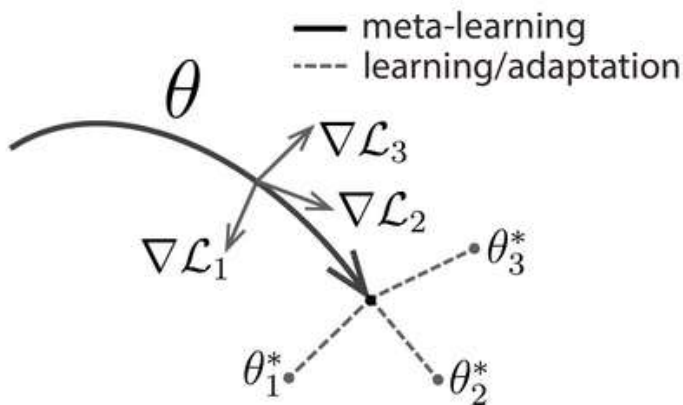


Diagram of the MAML approach.

Figure 3.1: Diagram of our model-agnostic meta-learning algorithm (MAML) which find optimal set of parameters that can be quickly adapted to different learning tasks i .source : [3] Figure-1

Algorithm 1: Model-Agnostic Meta-Learning

Data: $\rho(\tau)$: Tasks distribution
parameter: α, β : hyperparameters deciding the size of gradient descent step

- 1 initialize θ by a random process;
- 2 **while** *not done* **do**
- 3 Sample batch of tasks $\tau_i \rho(\tau)$; **end**
- 4 **for** *all* τ_i **do**
- 5 Taking all the K examples evaluate $\nabla_{\theta} \mathcal{L}_{\tau_i}(f_{\theta})$;
- 6 With gradient descent evaluate fine tuned parameters : $\theta'_i = \theta - \alpha \nabla_{\theta} \mathcal{L}_{\tau_i}(f_{\theta})$;
- 7 **end**
- 8 change $\theta \leftarrow \theta - \beta \sum_{\tau_i \rho(\tau)} \nabla_{\theta} \mathcal{L}_{\tau_i}(f_{\theta'_i})$;
- 9

lets say the the MAML is represented by the function f_{θ} with parameters θ . When the MAML is adapting to a new task \mathcal{T}_i we calculate a loss using the model prediction f_{θ} on the task \mathcal{T}_i and update θ to θ_i using one or more gradient update step,

$$\theta_i = \theta - \alpha \nabla_{\theta} \mathcal{L}_{\tau_i}(f_{\theta}) \quad (3.1)$$

Each task \mathcal{T}_i acts as a meta task for the MAML and its parameter θ are optimized aggregating the losses from all the tasks \mathcal{T}_i and optimizing f_{θ} with respect to θ

Mathematically the meta-objective is as follows:

$$\min_{\theta} \sum_{\tau_i \rho(\tau)} \mathcal{L}_{\tau_i}(f_{\theta'_i}) = \sum_{\tau_i \rho(\tau)} \mathcal{L}_{\tau_i}(f(f_{\theta - \alpha \nabla_{\theta} \mathcal{L}_{\tau_i}(f_{\theta})})) \quad (3.2)$$

Although the model objective function is calculated using each parameters θ_i for each task \mathcal{T}_i the meta-optimization is performed over the model parameters θ . And they are updated as follows:

$$\theta \leftarrow \theta - \beta \nabla_{\theta} \sum_{\tau_i \rho(\tau)} \mathcal{L}_{\tau_i}(f_{\theta'_i}) \quad (3.3)$$

where β is the meta-step size.

Chapter 4

Dataset for our Experiments

For our Experiments on MAML, we have used the Office-Home Dataset.

The Office-Home dataset is made to study object detection and domain adaptation algorithms using deep learning. object categories of 64 different kinds are present in the dataset sampled from things that are likely to be present in an office or home environment. For all 64 categories the images are available in 4 different domains - Art, Product, Clipart, Real World.



Figure 4.1: Here we see some samples of images present in the dataset; Art: artistic depictions of the objects, Product: images without background, Clipart: usual clipart images and Real-World: camera captured images in Real World setting. 16 of the 65 categories are shown here — sources : [1]

Object Categories :

Alarm Clock, Backpack, Batteries, Bed, Bike, Bottle, Bucket, Calculator, Calendar, Candles, Chair, Clipboards, Computer, Couch, Curtains, Desk Lamp, Drill, Eraser, Exit Sign, Fan, File Cabinet, Flipflops, Flowers, Folder, Fork, Glasses, Hammer, Helmet, Kettle, Keyboard, Knives, Lamp Shade, Laptop, Marker, Monitor, Mop, Mouse, Mug, Notebook, Oven, Pan, Paper Clip, Pen, Pencil, Postit Notes, Printer, Push Pin, Radio, Refrigerator, ruler, Scissors, Screwdriver, Shelf, Sink, Sneakers, Soda, Speaker, Spoon, Table, Telephone, Toothbrush, Toys, Trash Can, TV, Webcam

Figure 4.2: Object Categories, source : [1]

Chapter 5

Model Architecture

For our experiments we use the below Convolutional Neural Network Architecture :

1. **Convolutional Layer** : input channel - 3, output channel - 32, Kernel size - (3,3), padding - (1,1) with zeros.
2. **Batch Normalization Layer**
3. **Convolutional Layer** : input channel - 32, output channel - 32, Kernel size - (3,3), padding - (1,1) with zeros.
4. **Batch Normalization Layer**
5. **Convolutional Layer** : input channel - 32, output channel - 32, Kernel size - (3,3), padding - (1,1) with zeros.
6. **Batch Normalization Layer**
7. **Convolutional Layer** : input channel - 32, output channel - 32, Kernel size - (3,3), padding - (1,1) with zeros.
8. **Batch Normalization Layer**
9. **Fully Connected Linear Layer** : input size : 800, output size : 5

Chapter 6

MAML Experiment

For testing our MAML model. We use the officehome dataset. For our MAML experiments we ignore the domain shift aspect of our data and use any of the four domains available (Art, Clipart, Product, Real World) to train and test our model.

So for example, We choose 'Product' as our candidate model and then we only use the samples for training and both testing from that domain only.

We divide the sixty five classes in our Dataset into two sets:

- Training set (D_{train}) : containing 50 classes.
- Testing Set (D_{test}) : containing 15 classes.

While training our model we use the D_{train} dataset only.

Evaluation of the model is done on the D_{test} dataset.

6.0.1 Training :

We train our MAML to preform a 1-shot-5-way classification task. It means each task of the MAML is a classification of 5 classes randomly sampled from the Dataset D_{train} having 1 sample(or shot) per class.

The Algorithm is as shown below:

Algorithm 2: MAML for Few-Shot Supervised Learning

Data: $\rho(\tau)$: distribution over Tasks
parameter: α, β : step size hyperparameters

```
1 randomly initialize  $\theta$ ;  
2 while not done do end  
3   Sample batch of tasks  $\tau_i \rho(\tau)$ ;  
4   for all  $\tau_i$  do  
5     Sample  $K$  points of Data  $\mathcal{D} = \{x^{(j)}, y^{(j)}\}$  from  $\tau_i$ ;  
6     Evaluate  $\nabla_{\theta} \mathcal{L}_{\tau_i}(f_{\theta})$  using  $\mathcal{D}$  and  $\mathcal{L}_{\tau_i}$  where  $\mathcal{L}_{\tau_i}$  is either  
       Cross-Entropy Loss or Mean Squared Error(MSE);  
7     Compute adapted parameters with gradient descent:  $\theta'_i = \theta - \alpha$   
        $\nabla_{\theta} \mathcal{L}_{\tau_i}(f_{\theta})$ ;  
8     Sample datapoints  $\mathcal{D}'_i = \{x^{(j)}, y^{(j)}\}$  from  $i$  for the meta-update;  
9   end  
10  Change value of  $\theta \leftarrow \theta - \beta \nabla_{\theta} \sum_{\tau_i \rho(\tau)} \mathcal{L}_{\tau_i}(f_{\theta'_i})$  using each  $\mathcal{D}'_i$  and  $\mathcal{L}_{\tau_i}$   
    where  $\mathcal{L}_{\tau_i}$  is either Cross-Entropy Loss or Mean Squared Error(MSE);  
11
```

parameter terminologies:

- α = **learning rate**
- β = **meta-step size**
- Outer-loop iterations(line 2): **meta-iters**
- Task batch size(line 3): **meta-batch size**
- K (line 5) : **16** (1 for set \mathcal{D}_i and 15 for \mathcal{D}'_i)
- inner-loop iterations(line 6): **eval-iters**

6.0.2 Evaluation :

While evaluation we use the same Algorithms as Algorithm 2.

We use the D_{test} dataset to sample tasks in line 3 of Algorithm.

Since our MAML is trained to perform 1-shot-5-way classification task. While evaluating we sample 5 classes randomly and each having 2 sample each class(1 sample for fine tuning and 1 for testing our model performance).

We evaluate the model for 600 tasks and report the average accuracy.

Chapter 7

Results from Base MAML

Here We look at the results of our Experiment as explained in section 7.2 Using the base MAML Algorithm on our OfficeHome Dataset.

7.1 Congiguration for Base MAML :

- $\alpha = \text{learning rate} = 0.01$
- $\beta = \text{meta-step size} = 0.001$
- Outer-loop iterations(line 2): **meta-iters** = 60000
- Task batch size(line 3): **meta-batch size** = 3
- K(line 5) : **16** (1 for set \mathcal{D}_i and 15 for \mathcal{D}'_i)
- inner-loop iterations(line 6): **eval-iters** = 10

7.2 Accuracy plots in the four domains :

7.2.1 Procut Domain :

The final evaluation results (using best intermediate model after running 17000 meta-iterations) are :

- Train Dataset : accuracy = 73.1 %
- Test Dataset : accuracy = 49.2 %

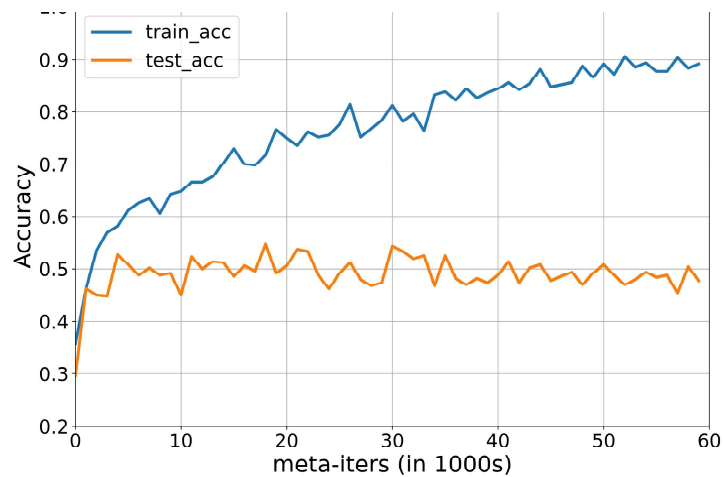


Figure 7.1: MAML Evaluation Accuracy vs meta-iters on Product Domain

7.2.2 Art Domain :

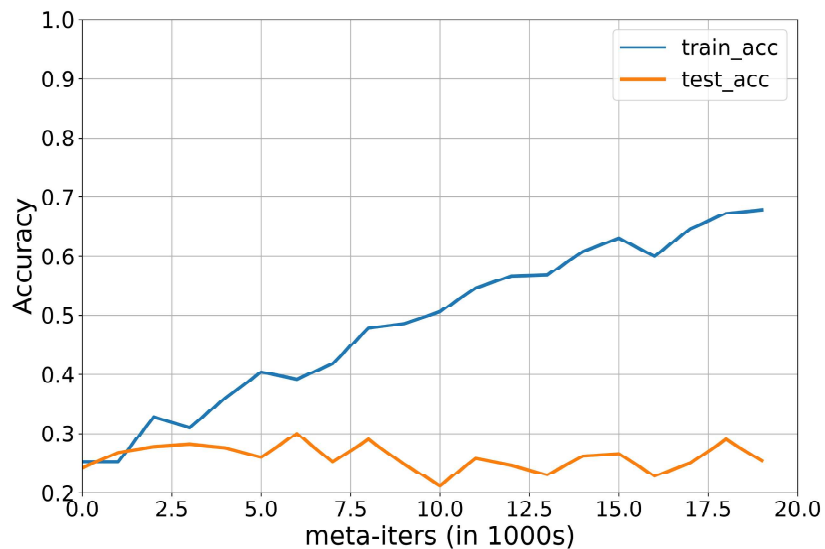


Figure 7.2: MAML Evaluation Accuracy vs meta-iters on Art Domain

The final evaluation results (using best intermediate model after running 6000 meta-iterations) are :

- Train Dataset : accuracy = 40.73%
- Test Dataset : accuracy = 26.60%

7.2.3 Clipart Domain :

The final evaluation results (using best intermediate model after running 5000 meta-iterations) are :

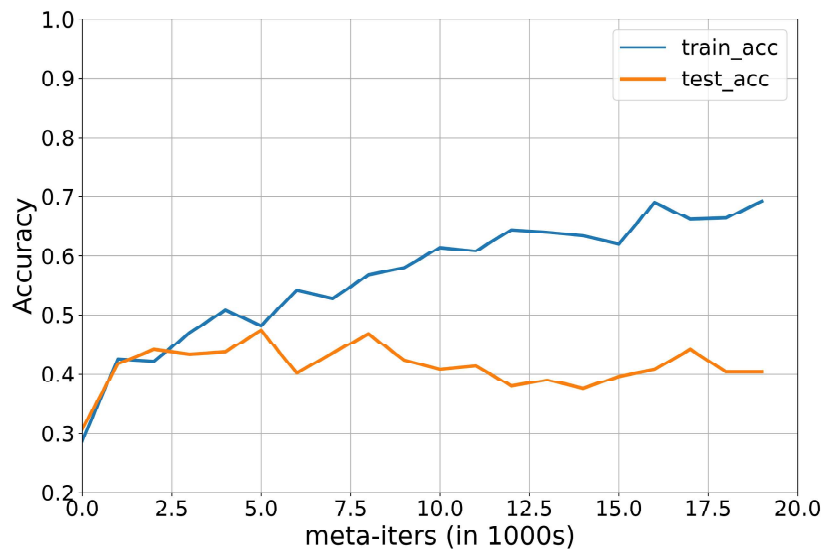


Figure 7.3: MAML Evaluation Accuracy vs meta-iters on Clipart Domain

- Train Dataset : accuracy = 52.63%
- Test Dataset : accuracy = 42.83%

7.2.4 Real World Domain :

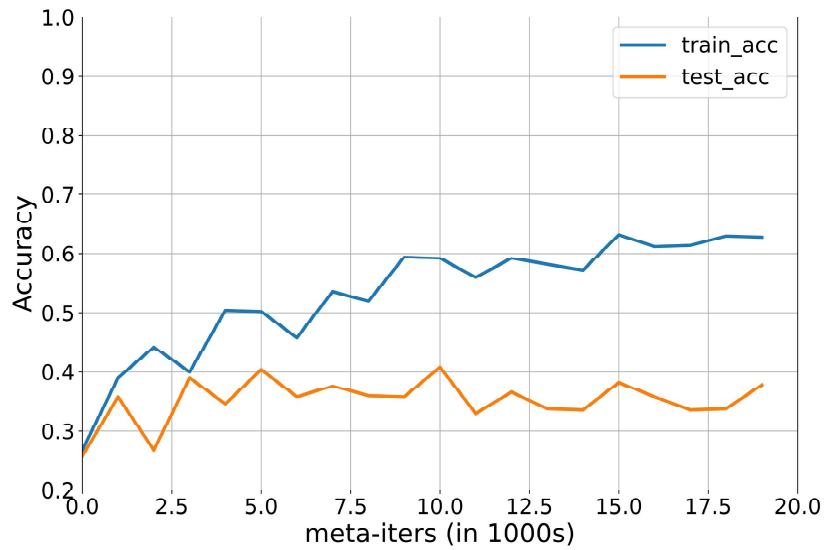


Figure 7.4: MAML Evaluation Accuracy vs meta-iters on Real World Domain

The final evaluation results (using best intermediate model after running 10000 meta-iterations) are :

- Train Dataset : accuracy = 56.26%
- Test Dataset : accuracy = 36.73%

Chapter 8

Modifications for Model Performance Enhancements

To improve our Model Performance compared to the baseline, we try several Improvements technique :

8.0.1 Dropout Layer:

Dropout is a form of regularization that has the effect of training many Neural Network architectures in parallel.

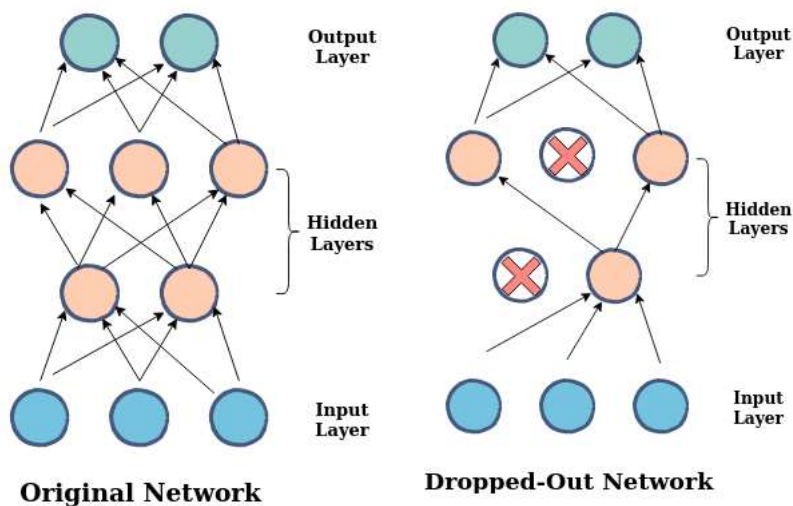


Figure 8.1: Original Network vs Dropout Network

While training some nodes in each layer are not used for predicting i.e they are "dropped out", this is done at random at every iteration. This has the effect of each layer looking different at every iteration with different number of nodes and connection. Hence, we get the effect of having different looking Network at every

iteration.

This has the effect of regularization since, the model is unable to rely on any specific node of a layer for prediction and hence has to learn a robust set of weights that doesn't give too much importance to any specific node.

8.0.2 Random Augmentation :

To overcome the over fitting to our training data, we apply augmentation techniques to our data to increase data points for training purpose. The augmentation is done online during sampling the data points line 5 of Algorithm 2 in section 7.1.

We use the following Augmentations :

- **Rotate** : Randomly rotate the image by an angle between -20 degrees to 20 degrees i.e. either clockwise or anticlockwise. around the centre of the image.
- **Shear along X & Y direction** : Randomly shear the image along X-direction(or Y) by selecting the shear value uniformly from the range $[-0.2, 0.2]$.
- **Translate along X & Y** : Randomly translate the image along X & Y direction by uniformly selecting the magnitude of translation from range $[-0.45, 0.45]$.
- **Color** : Randomly enhance the colour by factor between $[0.1, 1.9]$
- **Brightness** : Randomly enhance the colour by factor between $[0.1, 1.9]$

For all Augmentations we have used PIL library classes.

8.0.3 Label Smoothing :

Label Smoothing refers to changing hard target labels which are basically one-hot coded vectors with soft targets where the target values are a summation of values from an uniform distribution and the original one-hot hard target labels.

This improves the generalization of the multi-class classifier since it prevents the model from being too confident or too disappointed i.e. it has stops the loss from being extreme that is too high or too low.

As an example lets say we have a 4 class classification and our target label has 1 in the 0th index and 0 in all other index. What Label smoothing does is it puts

0.95 in 0th index and 0.05 in all other index. This improves the models robustness.

8.0.4 Pre-trained Weights :

We use our Network and train it on miniImageNet dataset.

The miniImageNet dataset contains 100 image classes randomly sampled from the The ImageNet Large Scale Visual Recognition Challenge - 2012. Each class comprises six hundred sample images of size 84x84. With a split of 64 training base class, 16 validation class and 20 novel classes.

We train the model for 40,000 iterations on the miniImageNet dataset and observe the performance on the validation set and use the one with highest accuracy score.

8.0.5 Learning Rate Scheduling :

This is a method to tweak (reduce or increase) the learnign rate while the model is being trained according to a pre-determined schedule.

We use the following learning rate schedulers :

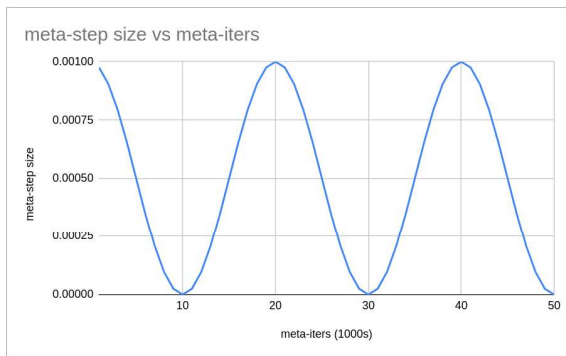


Figure 8.2: Cosine Annealing

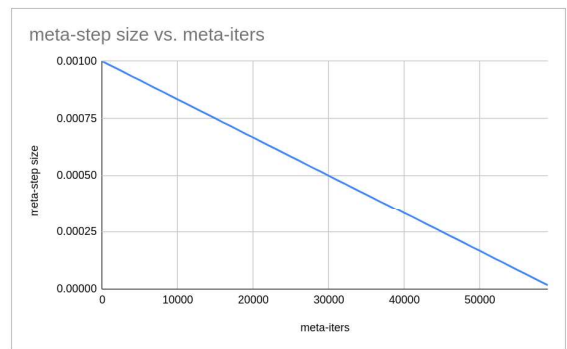


Figure 8.3: Linear Decaying

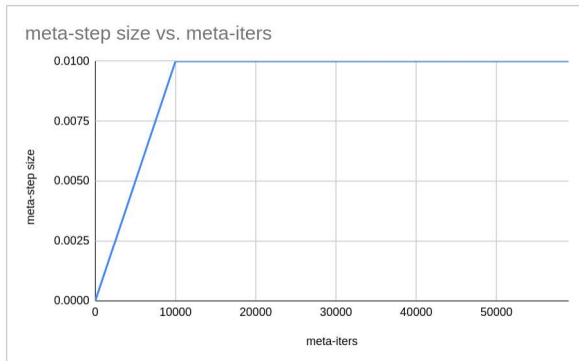


Figure 8.4: Warmup

Chapter 9

MAML performance post Enhancements :

In this chapter we will look at how the MAML performs after applying the enhancements we discussed in chapter 8.

First let us describe the enhancement configurations and their meanings:

9.1 Configuration terminology and their descriptions :

- **Rand-Augment-prob** : this parameter refers to the usage of Random Augmentation as described in section 8.0.2. e.g. **Rand-Augment-prob = 0.5** means there is a 50% probability of Random Augmentation being applied while sampling the images.
- **Dropout**: this parameter refers to the Dropout percentage we use in each layer. e.g **Dropout = 0.10**, implies a dropout percentage of 10% being applied.
- **Label Smoothing**: refers to use of label smoothing on the hard target labels. e.g. **Label smoothing = 0.05** refers to changing the indexes with 0 in the hard target to 0.05 and replacing the index with 1 with $(1 - 0.05) = 0.95$.
- **meta-step-size** : Refers to the meta step size parameter of the MAML algorithm , referred to by symbol β in algorithm 2.
- **LR** : refers to use of Learning Rate Scheduling. e.g. LR = Warm-Up refers to the use of Warm Up scheduler shown in Figure 8.4.
- **Pre-trained weights** : this refers to the use of pre-trained weights using the iniImageNet dataset as described in section 8.0.4

9.2 Accuracy Plots under various configurations:

We performed the experiment using the below mentioned enhancement configurations for our office-home dataset and selecting a specific domain (since here we are only looking at how the MAML performs and don't want our results to be distorted by domain shift)

The MAML performance in the following domains under the different enhancement configurations:

In the below figures, test_acc refers to the accuracy achieved using the test dataset classes. and train accuracy refers to the accuracy achieved using the training set classes.

9.2.1 Product Domain :

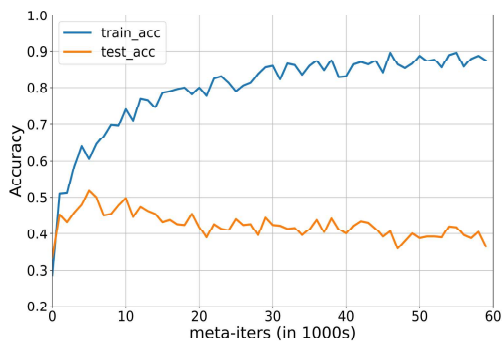


Figure 9.1: config 0

Rand-Augment-prob=0.5 Dropout=0.20
Label Smoothing=0.05 LR=Linear-Decay

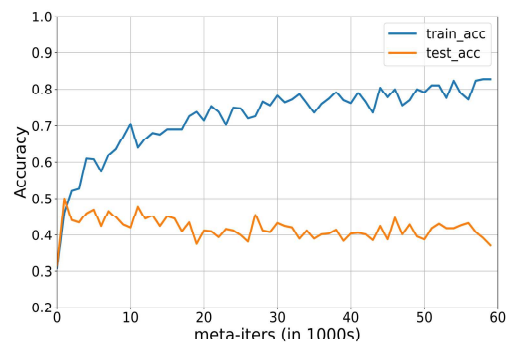


Figure 9.2: config 1

Rand-Augment-prob=0.5

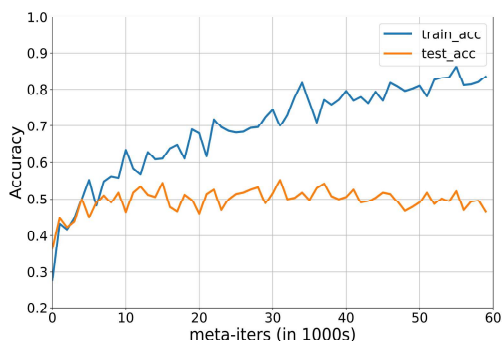


Figure 9.3: config 2

Rand-Augment-prob=0.5 Dropout=0.20
meta-step-size=0.0001

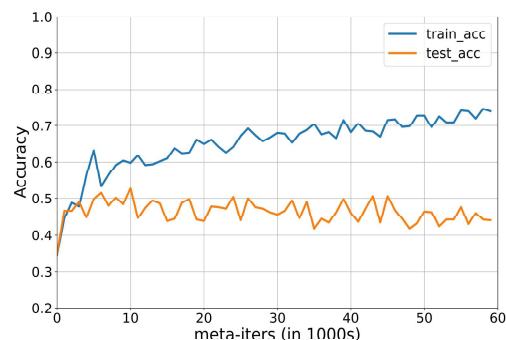


Figure 9.4: config 3

pre-trained Weights Rand-
Augment-prob=0.5 Dropout=0.10
Label-Smoothing=0.05

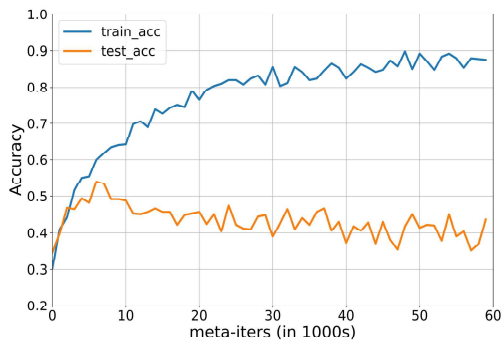


Figure 9.5: config 4

Rand-Augment-prob=0.5 Dropout=0.10
Label-Smoothing=0.05 LR=Warmp-Up

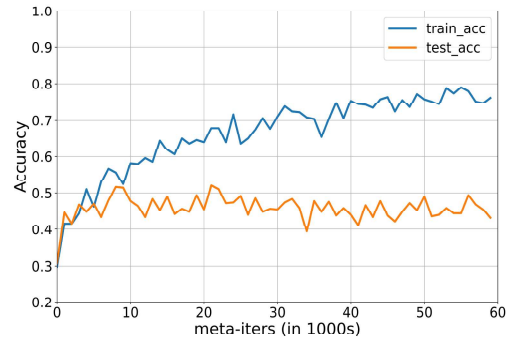


Figure 9.6: config 5

Rand-Augment-prob=0.5
Dropout=0.10 Label-Smoothing=0.05
meta-step-size=0.0001

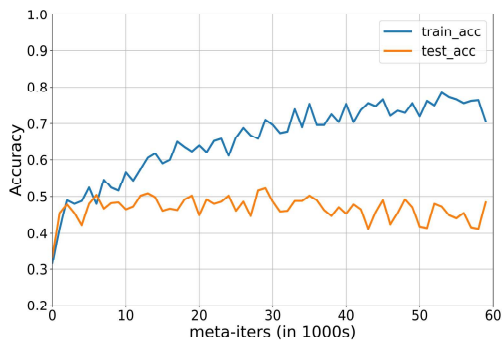


Figure 9.7: config 6

Rand-Augment-prob=0.5 Dropout=0.10
Label-Smoothing=0.05 meta-step-size=0.0001

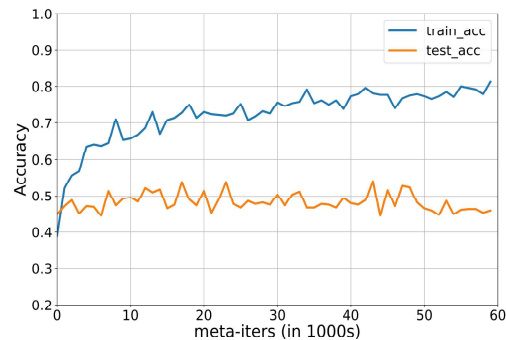


Figure 9.8: config 7

pre-trained weights

Here we can see, in Fig 9.1 adding Dropout and Learning Rate scheduling didn't solve our problem of over-fitting infactdue to the low learning rate at the later iterations, our model began performing worse on the test data classes.

in Fig. 9.3 we see reducing meta-step-size has a far better effect than using Learning Rate scheduling. Also we see from Fig 9.4 Label smoothing doesn't provide any increased benefits

In Fig.9.8 we see, that just using the pre-train weights makes the train_acc curve and test_acc curve much closer to each other, this leads us to believe the model is a more robust model and will best generalize to new data classes.

We conduct similar experiment for the other domains and have found the configurations perform similarly irrespective of the domain, We get the best performance from our configuration : 2 and configuration : 7 shown in Figure : 9.3 and Figure : 9.8 respectively.

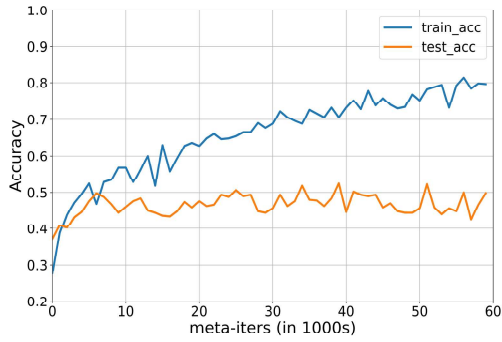


Figure 9.9: config 8
 Rand-Augment-prob=0.5 Dropout=0.10
 meta-step-size=0.0001

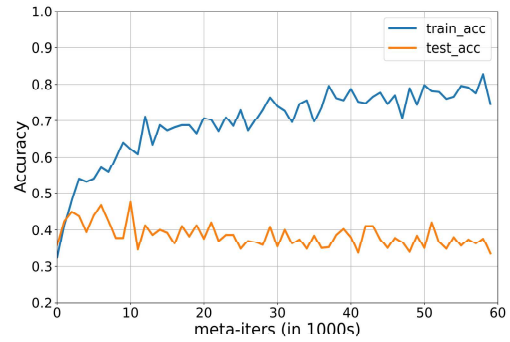


Figure 9.10: config 9
 Rand-Augment-prob=1

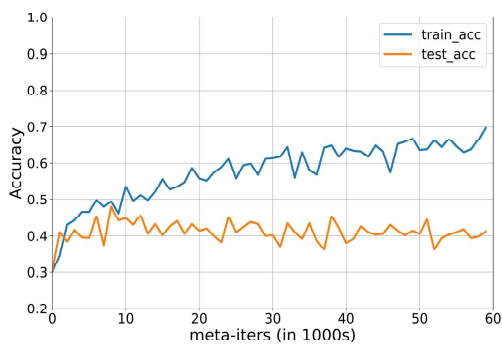


Figure 9.11: config 10
 Rand-Augment-prob=0.5 Dropout=0.25
 meta-step-size=0.0001

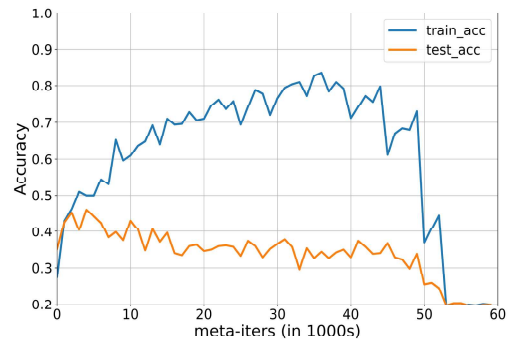


Figure 9.12: config 11
 Rand-Augment-prob=0.5
 Dropout=0.20 Label-Smoothing=0.05
 LR=WarmUp

Chapter 10

Discussion : MAML Enhancement Configuration

We see initially Base MAML is suffering from overfitting and we try to solve the problem using the methods mentioned in section 8.

Below we compare the results from our best 2 enhancement configurations from section 9 with base MAML configuration.

- Base MAML configuration : abbreviated as **Base** and is as mentioned in section 7.1
- configuration 2 : abbreviated as **config-2** having base MAML configuration with enhancement configuration - Rand-Augment prob=0.5, Dropout=0.20, meta-step-size = 0.0001
- configuration 7 : abbreviated as **config-7** having base MAML configuration with enhancement configuration - pre-trained weights, meta-step-size = 0.0001

Looking at the results in the four different domains:

10.1 Product Domain :

Configurations	train-set	test-set
Base	73.1%	49.1%
config-2	60.4%	48.6%
config-7	68.8%	50.30%

10.2 Art Domain :

Configurations	train-set	test-set
Base	40.73%	26.6%
config-2	39.36%	25.00%
config-7	59.9%	29.33%

10.3 Clipart Domain :

Configurations	train-set	test-set
Base	52.63%	42.83%
config-2	51.63%	40.76%
config-7	63.76%	42.30%

10.4 Real World Domain :

Configurations	train-set	test-set
Base	56.26%	36.73%
config-2	49.9%	36.30%
config-7	61.10%	37.73%

Hence, the enhancement configuration config-7 (using pre-trained weights from training our MAML network on miniimagenet dataset) gives us the best performance in terms of generalizing to new and unseen classes across any of the four domains.

Also the fact that we don't see a very high improvement using the various enhancement techniques suggests an inherently high contradictory nature between the classes of data objects present in the dataset. This fact is important to note and can be taken as a baseline reference point when we attempt domain adaptation using the same dataset.

Chapter 11

Model Agnostic Meta Learning with Gradient Surgery

In our experiment we use a method called Gradient Surgery [8] to combine the gradients in step 8 of Algorithm 1. i.e. the meta update step.

In this method we try to control the effect of contradicting tasks upon each other.

11.0.1 What are contradicting tasks ?

In the paper [8] it is hypothesized that an important issue in Meta Learning is the presence of contradicting gradients coming from the batch of task for each meta update in step 8 of Algorithm 1

Contradicting gradients for different tasks are those that point away from one another in the parameter space as shown by a negative inner product between the 2 gradients from 2 different tasks. This results in the resultant meta update being very small or none at all.

11.0.2 Gradient Surgery :

This method is used to nullify the effect of contradicting gradients which has a negative effect on the meta update as discussed above in section 4.1

We aim to solve this problem by taking 2 conflicting gradients say g_i and g_j as determined by a negative inner product. We simply replace g_i by $g_i = g_i - \frac{g_i \cdot g_j}{\|g_j\|^2} g_j$. that is we project g_i onto the normal plane of g_j . For gradients which have a positive inner product we do not alter them.

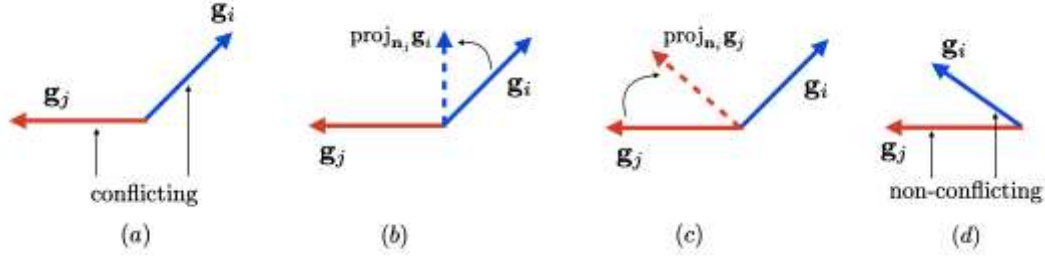


Figure 11.1: (a) conflicting gradients (b)projecting g_i onto the normal plane of g_j (c)projecting g_j onto the normal plane of g_i (d)non-conflicting gradients ; source : [8] Figure-2

Algorithm 3: Gradient Surgery MAML for Few-Shot Supervised Learning

```

1 Surgery( $\{\tau_K\}$ )
2  $g_k \leftarrow x \forall x \in \{\tau_K\}$ ;
3  $g_k^{Sur} \leftarrow g_k \forall k$ ;
4 for all  $i \in [0, \text{len}(\{\tau_K\})$  do
5   for  $j \sim_{\text{uniformly}} [0, \text{len}(\{\tau_K\}) \setminus i$  do
6     if  $g_j \cdot g_i < 0$  then
7        $g_i^{Sur} = g_i^{Sur} - \frac{g_i^{Sur} \cdot g_j}{\|g_j\|^2} g_j$ ;
8 return  $g^{Sur}$ ;
parameter:  $\alpha, \beta$  : hyperparameters that determine the size of step
Data:  $\rho(\tau)$  : distribution over Tasks
9 initialize  $\theta$  by a process that is random ;
10 while not done do
11   Sample batch of tasks  $\tau_i \sim \rho(\tau)$ ;
12   for all  $\tau_i$  do
13     Sample  $K$  points of Data  $\mathcal{D} = \{x^{(j)}, y^{(j)}\}$  from  $\tau_i$ ;
14     Evaluate  $\nabla_{\theta} \mathcal{L}_{\tau_i}(f_{\theta})$  using  $\mathcal{D}$  and  $\mathcal{L}_{\tau_i}$  where  $\mathcal{L}_{\tau_i}$  is either
15       Cross-Entropy Loss or Mean Squared Error(MSE);
16       With gradient descent compute fine tuned parameters:  $\theta'_i = \theta - \alpha$ 
17        $\nabla_{\theta} \mathcal{L}_{\tau_i}(f_{\theta})$ ;
18       For the meta-update sample points of data  $\mathcal{D}'_i = \{x^{(j)}, y^{(j)}\}$  from  $\tau_i$ ;
19   end
20    $\{gradList\} \leftarrow \nabla_{\theta} \mathcal{L}_{\tau_i}(f_{\theta'_i})$  using each  $\mathcal{D}'_i$  and  $\mathcal{L}_{\tau_i}$  ; where  $\mathcal{L}_{\tau_i}$  is either
21     Cross-Entropy Loss or Mean Squared Error(MSE);
22    $\{gradList\} \leftarrow \text{Surgery}(\{gradList\})$ ;
23   Update  $\theta = \theta - \beta \sum_{x \in \{gradList\}} x$ ;

```

Chapter 12

Result & Discussion : MAML with Gradient Surgery

We have seen how our best MAML performs with enhancements of pre-trained weights from training on miniimagenet dataset. This configuration overall gives us the best generalising ability to new unseen classes over all domains.

The below results are from using our MAML with Gradient Surgery and initializing with pre-trained weights on miniImageNet dataset :

Domain	train-set	test-set
Product	72.06%	50.56%
Art	61.70%	27.56%
Clipart	63.66%	43.80%
Real World	63.43%	37.46%

Hence we see the MAML with Gradient Surgery gives us a slight better performance for Product, Clipart and Real World Domain.

We see however the Art domain is particularly difficult for the model to classify images in. This is inline with our expectation since the Art domain has the most creative leeway in terms of representation of data classes, and even 2 images for the same class from the Art domain may look quite different

Hence, we will be using our Model Agnostic Meta Learner with Gradient Surgery and pre-trained weights and apply it for domain adaptation.

Chapter 13

Domain Adaptation using Meta Learning

We have two setting for our Domain Adaptation Experiment - 'Supervised' and 'Unsupervised'

1. Supervised Domain Adaptation : In the 'Supervised' setting our goal is to be able to train a model such that when it is shown samples of an object/class from a certain unseen domain it can quickly be trained on a few samples and be able to categorize that object.
2. Unsupervised Domain Adaptation : In the 'Unsupervised' setting our goal is to train our model such that upon showing a few samples of images for a certain object/class from certain easy to obtain domains it will be able to correctly classify these object in a different unseen domain

13.0.1 Meta Learning General Approach :

First discussing the part common to both the settings:

Our input space be \mathcal{X} and discrete label space be \mathcal{Y} . Let \mathcal{D} be a distribution over $\mathcal{X} \times \mathcal{Y}$. and our training dataset S_{train} comes from this distribution \mathcal{D} .

During the training phase our Meta Learning algorithm, has access to large number of labelled images belonging to many different classes, this dataset is called S_{train} . In this setting during training at every iteration i we sample K-shot-N-way classification task T_i ; meaning we sample N classes randomly from the labelled train dataset S_{train} and k shots for each class. Each of this task contain a small fine tuning training set called Support Set S_i (assumed to have labelled data points) and another set of data points (assumed to be unlabelled) called Query set Q_i .

While training the model is trained on the S_i set and then we calculate conditional probabilities of the classes in the Q_i set and calculate a loss based on this prediction. For multiple such tasks T_i we calculate this loss and aggregate them and then back-propagate the loss to update the model parameters.

For evaluating the model, the data comes from a different input space X' and label space y' to the S_{train} dataset. The testing dataset S_{test} comes from $X' \times Y'$.

Similar to the training mode, we sample tasks from the S_{test} dataset, where we have a small fine tuning set with labelled data and another evaluating set with unlabelled data for which the model predicts labels.

13.0.2 Supervised Domain Adaptation

In our experiment we have 4 domains - 'Art', 'Clipart', 'Product' and 'Real World'.

Algorithm 4: MAML for Few-Shot Supervised Domain Adaptation

parameter: α, β : hyperparameters to decide the size of gradient descent step

Data: $\rho(\tau)$: distribution over Tasks

Data: $D(\tau)$: List of Domains, test-domain

- 1 initialize θ by a random process ;
- 2 **while** *not done* **do**
- 3 Sample batch of tasks $\tau_i \rho(\tau)$; **end**
- 4 domain \leftarrow Sample batch of domains from $D(\tau) \setminus$ test-domain ;
- 5 **for** *all* τ_i **do**
- 6 Sample K points of Data $\mathcal{D} = \{x^{(j)}, y^{(j)}\}$ for each class in τ_i from sample domain for corresponding task τ_i ;
- 7 Evaluate $\nabla_{\theta} \mathcal{L}_{\tau_i}(f_{\theta})$ using \mathcal{D} and \mathcal{L}_{τ_i} where \mathcal{L}_{τ_i} is either Cross-Entropy Loss or Mean Squared Error(MSE);
- 8 With gradient descent evaluate fine tuned parameters : $\theta'_i = \theta - \alpha \nabla_{\theta} \mathcal{L}_{\tau_i}(f_{\theta})$;
- 9 For the meta-update sample points of data $\mathcal{D}'_i = \{x^{(j)}, y^{(j)}\}$ for classes in τ_i from corresponding domain;
- 10 **end**
- 11 Change $\theta \leftarrow \theta - \beta \nabla_{\theta} \sum_{\tau_i \rho(\tau)} \mathcal{L}_{\tau_i}(f_{\theta'_i})$ using every \mathcal{D}'_i and \mathcal{L}_{τ_i} where \mathcal{L}_{τ_i} is either Cross-Entropy Loss or Mean Squared Error(MSE);

12

After we have our S_{train} and S_{test} data-sets we proceed via the steps as stated in Algorithm 4.

Under the supervised setting we will have the following three different types of evaluation to test our learning Algorithm:

- Unseen Classes & Unseen Domain : For evaluation we sample classes from the S_{test} dataset which is unseen during training and the samples from every class is taken from the domain not seen during training. Here:
 1. We randomly sample N classes from the S_{test} dataset.
 2. For every class collect $(k + 1)$ shots where we use k shots to fine tune and keep one from each class for final evaluation to report accuracy.
- Seen Classes & Unseen domain : For evaluation we sample classes from the S_{train} dataset which is unseen during training and the samples from every class is taken from the domain not seen during training. Here:
 1. We randomly sample N classes from the S_{train} dataset.
 2. For every class collect $(k + 1)$ shots where we use k shots to fine tune and keep one from each class for final evaluation to report accuracy.
- Unseen Classes & Seen domain : For evaluation we sample classes from the S_{test} dataset which is unseen during training and the samples from every class is taken from either of the three domain seen during training. Here:
 1. We randomly sample N classes from the S_{test} dataset.
 2. For every class collect $(k + 1)$ shots where we use k shots to fine tune and keep one from each class for final evaluation to report accuracy.

Chapter 14

Domain Adaptation Results

14.0.1 Baseline

For Baseline we use the same MAML architecture as our Experiment. We pre-train our MAML network on the miniimagenet dataset.

The miniImageNet dataset contains 100 classes randomly chosen from ImageNet ILSVRC-2012 challenge with 600 images of size 84×84 pixels per class.

We mimic the evaluation methodologies used in our Supervised setting. We sample a N-Way-k-Shot task and train our pre-trained MAML network on data of k sample per class and calculate the accuracy on an evaluation set of 1 sample per class.

Domains	Art	Clipart	Product	Real World
Art	20.70%	19.70%	20.83%	20.23%
Clipart	19.96%	19.36%	19.90%	20.40%
Product	20.90%	19.63%	19.50%	20.56%
Real World	19.83%	20.73%	20.26%	21.43%

The above numbers are reported after running the evaluation 600 times and taking the average values.

14.0.2 Supervised Domain Adaptation

In Supervised setting the results of our 3 different evaluation strategies as explained in section 3.2, are as follows :

Our dataset officehome-65 contains 4 domains {Art, Clipart, Product, Real World}. Hence we have 4 configurations of our model each time keeping 1 domain as unseen and using the remaining 3 to build our models.

Below is a look at how domain adaptation is taking place using our Meta-Learner and Meta-Learner with Gradient Surgery.

Unseen Domain = {Real World}

Seen Domains during Training = {Art, Clipart, Product}

MAML :

1. Unseen Classes & Unseen Domain: 33.93%
2. Seen Classes & Unseen Domain : 50.23%
3. Unseen Classes & Seen Domain:
 - Art : 33.40%
 - Clipart : 34.03%
 - Product : 34.80%

GS MAML :

1. Unseen Classes & Unseen Domain: 35.90%
2. Seen Classes & Unseen Domain : 50.53%
3. Unseen Classes & Seen Domain:
 - Art : 37.03%
 - Clipart : 33.50%
 - Product : 34.97%

Unseen Domain = {Product}

Seen Domains during Training = {Art, Clipart, Real World}

MAML :

1. Unseen Classes & Unseen Domain: 45.36%
2. Seen Classes & Unseen Domain : 51.53%
3. Unseen Classes & Seen Domain:
 - Art : 52.03%
 - Clipart : 43.73%
 - Real World : 44.89%

GS MAML :

1. Unseen Classes & Unseen Domain: 43.40%
2. Seen Classes & Unseen Domain : 51.37%
3. Unseen Classes & Seen Domain:
 - Art : 49.37%
 - Clipart : 43.89%
 - Real World : 42.67%

Unseen Domain = {Clipart}

Seen Domains during Training = {Art, Product, Real World}

MAML :

1. Unseen Classes & Unseen Domain:
36.43%
2. Seen Classes & Unseen Domain :
51.90%
3. Unseen Classes & Seen Domain:
 - Art : 41.00%
 - Product : 36.83%
 - Real World : 36.76%

GS MAML :

1. Unseen Classes & Unseen Domain:
36.00%
2. Seen Classes & Unseen Domain :
52.43%
3. Unseen Classes & Seen Domain:
 - Art : 39.93%
 - Product : 35.90%
 - Real World : 36.26%

Unseen Domain = {Art}

Seen Domains during Training = {Clipart, Product, Real World}

MAML :

1. Unseen Classes & Unseen Domain:
32.33%
2. Seen Classes & Unseen Domain :
51.56%
3. Unseen Classes & Seen Domain:
 - Clipart : 32.06%
 - Product : 32.93%
 - Real World : 33.53%

GS MAML :

1. Unseen Classes & Unseen Domain:
30.7%
2. Seen Classes & Unseen Domain :
51.03%
3. Unseen Classes & Seen Domain:
 - Clipart : 32.10%
 - Product : 32.53%
 - Real World : 31.56%

Chapter 15

Discussion & Conclusion

Hence we see, the MAML algorithm out performs our Baseline model both for domain to domain adaptation and within domain.

The Baseline model scores indicate that the model has very limited predictive capabilities as we are seeing accuracy's around 20% for our 5-Way-1-Shot classification tasks. Even when looking at the predictive power within a single domain, for example being trained in the Product domain and evaluated in the Product domain we have an accuracy of 19.50%, the predictive power almost remains unchanged when evaluated for different domains (i.e. other than the one it is being trained on). This leads us to believe the model is unable to perform the 1-Shot learning in general and hence shows similar performance for both within domain and across domain evaluation.

The MAML algorithm improves the learning ability considerably. For example in looking at how the model performs in the Product domain (i.e. when it is trained by our supervised learning algorithm on the other domains - Art, Clipart, Real World) it gives us an accuracy of 51.53% in the unseen Product domain using seen classes from our S_{train} dataset. And when also using unseen classes(i.e. the classes in our S_{test} dataset) we get an accuracy of 45.36%. Eliminating domain shift and looking at the models performance in the previously seen domains just using unseen classes we see accuracy of 52.03% (Art) , 43.73 % (Clipart) and 44.89% (Real World). So, in our supervised learning setting the model is both able to generalize to new classes and new domains using 1-Shot learning.

The GS-MAML algorithm is not able to out perform the MAML algorithm in all cases. When looking at the performance in completely unseen domain and using unseen classes we see the GS-MAML slightly outperforms the MAML when evaluating in the Real World domain, in all other cases the MAML performs slightly better.

For future work it will be interesting to see how other methods can be used to tackle the inherently contradicting gradients we get from the MAML algorithm from different classes and domains which explains the algorithm for various configurations giving highest accuracy of about 51.37% for domain to domain adaptation. with a more robust way to combine the gradients it might be possible to break this ceiling.

Bibliography

- [1] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan, “Deep hashing network for unsupervised domain adaptation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5018–5027, 2017.
- [2] S. Thrun and L. Pratt, “Learning to learn: Introduction and overview,” 2017.
- [3] S. L. Chelsea Finn, Pieter Abbeel, “Model-agnostic meta-learning for fast adaptation of deep networks,” 2017.
- [4] K. C. A. K. F. P. Shai Ben-David, John Blitzer and J. nifer Wortman Vaughan, “A theory of learning from different domains,” 2009.
- [5] T. P. J.-Y. Z. P. I. K. S. A. A. Judy Hoffman, Eric Tzeng and T. Darrell, “Cycada: Cycle-consistent adversarial domain adaptation,” 2018.
- [6] M. Z. Muhammad Ghifary, W Bastiaan Kleijn and D. Balduzzi, “Domain general-ization for object recognition with multi-task autoencoders,” 2015.
- [7] M. G. Y. L. T. L. K. Z. Ya Li, Xinmei Tian and DachengTao, “Deep domain generalization via conditional invariant adversarial networks,” 2018.
- [8] A. G. S. L. K. H. C. F. Tianhe Yu, Saurabh Kumar, “Gradient surgery for multi-task learning,” 2020.